Apache Hadoop Cluster Operation to enable Security Experiments

Version 1.0, March 24, 2022

Abstract

This document provides step by step procedures to startup, run and shutdown the Apache Hadoop cluster for security tests and experiments.

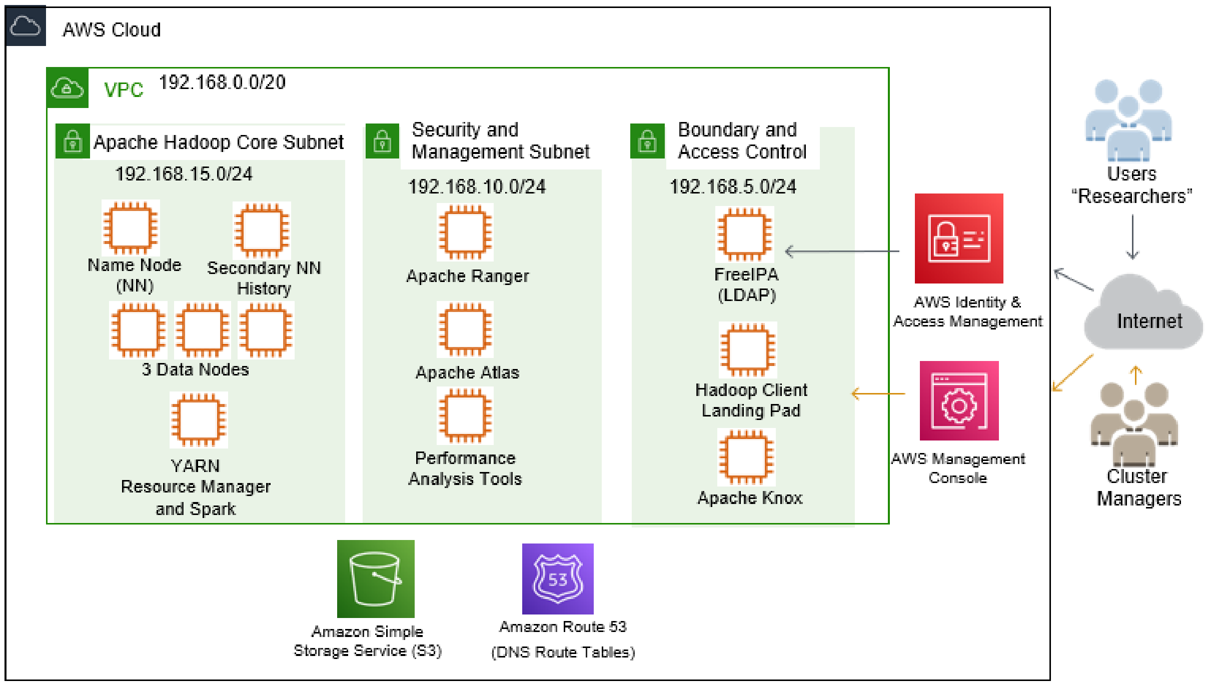
Anne Tall

anne.tall@gmail.com

Table of Contents

1 AWS EC2 Configuration Summary

The diagram and table below summarize the configuration of the EC2 instances for the Hadoop cluster:


AWS Cloud
VPC 192.168.0.0/20
Apache Hadoop Core Subnet
192.168.15.0/24
Name Node (NN)
Secondary NN History
3 Data Nodes
YARN Resource Manager and Spark
Security and Management Subnet
192.168.10.0/24
Apache Ranger
Apache Atlas
Performance Analysis Tools
Boundary and Access Control
192.168.5.0/24
FreeIPA (LDAP)
Hadoop Client Landing Pad
Apache Knox
AWS Identity & Access Management
AWS Management Console
Users "Researchers"
Internet
Cluster Managers
Amazon Simple Storage Service (S3)
Amazon Route 53 (DNS Route Tables)

Host Name	Primary SW	IP address	Subnet	Instance type	Elastic Block Storage (EBS) (GiB)	Fully Qualified Domain Name (FQDN)
pom	DataNode	192.168.15.131	15	m5.large	20	pom.sehadoop.test
flora	DataNode	192.168.15.132	15	m5.large	20	flora.sehadoop.test
alex	DataNode	192.168.15.133	15	m5.large	20	alex.sehadoop.test
babar	NameNode	192.168.15.135	15	m5.xlarge	10	babar.sehadoop.test
basil	2nd NameNode	192.168.15.115	15	m5.large	10	basil.sehadoop.test
celeste	Yarn	192.168.15.120	15	m5.large	10	celeste.sehadoop.test
zephir	Zookeeper	192.168.15.125	15	m5.large	10	zephir.sehadoop.test
pompadour	Ranger	192.168.10.110	10	m5.large	10	pompadour.sehadoop.test
troubadour	Atlas	192.168.10.115	10	m5.large	10	troubadour.sehadoop.test
cornelius	Knox	192.168.5.140	5	m5.large	10	cornelius.sehadoop.test
belle**	KMS-LDAP-FreeIPA	192.168.5.50	5	m4.xlarge	25	belle.sehadoop.test
truffles	Hive, HDFS Client (Landing Pad)	192.168.5.160	5	m5.large	10	truffles.sehadoop.test
izzy	Ambari / Management	192.168.10.105	10	m5.large	10	izzy.sehadoop.test

** belle was set up during previous experiments and is being reused in this cluster

2 Hadoop Start Up

2.1 Start Hadoop

The following command was previously executed on the NameNode (babar) as hdfs to start the Hadoop daemons:

```
start-all.sh
```

However, this start-all.sh command is deprecated, and the recommendation is to use different accounts to Start the HDFS and YARN, i.e., start HDFS with the hdfs account and start YARN with the yarn account. The command to start the MapReduce job history History Server is below.

on babar as hdfs

```
start-dfs.sh
```

on babar as hdfs (or as mapred)

```
mr-jobhistory-daemon.sh --config $HADOOP_CONF_DIR start historyserver
```

on celeste as yarn (or hdfs)

```
start-yarn.sh
```

Verify programs are running on babar as hdfs and as yarn if started yarn as yarn

```
for i in $(cat computers); do ssh $i "hostname -f; jps; echo -e '\n'";
done
```

Start the Spark History Server as hdfs on celeste

```
start-history-server.sh
```

2.2 Verify Status

On the NameNode, create a list of all the computers in the cluster in a (text) file titled “computers” (e.g., in /apache/hadoop) .

Execute the following command to verify node status:

```
for i in $(cat computers); do ssh $i "hostname -f; jps; echo -e '\n'"; done
```

2.3 View the Web Console

Using a web browser (e.g., Chrome), replace “<babar>” and “<celeste>” with the public IP address of the NameNode (babar) and ResourceManager (YARN) (celeste) as assigned in the AWS console. This IP address changes each time the instance is started unless it has been reserved on AWS.

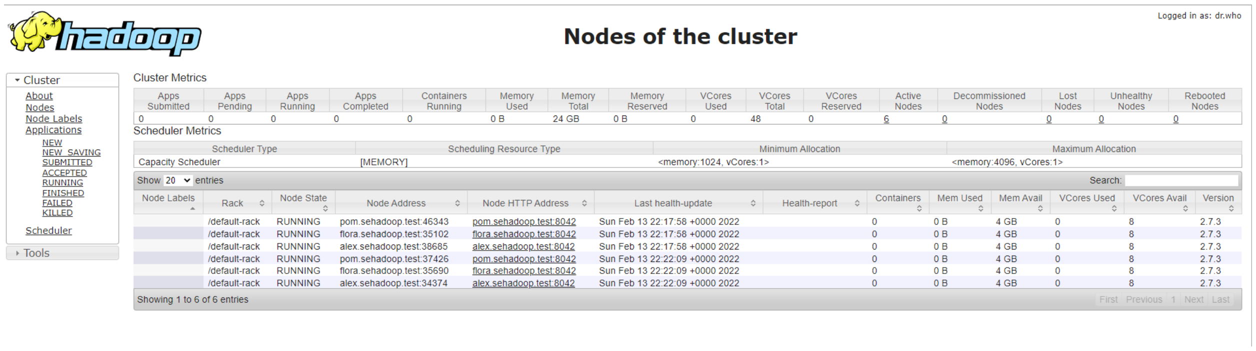
http://localhost:50070/dfshealth.html

Resource Manger status http://<celeste>:8088

NameNode status http://<babar>:50070

DataBlock Scanner Report http://<babar>:50075/blockScannerReport

Below are the screen shots of the Resource Manager (Nodes link) and NameNode Web User Interface (Web-UI)



Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	24 GB	0 B	0	48	0	6	0	0	0	0

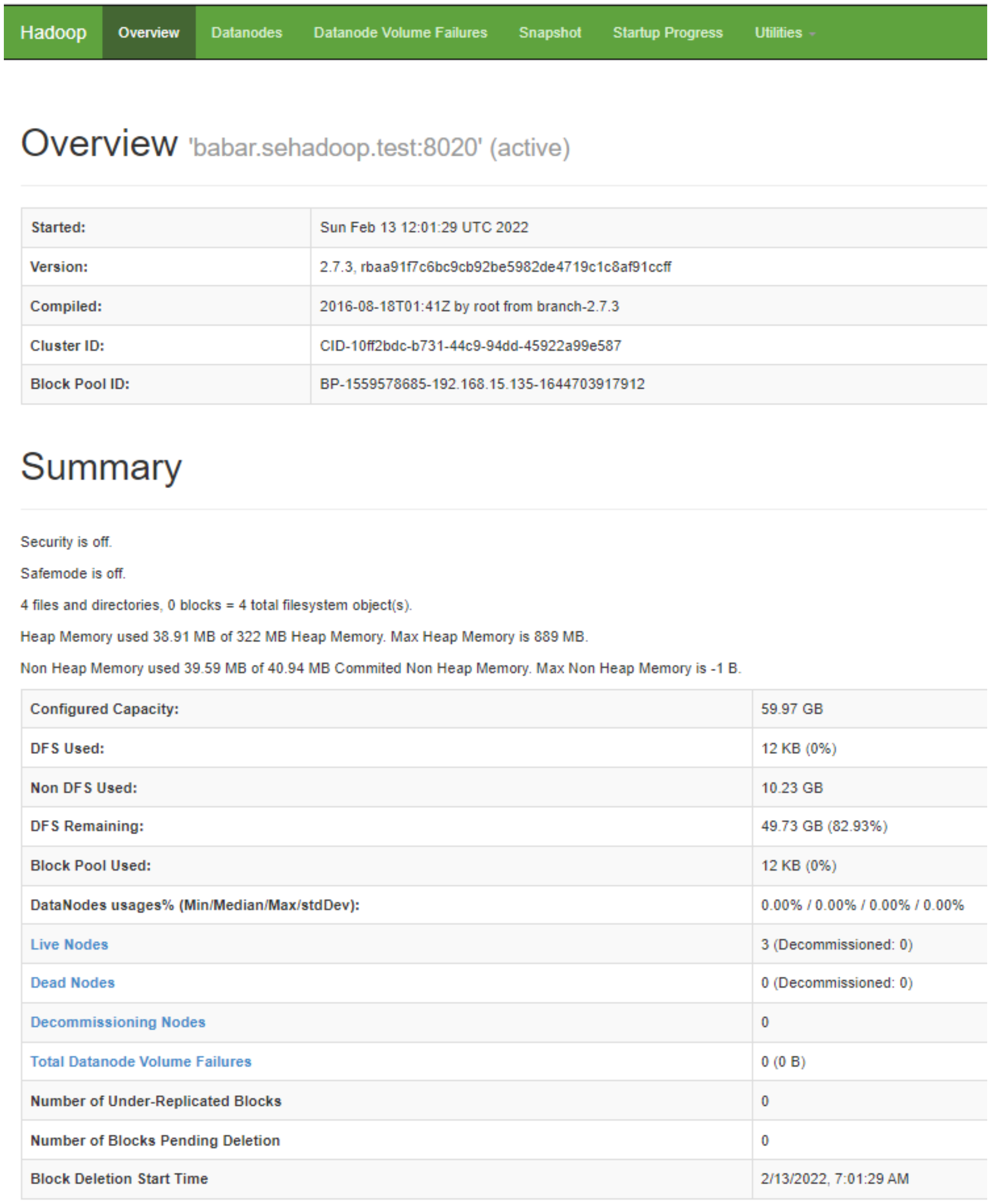
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:4096, vCores:1>

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	pom.sehadoop.test:46343	pom.sehadoop.test:8042	Sun Feb 13 22:17:58 +0000 2022		0	0 B	4 GB	0	8	2.7.3
	/default-rack	RUNNING	flora.sehadoop.test:35102	flora.sehadoop.test:8042	Sun Feb 13 22:17:58 +0000 2022		0	0 B	4 GB	0	8	2.7.3
	/default-rack	RUNNING	alex.sehadoop.test:38685	alex.sehadoop.test:8042	Sun Feb 13 22:17:58 +0000 2022		0	0 B	4 GB	0	8	2.7.3
	/default-rack	RUNNING	pom.sehadoop.test:37426	pom.sehadoop.test:8042	Sun Feb 13 22:22:09 +0000 2022		0	0 B	4 GB	0	8	2.7.3
	/default-rack	RUNNING	flora.sehadoop.test:35690	flora.sehadoop.test:8042	Sun Feb 13 22:22:09 +0000 2022		0	0 B	4 GB	0	8	2.7.3
	/default-rack	RUNNING	alex.sehadoop.test:34374	alex.sehadoop.test:8042	Sun Feb 13 22:22:09 +0000 2022		0	0 B	4 GB	0	8	2.7.3

Showing 1 to 6 of 6 entries



Overview 'babar.sehadoop.test:8020' (active)

Started:	Sun Feb 13 12:01:29 UTC 2022
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-10ff2bdc-b731-44c9-94dd-45922a99e587
Block Pool ID:	BP-1559578685-192.168.15.135-1644703917912

Summary

Security is off.

Safemode is off.

4 files and directories, 0 blocks = 4 total filesystem object(s).

Heap Memory used 38.91 MB of 322 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 39.59 MB of 40.94 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	59.97 GB
DFS Used:	12 KB (0%)
Non DFS Used:	10.23 GB
DFS Remaining:	49.73 GB (82.93%)
Block Pool Used:	12 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	3 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	2/13/2022, 7:01:29 AM

2.4 Stop Hadoop

Execute the following commands to stop HDFS, YARN, and shut down the cluster.

```
stop-all.sh
for i in $(cat computers); do ssh $i "sudo shutdown now; echo -e '\n'"; done
```

Note: The stop-all command is deprecated, to Stop HDFS, YARN and History Server, execute the following on babar as hdfs:)

```
stop-dfs.sh
```

```
mr-jobhistory-daemon.sh --config $HADOOP_CONF_DIR stop historyserver
```

as yarn / hdfs on celeste

```
stop-yarn.sh
```

Stop the Spark History Server as hdfs on celeste

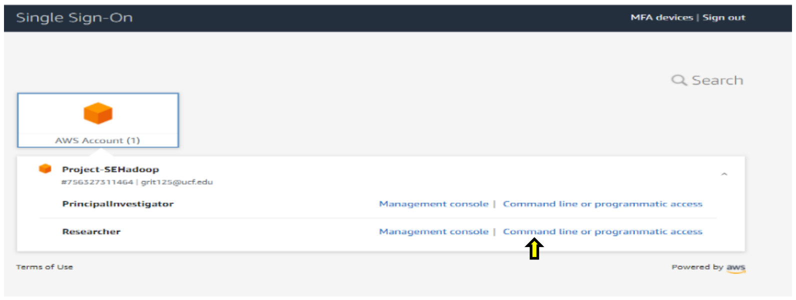
```
start-history-server.sh
```

3 Loading Data

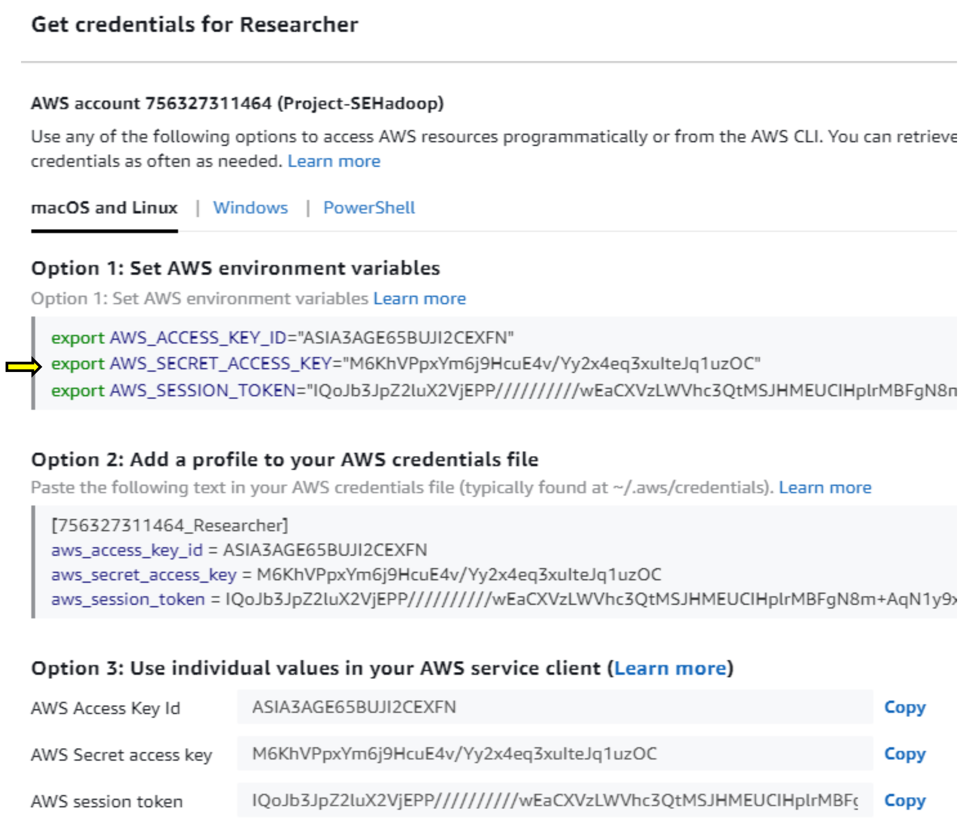
3.1 Moving Data to S3

To use the AWS CLI, a temporary key that expires in approximately one hour is created as follows:

(1) log into the console as usual, however, select the Command Line or programmatic access link, as indicated by the yellow arrow below, (instead of the Management Console).


Single Sign-On
MFA devices | Sign out
Search
AWS Account (1)
Project-SEHadoop
#756327311464 | grit125@ucf.edu
PrincipalInvestigator
Management console | Command line or programmatic access
Researcher
Management console | Command line or programmatic access
Terms of Use
Powered by aws

(2) Copy the first three export commands, as indicated by the yellow arrow below, then paste them on the command line.



Get credentials for Researcher

AWS account 756327311464 (Project-SEHadoop)

Use any of the following options to access AWS resources programmatically or from the AWS CLI. You can retrieve credentials as often as needed. Learn more

macOS and Linux | Windows | PowerShell

Option 1: Set AWS environment variables

Option 1: Set AWS environment variables Learn more

```
export AWS_ACCESS_KEY_ID="ASIA3AGE65BUJI2CEXFN"
export AWS_SECRET_ACCESS_KEY="M6KhVPpxYm6j9HcuE4v/Yy2x4eq3xulteJq1uzOC"
export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjEPP//////////wEaCXVzLWVhc3QtMSJHMEUCIHplrMBFgN8n
```

Option 2: Add a profile to your AWS credentials file

Paste the following text in your AWS credentials file (typically found at ~/.aws/credentials). Learn more

```
[756327311464_Researcher]
aws_access_key_id = ASIA3AGE65BUJI2CEXFN
aws_secret_access_key = M6KhVPpxYm6j9HcuE4v/Yy2x4eq3xulteJq1uzOC
aws_session_token = IQoJb3JpZ2luX2VjEPP//////////wEaCXVzLWVhc3QtMSJHMEUCIHplrMBFgN8m+AqN1y9x
```

Option 3: Use individual values in your AWS service client (Learn more)

AWS Access Key Id	ASIA3AGE65BUJI2CEXFN	Copy
AWS Secret access key	M6KhVPpxYm6j9HcuE4v/Yy2x4eq3xulteJq1uzOC	Copy
AWS session token	IQoJb3JpZ2luX2VjEPP//////////wEaCXVzLWVhc3QtMSJHMEUCIHplrMBF(	Copy

The following are examples of commands that can now be executed at the EC2 operating system command line interface to move data from the S3 bucket named "sehadoop-data" and the NameNode and Hadoop client:

```
aws s3 ls s3://sehadoop-data
aws s3 cp s3://<bucket name>/<folder>/<filename> <ec2filename>
```

examples:

```
aws s3 cp s3://sehadoop-data/Jobs /home/hdfs/Jobs --recursive
aws s3 cp s3://sehadoop-data/DataSet1/patients
/home/hdfs/DataSet1/patients
```

3.2 Moving HDFS Data to Local File System

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html

The following table contains a list of commonly used HDFS commands that are useful in navigating Hadoop file system.

Command	Description
hdfs dfs -ls hdfs dfs -ls ./data	display the list of Files and Directories in HDFS.
hdfs fsck /	check the health of the Hadoop file system.
hdfs dfs –mkdir /new_data	create a directory in HDFS.
hdfs dfs –touchz /directory/filename	create a file in HDFS with file size 0 bytes
hdfs dfs –du –s /directory/filename	check the file size.
hdfs dfs –cat /path/to/file_in_hdfs	reads a file on HDFS and prints the content of that file to the standard output.
hdfs dfs –text /directory/filename	takes a source file and outputs the file in text format.
hdfs dfs -copyFromLocal <localsrc> <hdfs destination> *Command:* hdfs dfs –copyFromLocal /home/hdfs/test /user/hdfs/test	copy the file from a Local file system to HDFS, identical to the -put command
hdfs dfs -copyToLocal <hdfs source> <localdst> *Command:* hdfs dfs –copyToLocal /user/hdfs/test /home/hdfs	copy the file from HDFS to Local File System
hdfs dfs -put <localsrc> <destination> *Command:* hdfs dfs –put /home/hdfs/test /user	copy single source or multiple sources from local file system to the destination file system.
hdfs dfs -get <src> <localdst> *Command:* hdfs dfs –get /user/test /home/hdfs	copy files from hdfs to the local file system.
hdfs dfs -count <path> *Command:* hdfs dfs –count /user	count the number of directories, files, and bytes under the paths that match the specified file pattern.
hdfs dfs -rm /user/hdfs/testfile	deletes a file, use -R to delete a directory and all contents of that directory

As hdfs on babar executed the following commands to move data to HDFS

```
hdfs dfs -copyFromLocal /home/hdfs/DataSet1/patients/* /DataSet1/patients
hdfs dfs -copyFromLocal /home/hdfs/DataSet1/social-media-users/*
/DataSet1/social-media-users
```

The following command provides a total count of files in each directory:

```
hdfs dfs -count /DataSet1/patients
```

Returns the value 1397

```
hdfs dfs -count /DataSet1/social-media-users
```

Returns the value 896

4 Running Apache Spark Programs

4.1.1 Dependencies - Bunsen

The Filter-Job-Synthea.py program developed for the SEHadoop experiments depends upon the installation of the Bunsen PySpark Tools. These tools are used to parse the HL7 formatted patient data.

Execute the following steps to install the Bunsen tools on celeste, babar, basil, pom, flora and alex.

as root in the /opt folder

(Note: this version of Bunsen (0.5.9) worked with the version of Synthea (2017-2018) that generated the HL7 data, I found that the later versions of Bunsen (1.4.7) did not work with later versions of Synthea (https://github.com/synthetichealth/synthea))

```
cd /opt

wget https://repo.maven.apache.org/maven2/com/cerner/bunsen/bunsen-
assembly/0.5.9/bunsen-assembly-0.5.9-dist.zip

unzip bunsen-assembly-0.5.9-dist.zip

~~rm bunsen~~

ln -s bunsen-assembly-0.5.9 bunsen

cp /opt/bunsen/jars/bunsen-spark-shaded-0.5.9.jar /apache/spark/jars/
```

Create a script for the Bunsen environment variables.

```
vi /etc/profile.d/bunsen.sh
```

put the following in this file

```
export PYTHONPATH=/opt/bunsen/python:$PYTHONPATH
```

```
chmod +x /etc/profile.d/bunsen.sh

source /etc/profile.d/bunsen.sh
```

4.1.2 Dependencies - python cryptography package and others

As root executed the following commands on celeste, pom, flora and alex to install the PySpark job dependencies

```
su root

python3 -m pip install --upgrade pip

python3 -m pip install pytz

yum install python3-devel cargo

python3 -m pip install setuptools_rust

python3 -m pip install cryptography
```

4.1.3 Running SEHadoop PySpark Jobs

The PySpark programs are designed to run sequentially with the output of each program providing input to the subsequent programs.

The programs were executed as hdfs from the hdfs home folder that contained the programs with the following commands

```
spark-submit Filter-Job-Synthea.py --jars /opt/bunsen/jars/bunsen-spark-
shaded-0.5.9.jar

spark-submit Filter-Job-SynSocial.py

spark-submit Join-Job.py

spark-submit Privacy-Job.py

spark-submit Analysis-Job-Hash.py

spark-submit Analysis-Job-Encrypt.py
```

4.1.4 Check the Hadoop Web UIs

To view the status of the programs, from my laptop browser, go to the ResourceManager webUI:

http://<<ipaddress of celeste>>:8088

To view the status of the file system, go to the NameNode webUI:

http://<ipaddress of babar>>:50070

5 Apache Ranger

Apache Ranger provides security policy management across the cluster.

There are three main components of Apache Ranger: Ranger-Admin console, Tag Sync and User Sync. These provide the management of HDFS, YARN and other Hadoop component service policies.

5.1 Apache Ranger Configuration

To run Apache Ranger Admin as ranger, executed the following, as root created directory

mkdir /var/run/ranger

chown -R ranger:hadoop /var/run/ranger

5.2 Install and Setup SOLR for Audit

Solr stores the audits and needs to be installed separately.

Reference: Apache Ranger Admin & Audits. Learn about Data security & Governance… | by Reetika Agrawal | Medium

The following commands and steps were executed to install and configure SOLR

```
cd /opt
```

```
mkdir solr
wget http://archive.apache.org/dist/lucene/solr/5.2.1/solr-5.2.1.tgz
tar xfz solr-5.2.1.tgz
ln -s solr-5.2.1 solr
chown - R root:hadoop solr
```

(1) Copy /apache/ranger/ranger-admin/contrib/solr_for_audit_setup to /opt/solr

```
cp -r /apache/ranger/ranger-admin/contrib/solr_for_audit_setup/ /opt/solr/
```

(2) list the contents of this folder and change permissions

```
ls -ltr
chmod +w install.properties
```

(3) change the install.properties in this folder

```
vi install.properties
```

change the following values

```
SOLR_INSTALL=true
SOLR_DOWNLOAD_URL=http://archive.apache.org/dist/lucene/solr/5.2.1/solr-
5.2.1.tgz
SOLR_RANGER_PORT=8983
```

note the following settings

```
SOLR_USER=solr
SOLR_GROUP=solr
SOLR_RANGER_DATA_FOLDER=/opt/solr/ranger_audit_server/data
SOLR_RANGER_HOME=/opt/solr/ranger_audit_server
```

(4) run ./setup.sh from this folder

```
./setup.sh
```

(5) once this is successful, execute the following (as solr) to start solr

```
/opt/solr/ranger_audit_server/scripts/start_solr.sh
```

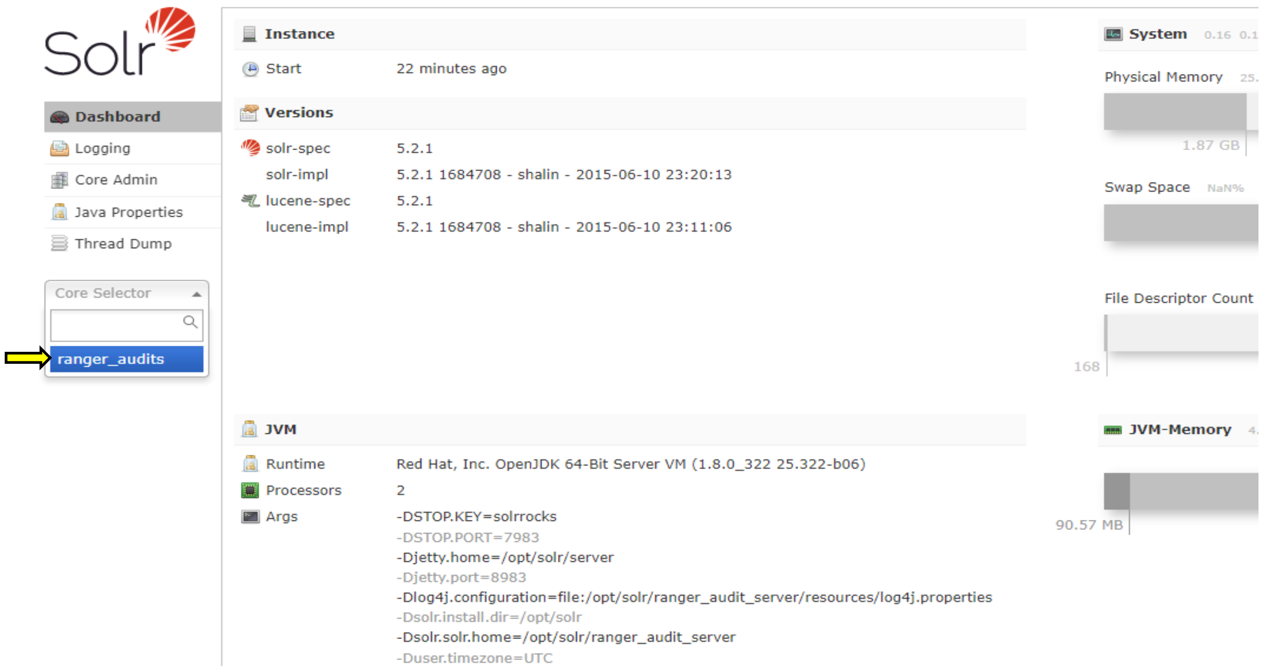
Execute the following as solr to shutdown Solr

```
/opt/solr/ranger_audit_server/scripts/stop_solr.sh
```

(6) check solr at the http://<solr host public ip>:8083

Note: Configure Ranger to use the following URL
http://pompadour.sehadoop.test:8983/solr/ranger_audits

At the WebUI, click on “Core Selector” and verify the ranger_audits database is available.



Instance
Start 22 minutes ago

Versions
solr-spec 5.2.1
solr-impl 5.2.1 1684708 - shalin - 2015-06-10 23:20:13
lucene-spec 5.2.1
lucene-impl 5.2.1 1684708 - shalin - 2015-06-10 23:11:06

JVM
Runtime Red Hat, Inc. OpenJDK 64-Bit Server VM (1.8.0_322 25.322-b06)
Processors 2
Args
```
-DSTOP.KEY=solrrocks
-DSTOP.PORT=7983
-Djetty.home=/opt/solr/server
-Djetty.port=8983
-Dlog4j.configuration=file:/opt/solr/ranger_audit_server/resources/log4j.properties
-Dsolr.install.dir=/opt/solr
-Dsolr.solr.home=/opt/solr/ranger_audit_server
-Duser.timezone=UTC
```

5.3 Apache Ranger HDFS Plugin

5.3.1 Enable the Apache Ranger HDFS Plugin

https://cwiki.apache.org/confluence/display/RANGER/Ranger+Installation+Guide#RangerInstallationGuide-Install/ConfigureRangerHDFSPlugin

On babar as hdfs

```
cd /apache
mkdir ranger
cd ranger
scp pompadour:/apache/ranger/ranger-2.1.0-hdfs-plugin.tar.gz
/apache/ranger
tar zxvf ranger-2.1.0-hdfs-plugin.tar.gz
ln -s /apache/ranger/ranger-2.1.0-hdfs-plugin /apache/ranger/ranger-hdfs-plugin
```

Link Hadoop configuration files

```
mkdir /apache/ranger/hadoop
mkdir /apache/ranger/hadoop/etc
ln -s /apache/hadoop/etc/hadoop /apache/ranger/hadoop/etc
ln -s /apache/hadoop/etc/hadoop /apache/ranger/hadoop

cd /apache/hadoop
ln -s /apache/hadoop/etc/hadoop /apache/hadoop/conf
```

Update following in the /apache/ranger/ranger/hdfs-plugin/install.properties file

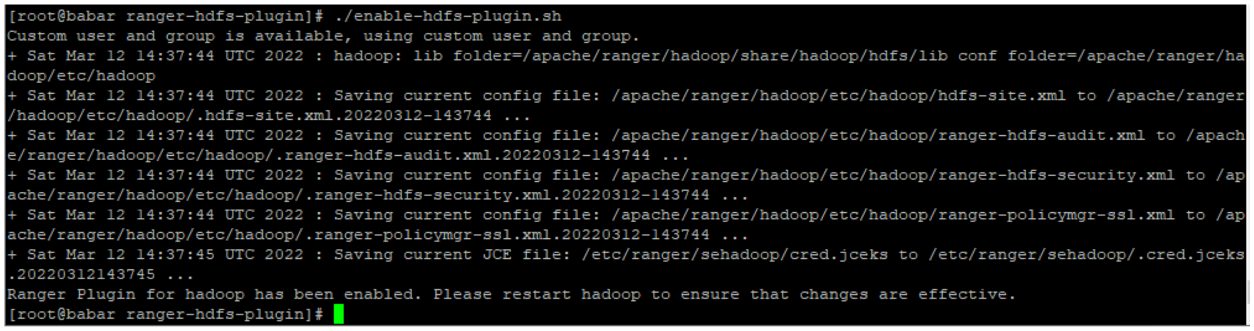
```
cd /apache/ranger/ranger-hdfs-plugin
cp install.properties install.properties-DEFAULT
vi install.properties
```

With the following values

```
POLICY_MGR_URL=http://pompadour.sehadoop.test:6080
SQL_CONNECTOR_JAR=/apache/ranger
REPOSITORY_NAME=sehadoop
XAAUDIT.SOLR.ENABLE=true
XAAUDIT.SOLR.URL=http://pompadour.sehadoop.test:8983/solr/ranger_audits
XAAUDIT.SOLR.USER=solr
XAAUDIT.SOLR.IS_ENABLED=true
XAAUDIT.SOLR.SOLR_URL=http://pompadour.sehadoop.test:8983/solr/ranger_audi
ts
```

Enable the HDFS plugin by executing the following command

```
su root
./enable-hdfs-plugin.sh
```



```
[root@babar ranger-hdfs-plugin]# ./enable-hdfs-plugin.sh
Custom user and group is available, using custom user and group.
+ Sat Mar 12 14:37:44 UTC 2022 : hadoop: lib folder=/apache/ranger/hadoop/share/hadoop/hdfs/lib conf folder=/apache/ranger/ha
doop/etc/hadoop
+ Sat Mar 12 14:37:44 UTC 2022 : Saving current config file: /apache/ranger/hadoop/etc/hadoop/hdfs-site.xml to /apache/ranger
/hadoop/etc/hadoop/.hdfs-site.xml.20220312-143744 ...
+ Sat Mar 12 14:37:44 UTC 2022 : Saving current config file: /apache/ranger/hadoop/etc/hadoop/ranger-hdfs-audit.xml to /apach
e/ranger/hadoop/etc/hadoop/.ranger-hdfs-audit.xml.20220312-143744 ...
+ Sat Mar 12 14:37:44 UTC 2022 : Saving current config file: /apache/ranger/hadoop/etc/hadoop/ranger-hdfs-security.xml to /ap
ache/ranger/hadoop/etc/hadoop/.ranger-hdfs-security.xml.20220312-143744 ...
+ Sat Mar 12 14:37:44 UTC 2022 : Saving current config file: /apache/ranger/hadoop/etc/hadoop/ranger-policymgr-ssl.xml to /ap
ache/ranger/hadoop/etc/hadoop/.ranger-policymgr-ssl.xml.20220312-143744 ...
+ Sat Mar 12 14:37:45 UTC 2022 : Saving current JCE file: /etc/ranger/sehadoop/cred.jceks to /etc/ranger/sehadoop/.cred.jceks
.20220312143745 ...
Ranger Plugin for hadoop has been enabled. Please restart hadoop to ensure that changes are effective.
[root@babar ranger-hdfs-plugin]#
```

ranger-hdfs-audit.xml files are created in /apache/hadoop/etc/hadoop folder

Copy the plugin jar files to the hadoop folder

```
cp -r /apache/ranger/ranger-hdfs-plugin/lib/*
/apache/hadoop/share/hadoop/hdfs/lib/
```

5.3.2 Start Apache Ranger to Conform HDFS Plugin is Operating

as Ranger on Pompadour

```
/apache/ranger/ranger-admin/ranger-admin start
```

Start HDFS as hdfs on babar and start YARN as yarn on celeste

5.3.3 Create HDFS Service on Apache Ranger

Go to the Apache Ranger WebUI

Create an HDFS Service

Service Name = sehadoop

Display Name = sehadoop

Username = hdfs

Password = `<xxx replace with password xxx>`

Namenode URL = hdfs://192.168.15.135:8020

Click on Test Connection to verify it connects successfully

Then create the service.

5.3.4 Create Users in Apache Ranger

Create local users on Celeste and Babar and Users in Apache Ranger that are authorized to execute the PySpark programs which access the DataSet1 data

name	**group on Ranger**	**group on OS (celeste and babar)**
bob	researchr	hadoop_users
lang	researchr	hadoop_users
chris	researchr	hadoop_users
aki	researchr	hadoop_users

Create OS users and groups on celeste and babar , also added hdfs and yarn to the hadoop_user group

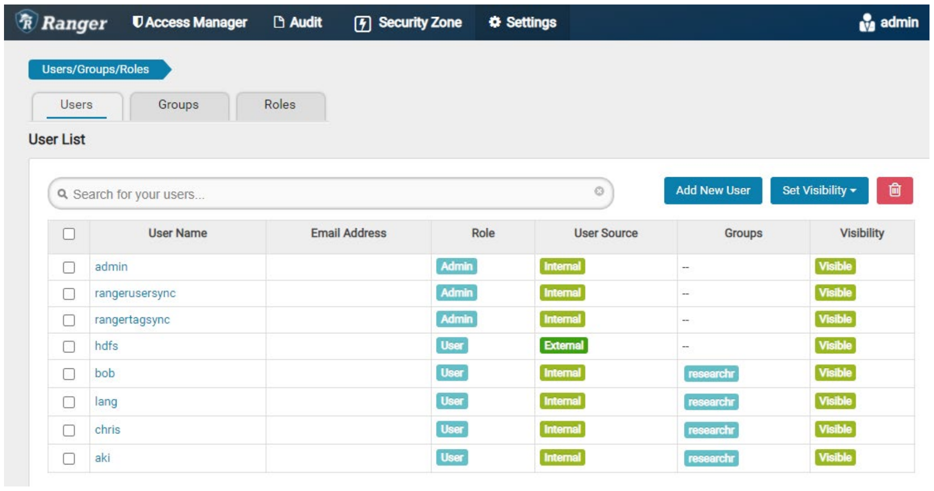
```
sudo groupadd <groupname>

sudo useradd -g <groupname> <username>

sudo passwd <username>
```

Under the Settings drop down, select Users/Groups/Roles

Select the Groups tab and create the group, then select users and add these users



User List

	User Name	Email Address	Role	User Source	Groups	Visibility
☐	admin		Admin	Internal	--	Visible
☐	rangerusersync		Admin	Internal	--	Visible
☐	rangertagsync		Admin	Internal	--	Visible
☐	hdfs		User	External	--	Visible
☐	bob		User	Internal	researchr	Visible
☐	lang		User	Internal	researchr	Visible
☐	chris		User	Internal	researchr	Visible
☐	aki		User	Internal	researchr	Visible

5.3.5 Create HDFS policies in Apache Ranger

Select Access Manger and Resource Based Policies from the drop down,

Select sehadoop under HDFS

Select Add New Policy from the right

Policy Name=dataset_open_policy

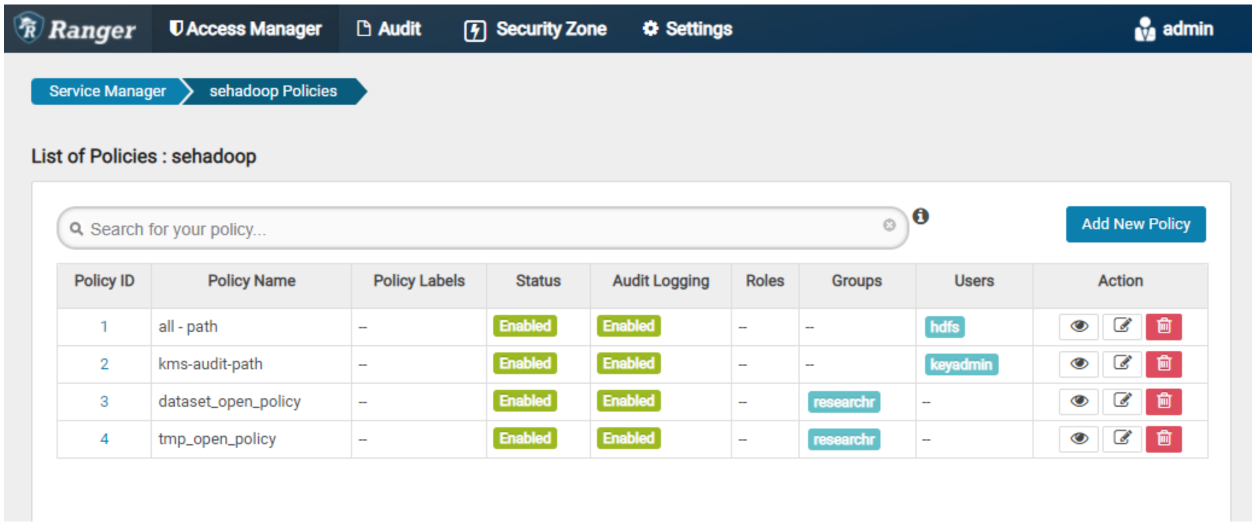
Resource Path=/DataSet1/

Allow Conditions:

Select Group or Select User

Add Permissions

Created a second policy to provide the group access to the /tmp folder



List of Policies : sehadoop

Policy ID	Policy Name	Policy Labels	Status	Audit Logging	Roles	Groups	Users	Action
1	all - path	--	Enabled	Enabled	--	--	hdfs	
2	kms-audit-path	--	Enabled	Enabled	--	--	keyadmin	
3	dataset_open_policy	--	Enabled	Enabled	--	researchr	--	
4	tmp_open_policy	--	Enabled	Enabled	--	researchr	--	

5.3.6 Execute PySpark Jobs Using Ranger Users

Copy the Jobs folder to the home directory of bob, then execute

```
spark-submit Filter-Job-SynSocial.py
```

Error Messages

```
/usr/local/bin/python3.9: can't open file '/tmp/hsperfdata_bob/Filter-Job-
SynSocial.py': [Errno 2] No such file or directory
```

Created the folder /tmp/hsperfdata_bob

```
ERROR DiskBlockManager: Failed to create local dir in /apache/spark/spark-temp.
Ignoring this directory.

java.io.IOException: Failed to create a temp directory (under /apache/spark/spark-
temp) after 10 attempts!
```

On Celeste

```
sudo chown -R hdfs:hadoop_users /apache/spark/spark-temp

sudo chown -R hdfs:hadoop_users /apache/spark/logs

sudo chown -R hdfs:hadoop_users /apache/hadoop/logs

sudo chown -R hdfs:hadoop_users /apache/hadoop/log

On Babar

sudo chown -R hdfs:hadoop_users /apache/hadoop/logs

sudo chown -R hdfs:hadoop_users /apache/hadoop/log


ERROR SparkContext: Error initializing SparkContext.

org.apache.hadoop.security.AccessControlException: Permission denied: user=bob,
access=WRITE,
inode="/user/bob/.sparkStaging/application_1647097259287_0001":hdfs:supergroup:drwxr-
xr-x
```

5.4 Apache Ranger YARN Plugin

Apache Ranger has the ability to manage the ACLs for the YARN queues.

https://cwiki.apache.org/confluence/plugins/servlet/mobile?contentId=56067021#content/view/56067021

NOTE: BUG Fix - The Apache Ranger plugin installs the following property in the yarn-site.xml file. However this property does not appear to be compatible with Hadoop version 2.7.3, so I removed it:

```
<property>
        <name>yarn.authorization-provider</name>

<value>org.apache.ranger.authorization.yarn.authorizer.RangerYarnAuthorizer</value>
    </property>
```

5.4.1 Enable the Apache Ranger YARN Plugin

as hdfs on pompadour

```
scp ranger-2.1.0-yarn-plugin.tar.gz celeste:/apache/ranger
```

as hdfs on celeste

```
cd /apache/ranger
tar zxvf ranger-2.1.0-yarn-plugin.tar.gz
ln -s /apache/ranger/ranger-2.1.0-yarn-plugin /apache/ranger/ranger-yarn-plugin
```

Link Hadoop configuration files

```
mkdir /apache/ranger/hadoop
mkdir /apache/ranger/hadoop/etc
ln -s /apache/hadoop/etc/hadoop /apache/ranger/hadoop/etc
ln -s /apache/hadoop/etc/hadoop /apache/ranger/hadoop
```

Update install.properties file

```
cd ranger-yarn-plugin
cp install.properties install.properties-DEFAULT
vi install.properties
```

Changed the following values in this file:

```
POLICY_MGR_URL=http://pompadour.sehadoop.test:6080
SQL_CONNECTOR_JAR=/apache/ranger
REPOSITORY_NAME=sehadoop-yarn
XAAUDIT.SOLR.ENABLE=true
XAAUDIT.SOLR.URL=http://pompadour.sehadoop.test:8983/solr/ranger_audits
XAAUDIT.SOLR.USER=solr
XAAUDIT.SOLR.IS_ENABLED=false
XAAUDIT.SOLR.SOLR_URL=http://pompadour.sehadoop.test:8983/solr/ranger_audi
ts
```

```
su root

./enable-yarn-plugin.sh
```

Copy the plugin jar files to the hadoop folder

```
cp -r /apache/ranger/ranger-yarn-plugin/lib/*
/apache/hadoop/share/hadoop/hdfs/lib

cp -r /apache/ranger/ranger-yarn-plugin/lib/*
/apache/hadoop/share/hadoop/yarn/lib


scp -r /apache/ranger/ranger-yarn-plugin/lib/*
babar:/apache/hadoop/share/hadoop/hdfs/lib


scp -r /apache/ranger/ranger-yarn-plugin/lib/*
babar:/apache/hadoop/share/hadoop/yarn/lib
```

5.4.2 Start Apache Ranger to Conform YARN Plugin is Operating

as Ranger on Pompadour

```
/apache/ranger/ranger-admin/ranger-admin start
```

Start HDFS as hdfs on babar and start YARN as yarn on celeste

5.4.3 Create YARN Service on Apache Ranger

At the Ranger Admin WebUI, Service Manager tab

selected the YARN folder

Service Name = sehadoop-yarn

Username = yarn

Password = `<xxx replace with password xxx>`

YARN REST URL = http://celeste.sehadoop.test:8088

selected Add

5.4.4 Create YARN policies in Apache Ranger

Select the sehadoop-yarn service under the YARN folder on the main menu

selected Add New Policy

Policy Name = yarn_policy

Queue = 1

Select Group = researchr

Permissions = submit-app

Add

6 Apache Atas

Apache Atlas provides metadata management in the cluster.

After installing and starting Apache Atlas, the following actions were executed to assign and manage metadata for data stored in HDFS and the jobs submitted to the YARN Capacity Scheduler.

For Apache Atlas, the term "Type" refers to a definition of metadata objects. A type represents the attributes that define the properties for a metadata object. It is analogous to the concept of a "class" in object oriented programming or a 'table schema' in relational databases.

Types have metatype of 'Entity', 'Struct', 'Classification' or 'Relationship', which refer to their collection of attributes. Each attribute has a name (e.g. 'name') and some other associated properties. A property can be referred to using an expression type_name.attribute_name.

The term "Entity" refers to a specific instance of a Type. Entities have specific values assigned to the attributes. The analogy to object oriented program is that an Entity is like an "object" of a certain "class" or actual values in the 'table schema' in a relational database.

The following sections describe the steps followed to manage metadata on Atlas. This is based upon the Udemy Apache Atlas course. The example files were updated on my laptop using Sublime text editor and then copied to the /home/atlas folder using WinSCP graphical user interface (located at C:\Program Files (x86)\WinSCP).

Note: detail are here https://asf.alaska.edu/how-to/data-recipes/moving-files-into-and-out-of-an-aws-ec2-instance-windows/

```
sudo chown -R atlas:hadoop /home/atlas/atlas-files
```

To more easily view the results of these command, install jq on Apache Atlas

```
yum install jq
```

6.1 Install Sample Data

Start Apache Atlas, then execute the quick start to load sample Types and Entities that are used as the basis for defining additional entities, as atlas execute the following on troubadour

```
/apache/atlas/bin/atlas_start.py
```

```
/apache/atlas/bin/quick_start.py
```

Then go to the Atlas WebUI at http://<ip address of atlas>:21000

6.2 Apache Type and Entity Definitions using the Apache API

Types and Entities are defined in JSON files that are added to Atlas using the Atlas API for the format of the file and curl commands.

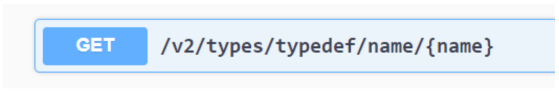
The template for the Type definition files is at Apache Atlas API site:

https://atlas.apache.org/api/v2/index.html

https://atlas.apache.org/api/v2/ui/index.html

in the TypeREST section, the

GET /v2/types/typedef/name/{name} API command was used



for example, to obtain the DB Type definition, this is /v2/types/typedef/name/DB

this can be obtained through the Apache Ranger REST API by executing the following command

```
curl -X GET -u admin:admin
http://localhost:21000/api/atlas/v2/types/typedef/name/DB
```

6.2.1 Create the hdfs_folder Type json file

Updated the TypedefMySQLDB.json and saved it as TypedefHDFSFolder.json

changed the following lines in the examples

Original	**Update**
“name”: “mysql_db”,	“name”: “hdfs_folder”,
"description": "MySQL DB",	"description": "HDFS Folder",
"name": "locationUri",	"name": "path",

6.2.2 Create of the hdfs_folder Type

as atlas on troubadour

Create the hdfs_folder type definition using the Atlas API TypeREST POST command (there is only one POST command) (this is a “Bulk create API for all atlas type definitions, only new definitions will be created”) as follows :

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-control:
no-cache'  -H 'content-type: application/json'
http://localhost:21000/api/atlas/v2/types/typedefs -d @TypedefHDFSFolder.json
```

check the results using the GET command then pipe the results of the to jq as follows

```
curl -X GET -u admin:admin
http://localhost:21000/api/atlas/v2/types/typedef/name/hdfs_folder |jq
```

the GUID, creation date, and other information, such as the supertype attributes are added

6.2.3 Verify the creation of this Type on the Atlas WebGUI

http://<ip address of atlas>:21000

6.2.4 Review the Relationships

the DataSet Type includes input and output from processes, and hdfs_folder Type inherits from the DataSet Type these do not need to be added

For background, this is some information about Relationships:

> There are three Relationship Categories: ASSOCIATION, AGGREGATION, and COMPOSITION
>
> The Relationship category determines the style of relationship around containment and lifecycle. UML terminology is used for the values.
>
> ASSOCIATION is a relationship with no containment.
>
> COMPOSITION and AGGREGATION are containment relationships.
>
> The difference being in the lifecycles of the container and its children. In the COMPOSITION case, the children cannot exist without the container. For AGGREGATION, the life cycles of the container and children are totally independent.

6.2.5 Create Entities of the hdfs_folder Type json file

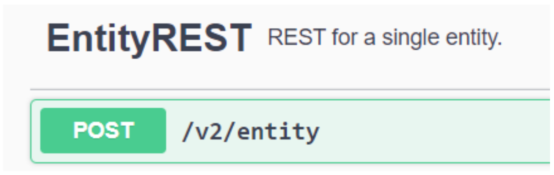
The EntityREST list of APIs is used to create entities

use the GET /v2/entity/guid/{guid} to Fetch a complete definition of an entity given its GUID

to use as a template

The GUID is the alpha-numeric string at the end of the URL

then use the POST /v2/entity to create a new entity or update existing entities in Atlas



The qualifiedName field needs to be unique to all entities managed by Atlas

so for each folder the following values were used to update the example file

JSON file name	Name and display name	qualifiedName (unique)	Owner	Description
patientsHDFSFolder.json	patients, patients folder	DataSet1_patients@sehadoop	hdfs	patients medical records
social-media-usersHDFSFolder.json	social-media-users, social-media-users folder	DataSet1_social-media-users@sehadoop	hdfs	social media users messages
patientsDFDataHDFSFolder.json	patientsDFData	DataSet1_patientsDFData@sehadoop	hdfs	patient data filtered and in DataFrame format
social-media-usersDFDataHDFSFolder.json	social-media-usersDFData	DataSet1_social-media-usersDFData@sehadoop	hdfs	social media data filtered and in DataFrame format
joinedDFDataHDFSFolder.json	joinedDFData	DataSet1_joinedDFData@sehadoop	hdfs	joined data
hashedDFDataHDFSFolder.json	hashedDFData	DataSet1_hashedDFData@sehadoop	hdfs	hashed data
encryptedDFDataHDFSFolder.json	encryptedDFData	DataSet1_encryptedDFData@sehadoop	hdfs	encrypted data
KEYHDFSFolder.json	KEY KEY folder	DataSet1_KEY@sehadoop	hdfs	key used for encryption of encryptedDFData
analysis-hashHDFSFolder.json	analysis-hash	DataSet1_analysis-hash@sehadoop	hdfs	results from analyzing hashed data
analysis-hash-statsHDFSFolder.json	analysis-hash-stats	DataSet1_analysis-hash-stats@sehadoop	hdfs	statistics on hashed data
analysis-encryptHDFSFolder.json	analysis-encrypt	DataSet1_analysis-encrypt@sehadoop	hdfs	results from analyzing encrypted data
analysis-encrypt-statsHDFSFolder.json	analysis-encrypt-stats	DataSet1_analysis-encrypt-stats@sehadoop	hdfs	statistics on encrypted data

6.2.6 Post the entity file

execute the following command for each entity json file to post it

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-control: no-cache'  -H 'content-type: application/json' http://localhost:21000/api/atlas/v2/entity -d @patientsHDFSFolder.json |jq
```

6.2.7 Create a Spark Programs Type based upon the Load Process Type

```
curl -X GET -u admin:admin
http://localhost:21000/api/atlas/v2/types/typedef/name/LoadProcess |jq
```

Follow a similar approach to create Spark Program Type use the type example IngestionProcess

Updated the IngestionProcess.json and saved it as SparkProcess.json changed the following lines in the examples

Original	Update
"name": "IngestionProcess",	“name”: “SparkProcess”,
"description": "Ingestion Process",	" "description": "Spark Process",

Post this new process as follows:

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-control: no-cache'  -H 'content-type: application/json' http://localhost:21000/api/atlas/v2/types/typedefs -d @SparkProcess.json |jq
```

6.2.8 Create an Entity for each Spark Process of Type SparkProcess

use a Load Process as an example

update the SalesIngestionProcess.json file for each program with the attributes listed below

JSON File Name	owner	outputs attributes	qualifiedName (unique)	inputs attributes	description	Name and display name
filter-job-syntheaProcess.json	hdfs	guid of patientsDFData typeName=hdfs_folder qualifiedName = DataSet1_patientsDFData@sehadoop	filter-job-synthea@sehadoop	guid of patients typeName=hdfs_folder	filters medical record data	Filter-Job-Synthea.py Filter-Job-Synthea
filter-job-synsocialProcess.json	hdfs	guid of social-media-usersDFData typeName=hdfs_folder	filter-job-synsocial@sehadoop	guid of social-media-users typeName=hdfs_folder	filters social media data	Filter-Job-SynSocial.py, Filter-Job-SynSocial
join-jobProcess.json	hdfs	typeName=hdfs_folder guid of joinedDFData qualifiedName = DataSet1_joinedDFData@sehadoop	join-job@sehadoop	2 input files: typeName=hdfs_folder guid of social-media-usersDFData and patientsDFData	joins social media data and medical records	Join-Job.py Join-Job
privacy-jobProcess.json	hdfs	3 output files: typeName=hdfs_folder guid of KEY, hashedDFData, and encryptedDFData	privacy-job@sehadoop	typeName=hdfs_folder guid of joinedDFData	encrypts and hashes sensitive data	Privacy-Job.py Privacy-Job
analysis-job-encryptProcess.json	hdfs	2 output files: typeName=hdfs_folder guid of analysis-encrypt and analysis-encrypt-stats	analysis-job-encrypt@sehadoop	typeName=hdfs_folder guid of encryptedDFData	analyzes encrypted files	Analysis-Job-Encrypt.py Analysis-Job-Encrypt
analysis-job-hashProcess.json	hdfs	2 output files: typeName=hdfs_folder guid of analysis-hash and analysis-hash-stats	analysis-job-hash@ sehadoop	typeName=hdfs_folder guid of hashedDFData	analyzes hashed files	Analysis-Job-Hash.py Analysis-Job-Hash

The number of inputs and outputs would need to correspond to the data processing lifecyle

Note guid is the alpha-number at the end of the URL when the entity is displayed on the Atlas WebUI.

Create an entity for each of these processes by executing the following command

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-
control: no-cache'  -H 'content-type: application/json'
http://localhost:21000/api/atlas/v2/entity -d @filter-job-
synsocialProcess.json |jq
```

6.2.9 Check the Atlas WebUI to verify the Lineage is properly configured

http://<ip address of atlas>:21000

6.2.10 Define and POST the HIPAA Classification based upon the example

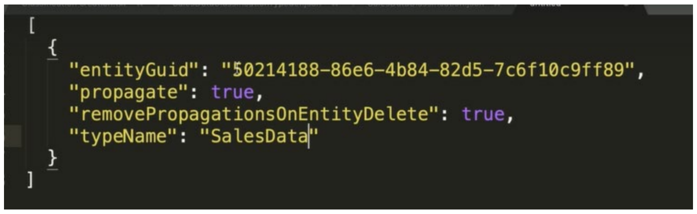
Created the HIPAAClassification.json file based upon the example

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-
control: no-cache'  -H 'content-type: application/json'
http://localhost:21000/api/atlas/v2/types/typedefs -d
@HIPAAClassification.json |jq
```

Execute the same steps for Classification Sensitive and Redacted

6.2.11 Assign Classifications to Data

A json file can be created to apply the classification to multiple entities, e.g., titled patientsHIPAAClassification.json with the following content. Repeat within the braces [ ] common separated items contained in the curved brackets {} and updated for each entities GUID



```
[
  {
    "entityGuid": "50214188-86e6-4b84-82d5-7c6f10c9ff89",
    "propagate": true,
    "removePropagationsOnEntityDelete": true,
    "typeName": "SalesData"
  }
]
```

An example file illustrates how to do this for HIPAA. This requires the GUID for each entity.

Apply the classification using the POST command

```
curl -X POST -u admin:admin -H 'accept: application/json'  -H 'cache-
control: no-cache'  -H 'content-type: application/json'
http://localhost:21000/api/atlas/v2/entity/guid/<guid of entity>
classifications -d @patientsHIPAAClassification.json
```

Alternatively, the Apache WebUI can be used to assign the classifications.

https://docs.cloudera.com/runtime/7.2.6/atlas-working-with-classifications/topics/atlas-creating-classifications.html

(note: the POSTman tool helps with managing these CURL commands)

6.2.12 Correcting Errors

Note - If there is an error with the curl POST command, for entities - use the following DELETE command

to delete a Type Definition

```
curl -X DELETE -u admin:admin
http://localhost:21000/api/atlas/v2/types/typedef/name/mysql_db
```

to delete an Entity Definition

```
curl -X DELETE -u admin:admin
http://localhost:21000/api/atlas/v2/entity/guid/65a97440-f7d0-40e4-
802e-415cb3a6f6b0
```

remove types and entities marked as deleted

```
curl -X PUT -u admin:admin
'http://localhost:21000/api/atlas/admin/purge' -H 'Content-Type:
application/json' -d '["6b8ee3a4-c0e5-4861-a3d9-9b1e977610d9",
"65a97440-f7d0-40e4-802e-415cb3a6f6b0"]' |jq
```

references:
https://docs.cloudera.com/runtime/7.2.10/atlas-reference/topics/atlas-reference-purging-deleted-entities-delete-api.html

https://issues.apache.org/jira/browse/ATLAS-3477

6.3 Connecting Atlas to Ranger

The following steps were executed so the metadata managed by Atlas is input to Ranger. This enables Ranger to use this metadata as part of the security policies. The Ranger Tag Sync module facilitates this interface to Atlas. These commands were executed as atlas on troubadour

6.3.1 stop Apache Atlas

```
atlas_stop.py
```

6.3.2 Install Kafka on Atlas so it can connect to Ranger

```
cd /apache
wget https://archive.apache.org/dist/kafka/1.1.0/kafka_2.11-1.1.0.tgz
tar xvf kafka_2.11-1.1.0.tgz
```

6.3.3 Configure Kafka properties

```
ln -s kafka_2.11-1.1.0 kafka
```

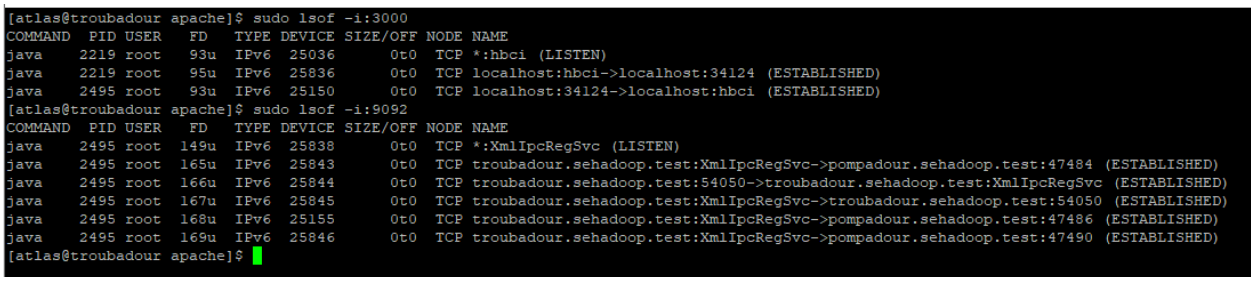
```
cd /apache/kafka/config
vi zookeeper.properties
clientPort=3000 #default value is 2181
vi server.properties
zookeeper.connect=localhost:3000 #default value is 2181
```

6.3.4 Start Kafka

verify the port kafka is running on using the following command

```
sudo yum install lsof
sudo lsof -i:3000
sudo lsof -i:9092 (kafka boot strap server)


sudo /apache/kafka/bin/zookeeper-server-start.sh -daemon
/apache/kafka/config/zookeeper.properties
sudo /apache/kafka/bin/kafka-server-start.sh -daemon
/apache/kafka/config/server.properties
```



```
[atlas@troubadour apache]$ sudo lsof -i:3000
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java    2219 root   93u  IPv6  25036      0t0  TCP *:hbci (LISTEN)
java    2219 root   95u  IPv6  25836      0t0  TCP localhost:hbci->localhost:34124 (ESTABLISHED)
java    2495 root   93u  IPv6  25150      0t0  TCP localhost:34124->localhost:hbci (ESTABLISHED)
[atlas@troubadour apache]$ sudo lsof -i:9092
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
java    2495 root  149u  IPv6  25838      0t0  TCP *:XmlIpcRegSvc (LISTEN)
java    2495 root  165u  IPv6  25843      0t0  TCP troubadour.sehadoop.test:XmlIpcRegSvc->pompadour.sehadoop.test:47484 (ESTABLISHED)
java    2495 root  166u  IPv6  25844      0t0  TCP troubadour.sehadoop.test:54050->troubadour.sehadoop.test:XmlIpcRegSvc (ESTABLISHED)
java    2495 root  167u  IPv6  25845      0t0  TCP troubadour.sehadoop.test:XmlIpcRegSvc->troubadour.sehadoop.test:54050 (ESTABLISHED)
java    2495 root  168u  IPv6  25155      0t0  TCP troubadour.sehadoop.test:XmlIpcRegSvc->pompadour.sehadoop.test:47486 (ESTABLISHED)
java    2495 root  169u  IPv6  25846      0t0  TCP troubadour.sehadoop.test:XmlIpcRegSvc->pompadour.sehadoop.test:47490 (ESTABLISHED)
[atlas@troubadour apache]$
```

6.3.5 Edit Apache Atlas application Properties File

```
vi /apache/atlas/conf/atlas-application.properties
```

verify and change the following properties

```
atlas.notification.embedded=false

atlas.kafka.zookeeper.connect=localhost:3000 #default value is 9026

atlas.kafka.boostrap.servers=localhost:9092 # default value is 9027
```

6.3.6 Start Apache Atlas

atlas_start.py

6.3.7 Update/confirm the ports in the Ranger Tag Sync install.properties file

on pompadour as ranger /apache/ranger/ranger-tagsync/install.properties

```
chmod +w install.properties
vi /apache/ranger/ranger-tagsync/install.properties
```

change / confirm configuration of the following values

```
TAG_SOURCE_ATLAS_KAFKA_BOOTSTRAP_SERVERS =
troubadour.sehadoop.test:9092 # from 6667

TAG_SOURCE_ATLAS_KAFKA_ZOOKEEPER_CONNECT =
troubadour.sehadoop.test:3000 # from 2181

TAG_SOURCE_ATLASREST_ENDPOINT = http://troubadour.sehadoop.test:21000

TAG_SOURCE_ATLASREST_USERNAME =admin

TAG_SOURCE_ATLASREST_PASSWORD =admin
```

`TAGSYNC_ATLAS_TO_RANGER_SERVICE_MAPPING=sehadoop` # this is the value of the HDFS service created in Ranger

```
#User and group for the tagsync process

unix_user=ranger

unix_group=hadoop

#change password of rangerTagsync user. Please note that this password
should be as per rangerTagsync user in ranger

rangerTagsync_password=<xxx replace with password xxx>
```

6.3.8 Start Apache Ranger TagSync

on Pompadour as Ranger execute the following commands to start the TagSync service

```
sudo -E sh setup.sh
sudo -E sh ranger-tagsync-services.sh start
```

7 Hadoop Configuration Updates

7.1 Create HDFS Users

The following Linux OS, local accounts were created

name	group
raj	researchl
jose	researchl
alice	researchl
ning	researchl

Create OS users and groups

```
groupadd <groupname>
useradd -g <groupname> <username>
passwd <username>
```

Create the following HDFS folders for user's home directory and temp storage

```
hdfs dfs -mkdir /user/<username>
hdfs dfs -mkdir /user/<username>/tmp/hadoop-<username>
```

Change the owner and group permissions on these files

```
hdfs dfs –chown –R <username>:<groupname> /user/<username>
```

Refresh the user and group mappings to let the NameNode know about the new user:

```
hdfs dfsadmin -refreshUserToGroupsMappings
hdfs dfsadmin -refreshServiceAcl
```

Note that if a folder with the user name needs to exist in the /user folder in hdfs for the user to have permission to submit jobs. This folder does not need to be granted permissions (i.e., using chown), if the user exists in Ranger.

7.2 Configuring YARN Job Scheduling

YARN has three types of job submission schedulers: First In First Out (FIFO), Capacity Scheduler, and Fair Scheduler. Capacity Scheduler is the default policy.

For FIFO, when jobs are submitted, they are put in the queue. Jobs are executed in order of arrival without any sense of importance. There are not any options to configure Access Control Lists (ACLs) with the FIFO scheduler, so everyone can submit jobs to YARN.

Capacity scheduler is based upon using queues. The queues would be assigned to groups/organizations. The total available resources is divided up between the different queues. This helps to ensure that resources are available to small jobs and that they are not stuck waiting for the completion of a large job.

In the Fair Scheduler configuration, an equal share of resources is allocated to each job. The configuration is based upon pools of resources with users assigning jobs to each pool. This configuration appears more high fidelity and complex in that minim and maximum memory (RAM), for example, can be allocated to the pools/queues.

The type and configuration of job submission scheduler used by YARN is defined by properties in the in the yarn-site.xml. To control access to submit jobs to YARN queues (i.e., to enable YARN queue ACLs) the following property needs to be set in yarn-site.xml, the default value is false.

```
 <property>
<name>yarn.acl.enable</name>
<value>true</value>
</property>
```

7.2.1 FIFO Properties

```
<property>
<name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fifo.FifoSched
uler</value>
</property>
```

7.2.2 Capacity Scheduler Properties

https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html

```
<property>
<name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.Capac
ityScheduler</value>
</property>
```

Queue properties are set in the `capacity-scheduler.xml` configuration file to define the queues, queue hierarchies, resource allocations to queues, queue access control lists (ACLs) and other properties.

The default values in `capacity-scheduler.xml` (not changed) are listed below:

```
<configuration>

  <property>
    <name>yarn.scheduler.capacity.maximum-applications</name>
    <value>10000</value>
    <description>
      Maximum number of applications that can be pending and running.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.maximum-am-resource-percent</name>
    <value>0.1</value>
    <description>
      Maximum percent of resources in the cluster which can be used to run
      application masters i.e. controls number of concurrent running
      applications.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.resource-calculator</name>

<value>org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator</value>
    <description>
      The ResourceCalculator implementation to be used to compare
      Resources in the scheduler.
      The default i.e. DefaultResourceCalculator only uses Memory while
      DominantResourceCalculator uses dominant-resource to compare
      multi-dimensional resources such as Memory, CPU etc.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default</value>
    <description>
      The queues at the this level (root is the root queue).
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.capacity</name>
    <value>100</value>
    <description>Default queue target capacity.</description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.user-limit-factor</name>
    <value>1</value>
    <description>
      Default queue user limit a percentage from 0.0 to 1.0.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.maximum-capacity</name>
    <value>100</value>
    <description>
      The maximum capacity of the default queue.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.state</name>
    <value>RUNNING</value>
    <description>
      The state of the default queue. State can be one of RUNNING or STOPPED.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.acl_submit_applications</name>
    <value>*</value>
    <description>
      The ACL of who can submit jobs to the default queue.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.acl_administer_queue</name>
    <value>*</value>
    <description>
      The ACL of who can administer jobs on the default queue.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.node-locality-delay</name>
    <value>40</value>
    <description>
      Number of missed scheduling opportunities after which the
CapacityScheduler
      attempts to schedule rack-local containers.
      Typically this should be set to number of nodes in the cluster, By
default is setting
      approximately number of nodes in one rack which is 40.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.queue-mappings</name>
    <value></value>
    <description>
      A list of mappings that will be used to assign jobs to queues
      The syntax for this list is [u|g]:[name]:[queue_name][,next mapping]*
      Typically this list will be used to map users to queues,
      for example, u:%user:%user maps all users to queues with the same name
      as the user.
    </description>
  </property>
  <property>
    <name>yarn.scheduler.capacity.queue-mappings-override.enable</name>
    <value>false</value>
    <description>
      If a queue mapping is present, will it override the value specified
      by the user? This can be used by administrators to place jobs in queues
      that are different than the one specified by the user.
      The default is false.
    </description>
  </property>
</configuration>
```

7.2.3 Fair Scheduler

https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-site/FairScheduler.html

```
<property>
<name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairSched
uler</value>
</property>
```

A queue allocation file, in XML format, is defined as described at the link above. The name of this file is specified in a property contained in yarn-site.xml. This can also include ACL, which define who is authorized to submit jobs to the queues.

8 Operating System Configuration Updates

The following commands and changes were executed to prepare the operating system for Hadoop installation.

8.1 Create User Accounts and Change Default Passwords

The default OS account is CentOS. The password for this account and root was changed and the hdfs account and hadoop group were created for installation of Hadoop software.

These are local, operating system accounts. Network directory (Free IPA, LDAP) managed accounts are addressed in following sections, (Section 6.1).

```
sudo passwd centos
sudo passwd root
su root
useradd -m hdfs   #creates home directory for hdfs
passwd hdfs
groupadd hadoop
usermod -aG hadoop hdfs
usermod -aG wheel hdfs
```

Confirmed the wheel group is enabled by default by ensuring wheel is not commented out in the sudoers file, by executing the following command `cat /etc/sudoers`.

8.2 Update the Operating System

Update the operating system.

```
yum -y update
```

8.3 Other Package Installations

Install the following additional packages to support the execution of MapReduce and Spark programs and other Hadoop ecosystem components.

```
yum install -y yum-priorities #ensures proper package installation priorities
yum install -y wget curl unzip tar gcc* rpm-build
yum group install -y "Development Tools"
yum install -y epel-release #extra Package for Enterprise Linux
yum install -y dkms kernel-devel
yum install -y python-devel
```

9 Directory Services - FreeIPA LDAP

FreeIPA was set up and configured on belle to manage accounts on all the computers in the cluster.

https://community.cloudera.com/t5/Community-Articles/Ambari-2-4-Kerberos-with-FreeIPA/ta-p/247653

https://mapredit.blogspot.com/2016/10/freeipa-and-hadoop-distributions-hdp-cdh.html

9.1 Hadoop Service Instance Kerberos Configuration

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SecureMode.html

Each Hadoop Service instance needs to be configured with its Kerberos principal and keytab file location.

The general format of a Service principal is `ServiceName/_HOST@REALM.TLD`.
e.g. `dn/_HOST@EXAMPLE.COM`.

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SecureMode.html

nn/babar.sehadoop.test@SEHADOOP.TEST

host/babar.sehadoop.test@SEHADOOP.TEST

sn/basil.sehadoop.test@SEHADOOP.TEST

dn/pom.sehadoop.test@SEHADOOP.TEST

dn/flora.sehadoop.test@SEHADOOP.TEST

dn/alex.sehadoop.test@SEHADOOP.TEST

rm/celeste.sehadoop.test@SEHADOOP.TEST

nm/pom.sehadoop.test@SEHADOOP.TEST

nm/flora.sehadoop.test@SEHADOOP.TEST

nm/alex.sehadoop.test@SEHADOOP.TEST

jhs/babar.sehadoop.test@SEHADOOP.TEST

on belle at the command line

```
ipa user-add hdfs

ipa user-add yarn

ipa user-add mapred
```

the format for creating the key tab file is:

```
ktutil: add_entry -password -p principal_name -k number -
e encryption_type for each encryption type
```

for example

```
ktutil: add_entry -password -p nn/babar.sehadoop.test@SEHADOOP.TEST

-k 1 -e aes256-cts-hmac-sha1-96
```

write to the key tab file as follows

```
ktutil: wkt /etc/security/keytab/nn.service.keytab
```

9.2 User Account Kerberos Configuration

kinit hdfs

kinit yarn

kinit mapred

9.3 Core-site.xml Configuration

```
<property>
    <name>hadoop.security.authentication</name>
    <value>kerberos</value>
</property>
<property>
    <name>hadoop.security.authorization</name>
    <value>true</value>
</property>
```

9.4 SSH Configuration

Change Kerberos Options to yes in the SSH configuration file.