﻿1. Log in to AWS using Amazon / AWS credentials

2. Select your data center

3. Select VPC under Networking

4. Select Start VPC wizard 10.0.0.0/24

5. After creating the VPC, select "Modify Auto-Assign Public IP" to enable in subnets section > Subnet actions

6. Create a all-trafic security group associated with the VPC

allowing all traffic inbound and outbound 0.0.0.0/0

7. Generate security credentials Cloudera Director supports IAM users

and Amazon prefers them, however we will use credentials

8. Download the access key and secret access key on your local machine and paste in launch-cluster.sh

9. Create a keypair Go to ec2 dashboard name it security

10.Download the keypair and give it 400 permissions chmod 400 keyfile.

THESE SETTING ARE USED ONLY ONCE, LATER YOU CAN LAUNCH USING THESE SETTINGS

11.launch instance with ami-c318a8a8

Compute optimized c3.large 2gb Ram

Make sure select correct vpc.

12.Edit aws.simple.conf $> grep REPLACE-ME \*

12.a) name: CloudAge-Hadoop-Security

12.b) AWSAccessKeyId=AKIAJLSVLAOTSO7P6RTA

AWSSecretKey=ZWiUoy2qG+vD8s7A/Y5TzIzWQqPcCoETkkEB8cgj

publishAccessKeys: true

12.c) region: us-east-1

12.d) subnetId: subnet-e2440ec9

12.e) type: m3.xlarge

12.f) count: 5

check AMI ami-8767d1ec

privateKey: /home/ec2-user/security.pem

publishAccessKeys: true

13. copy public DNS Address in the file called launcher

echo ec2-52-204-118-45.compute-1.amazonaws.com > launcher

14. sh prelaunch.sh

after cluster is ready.

15. open vi editor name it cluster

i-04862e24aaa101946: ec2-54-85-55-177.compute-1.amazonaws.com

i-04a7663c9dbab0941: ec2-54-174-233-99.compute-1.amazonaws.com

i-05d2d46e1671e18b9: ec2-54-209-158-72.compute-1.amazonaws.com

i-06a0c94a83326030b: ec2-54-197-29-155.compute-1.amazonaws.com

i-07473167f1e6938e9: ec2-52-23-205-210.compute-1.amazonaws.com

i-07473167f1e6938e9: ec2-52-23-205-210.compute-1.amazonaws.com

Press esc and at colon type 1,$ s/.\*: //g This will remove everything from line 1 upto the colon and replacing nothing with global scope.

16. echo ec2-34-207-143-240.compute-1.amazonaws.com > cm

17. connect to cloudera-manager paste in the browser ec2-54-198-210-80.compute-1.amazonaws.com:7180

18. ssh -i security.pem ec2-user@`cat cm`

19. The url is the cm hostname:7180

the username and password is admin/admin

Cloudera Manager, or CM

Shows you service health and charts

Lets you start and stop services

Tells you dependent services and warns you if you do things out of order

It lets you stop services from management by "deleting" them

It also lets you add services

It surfaces role assignment information and lets you assign roles, but it

makes a guess as to the proper assignments

we will install hue and oozie it should be oozie and hue.

CM lets you change the configurations of the services. This automatically

edits the corresponding hadoop config xml files and deploys them, tracking

the changes over time.

we will uncheck "Check HDFS Permissions" which causes hdfs-site.xml

to get regenerated with dfs.permissions set to false.

CM lets you know when configurations are stale and services need restarting

CM makes "safety valves" available where you can provide xml snippets for

properties that are not explicitly available in the UI

we can put a bit of text in a Safety Valve

http://www.cloudera.com/products/cloudera-manager.html

http://www.cloudera.com/documentation/enterprise/latest/topics/cm\_intro\_primer.html

20.######### REStart cluster that has hdfs permissions DISabled ######After Finish ####

hdfs > instance > datanode > webUI

21.echo ec2-54-226-26-85.compute-1.amazonaws.com > host

22. ssh -i ./security.pem ec2-user@`cat host`

sudo useradd jinga

sudo passwd jinga

su jinga

hadoop fs -mkdir /user/jinga

vi file

This a test for security

This a test for security

This a test for security

This a test for security

:wq

hadoop fs -put file .

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 10 10

hadoop fs -cat file

hdfs fsck /user/jinga -files -blocks -locations

make sure you are on the same node that is mentioned in the replica placements.

find /data0/dfs /data1/dfs -name \*blk\_1073742640\*

cat /data1/dfs/dn/current/BP-86281392-10.0.0.133-1496236360799/current/finalized/subdir0/subdir3/blk\_1073742640

See the unencrypted text

now exit from jinga

In another window, ssh to the host again with ec2-user install

sudo yum install libpcap -y && sudo yum install wireshark -y

sudo dumpcap -i eth0

^C

on another window of jinga user create

vi files

Jinga you are playing with security.

Jinga you are playing with security.

Jinga you are playing with security.

Jinga you are playing with security.

Jinga you are playing with security.

Jinga you are playing with security.

:wq

In the first window, edit a file and place it into hdfs

hadoop fs -mkdir /user/ec2-user/

hadoop fs -put files .

^C

sudo tshark -r /tmp/wireshark\_eth0\_20170531200415\_A8xgn3 -V | grep -i Jinga

There's data going over the wire unencrypted between HDFS clients and processes

There's data going over the wire unencrypted between HDFS processes on different

nodes.

HDFS permissions need to be re-enabled

Two windows

Two files with hostnames in them

Terminal 1: ssh to host1

ssh -i security.pem ec2-user@`cat host1`

Try to do

hadoop fs -mkdir /user/bingo

and see a permission denied error

cat /etc/passwd

sudo -i

su - hdfs

HDFS is the root user for HDFS

So now you can create home directories for users

hadoop fs -mkdir /user/usera /user/userb

hadoop fs -chown usera:usera /user/usera

hadoop fs -chown userb:userb /user/userb

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar randomwriter /user/usera/bingo

Login to another host

Now add and become usera

exit to ec2-user

sudo sh

su - usera

run a job:

While that's running, ssh to host2 in the other window

add another user with the same name

sudo useradd usera

sudo sh

su - usera

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar randomwriter /user/usera/bar

no authentication, so no way of knowing if it's the same

user on both hosts.

The only reason this works is because the user name exists on both hosts.

ssh to a host

launch the hive shell:

[usera@ip-10-0-0-227 ~]$ hive

From hive, create two tables:

create table beauty (a string, b string, c string) row format delimited fields terminated by '\t';

create table ugly (a string, b string, c string) row format delimited fields terminated by '\t';

show tables;

describe ugly;

describe beauty;

exit;

You can find out where the files backing the stable are stored in HDFS

Add data to a pair of files tab delimited dataset and datasets

vi dataset

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

JIO Maro JIO Maro JIO Maro JIO

:wq

vi datasets

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

fellow mellow jellow

:wq

Then back in hive

hive

load data local inpath 'dataset' into table ugly;

select \* from ugly;

load data local inpath 'datasets' into table beauty;

select \* from beauty;

exit;

Add the data to hive dataset using:

hadoop fs -put datasets /user/hive/warehouse/ugly

hive> select \* from ugly;

Edit a pair of simple scripts (script1.sh, script2.sh)

add file script1.sh; echo hello world

exit;

vi script1.sh

echo hello world

:wq

hive> add file script1.sh;

hive> from beauty select transform(a) using 'script1.sh' as (data);

vi script2.sh

hive -e 'drop table ugly;'

:wq

hive> add file script2.sh;

hive> from beauty select transform(a) using 'script2.sh' as (data);

hive> show tables;

exit;

beeline is recommended to dedicatedly secure the cluster.

23. ssh -i security.pem ec2-user@`cat cm`

&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&----MIT Kerberoz ----------&&&&&&&&&&&&&&&&&&&&&&&

sudo yum install -y krb5-server

yum list installed "krb?-\*"

hostname -f (ip-10-0-0-240.ec2.internal)

sudo vi /etc/krb5.conf

[libdefaults]

default\_realm = HADOOPSECURITY.COM

dns\_lookup\_realm = false

dns\_lookup\_kdc = false

ticket\_lifetime = 24h

renew\_lifetime = 7d

forwardable = true

default\_tgs\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

default\_tkt\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

permitted\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

[realms]

HADOOPSECURITY.COM = {

kdc = ip-10-0-0-249.ec2.internal

admin\_server = ip-10-0-0-249.ec2.internal

max\_renewable\_life = 7d

}

:wq

1,$ s/EXAMPLE.COM/HADOOPSECURITY.COM/g

sudo vi /var/kerberos/krb5kdc/kadm5.acl

\*/admin@HADOOPSECURITY.COM \*

:wq

sudo vi /var/kerberos/krb5kdc/kdc.conf

[kdcdefaults]

kdc\_ports = 88

kdc\_tcp\_ports = 88

[realms]

HADOOPSECURITY.COM = {

#master\_key\_type = aes256-cts

acl\_file = /var/kerberos/krb5kdc/kadm5.acl

dict\_file = /usr/share/dict/words

admin\_keytab = /var/kerberos/krb5kdc/kadm5.keytab

supported\_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal

max\_renewable\_life = 7d

}

:wq

sudo kdb5\_util create

"Give a master password"

sudo service krb5kdc start

sudo service kadmin start

exit to Kondwa from DataCenter

to localmachine in the working directory.

scp -i ./security.pem ec2-user@`cat cm`:/etc/krb5.conf ./

sh ./clustercmd.sh sudo yum install krb5-workstation -y

./putnmove.sh ./krb5.conf /etc/

http://www.oracle.com/technetwork/java/javase/downloads/index.html

./putnmove.sh UnlimitedJCEPolicy/US\_export\_policy.jar /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security

./putnmove.sh UnlimitedJCEPolicy/local\_policy.jar /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security

ssh -i security.pem ec2-user@`cat cm`

sudo kadmin.local

addprinc cm/admin

exit

ssh -i security.pem ec2-user@`cat host`

kinit cm/admin

"password"

klist -l

EXIT

ssh -i security.pem ec2-user@`cat cm`

ADMINISTRATION ENABLE KERBEROZ complete the wizard and DONE

Kerberos Security Realm

HADOOPSECURITY.COM

aes256-cts-hmac-sha1-96

aes128-cts-hmac-sha1-96

arcfour-hmac-md5

cm public or private dns (hostname -f)

cm/admin

password

RESTART CLUSTER.

ps -eaf | grep java

check log files of the host, hdfs > datanode> Logfiles

ssh -i security.pem ec2-user@`cat host`

hadoop fs -ls /user

exit to local machine

./clustercmd.sh sudo useradd user1 -u 1001

./clustercmd.sh sudo useradd user2 -u 1002

./clustercmd.sh sudo useradd admin -u 1003

./clustercmd.sh sudo useradd jinga -u 1004

ssh -i security.pem ec2-user@`cat cm`

sudo kadmin.local

addprinc user1

Enter password for principal "user1@HADOOPSECURITY.COM":

addprinc user2

Enter password for principal "user2@HADOOPSECURITY.COM":

addprinc admin

Enter password for principal "admin@HADOOPSECURITY.COM":

addprinc jinga

Enter password for principal "jinga@HADOOPSECURITY.COM":

addprinc hdfs

Enter password for principal "hdfs@HADOOPSECURITY.COM":

to be continued...

----------------------------------------------------------------------------------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%AUTHORIZATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Starting with: A kerberized cluster with a gateway node

To emphasize the need for authorization using a project

like Apache Sentry by demonstrating what is \*not\* provided when Sentry is absent.

Specifically, to show that even though we have authentication with

Kerberos, we do not have any authorization. For example, after authentication any user has complete admin access over databases and tables in the SQL interfaces (hive and impala).

To illustrate this, set up a cluster with Kerberos authentication using MIT Kerberos. designate one host as a gateway node and connect to gw.

add users user1, user2, admin, jinga via AD or via MIT.

./clustercmd.sh sudo useradd user1

./clustercmd.sh sudo useradd user2

./clustercmd.sh sudo useradd admin

./clustercmd.sh sudo useradd jinga

sudo kadmin.local

addprinc jinga

Log in as jinga to hive server on gateway node

sudo su

su jinga

kinit jinga

password

beeline

!connect jdbc:hive2://ip-10-0-0-133.ec2.internal:10000/default;principal=hive/ip-10-0-0-133.ec2.internal@HADOOPSECURITY.COM

create database database1;

drop database database1;

^z

However, jinga shouldn’t have any privileges. We need a framework for authorization that works not just for hive and impala but also for all of hdfs.

For more information:

https://blog.cloudera.com/blog/2014/02/migrating-from-hive-cli-to-beeline-a-primer/

https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients

--------------------------------SENTRY-----------------------------------------------

Starting with: A kerberized cluster with a gateway node

Here we want to enable Sentry.

set up a cluster with Kerberos authentication using MIT Kerberos. designat one host as a gateway node.

ssh -i security.pem ec2-user@`cat gw`

MAKE SURE addED users user1, user2, admin, jinga via AD via MIT.

add a Sentry service.

Adding the sentry server and enabling the Sentry service

make the user "admin" an admin user in sentry

(config option in cm)

sentry service has to be enabled

impersonation has to be disabled

yarn minimum userid

enable sentry in hive

enable sentry in impala

restart hue impala and oozie

restart the cluster

Log in as jinga, try to create a database or a table, you see no valid

privileges error see the debug message

Log in as jinga to hive server on gateway node

sudo su

su jinga

kinit

beeline

!connect jdbc:hive2://ip-10-0-0-160.ec2.internal:10000/default;principal=hive/ip-10-0-0-160.ec2.internal@HADOOPSECURITY.COM

create database database1;

cm> hive> configuration > search log4j

enable debugging in the Sentry service for HiveServer2.

log4j.logger.org.apache.sentry=DEBUG

Try to use a database

For more information:

http://blog.cloudera.com/blog/2014/05/how-to-configure-jdbc-connections-in-secure-apache-hadoop-environments/

http://sentry.apache.org/

Starting with: A kerberized cluster with a gateway node

set up a cluster with Kerberos authentication using

MIT Kerberos. designat one host as a gateway node.

add users user1, user2, admin, jinga via AD or via MIT.

look in the directory containing hive-contrib.jar set in hive aux jars

in Cloudera Manager as set in the hive aux directory.

(/opt/cloudera/parcels/CDH/lib/hive/lib)

enable Sentry by adding a Sentry service in CM and make sure the "admin"

user is an admin user in sentry

use Sentry in a similar way to what we would do with a real data set.

sample dataset we're using is UFO dataset, so from our working

files folder, we'll copy dataset.csv over to the gateway node.

Then we'll ssh to the gateway node and upload it to HDFS.

kdestroy

"login" as admin by obtaining a kerberos ticket

sudo kadmin.local

addprinc hdfs

exit

kinit hdfs

sudo su

su hdfs

cd

And upload the file

wget https://s3.amazonaws.com/securityhadoop/dataset.csv

hadoop fs -put dataset.csv /user/hive

hadoop fs -chown hive:hive /user/hive/dataset.csv

kinit admin

And then launch beeline and connect to hive server 2

beeline

!connect jdbc:hive2://ip-10-0-0-160.ec2.internal:10000/default;principal=hive/ip-10-0-0-160.ec2.internal@HADOOPSECURITY.COM

create role admin\_role;

grant all on server server1 to role admin\_role;

grant all on database default to role admin\_role;

grant role admin\_role to group admin;

Confirm that you can create a simple table:

create table foo(a string);

show tables;

If that works, Sentry has applied your privileges.

If that works, Sentry has applied your privileges.

Now, let's create a table describing the dataset data using the regex serde.

https://community.hortonworks.com/articles/58591/using-regular-expressions-to-extract-fields-for-hi.html

create table sightex (

group1 string, group2 string, group3 string, group4 string,

group5 string, group6 string, group7 string, group8 string,

group9 string, group10 string)

row format serde 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'

with serdeproperties (

"input.regex"="(\\d\*),(\\d\*),(\".\*\"),((\".\*\")|([^,]\*)),((\".\*\")|([^,]\*)),(\".\*\")"

) stored as textfile;

load data inpath '/user/hive/dataset.csv' into table sightex;

This is backed by regex serde so is slow and not supported by Impala,

so let's create a copy of the table.

create table sightings\_parquet as select group1 as sightex, group2 as reported, group3 as loc, group4 as shape, group7 as duration, group10 as description from sightex where group1 is not null;

create role analyst;

grant select on table sightings\_parquet to role analyst;

grant role analyst to group user1;

Now, let's create a view on sightings\_parquet that excludes the description

column and allow user2 to read that view.

create view sightings\_ltd as select sighted, reported, loc, shape, duration from sightings\_parquet;

create role ltd\_reader;

grant select on sightings\_ltd to role ltd\_reader;

grant role ltd\_reader to group user2;

So imagine UFO sightings in New Jersey have particular interest to a different

community of users. Let's create a derived table with UFO Sightings from New

Jersey and make that readable by jinga

create database sightings\_parquet;

create table jersey as select \* from sightings\_parquet where loc LIKE "%NJ%";

create role nj;

grant select on jersey to role nj;

grant role nj to group jinga;

So to review our permissions, admin can do anything, user1 can read

from the complete sightings\_parquet table, user2 can read from a limited

view of the table that excludes the sensitive column description, even

though that view is never materialized, and jinga can read from the new jersey

table.

Now query with hue to see sentry in action

select description from sightings\_parquet limit 20;

For more information:

http://sentry.apache.org/

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%enable HDFS extended ACLs%%%%%%%%%%%%%%

https://s3.amazonaws.com/securityhadoop/dataset.csv

Login to GW Node

we must create users.

Create a file file1 with data in it to admin's home dir

Create a file file2 with data in it to admin's home dir

without having to make user1 an owner, we can change the rw permissions

of files1 and files2 so that user1 can read file1 and user2 can read file2

kinit admin

cp dataset.csv file1

cp dataset.csv file2

hdfs dfs -put file1

hdfs dfs -put file2

hdfs dfs -ls

hdfs dfs -chmod 600 file1 file2

hdfs dfs -ls file2 file1

login to cm > hdfs > config> search acl > enable.

Redeploy client config and restart cluster,

hdfs dfs -getfacl /user/admin/file1

hdfs dfs -getfacl /user/admin/file2

hdfs dfs -setfacl -m user:user1:rw- /user/admin/file1

hdfs dfs -setfacl -m user:user2:rw- /user/admin/file2

hdfs dfs -getfacl /user/admin/file1

hdfs dfs -getfacl /user/admin/file2

kdestroy

kinit user1

login to user1

hdfs dfs -cat /user/admin/file1

hdfs dfs -cat /user/admin/file2 (no Permission)

For more information:

http://www.cloudera.com/documentation/enterprise/latest/topics/cdh\_sg\_hdfs\_ext\_acls.html

https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html#ACLs\_Access\_Control\_Lists

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%enable HDFS Extended ACL sync on Sentry.

This will enable non-SQL processing frameworks such as

MapReduce, Pig and Spark to access the backing files

for our data sets according to their Sentry privileges.

CM -> HDFS -> Configuration -> Search for ACL

Click ACL and Sentry sync

Deploy client config

Restart cluster

kinit hdfs

hadoop fs -getfacl /user/hive/warehouse/sightex

hadoop fs -getfacl /user/hive/warehouse/dataset

hadoop fs -getfacl /user/hive/warehouse/jersey

For more information:

http://www.cloudera.com/documentation/enterprise/latest/topics/sg\_hdfs\_sentry\_sync.html

https://blog.cloudera.com/blog/2015/01/new-in-cdh-5-3-apache-sentry-integration-with-hdfs/

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Encryption Zone%%%%%%%%%%%%%%%%%%%%%%%

Login or su to these users on one of the hosts in your cluster. These directions will help to verify KMS is setup to encrypt files.

Create a key and directory.

su <KEY\_ADMIN\_USER>

hadoop key create mykey1

hadoop fs -mkdir /tmp/zone1

Create a zone and link to the key.

su hdfs

hdfs crypto -createZone -keyName mykey1 -path /tmp/zone1

Create a file, put it in your zone and ensure the file can be decrypted.

su <KEY\_ADMIN\_USER>

echo "Hello World" > /tmp/helloWorld.txt

hadoop fs -put /tmp/helloWorld.txt /tmp/zone1

hadoop fs -cat /tmp/zone1/helloWorld.txt

rm /tmp/helloWorld.txt

Ensure the file is stored as encrypted.

su hdfs

hadoop fs -cat /.reserved/raw/tmp/zone1/helloWorld.txt

hadoop fs -rm -R /tmp/zone1

Starting with a cluster that has kerberos enabled and a gateway role defined.

First, enable the cluster to use AES-NI for performance:

0. hadoop checknative

1. Create a directory for libcrypto.so.1: /var/lib/hadoop/extra/native

./clustercmd.sh sudo mkdir -p /var/lib/hadoop/extra/native/

2. Place libcrypto.so.1 in a directory where hadoop can see it

./clustercmd.sh sudo cp /usr/lib64/libcrypto.so.1.0.1e /var/lib/hadoop/extra/native/libcrypto.so

3. ssh -i security.pem ec2-user@`cat gw`

hadoop checknative again

Create a Java KMS

Associate it with kerberos authentication

java kms > config > authentication type > kerberoz

Make sure HDFS service is associated with the KMS

hdfs > configuration > KMS Service > keystore kms java

Deploy a client configuration

Restart cluster

As the user HDFS

kinit hdfs

hadoop key create mykey

hadoop fs -mkdir /zone

hdfs crypto -createZone -keyName mykey -path /zone

hdfs crypto -listZones

hdfs dfs -put dataset.csv /zone

hdfs dfs -mv /zone/dataset.csv /user/admin

kinit admin

hadoop distcp -skipcrccheck -update /zone/dataset.csv /user/admin

For more information:

http://www.cloudera.com/documentation/enterprise/5-3-x/topics/cdh\_sg\_hdfs\_encryption.html

http://blog.cloudera.com/blog/2015/01/new-in-cdh-5-3-transparent-encryption-in-hdfs/

HDFS supports extended ACLs, which offer POSIX access control lists to the Hadoop environment.

Access Control Lists let you define finer grained access control beyond the traditional access control

in UNIX which limits access control to the three levels of user, the group, and others.

Extend HDFS extended ACLs in Cloudera Manager and restart the cluster.

%%%%%%%%%%%%%%%%%%%% enable SSL:%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

To enable "wire encryption" including RPC protection, secure data transfer, SSL and encrypted shuffle

after the keys have been established, in CM do the following:

STOP the cluster for simplicity

1. hadoop.rpc.protection set to PRIVACY

2. dfs.encrypt.data.transfer

3. dfs.data.transfer.protection is \*unset\* (it is superceded by dfs.data.transfer.protection)

Check hadoop.ssl.enabled and note how CM higlhights the required information

set ssl.server.keystore.location to /opt/hadoop/security/jks/keystore.jks

keystore passwords are password:

ssl.server.keystore.password

ssl.server.keystore.keypassword

ssl.client.truststore.location is /opt/hadoop/security/jks/truststore.jks

password is password

we can also enable HTTP web consoles for SPNEGO auth, but this requires

our web browser to have a kerberos ticket for the realm. In our setup

this isn't easily possible unless your web browser is on the same VPC

as the cluster due to the DNS setup of EC2, so we'll skip that for our purposes.

YARN service

set keystore file ssl.server.keystore.location to /opt/hadoop/security/jks/keystore.jks

set keystore password ssl.server.keystore.password and ssl.server.keystore.keypassword

we can also enable ssl for HTTFS, and Cloudera Manager but we'll leave that

out for now.

http://www.cloudera.com/documentation/enterprise/latest/topics/cm\_sg\_ssl\_yarn\_mr\_hdfs.html

we do the work required to create certificates signed

with a self-signed root CA.

To make this easier, the complete copy paste is scripted for you.

clustercmd.sh hostname -f | strings > private-cluster

scp -i security.pem private-cluster ec2-user@`cat cm`:~/

Copy prep-ssl.sh to cm host

scp -i security.pem prep-ssl.sh ec2-user@`cat cm`:~/

On CM host, we do an sh -x so we can see what happens:

sh -x ./prep-ssl.sh

Now exit out of CM and copy the tar file it generated to your local host.

scp -i ./security.pem ec2-user@`cat cm`:~/certs.tar ./

Now distribute the tar file to the cluster and extract

./putnmove.sh certs.tar

./clustercmd.sh tar xvf certs.tar

./clustercmd.sh sudo mv /opt/hadoop /opt/hadoop.old

./clustercmd.sh sudo mv opt/hadoop /opt/hadoop

./putnmove.sh createsymlinks.sh

./clustercmd.sh sudo sh createsymlinks.sh

Update the JVM jssacerts

./clustercmd.sh sudo cp /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security/cacerts /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security/cacerts.old

./clustercmd.sh sudo cp /opt/hadoop/security/truststore/jssecacerts /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security/cacerts

For more information

https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

https://www.openssl.org/docs/

https://blog.talpor.com/2015/07/ssltls-certificates-beginners-tutorial/

http://www.cloudera.com/documentation/enterprise/latest/topics/cm\_sg\_ssl\_yarn\_mr\_hdfs.html

we go over the work required to create certificates signed

with a self-signed root CA. The hadoop ecosystem is written in a variety

of different languages, including Java, Python, and C.

Java has its own format for SSL keys, and the non-java processes rely

on x509 / PEM format SSL keys and certificates that

requires their SSL certificates in a different format, such as Java (jks)

and x509.

So this can be a little daunting, but basically what we're doing is using

a combination of the java keytool and openssl to create a self-signed root

CA, generate the required certificates, sign them, and convert them to the

different formats using an intermediate format (PKCS12). Then we

distribute them throughout the cluster.

sudo mkdir -p /opt/hadoop/security/ca-certs /opt/hadoop/security/jks /opt/hadoop/security/tmp /opt/hadoop/security/certs /opt/hadoop/security/truststore /opt/hadoop/security/x509

OpenSSL can be used to generate a root Certificate Authority

sudo openssl genrsa \

-out /opt/hadoop/security/ca-certs/rootCA.key \

-aes256 \

-passout pass:password 2048

And then you can convert rootCA key to pem format

sudo openssl req -x509 \

-new \

-nodes \

-key /opt/hadoop/security/ca-certs/rootCA.key \

-days 4000 \

-out /opt/hadoop/security/ca-certs/rootCA.pem \

-passin pass:password \

-passout pass:password \

-subj '/C=IN/ST=Pune/L=SF/O=Cloudage/OU=PS/CN=Admin\/emailAddress=support@cloudage.com.co'

Generate the java keystore file for every node in the cluster:

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-genkey \

-alias thishost \

-keyalg RSA \

-keystore \

/opt/hadoop/security/jks/thishost-keystore.jks \

-keysize 2048 \

-dname "CN=thishost, OU=PS, O=CloudAge, L=Pune, S=MAHARASHTRA, C=IN" \

-storepass password \

-keypass password

Generate a temporary p12 file from the java keystore file:

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-importkeystore \

-srckeystore /opt/hadoop/security/jks/thishost-keystore.jks \

-srcstorepass password \

-srckeypass password \

-destkeystore /opt/hadoop/security/tmp/thishost-keystore.p12 \

-deststoretype PKCS12 \

-srcalias thishost \

-deststorepass password \

-destkeypass password

Generate unsigned keys from the p12 file

sudo openssl pkcs12 \

-in /opt/hadoop/security/tmp/thishost-keystore.p12 \

-passin pass:password \

-nocerts \

-out /opt/hadoop/security/x509/thishost-unsignedkey.pem \

-passout pass:password

Generate the CSR (certificate signing request) from the

java keystore file:

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool -certreq \

-alias thishost \

-keystore /opt/hadoop/security/jks/thishost-keystore.jks \

-file /opt/hadoop/security/certs/thishost.csr \

-storepass password \

-keypass password

Sign the cert

sudo openssl x509 -req \

-in /opt/hadoop/security/certs/thishost.csr \

-CA /opt/hadoop/security/ca-certs/rootCA.pem \

-CAkey /opt/hadoop/security/ca-certs/rootCA.key \

-CAcreateserial \

-out /opt/hadoop/security/certs/thishost.pem \

-days 4000 \

-passin pass:password

Make a privileged CA truststore starting with the one included in the JDK

sudo cp /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security/cacerts /opt/hadoop/security/truststore/jssecacerts

Import rootCA.pem file into the CA truststore

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-noprompt \

-importcert \

-trustcacerts \

-alias rootCA \

-file /opt/hadoop/security/ca-certs/rootCA.pem \

-keystore /opt/hadoop/security/truststore/jssecacerts \

-storepass changeit

Import the root ca certificate file into the java keystore

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-noprompt \

-importcert \

-trustcacerts \

-alias rootCA \

-file /opt/hadoop/security/ca-certs/rootCA.pem \

-keystore /opt/hadoop/security/jks/thishost-keystore.jks \

-storepass password \

-keypass password

create hue truststore. Because we're using a CA signed

cert, we put the CA cert chain into the hue truststore

In this dummy environment, the CA cert chain only has the root CA

cat /opt/hadoop/security/ca-certs/rootCA.pem >> /tmp/ca-truststore.pem

Add the server private key and signed certificate to a NEW temporary

pkcs12 store

sudo rm /opt/hadoop/security/tmp/\*

sudo openssl pkcs12 \

-export \

-out /opt/hadoop/security/tmp/thishost-keystore.p12 \

-inkey /opt/hadoop/security/x509/thishost-unsignedkey.pem \

-in /opt/hadoop/security/certs/thishost.pem \

-CApath /opt/hadoop/security/ca-certs \

-name thishost \

-passin pass:password \

-passout pass:password

Now that we have a signed certificate in pckcs12 format, we want to rewrite

the java ones to be signed.

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-importkeystore \

-alias thishost \

-srckeystore /opt/hadoop/security/tmp/thishost-keystore.p12 \

-srcstoretype PKCS12 \

-srcstorepass password \

-srckeypass password \

-deststorepass password \

-destkeypass password \

-destkeystore /opt/hadoop/security/jks/thishost-keystore.jks

export keys and certs x509 and rsa via new p12 files. We show

this here because you can't always assume the p12 file already exists

even though we just created it. Sometimes you'll start with a java keystore

and need to generate keys and certs of different types from jks

sudo rm /opt/hadoop/security/tmp/\*

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-importkeystore \

-srckeystore /opt/hadoop/security/jks/thishost-keystore.jks \

-srcstorepass password \

-srckeypass password \

-destkeystore /opt/hadoop/security/tmp/thishost-keystore.p12 \

-deststoretype PKCS12 \

-srcalias thishost \

-deststorepass password \

-destkeypass password

sudo openssl pkcs12 -in /opt/hadoop/security/tmp/thishost-keystore.p12 \

-passin pass:password \

-nokeys \

-out /opt/hadoop/security/x509/thishost-cert.pem

sudo openssl pkcs12 -in /opt/hadoop/security/tmp/thishost-keystore.p12 \

-passin pass:password \

-nocerts \

-out /opt/hadoop/security/x509/thishost.key

sudo openssl rsa \

-in /opt/hadoop/security/x509/thishost.key \

-passin pass:password \

-out /opt/hadoop/security/x509/thishost-keynopw.pem

Build truststores

Add root CA cert to truststore:

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-noprompt \

-importcert \

-trustcacerts \

-alias rootCA \

-file /opt/hadoop/security/ca-certs/rootCA.pem \

-keystore /opt/hadoop/security/jks/truststore.jks \

-storepass password

For each host add the cert to the truststore

sudo /usr/java/jdk1.7.0\_67-cloudera/bin/keytool \

-noprompt \

-importcert \

-trustcacerts \

-alias thishost \

-file /opt/hadoop/security/certs/thishost.pem \

-keystore /opt/hadoop/security/jks/truststore.jks \

-storepass password

Once you've done this you can study the individual commands and get an appreciation for how to use SSL tools to convert keys and create an SSL environment for a cluster.

For more information:

https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

https://www.openssl.org/docs/

https://blog.talpor.com/2015/07/ssltls-certificates-beginners-tutorial/

http://www.cloudera.com/documentation/enterprise/latest/topics/cm\_sg\_ssl\_yarn\_mr\_hdfs.html

Data put into that encryption zone is encrypted transparently and automatically.

So do

hadoop fs -put sightings.csv /zone

The data gets encrypted into /zone

However, if we try to move it out of an encryption zone, it fails:

hadoop fs -mv /zone/sightings.csv /user/jinga

That's going to fail because you can't arbitrarily moving data in and out

of an encryption zone or between encryption zones.

This is because hadoop fs -mv is a namenode operation that merely

renames the files associated with the blocks. HDFS encryption actually

encrypts the blocks, so when data is moved in and out of an encryption

zone it's computationally expensive.

kinit jinga

hadoop distcp /zone/dataset.csv /user/jinga/

Fails because of the checksum check but that's to be expected

hadoop distcp -skipcrccheck -update /zone/dataset.csv /user/jinga

kinit hdfs (or chmod 777 /zone)

hadoop distcp -skipcrccheck -update /user/jinga/dataset.csv /zone/data

For more information:

http://blog.cloudera.com/blog/2015/01/new-in-cdh-5-3-transparent-encryption-in-hdfs/

http://www.cloudera.com/documentation/enterprise/5-3-x/topics/cdh\_sg\_hdfs\_encryption.html

After setting the following in the YARN service, deploy client configuration

ssl.server.keystore.location

ssl.server.keystore.password

ssl.server.keystore.keypassword

deploy client configuration and start ONLY HDFS, Zookeeper, and KMS

As a basic test to see if this worked, check HTTPS on the web ui

Notice the web UI is HTTPS

Now start YARN service. If there are issues expect failures here

Run a pi job from the gateway node

kinit jinga

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-maeduce/hadoop-mapreduce-examples.jar pi 10 10

Another failure point here is the encrypted shuffle. If MR jobs fail then SSL

isn't correctly enabled.

For more information:

http://www.cloudera.com/documentation/enterprise/latest/topics/cm\_sg\_ssl\_yarn\_mr\_hdfs.html

Before we begin, let’s note the IP of the Active Directory server and the IP

of an Impala server.

AD server ec2-54-164-173-81.compute-1.amazonaws.com

hostname ip-10-0-0-160.ec2.internal

To join a windows server with Active Directory, launch the following AMI:

AMI Name: Windows Server with Tableau and Impala ODBC Driver

AMI ID: ami-38a8e052 (US EAST)

Administrator/Passw0rd!

And make it a member of the same VPC and subnet as follows:

After connect to the new host

Navigate to control panel -> Network and internet -> network connections

Click on the ethernet connection that is active. Properties -> IPv4

Configure the DNS server so that the preferred DNS server is the

AD server

System properties -> change -> make a member of hadoopsecurity.local domain

Add users to remote desktop to allow them to connect

Control panel -> system & security -> remote desktop

Add

admin@hadoopsecurity.local

user1@hadoopsecurity.local

user2@hadoopsecurity.local

jinga@hadoopsecurity.local

Create a system DSN for Impala using the ODBC driver setup

If you’ve configured SSL on your cluster for Impala, you need to make sure

you can copy the cacerts file from the cluster over to the client host for

proper SSL authentication. You can do that using the redirect option to remote

desktop. Then copy the rootCA.pem file over to the desktop of the Windows

2012 server then to C:\Program Files\Cloudera ODBC Driver for Impala\lib\cacerts.pem

Authorization Kerberos

Realm hadoopsecurity.local

hostname internal hostname of server running impalad

The test connections is expected to fail when logged in as administrator

as this user doesn't have a kerberos id. After creating the DSN log out

and log in as admin@hadoopsecurity.local (using the password you defined

when you added that user).

Check test connections for the DSN when logged in as that user and refer

to the impalad or hs2 logs in Cloudera Manager to debug

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Active Directory Kerberos\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

From the AWS console, launch a Microsoft Server 2012 R2 Base AMI

into the same VPC with the same security group as the cluster

Use Microsoft remote desktop to connect, after you've obtained

the administrator password from the EC2 console. NOTE you might

need to update your microsoft remote desktop client.

server manager

local server

computer name

change to HADOOP-AD

Click add roles and features

Add the DNS server (disregard static IP)

Even if you click allow restart, it won't restart

Restart server at this point

After restart:

Add roles and features

Active Directory Domain Services

Click promote to directory server

Click add new forest

use HADOOPSECURITY.LOCAL as the domain

THE FOLLOWING STEPS ARE REQUIRED FOR LDAPS

AND THEY'RE REQUIRED FOR HUE SETUP AS WELL AS

CM KERBERIZATION. They're also covered in the

Now it's time to configure AD for use with CM

Rightclick on the domain, add new OU

Add user cm to OU

Rightclick on domain, delegate control wizard

Assign create user privs to cm

Next walk through Add roles and services to install AD cert services

Click next through wizard, making sure to install a CA

Click on configure AD cert services

Keep defaults on everything

Click Certification Authority Role

Setup type Enterprise CA. This is critical.

Root CA is selected

Keep default encryption types

NOTE the distinguished name suffix

DC=hadoopsecurity,DC=local

:q!CN=hadoopsecurity-HADOOP-AD-CA,DC=hadoopsecuritycurity,DC=local

Active Directory will serve LDAPS after reboot

STEPS ARE REQUIRED FOR LDAPS AND THEY'RE REQUIRED FOR HUE SETUP AS WELL AS

CM KERBERIZATION.

If you have an AD server running on this host and it's showing as green

you've completed this step.

user : cm

Password : Manchester1

If you have an AD server running on this host and it's showing as green

you've completed this step.

For more informaiton:

http://www.serverwatch.com/tutorials/article.php/1474461/Active-Directory-Tutorial-A-Quick-Start--Set-Up-Guide.htm

STEPS ARE REQUIRED FOR LDAPS AND THEY'RE REQUIRED FOR HUE SETUP AS WELL AS CM KERBERIZATION.

AD server in the hosts file on every node in the cluster.

putnmove.sh hosts /etc/hosts

Install openldap-clients and krb5-workstation if it's not there.

./clustercmd.sh sudo yum install openldap-clients -y

./clustercmd.sh sudo yum install krb5-workstation -y

Distribute the JCE policy file.

./putnmove.sh UnlimitedJCEPolicy/US\_export\_policy.jar /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security

./putnmove.sh UnlimitedJCEPolicy/local\_policy.jar /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security

Pull over krb5.conf and edit it

scp -i ./security.pem ec2-user@`head -1 cluster`:/etc/krb5.conf ./

1. The logging section is removed for simpilicity

2. The domain\_realm section is removed for simplicity. This section becomes

useful if you have multiple kerberos realms.

3. EXAMPLE.COM is replaced with the realm name HADOOPSECURITY.LOCAL

4. kerberos.example.com is replaced by the hostname of the KDC

(hadoop-ad.hadoopsecurity.local)

5. Then we add the following supported encryption types:

default\_tgs\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

default\_tkt\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

permitted\_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac-md5

Then copy krb5.conf to every node in the cluster

./putnmove.sh krb5.conf /etc/

ssh to a host

Perform a kinit cm

klist

kdestroy

Try an openssl connection

openssl s\_client -connect hadoop-ad.hadoopsecurity.local:636

hadoopsecurity-HADOOP-AD-CA

DC=hadoopsecurity,DC=local

CN=hadoopsecurity-HADOOP-AD-CA,DC=hadoopsecurity,DC=local

Cloudera Manager enable kerberos wizard

Select Active Directory

kdc server host

hadoop-ad.hadoopsecurity.local

Kerberos security realm

HADOOPSECURITY.LOCAL

encryption types from krb5.conf

Active directory suffix

ou=hadoop,DC=hadoopsecurity,DC=local

http://www.cloudera.com/documentation/enterprise/latest/topics/cdh\_sg\_hadoop\_security\_active\_directory\_integrate.html

Now that we've enabled kerberos authentication,

we can try to use it by ssh to a host in the cluster and running an hdfs client command

hadoop fs -ls /user

And we get an error that there are no valid credentials provided.

enabling users to use the secure cluster.

minimum user id in yarn should be 1

First, Add the users to every node in the cluster:

./clustercmd.sh sudo useradd tonga

./clustercmd.sh sudo useradd bonga

./clustercmd.sh sudo useradd monga

./clustercmd.sh sudo useradd admin

add the principal to kerberos. On the AD host:

Users and computers

Add users

tonga

bonga

monga

hdfs

admin

add the hdfs user so you can perform actions as hdfs (who is root in Hadoop)

Then on a cluster host

kinit hdfs

hadoop fs -mkdir /user/tonga

hadoop fs -mkdir /user/bonga

hadoop fs -mkdir /user/monga

hadoop fs -mkdir /user/admin (already exits)

hadoop fs -chown tonga:tonga /user/tonga

hadoop fs -chown bonga:bonga /user/bonga

hadoop fs -chown monga:monga /user/monga

hadoop fs -chown admin:admin /user/admin

https://www.centrify.com/solutions/big-data-security/hadoop/

(automation tool for user managment in the hadoop cluster)

kdestroy

To destroy the kerberos ticket, effectively logging out hdfs.

https://s3.amazonaws.com/securityhadoop/dataset.csv

hadoop fs -put dataset.csv

And you'll see it fails. kinit tonga and try again:

kinit tonga

hadoop fs -put dataset.csv

Run a pi job

It is the kerberos ticket, not the system user, who determines who owns

the file or who is performing the operation.

http://www.roguelynn.com/words/explain-like-im-5-kerberos/

install hue and oozie on the same server

and

create gateway node

In Cloudera Manager for the hue service

Enable ldap authentication

set:

backend to

desktop.auth.backend.ldapBackend

ldap\_url to

ldaps://hadoop-ad.hadoopsecurity.local

start\_tls checked

create LDAP users on login checked

LDAP search base should be:

dc=hadoop-ad,dc=hadoopsecurity,dc=local

LDAP bind user

cm

LDAP bind password

Manchester1

Set NT domain to

hadoopsecurity.local

Hue has import from ldap settings in the ui for user management.

These steps are unnecessary if ldap auth is enabled to create user on login

http://gethue.com/

http://gethue.com/how-to-configure-hue-in-your-hadoop-cluster/