Apache Hadoop Cluster Installation and Configuration to enable Security Experiments

Version 1.0, March 24, 2022

Abstract

This document provides step by step procedures to set up an Apache Hadoop cluster for security testing and experimentation.

Anne Tall

anne.tall@gmail.com

Table of Contents

1 AWS EC2 Instance Configuration 1
1.1 Virtual Private Cloud (VPC) Network Configuration 1
1.2 Network Address Translation 2
1.3 Launching Instances 2
1.3.1 Step 1: Choose an Amazon Machine Image (AMI) 3
1.3.2 Step 2: Choose an Instance Type 3
1.3.3 Step 3: Configure Instance Details 4
1.3.4 Step 4: Add Storage 5
1.3.5 Step 5 - Add Tags 5
1.3.6 Step 6 - Configure Security Group 5
1.3.7 Step 7 - Review Instance and Launch 6
1.4 AWS Route 53 DNS Service 7
1.5 AWS S3 Interface 8
1.5.1 AWS CLI installation 8
1.5.2 Assign the IAM role to the Instance 9
1.5.3 EC2 - S3 Interface Commands 9
1.6 AWS EC2 Configuration Summary 9
2 Operating System Configuration 10
2.1 Create User Accounts and Change Default Passwords 11
2.2 Update the Operating System 11
2.3 Change the Hostname 11
2.4 Confirm the IP Address 12
2.5 Update the Hosts File 12
2.6 Install JAVA 12
2.7 Install JCE for Kerberos 13
2.8 Set System-Wide Open File Count 13
2.9 Set Memory Swappiness 14
2.10 Set User File Handle Limits 14
2.11 Disable Transparent Hugh Page Compaction 15
2.12 Turn off BIOS Power Saving Options 16
2.13 Confirm Default File Permissions Mask 16
2.14 Disable SELinux 17

2.15 Turn on NTPD 17
2.16 Disable IPTables Firewall 17
2.17 Turn on Name Service Cache Daemon (NSCD) 17
2.18 Other Package Installations 18
2.19 Install Python3 18
2.20 Install Scala 19
2.21 Configure SSH 20
2.21.1 Update SSH Configuration File 20
2.21.2 Configure Passwordless SSH Connection 20
3 Hadoop Software Installation 21
4 Primary Hadoop Configuration Files and Settings 22
4.1 hadoop-env.sh 23
4.2 core-site.xml 24
4.3 hdfs-site.xml 25
4.4 mapred-site.xml 26
4.5 yarn-site.xml 27
4.6 log4j.properties 28
4.7 hadoop-policy.xml 29
4.8 master and slave files 29
5 Hadoop Start Up 30
5.1 Format HDFS 30
5.2 Start Hadoop 30
5.3 Verify Status 30
5.4 View the Web Console 30
5.5 Create HDFS Users 32
5.6 Stop Hadoop 32
6 Additional Hadoop Configuration Settings 32
6.1 HTTPS Settings 33
6.2 Kerberos 34
6.3 Stale Configuration Page in Cloudera Cluster 34
6.4 Speculative Execution 34
6.5 Data Node Commissioning and Decommissioning 35
6.6 Trash Functionality 36

6.7 Rack Awareness 36
6.8 High Availability 37
6.9 Cluster Federation Architecture 37
7 Additional Hadoop Ecosystem Component Configurations 37
7.1 Apache Spark 37
7.1.1 Download Spark 38
7.1.2 Configure Spark Environment Variables 39
7.1.3 (Optional) Updated the Spark log level in log4j 39
7.1.4 Modify the Workers File 39
7.1.5 Update spark-env.sh 39
7.1.6 Update spark-defaults.conf 40
7.1.7 Incorporated Configurations for Dynamic Resource Allocation 40
7.1.8 Update yarn-site.xml 41
7.1.9 Change YARN Heap Size 42
7.1.10 Verify the Spark Shell 42
7.1.11 Test Spark by Running an Example Program 43
7.1.12 Check the ResourceManager Web UI 44
7.2 Apache Ranger 45
7.2.1 Create the ranger account and environment variables 45
7.2.2 Download Ranger 46
7.2.3 Install MySQL 47
7.2.4 Install and Configure Solr for Ranger Audits 48
7.2.5 Configure Ranger Install.Properties Files 48
7.2.6 Execute Ranger Setup 50
7.2.7 Start Ranger 51
7.2.8 Login to the Ranger WebUI 51
7.2.9 Stop Ranger 51
7.3 Apache Atas 51
7.3.1 Create the atlas account 51
7.3.2 Install maven 52
7.3.3 Download Atlas 52
7.3.4 Configure and Start Atlas 52
7.3.5 Login to the Atlas WebUI 52

7.3.6 Stop Atlas ... 53
7.4 Directory Services - FreeIPA LDAP ... 53
7.4.1 Install FreeIPA Server ... 53
7.4.2 Configure FreeIPA Server ... 53
7.4.3 Log Into the WebUI ... 54
7.4.4 Verify FreeIPA is Running ... 54
7.4.5 Install the FreeIPA Clients ... 54
7.4.6 Configure the FreeIPA Clients ... 55
8 Footnotes and Tips ... 55
8.1 Deleting EC2 Instances ... 55
8.2 SSH Operations ... 55
8.3 Operating System Scripts ... 55
8.4 HDFS DFS Warning Messages ... 56
8.5 Group Settings /apache ... 56

1 AWS EC2 Instance Configuration

The Hadoop cluster software was installed on Amazon Web Services (AWS) Elastic Cloud Compute (EC2) instances, which are virtual machines running in Amazon's data center.

1.1 Virtual Private Cloud (VPC) Network Configuration

An AWS Virtual Private Cloud (VPC) network, titled "SEHadoop," was defined on AWS and used to interconnect the EC2 instances.

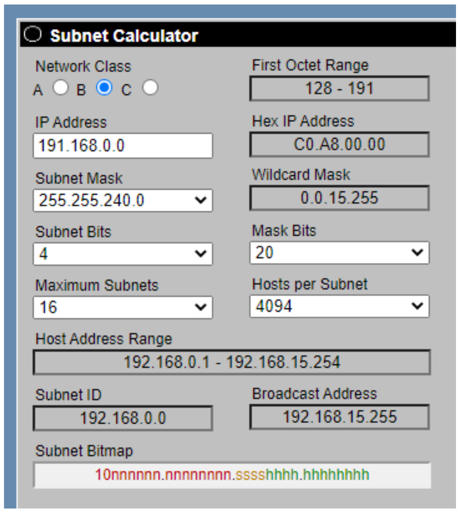
All the EC2 Instances send and receive data from computers on the Internet (primarily through an AWS Internet gateway that supports private to public IP translation), for example, to access to software, such as YUM updates.

The EC2 instances are grouped into three subnets, based upon common functionality. The core Hadoop components, that is the Name Node, Secondary Name Node, YARN, and all Data Nodes will be on one subnet. Security and management components, that is Apache Ranger and Apache Atlas, (and potentially Apache Ambari) are on a different subnet. The third subnet contains the user-facing applications, that is the Directory server (Lightweight Directory Access Protocol [LDAP]), a Command Line Interface (CLI) Hadoop client that serves as a "landing pad" for users to submit jobs (Spark and MapReduce) and execute Hadoop commands, and potentially Knox proxy server and Hive SQL interface.

The intent of grouping instances with common functionality into subnets is to support potential, future experiments. For example, to analyze protocol filtering to determine the functional limitations, security benefits, and performance tradeoffs in controlling communications between subnets.

The subnets use private IP addresses. AWS provides the Internet gateway that preforms the private to public IP address translation (NAT). In AWS the term "private network" refers to the group of computers (networked on a subnet) that cannot communicate with computers on the Internet, whereas computers with private IP addresses can get to the internet to send/receive data using a NAT service provided by the AWS Internet Gateway. LINK.

- VPC details:
    - Name - SEHadoop-vpc (Security Enhanced Hadoop)
    - IPv4 CIDR Block 192.168.0.0/20 (4,096 devices) ADS
    - Default tenancy and No IPv6 CIDR Block
- Subnets:
    - Name Tag – SEHadoop-subnet5, SEHadoop-subnet10, SEHadoop-subnet15, (subnet5 - Hadoop Core, subnet10 - Security/Management, subnet15 - User Interface)
    - Availability Zone – default, No Preference
    - IPv4 CIDR Block – 192.168.5.0/24 (192.168.10.0/24, 192.168.15.0/24)(254 devices) (see the Subnet Calculator block for the IP address range details)
- Internet Gateway attach it to the SEHadoop-vpc
    - Name Tag – SEHadoop-vpc-igw
    - Right Click on State and attach it to the vpc
- Route Table (Routes Tab)
    - Edit Routes / Add Routes: Destination 0.0.0.0/0 , Target Internet Gateway SeHadoop-vpc-igw


Subnet Calculator

Network Class A ○ B ◉ C ○

First Octet Range: 128 - 191

IP Address: 191.168.0.0

Hex IP Address: C0.A8.00.00

Subnet Mask: 255.255.240.0

Wildcard Mask: 0.0.15.255

Subnet Bits: 4

Mask Bits: 20

Maximum Subnets: 16

Hosts per Subnet: 4094

Host Address Range: 192.168.0.1 - 192.168.15.254

Subnet ID: 192.168.0.0

Broadcast Address: 192.168.15.255

Subnet Bitmap: 10nnnnnn.nnnnnnnn.sssshhhh.hhhhhhhh

1.2 Network Address Translation

AWS provides additional information about Network Address Translation (NAT).

Converting from private IP addresses to Internet accessible, public IP addresses, the NAT function, is built into the VPC.

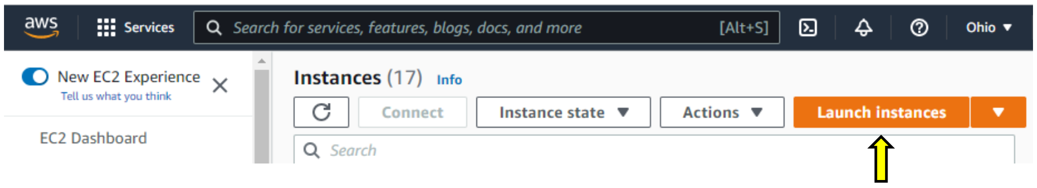
NAT is automatically set up with EC2 instance instantiation with Amazon domain names, AWS Dynamic Public IP addresses, and user selected private IP address. More information is at this link:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html

1.3 Launching Instances

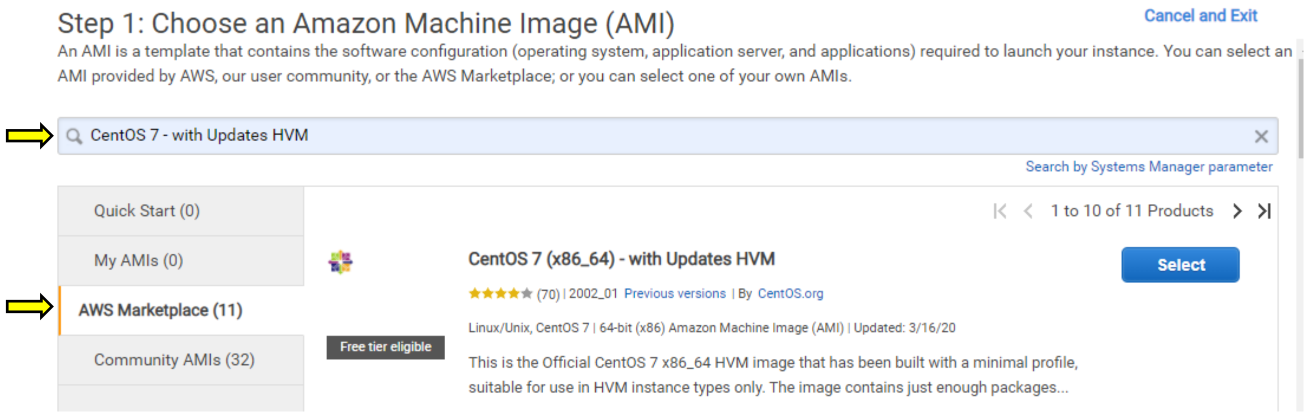
The following steps were executed to create the EC2 instances for the Hadoop cluster:

From the EC2 service dashboard, select “Launch Instance”, as indicated by the yellow arrow.



1.3.1 Step 1: Choose an Amazon Machine Image (AMI)

Hadoop was developed to run on the Linux operating system (OS). There are versions of Hadoop that run on Windows, however, for production, operational environments, Linux is the preferred OS. The CentOS 7 operating system was chosen because it is a free, open-source version of the RedHat operating system, which is commonly used in U.S. government and business, (that I work with).



Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

CentOS 7 - with Updates HVM

Search by Systems Manager parameter

Quick Start (0)

My AMIs (0)

AWS Marketplace (11)

Community AMIs (32)

1 to 10 of 11 Products

CentOS 7 (x86_64) - with Updates HVM

Select

★★★★★ (70) | 2002_01 Previous versions | By CentOS.org

Free tier eligible

Linux/Unix, CentOS 7 | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 3/16/20

This is the Official CentOS 7 x86_64 HVM image that has been built with a minimal profile, suitable for use in HVM instance types only. The image contains just enough packages...

Search for CentOS 7, by typing in "CentOS 7 - with Updates HVM" in the search bar, hitting return, and then select the "AWS Marketplace on the left tab. Click on the OS, and then click on the blue "select" icon. After making this selection, a pop up is displayed with more information about the OS and links to additional information, such as the default login user is "centos" and resource links, see list below. There are also pricing details with hourly fees for different instance types. The fees for the m5 and m4 instance types used are listed in the table below:

m5 Instance Type	Software	EC2	Total (per hour)
m5.large	$0.00	$0.096	$0.096/hr
m5.xlarge	$0.00	$0.192	$0.192/hr
m4.xlarge	$0.00	$0.20	$0.20/hr

The following resource links are available for more information:

https://wiki.centos.org/Cloud/AWS

https://wiki.centos.org/TipsAndTricks

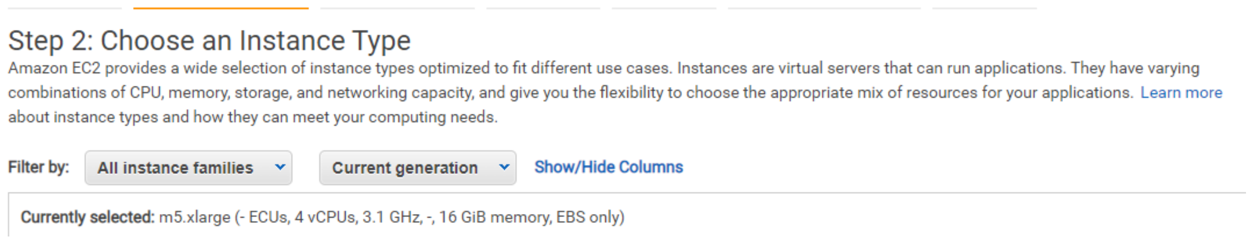
https://wiki.centos.org/FAQ

https://www.centos.org/forums/viewforum.php?f=44

Click "Continue" in the lower right corner of the pop-up.

1.3.2 Step 2: Choose an Instance Type

Select the instance type from the list based upon the table below.


Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance families | Current generation | Show/Hide Columns

Currently selected: m5.xlarge (- ECUs, 4 vCPUs, 3.1 GHz, -, 16 GiB memory, EBS only)

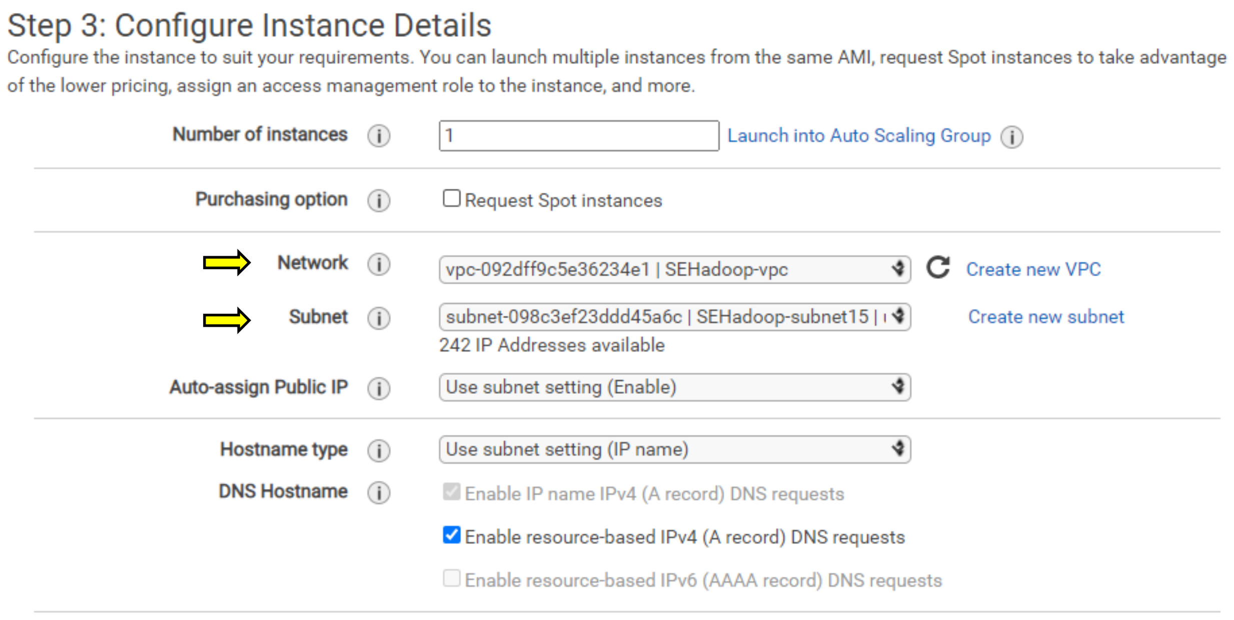
The drop down “All instance families” filter ccan be used to reduce the list to a certain family, such as “m5.”

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
m5	m5.large	2	8	EBS only	Yes	Up to 10 Gigabit	Yes
m5	m5.xlarge	4	16	EBS only	Yes	Up to 10 Gigabit	Yes
m4	m4.xlarge	4	8	EBS only	Yes	Moderate	Yes

Click on the “Next: Configure Instance Details” selection in the lower right corner.

1.3.3 Step 3: Configure Instance Details

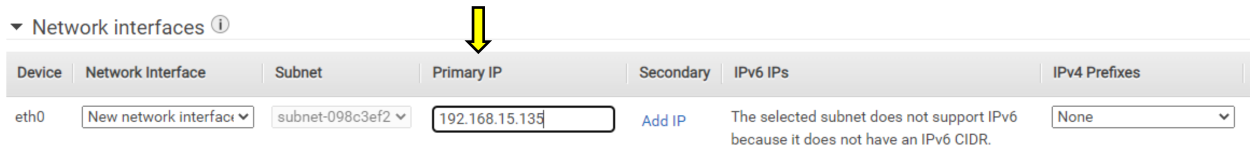
In step 3, the Network selected from the drop down is the SEHadoop-vpc previously created. The Subnet selected, either SEHadoop-subnet5, SEHadoop-subnet10 or SEHadoop-subnet 15, corresponds the subnet assignments previously defined. Default values were used for the other settings in the options at the top of screen. As shown in the figure below.


Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1 — Launch into Auto Scaling Group
Purchasing option	☐ Request Spot instances
Network	vpc-092dff9c5e36234e1 | SEHadoop-vpc — Create new VPC
Subnet	subnet-098c3ef23ddd45a6c | SEHadoop-subnet15 | — Create new subnet; 242 IP Addresses available
Auto-assign Public IP	Use subnet setting (Enable)
Hostname type	Use subnet setting (IP name)
DNS Hostname	☑ Enable IP name IPv4 (A record) DNS requests; ☑ Enable resource-based IPv4 (A record) DNS requests; ☐ Enable resource-based IPv6 (AAAA record) DNS requests

Scroll down and at the bottom of the screen, in the network interface section, add the private IP address in the “Primary IP” box as shown in the figure below.

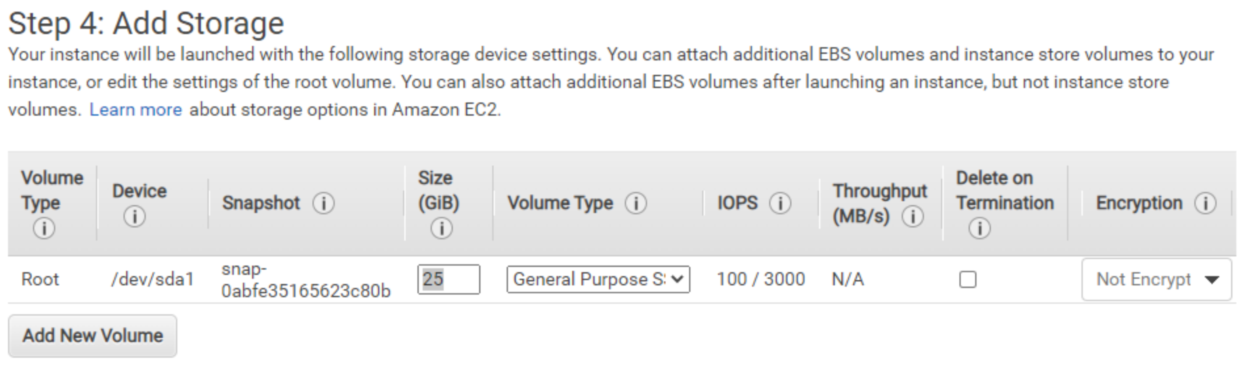


Network interfaces

Device	Network Interface	Subnet	Primary IP	Secondary	IPv6 IPs	IPv4 Prefixes
eth0	New network interface	subnet-098c3ef2	192.168.15.135	Add IP	The selected subnet does not support IPv6 because it does not have an IPv6 CIDR.	None

1.3.4 Step 4: Add Storage

The amount of Elastic Block Storage (EBS) assigned to each instance was



Step 4: Add Storage

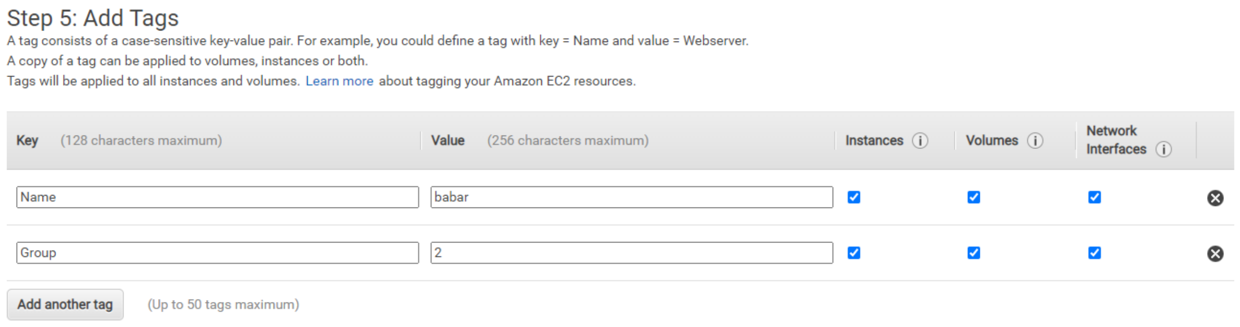
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0abfe35165623c80b	25	General Purpose S	100 / 3000	N/A	☐	Not Encrypt

Add New Volume

1.3.5 Step 5 - Add Tags

Add two tags. One tag with the key "Name" and the name of the computer. The second tag with the key "Group" and the number 2 for later group of the AWS instances created on the EC2 management dashboard. These tags are optional, however they are helpful in sorting the instances on the dashboard.



Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

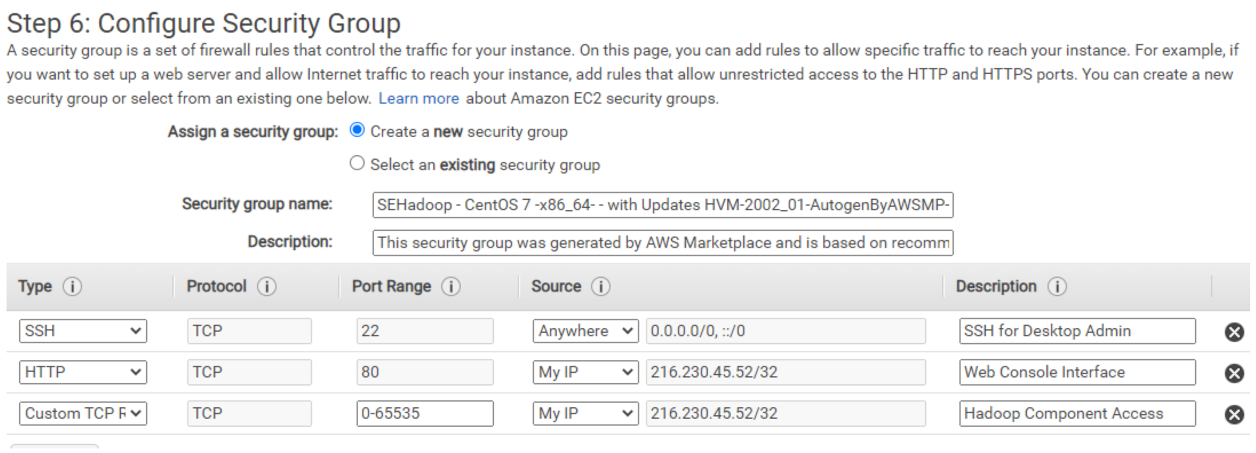
Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes	Network Interfaces
Name	babar	✓	✓	✓
Group	2	✓	✓	✓

Add another tag (Up to 50 tags maximum)

1.3.6 Step 6 - Configure Security Group

Create and reuse the created security group to limit inbound traffic and allow all outbound traffic.

The name of the security group is "SEHadoop - CentOS7 -x86_64 -- with Updates HVM-2002_01-AutogentByAWSMP" as shown below.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: SEHadoop - CentOS 7 -x86_64- - with Updates HVM-2002_01-AutogenByAWSMP-

Description: This security group was generated by AWS Marketplace and is based on recomm

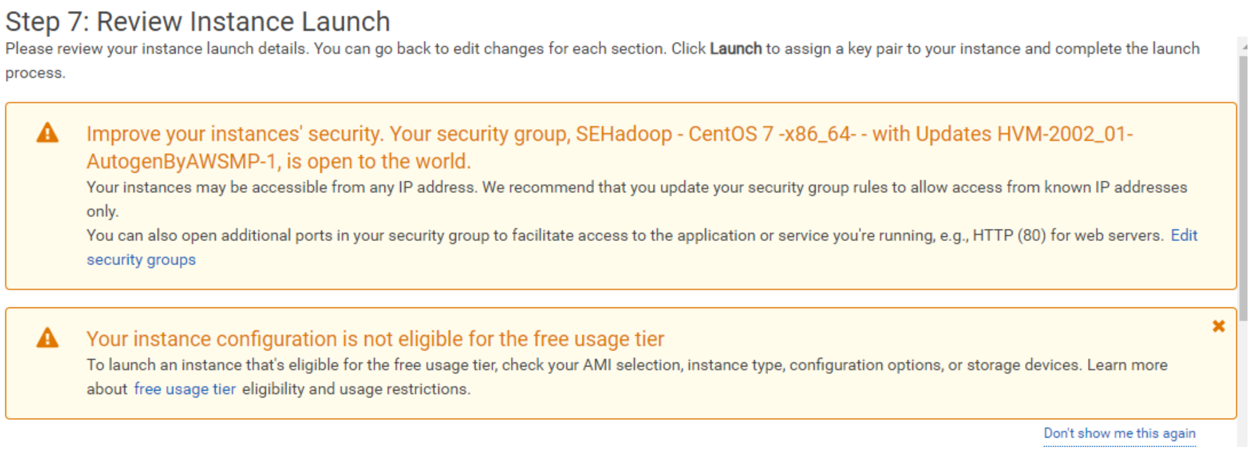
Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	SSH for Desktop Admin
HTTP	TCP	80	My IP	216.230.45.52/32	Web Console Interface
Custom TCP F	TCP	0-65535	My IP	216.230.45.52/32	Hadoop Component Access

For inbound traffic, the following was included in this new security group.

Type	Protocol	Port Range	Source	Comment/Note
SSH	TCP	22	Anywhere (0.0.0.0/0)	(default) (will change to my IP)
HTTP	TCP	80	Anywhere	(default) (will change to my IP)
Custom TCP Rule	TCP	0-65535	my IP address	Hadoop Component Access
Custom TCP Rule	TCP	0-65535	192.168.0.0/0	Private Hadoop Connections

Click on the “Review and Launch” selection in the lower right corner.

1.3.7 Step 7 - Review Instance and Launch



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, SEHadoop - CentOS 7 -x86_64- - with Updates HVM-2002_01-AutogenByAWSMP-1, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

Your instance configuration is not eligible for the free usage tier

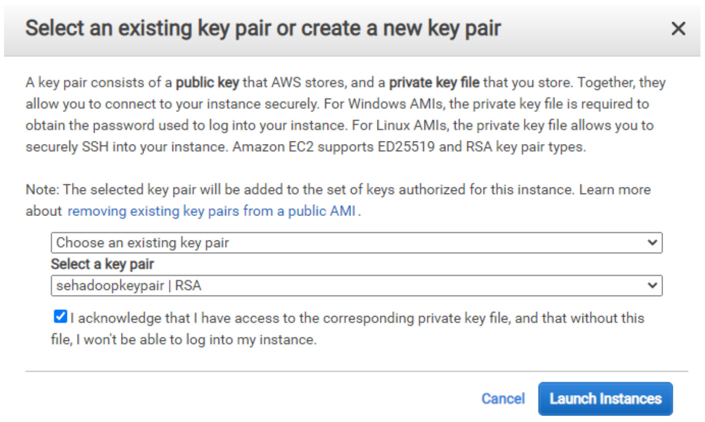
To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about free usage tier eligibility and usage restrictions.

Don't show me this again

Click on the “Launch” selection in the lower right corner.

At the pop-up selected the exiting key pair “sehadoopkeypair | RSA” that was previously created.

Instructions for a converting a new key into a version that can be used by Windows Putty is described at this link - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html .



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Choose an existing key pair

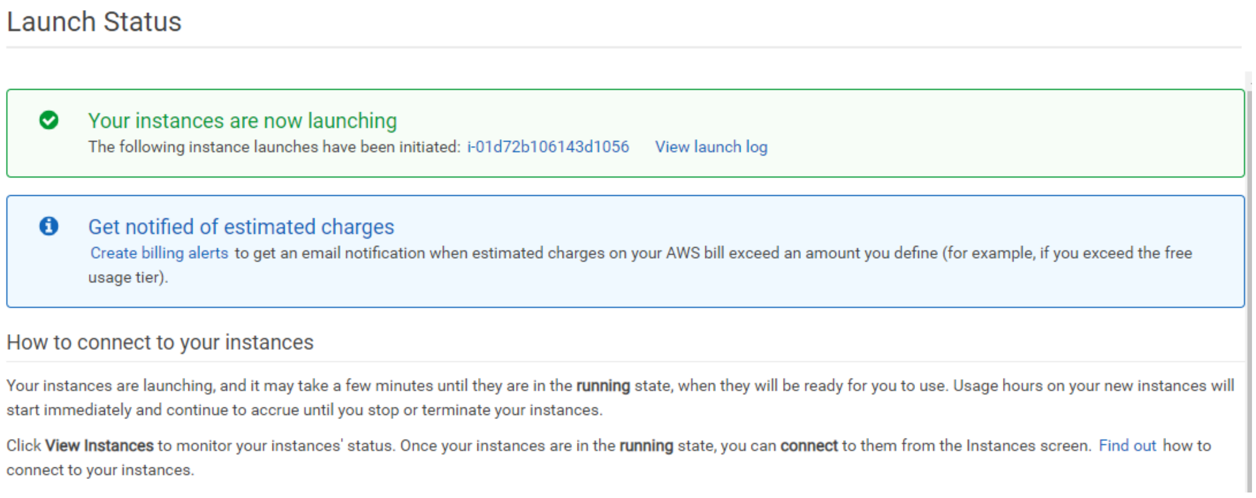
Select a key pair

sehadoopkeypair | RSA

☑ I acknowledge that I have access to the corresponding private key file, and that without this file, I won't be able to log into my instance.

Click on the box in front of the acknowledgement, then click on the “Launch Instances” selection in the lower right corner.

The instance will launch and options for additional information will be displayed.



Your instances are now launching

The following instance launches have been initiated: i-01d72b106143d1056 View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

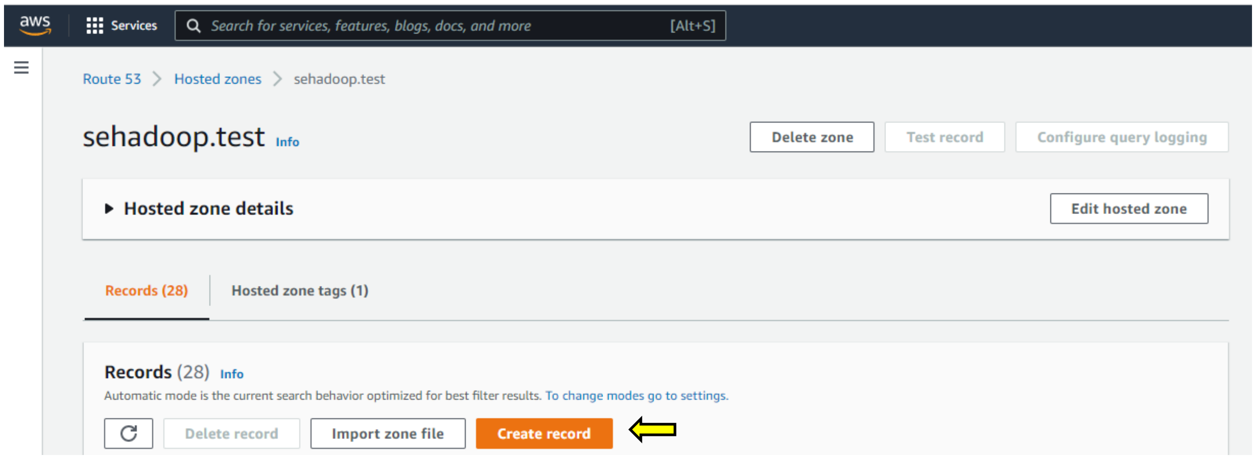
How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

1.4 AWS Route 53 DNS Service

I configured SEHadoop in the AWS Route 53 hosted zone, domain name service (DNS) to support host name, Fully Qualified Domain Name (FQDN) to private IP address resolution. This provides a central location to manage each hostname rather or in addition to the configuration in the /etc/hosts file. There is a small additional monthly charge for DNS (about $0.50 per month). An “A” type record is created for each host, by selecting “Create Record” as shown in the screen shot below.



The values entered from the table below are:

- Host Name for the "Record Name" prior to the Fully Qualified Domain Name (FQDN)
- IP address for the "Value"

The default values in Record Type (A), TTL (300) and Routing policy (Simple routing) were not changed.

1.5 AWS S3 Interface

A connection to the AWS Simple Storage Service (S3) was used to support moving data into and out of the EC2 cluster. To move files between an EC2 instance and an S3 bucket the AWS Command Line Interface (CLI) has to be installed. Also an IAM role has to be created and selected in the security settings for the EC2 instance.

On my cluster, the created IAM role is "SEHadoopS3Role"

1.5.1 AWS CLI installation

Reference: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

The AWS CLI was installed on the NameNode and HDFS Client Landing Pad as root, at the home directory. The software is installed in `/usr/local/bin/aws` folder.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
aws --version
```

Set the AWS home variable and path

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

Enter the following in this file:

```
export PATH=/usr/local/bin/aws:$PATH
```

close the file and change the permissions.

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

reboot or type the following

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

1.5.2 Assign the IAM role to the Instance

At the EC2 instance console execute the following actions:

- Select instance, right click and select "Actions" in the drop down, select "Security" then "Modify IAM role"
- Select the IAM role we just created
- Select "save"
- (or go to the "Security" setting on the EC2 instance from the "Actions" dropdown menu and select "Modify IAM Role" , then select the created IAM role from the drop down list)

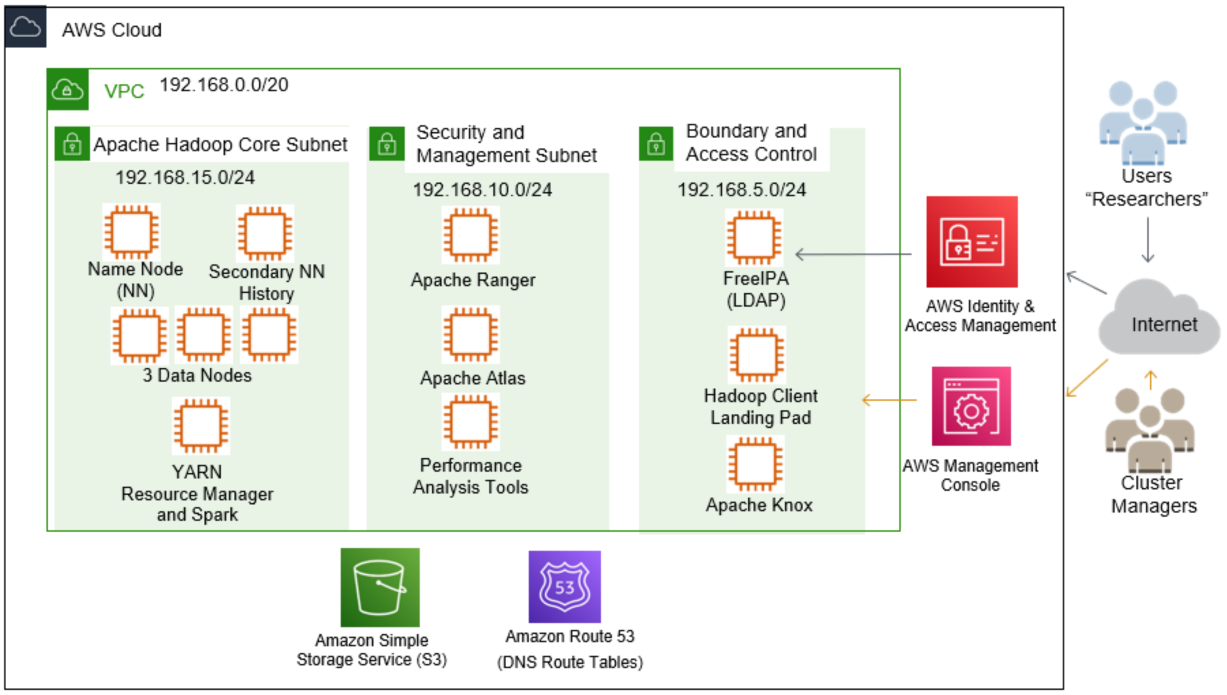
1.5.3 EC2 - S3 Interface Commands

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

```
aws s3 ls
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/
```

1.6 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 Operating System Configuration

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

2.1 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.

```
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

useradd -m yarn
passwd yarn
usermod -aG hadoop yarn
usermod -aG wheel yarn
```

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`.

2.2 Update the Operating System

Update the operating system.

```
yum -y update
```

2.3 Change the Hostname

Change the hostname in the `/etc/hostname` in the hostname file.

```
vi /etc/hostname
```

Remove the current name and change the host name to the Fully Qualified Domain Name (FQDN) in the following format:

```
hostname.sehadoop.test
```

Where “`hostname`” is the host name listed in the previous table.

This replaces the AWS assigned name that is in the format “ip-192-168-10-5.us-east-2.compute.internal.”

Confirm the name has been changed by executing the following command:

```
hostnamectl status
```

2.4 Confirm the IP Address

Execute the following command and confirm the private IP address of the computer is set correctly in accordance with the list in above, Section 1.4.

```
ifconfig
```

2.5 Update the Hosts File

Add the IP addresses, FQDNs and short alias names in the hosts file for all the instances in the cluster.

```
vi /etc/hosts
```

Add the following below the contents of this file:

```
192.168.15.131      pom.sehadoop.test              pom
192.168.15.132      flora.sehadoop.test            flora
192.168.15.133      alex.sehadoop.test             alex
192.168.15.135      babar.sehadoop.test            babar
192.168.15.115      basil.sehadoop.test            basil
192.168.15.120      celeste.sehadoop.test          celeste
192.168.15.125      zephir.sehadoop.test           zephir
192.168.10.110      pompadour.sehadoop.test        pompadour
192.168.10.115      troubadour.sehadoop.test       troubadour
192.168.5.140       cornelius.sehadoop.test        cornelius
192.168.5.50        belle.sehadoop.test            belle
192.168.5.160       truffles.sehadoop.test         truffles
192.168.10.105      izzy.sehadoop.test             izzy
```

2.6 Install JAVA

Download, install and configure JAVA 1.8 using the following commands:

```
yum -y install java-1.8.0-openjdk-devel
```

Confirm by typing the following command

```
java -version
```

Confirm installation with the following commands:

```
which java
sudo update-alternatives --config java
```

Set the JAVA home variable and path

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

Enter the following in this file:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk

export PATH=$JAVA_HOME/bin:$PATH
```

close the file and change the permissions.

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

reboot or type the following

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

confirm with the following command

```
echo $PATH

echo $JAVA_HOME
```

2.7 Install JCE for Kerberos

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on all hosts in the cluster. If you are using OpenJDK, some

distributions of the OpenJDK (such as RHEL/CentOS and Ubuntu) come with unlimited strength JCE automatically and therefore, installation of JCE is not required. For Java 8 Update 144 and earlier, you need to install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy files:

1. Download the unlimited strength JCE policy files from here: https://www.oracle.com/java/technologies/javase-jce8-downloads.html

```
wget --no-check-certificate --no-cookies --header "Cookie:
oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-
pub/java/jce/8/jce_policy-8.zip"
```

2. Extract the downloaded file

```
unzip -o -j -q jce_policy-8.zip -d
/usr/jdk64/jdk1.8.0_40/jre/lib/security/
```

3. Replace the existing policy JAR files in $JAVA_HOME/jre/lib/security with the extracted unlimited strength policy JAR files

2.8 Set System-Wide Open File Count

(Reference Edureka Presentation 3, Slide 5)

By default, the number of open files on Linux is 1024 for each user. If this default value is not changed, errors may occur, e.g., java.io.FileNotFoundException: (Too many open files)

The open file limit needs to be set to unlimited or a higher number like 32832, by executing the following command:

```
ulimit -Hn 32832
```

To persist this setting, edit the system wide file `/etc/sysctl.conf`, as follows:

```
vi /etc/sysctl.d/99-sysctl.conf
```

then add the following at the end of this file

```
fs.file-max=6544018
```

2.9 Set Memory Swappiness

Swappiness is a Linux kernel parameter which specifies how much RAM to swap. Default is 60. The higher the value of the swappiness parameter, the more aggressively the kernel will swap.

In Hadoop, maximizing data in-memory is ideal, so tune the OS so it will do memory swap only when there is an Out of Memory (OOM situation). Set the vm.swappiness kernel parameter to as minimum as possible, e.g. one or 10 (it cannot be zero (0)). This needs to be changed in the `/etc/sysctl.conf` file.

```
vi /etc/sysctl.d/99-sysctl.conf
```

add the following to this file

```
vm.swappiness=10
```

2.10 Set User File Handle Limits

Reference Edureka Video Lecture - III-Hadoop Cluster Setup and Working, (time 6min:45sec)

The following configures the number of files a particular user can have open.

```
vi /etc/security/limits.conf
```

add the following to the end of this file, directly above the “End of file” comment:

```
* soft nofile 32768
* hard nofile 32768
* soft nproc 32768
* hard nproc 32768
```

after reboot, confirm this setting by typing

```
ulimit -a
ulimit -n
```

This same change needs to be made in `/etc/security/limits.d/20-nproc.conf` as well

add a comment to the current variable line and the values above following this line

```
root soft nproc unlimited
```

The basic syntax of the limits.conf file is :

<domain><type><item><value>

The username is set in the domain field. In the type field, the type of limits that are applied to the mentioned domain are set. This field can be set as one of two values, soft or hard:

- Hard: The user can not cross the mentioned values.

- Soft: User can cross the mentioned value till previse Hard value.

(If running at start up with SystemD, the number of processes and files settings also needs to be configured in /etc/system/system/<app>.service)

2.11 Disable Transparent Hugh Page Compaction

Linux Transparent Huge Pates (THP) conceals the complexity of using Huge Pages and automates the creation of contiguous memory space which assists application performance. However, in Hadoop clusters it is detrimental to performance since the defragmentation process puts a heavy load on the processor. This is a necessary change ("HARD" requirement) for the Cloudera installations.

It is enabled by default in some Linux Operating Systems, but not all. For CentOS 7, to determine if it running, execute the following command:

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

If the setting returned in brackets is "never" then it does not need to be disabled, however if the setting in brackets is "always" then it must be disabled. For example, the following indicates defrag needs to be disabled.

```
[always] madvise never
```

If it is enabled, it needs to be disabled by executing the following command in CentOS:

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

To make this change permanent, add the `transparent_hugepage=never` kernel parameter option to the grub2 configuration file. Append or change the `transparent_hugepage=never` kernel parameter on the GRUB_CMDLINE_LINUX option in /etc/default/grub file. Make a backup of the file then make changes.

```
cp /boot/grub2/grub.cfg /boot/grub2/grub.cfg-BACKUP
vi /etc/default/grub
```

Make the change indicated in bold to this file.

```
GRUB_TIMEOUT=1
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL="serial console"
GRUB_SERIAL_COMMAND="serial --speed=115200"
GRUB_CMDLINE_LINUX="console=tty0 crashkernel=auto console=ttyS0,115200
transparent_hugepage=never"
GRUB_DISABLE_RECOVERY="true"
```

Rebuild the `/boot/grub2/grub.cfg` file by running the `grub2-mkconfig -o` command. Before rebuilding the GRUB2 configuration file, ensure the existing file there is backup of the existing `/boot/grub2/grub.cfg`.

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

After reboot of the system, confirm changes are in effect.

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

2.12 Turn off BIOS Power Saving Options

(Not applicable to running on AWS EC2)

2.13 Confirm Default File Permissions Mask

The umask setting is set to 0022, verify with the following command and change if needed in the bashrc file.

```
umask # value returned 0022
echo “umask 0022” >> ~/.bashrc
```

2.14 Disable SELinux

SELinux needs to be disabled. Edit the /etc/selinux/config file and change the SELINUX setting to disabled. The instance has to be rebooted for this to take effect.

Confirm it is disabled by typing the getenforce command

```
vi /etc/selinux/config
```

change `SELINUX=enforcing` to `SELINUX=disabled`

save the file, confirm after reboot using the following command

```
getenforce
```

2.15 Turn on NTPD

Enable the Network Time Protocol Daemon (NTPD) so all the clocks in the computer are synchronized

https://aws.amazon.com/blogs/aws/keeping-time-with-amazon-time-sync-service/

Execute the following commands for AWS EC2 instances:

```
yum install -y ntp
systemctl enable ntpd
```

Added the following line to the end of the NTP.conf file

(The IP address below is for the Amazon Time Sync service)

```
vi /etc/ntp.conf
server 169.254.169.123 prefer iburst
```

confirm it’s running with the following command

```
timedatectl
```

2.16 Disable IPTables Firewall

IPTables are turned off by default. The following commands are used to confirm and turn off autostart of IPTables and IP6Tables firewalls services

```
service iptables stop
chkconfig iptables off
service ip6tables stop
chkconfig ip6tables off
```

2.17 Turn on Name Service Cache Daemon (NSCD)

Reference Edureka Video Lecture - III-Hadoop Cluster Setup and Working, (time 6min:30sec)

Name service cache daemon (nscd) caches name service lookups and can improve performance with name resolution services, such as Domain Name Service (DNS). The `ncsd -g` command displays the configuration of the nscd configuration `/etc/nscd.conf`. The positive time to live (ttl) in seconds and the keep hot count values are checked. Values are set at `positive ttl:120` and `keep hot count: 20`. This will reduce the frequency of name service lookups.

```
yum install -y nscd
service nscd start
chkconfig --level 345 nscd on
nscd -g #displays the nscd configuration to confirm
```

In the chkconfig command above 3, 4, and 5 refer to run levels; chkconfig is used to configure a service to be started (or not) in a specific run level.

2.18 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
```

2.19 Install Python3

Python 2.7.5 is used by CentOS, however Python 3.x is required for Spark, so both versions of Python need to be installed.

Reference - https://tecadmin.net/install-python-3-9-on-centos/

https://support.datafabric.hpe.com/s/article/How-do-I-install-and-configure-custom-Python-version-for-Spark?language=en_US

as root

```
cd /usr/local/bin
yum install -y gcc openssl-devel bzip2-devel libffi-devel zlib-devel
wget https://www.python.org/ftp/python/3.9.7/Python-3.9.7.tgz
tar xzf Python-3.9.7.tgz
cd Python-3.9.7
./configure --enable-optimizations
make altinstall
ln -s /usr/local/bin/python3.9 /usr/local/bin/python3
```

```
vi /etc/profile.d/python3.sh
```

put the following in this file

```
export PATH=$PATH:/usr/local/bin/python3.9:usr/local/bin/python3
```

wq! - exit and safe the file just created

```
chmod +x /etc/profile.d/python3.sh
source /etc/profile.d/python3.sh
```

Confirm installation

```
python3 -V
python3 #starts shell, exit shell by typing exit()
```

2.20 Install Scala

Scala is required for Spark and needs to be installed on all the nodes.

```
su root
cd /usr/lib
wget http://downloads.lightbend.com/scala/2.11.12/scala-2.11.12.tgz
tar -zxvf scala-2.11.12.tgz -C /usr/lib
ln -s /usr/lib/scala-2.11.12 /usr/lib/scala
vi /etc/profile.d/scala.sh
```

put the following in this file

```
export SCALA_HOME=/usr/lib/scala
export PATH=$PATH:/usr/lib/scala/bin
```

exit the file (wq!)

```
chmod +x /etc/profile.d/scala.sh
source /etc/profile.d/scala.sh
```

exit (from root), confirm installation by typing

```
scala -version
```

confirm scala installation by typing “scala” at the command prompt,

exit the shell by typing :quit or ctrl-D

2.21 Configure SSH

After setting up all the systems, configure passwordless SSH connections between nodes. Create the SSH Key on the NameNode, babar. Reference - https://www.ssh.com/academy/ssh/copy-id

(Check on the passwordless SSH configuration in the High Availability configuration - may need to be bi-directional between all components.That is, the key generation command (below) needs to be executed on all nodes using the hdfs account)

2.21.1 Update SSH Configuration File

As root, edit the ssh configuration file to enable login using passwords

```
sudo vi /etc/ssh/sshd_config
```

remove the comment character (#) in front of `PubkeyAuthentication yes`

change

`AuthorizedKeysFile .ssh/authorized_keys`

to

`AuthorizedKeysFile %h/.ssh/authorized_keys`

remove the comment character in front of `PasswordAuthentication yes`

add a comment character to the front of `PasswordAuthentication no`

remove the comment character in front of `ChallengeResponseAuthentication yes`

add a comment character to the front of `ChallengeResponseAuthentication no`

Once the changes are made reload ssh

```
sudo systemctl restart sshd
```

2.21.2 Configure Passwordless SSH Connection

Execute the command below, (as hdfs on babar) (and as hdfs on celeste, yarn on babar and celeste) accept default location, hdfs home directory, for saving the key pair.

```
ssh-keygen -t rsa -P ""  #the quotes need to be in text format
```

Copy the generated key to the master node's authorized keys

```
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Copy the Master node's ssh key to all the data nodes / slave's authorized keys

```
ssh-copy-id -i /home/hdfs/.ssh/id_rsa.pub hdfs@datanode
```

```
ssh-copy-id -i /home/hdfs/.ssh/id_rsa.pub hdfs@pom
```

Note: the permissions / mode on authorized_keys should be 600 (default is 664), use ssh -v hdfs@host to debug ssh issues

Repeat these steps as yarn on celeste.

3 Hadoop Software Installation

Hadoop version 2.7.3 was used for this set up. Information about this version is at this web site: https://hadoop.apache.org/docs/r2.7.3/index.html

The most common location to install Hadoop is in the folder /usr/local/hadoop. However, it can also be installed in other locations, such as a folder at the root level, e.g., titled "apache".

Hadoop version 2.7.3 is installed in the folder /apache/hadoop using the hdfs account by executing the following commands.

```
sudo mkdir /apache
sudo chown -R hdfs:hadoop /apache
cd apache
wget http://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
tar xfz hadoop-2.7.3.tar.gz
```

Create a sim-link to hadoop

```
ln -s hadoop-2.7.3 hadoop
```

The following folders will be created automatically by Hadoop based upon the hdfs-site.xml file. Alternatively, the following commands create the folders on the Linux OS:

```
mkdir /apache/hadoop/hadoop-2.7.3/hdata/namenode
mkdir /apache/hadoop/hadoop-2.7.3/hdata/datanode
```

Create a start up Hadoop home variable and path, as root

```
sudo vi /etc/profile.d/hadoop.sh
```

put the following in this file:

```
#Hadoop Startup Environment Variables
#These environment variables are also set in hadoop-env.sh
export HADOOP_HOME=/apache/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_LOG_DIR=$HADOOP_HOME/log
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.libary.path=$HADOOP_HOME/lib/native"
#export HADOOP_HEAPSIZE_MAX=1000    #set to  default = 1G
#export HADOOP_NAMENODE_OPTS="-Xmx5g"
export HADOOP_PID_DIR=$HADOOP_HOME/hadoop2_data/hdfs/pid
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

#from Hadoop version 2.10.1
export PDSH_RCMD_TYPE=ssh
```

Make this file executable.

```
sudo chmod +x /etc/profile.d/hadoop.sh
```

Activate the changes and confirm with the following command

```
source /etc/profile.d/hadoop.sh
echo $PATH
```

4 Primary Hadoop Configuration Files and Settings

Hadoop configuration files are located in the hadoop folder in the /etc/hadoop folder, i.e., /apache/hadoop/hadoop/etc/hadoop. There are approximately 8 primary configuration files for Hadoop core. They are listed in the table below. Not all files are changed at initial installation as noted. The initial settings added to these files for initial installation are described in the following sections.

File	Description
hadoop-env.sh	environment variables used in the scripts to run Hadoop, variables that affect the JDK used by the Hadoop daemons (in /bin/hadoop), a key variable in this file is $JAVA_HOME
core-site.xml	configuration settings for Hadoop Core, such as I/O settings that are common to HDFS and MapReduce, identifies where the NameNode runs in the cluster
hdfs-site.xml	configurations for the HDFS daemons, NameNode, Secondary NameNode, and DataNodes
yarn-site.xml	configuration settings for YARN, Resource Manager, Node Manager, Application Master and Containers
yarn-env.sh	location where yarn configuration info is stored, e.g., PID, (No changes were made to the default configuration of this file)
mapred-site.xml	configuration settings for MapReduce applications
mapred-env.sh	site specific customization of Hadoop daemons for MapReduce processing, (No changes were made to the default configuration of this file)
log4j.properties	for customizing logging configurations For example, set logging only for ERRORs.
hadoop-policy.xml	access control lists for Hadoop services, (No changes were made to the default configuration of this file during initial set up)

masters	lists the secondary NameNode (one machine per line)
slaves	lists the machines that run a DataNode and NodeManager (one per line)

After configuring these files on the NameNode (babar), (as hdfs), copied the files to all the other hosts using the scp command.

```
scp $HADOOP_HOME/etc/hadoop/hadoop-env.sh
basil:$HADOOP_HOME/etc/hadoop/hadoop-env.sh

scp $HADOOP_HOME/etc/hadoop/core-site.xml
basil:$HADOOP_HOME/etc/hadoop/core-site.xml

scp $HADOOP_HOME/etc/hadoop/hdfs-site.xml
basil:$HADOOP_HOME/etc/hadoop/hdfs-site.xml

scp /$HADOOP_HOME/etc/hadoop/yarn-site.xml
basil:$HADOOP_HOME/etc/hadoop/yarn-site.xml

scp $HADOOP_HOME/etc/hadoop/mapred-site.xml
basil:$HADOOP_HOME/etc/hadoop/mapred-site.xml

scp $HADOOP_HOME/etc/hadoop/log4j.properties
basil:$HADOOP_HOME/etc/hadoop/log4j.properties

scp $HADOOP_HOME/etc/hadoop/slaves basil:$HADOOP_HOME/etc/hadoop/slaves
```

Created a file with all the computer names except the NameNode titled "`computers-nn`" to expedite the copying of files by executing the following commands:

```
for i in $(cat computers-nn); do scp $HADOOP_HOME/etc/hadoop/hadoop-env.sh
$i:$HADOOP_HOME/etc/hadoop/hadoop-env.sh; done


for i in $(cat computers-nn); do scp $HADOOP_HOME/etc/hadoop/core-site.xml
$i:$HADOOP_HOME/etc/hadoop/core-site.xml; done
```

4.1 hadoop-env.sh

Add the following environment variables to this file, directly below the "Set Hadoop-specific environment variables here." comment:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk

export HADOOP_HOME=/apache/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_LOG_DIR=$HADOOP_HOME/log
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.libary.path=$HADOOP_HOME/lib/native"
#export HADOOP_HEAPSIZE_MAX=1000    #set to  default = 1G
#export HADOOP_NAMENODE_OPTS=”-Xmx5g”
export HADOOP_PID_DIR=$HADOOP_HOME/hadoop2_data/hdfs/pid
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

#from Hadoop version 2.10.1
export PDSH_RCMD_TYPE=ssh
```

Reference: regarding HADOOP_OPTS, should `/native` be at the end of this environment variable?
https://www.edureka.co/community/110/hadoop-unable-native-hadoop-library-your-platform-warning

4.2 core-site.xml

The following properties were added between the `<configuration>` and `</configuration>` lines to the core-site.xml file. By default, there are no properties in this file. A description of the properties is at this link:

https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/core-default.xml

```
<property>
     <name>fs.defaultFS</name>
     <value>hdfs://babar.sehadoop.test:8020</value>
     <description>URI of the NameNode, default port is 8020</description>
</property>
<property>
      <name>hadoop.tmp.dir</name>
      <value>/tmp/hadoop-${user.name}</value>
      <description>Hadoop creates this folder on HDFS</description>
</property>
<property>
      <name>io.file.buffer.size</name>
      <value>131072</value>
</property>

<property>
     <name>hadoop.security.authentication</name>
     <value>simple</value>
     <description>if set to Kerberos, enables Kerberos auth</description>
</property>
<property>
     <name>hadoop.security.authorization</name>
     <value>true</value>
     <description>enables RPC service-level authorization</description>
</property>
<property>
     <name>hadoop.proxyuser.superuser.hosts</name>
     <value>*</value>
     <description>allow all hosts superuser impersonation access</description>
</property>
<property>
```

```
     <name>hadoop.proxyuser.superuser.groups</name>
     <value>*</value>
     <description>allow all groups superuser impersonation access</description>
</property>
```

4.3 hdfs-site.xml

The following properties were added between the `<configuration>` and `</configuration>` lines to the hdfs-site.xml file. By default, there are no properties in this file. A description of the properties is at this link:

https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

```
<property>
     <name>dfs.replication</name>
     <value>3</value>
      <description>the number of block replications created</description>
</property>
<property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/apache/hadoop/hdata/namenode</value>
      <description>Hadoop creates this folder</description>
</property>
<property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/apache/hadoop/hdata/datanode</value>
      <description>Hadoop creates this folder</description>
</property>
<property>
     <name>dfs.namenode.secondary.http-address</name>
     <value>basil.sehadoop.test:50090</value>
     <description>URI of the 2ndNameNode, default port is 50090</description>
</property>
<property>
        <name>dfs.permissions.enabled</name>
        <value>true</value>
    </property>
    <property>
         <name>dfs.permissions.superusergroup</name>
         <value>supergroup</value>
         <description>default value is supergroup</description>
    </property>
<property>
         <name>dfs.namenode.acls.enabled</name>
         <value>true</value>
    </property>
    <property>
         <name>dfs.cluster.administrators</name>
         <value>hdfs hadoop</value>
         <description>value format is user1,user2 group1,group2</description>
    </property>
```

4.4 mapred-site.xml

Copy mapred-site.xml.template to mapred-site.xml.

```
cp /apache/hadoop/hadoop/etc/hadoop/mapred-site.xml.template
/apache/hadoop/hadoop/etc/mapred-site.xml
```

The following properties were added between the `<configuration>` and `</configuration>` lines to the mapred-site.xml file. By default, there are no properties in this file. A description of the properties is at this link:

https://hadoop.apache.org/docs/r2.7.3/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml

```
<property>
     <name>mapreduce.framework.name</name>
     <value>yarn</value>
      <description>local, classic or yarn, default port is 8021</description>
</property>
<property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
    <value>2048</value>
    <description>Must be larger than both map + reduce
memory.mb</description>
</property>
<property>
    <name>yarn.app.mapreduce.am.command-opts</name>
    <value>-Xmx1024m</value>
</property>
<property>
    <name>mapreduce.map.memory.mb</name>
    <value>1024</value>
    <description>equal to yarn.scheduler.minimum-allocation-mb</description>
</property>
<property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>1024</value>
    <description>equal to yarn.scheduler.minimum-allocation-mb</description>
</property>
<property>
      <name>mapreduce.jobtracker.address</name>
      <value>hdfs://celeste.sehadoop.test:8021</value>
      <description>replaces deprecated mapred.job.tracker</description>
</property>
<property>
      <name>mapreduce.task.io.sort.factor</name>
      <value>10</value>
      <description>streams merged at once while sorting files,
      default 100</description>
</property>
<property>
      <name>mapreduce.task.io.sort.mb</name>
      <value>100</value>
      <description>memory-limit while sorting data, 512 default</description>
</property>
```

4.5 yarn-site.xml

The following properties were added between the `<configuration>` and `</configuration>` lines to the yarn-site.xml file. By default, there are no properties in this file. A description of the properties is at this link:

https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-common/yarn-default.xml

```
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>192.168.15.120</value>
    ~~<value>celeste.sehadoop.test</value>~~
</property>
<property>
      <name>yarn.resourcemanager.address</name>
      <value>celeste.sehadoop.test:8032</value>
</property>
<property>
     <name>yarn.resourcemanager.resource-tracker.address</name>
     <value>celeste.sehadoop.test:8031</value>
</property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>celeste.sehadoop.test:8030</value>
</property>
<property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>celeste.sehadoop.test:8033</value>
</property>
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>celeste.sehadoop.test:8088</value>
</property>
<property>
     <name>yarn.nodemanager.aux-services</name>
     <value>mapreduce_shuffle</value>
      <description>identifies auxiliary service</description>
</property>
<property>
     <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
      <description>the class name for aux service, shuffle</description>
</property>
<property>
      <name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.Capac
ityScheduler</value>
```

```
	<description>CapacityScheduler and FairScheduler recommended</description>
</property>
<property>
	<name>yarn.application.classpath</name>
	<value>$HADOOP_CONF_DIR, $HADOOP_COMMON_HOME/share/hadoop/common/*,
$HADOOP_COMMON_HOME/share/hadoop/common/lib/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*</value>
	   <description>classpaths for typical applications</description>
</property>
<property>
	<name>yarn.scheduler.minimum-allocation-mb</name>
	<value>1024</value>
</property>
<property>
	<name>yarn.scheduler.maximum-allocation-mb</name>
	<value>4096</value>
</property>
<property>
	<name>yarn.scheduler.maximum-allocation-vcores</name>
	<value>1</value>
	<description>1 is the default value</description>
</property>
<property>
	<name>yarn.nodemanager.resource.memory-mb</name>
	<value>4096</value>
	<description>total MB resources on NodeManager to run containers, -1 is auto calculated, 8192 MB default</description>
</property>
<property>
	<name>yarn.nodemanager.resource.vmem-pmem-ratio</name>
	<value>2.1</value>
	<description>ratio of virtual and physical memory</description>
</property>
<property>
	<name>yarn.nodemanager.pmem-check-enabled</name>
	<value>false</value>
</property>
<property>
	<name>yarn.nodemanager.vmem-check-enabled</name>
	<value>false</value>
</property>
```

4.6 log4j.properties

The $HADOOP_HOME/etc/hadoop/log4j.properties file defines the logging related properties in the Hadoop cluster.

The default logging level is INFO and destination is console, as specified by the following property in this file. This setting results in verbose output with a lot of messages on the console and can lead to confusion in finding the actual issue messages that require attention.

```
hadoop.root.logger=INFO,console
```

This setting can be changed to "ERROR", as follows, so that only error logs are displayed on the console. (This was done after installation was confirmed.)

```
hadoop.root.logger=ERROR,console
```

The default path for log storage is $HADOOP_HOME/logs.

4.7 hadoop-policy.xml

Like the other files, the properties are added between the `<configuration>` and `</configuration>` lines to the hadoop-policy.xml file. By default, many of the properties are set with the wildcard "*" indicating all users/groups are permitted. A description of the properties is at this link:

Apache Hadoop 3.3.1 – Service Level Authorization Guide

All of the properties in this file have the value "*" which is the wild card that does not limit access. None of the values in this file were changed. Two example properties are listed below.

```
<property>
     <name>security.job.client.protocol.acl</name>
     <value>*</value>
     <description>users groups given access to submit jobs</description>
</property>
<property>
     <name>security.datanode.protocol.acl</name>
     <value>*</value>
     <description>users groups access to datanodes</description>
</property>
```

4.8 master and slave files

These files contain the hostnames or IP addresses of instances/computers in the cluster.

List the secondary NameNode (basil) in the masters file on the NameNode (babar).

List the NameNode (babar) in the masters file on the DataNodes (pom, flora, alex).

```
vi /apache/hadoop/hadoop/etc/hadoop/masters
```

`basil.sehadoop.test` or `babar.sehadoop.test`

List all the DataNode in the slaves file, one per line on all computers in the cluster

```
vi /apache/hadoop/hadoop/etc/hadoop/slaves
```

```
pom.sehadoop.test
flora.sehadoop.test
alex.sehadoop.test
```

5 Hadoop Start Up

5.1 Format HDFS

AFTER making the configuration changes to the Hadoop configuration files listed in the previous section, Section 4, execute the Hadoop format command.

```
cd /apache/hadoop/bin
./hdfs namenode -format
```

This formats HDFS and is only executed once, as part of set up. The FSimage is initialized. If this command is run on an existing HDFS, the stored data will be lost.

Look for the following message to confirm success:

`INFO common.Storage: Storage directory /apache/hadoop/hdata/`**`namenode has been successfully formatted.`**

5.2 Start Hadoop

Execute the following command on the NameNode to start the Hadoop daemons:

```
start-all.sh
```

5.3 Verify Status

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

Execute the following command to verify node status:

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

5.4 View the Web Console

Using a web browser (e.g., Chrome), replace “hostIP” 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.

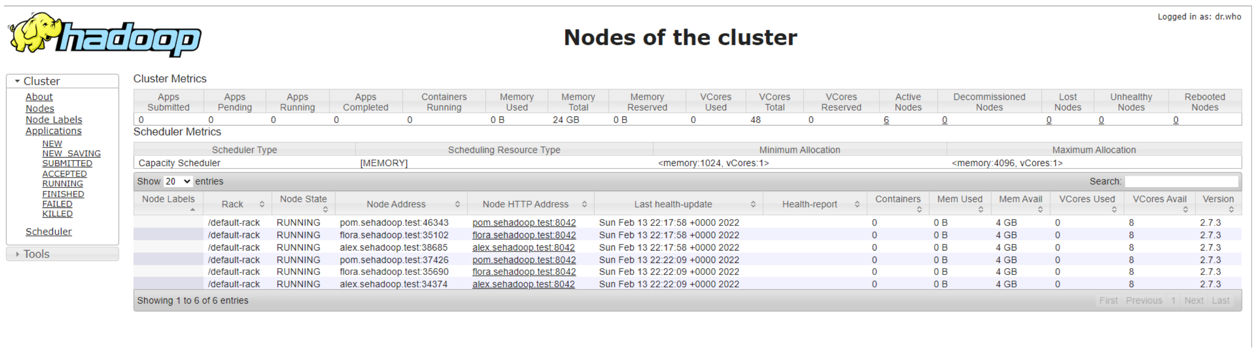
http://localhost:50070/dfshealth.html

Resource Manger status http://hostIP:8088

NameNode status http://hostIP:50070/dfshealth.jsp

DataBlock Scanner Report http://hostIP:50075/blockScannerReport

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



Logged in as: dr.who

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

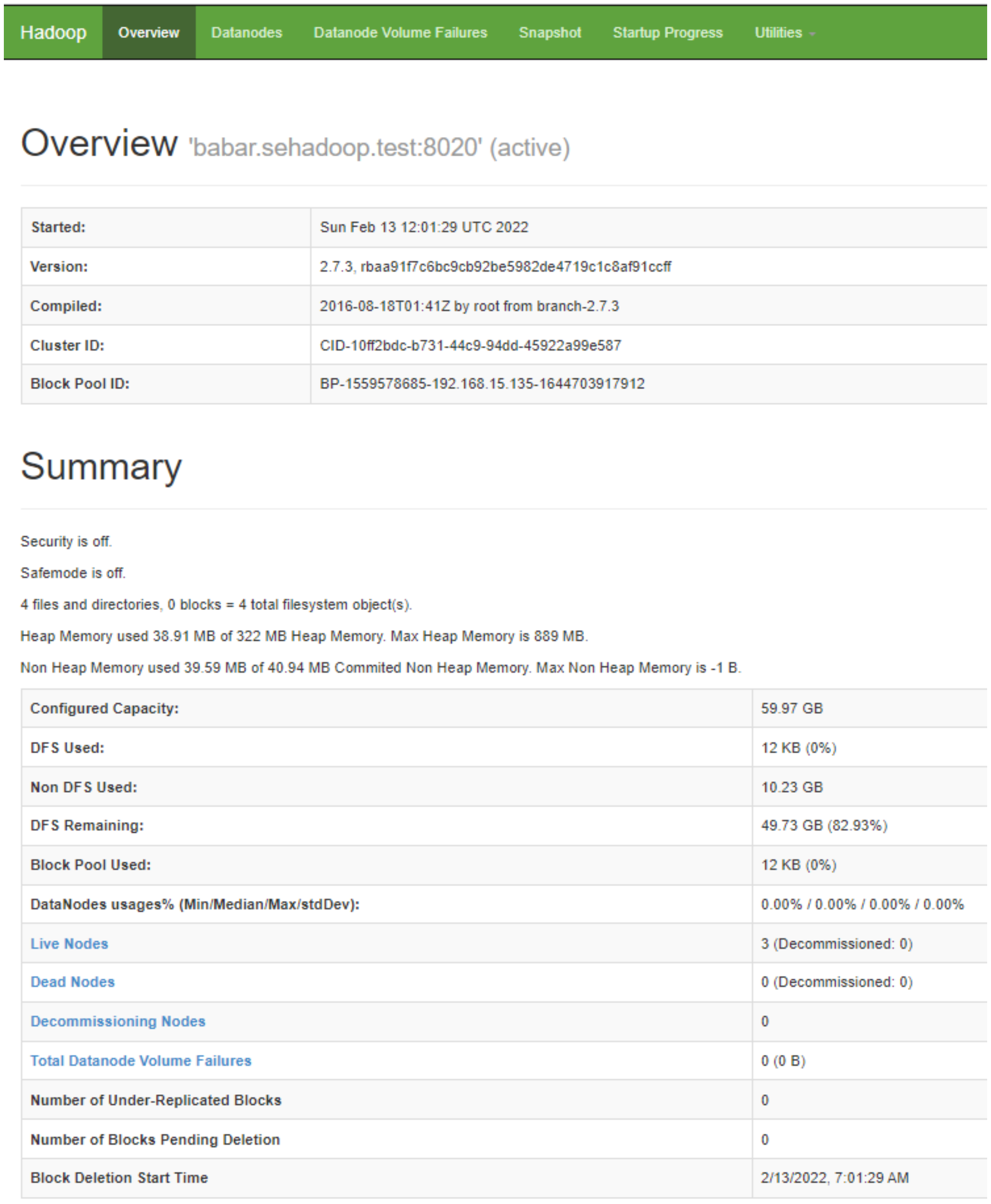
Scheduler Metrics

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

5.5 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
```

5.6 Stop Hadoop

Execute the following commands to shut down the cluster.

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

6 Additional Hadoop Configuration Settings

After initial Hadoop set up, the following additional configuration settings may be added to enhance the Hadoop cluster. These enhancements include improving the security, reliability, and performance of the cluster. Testing the performance and security of Hadoop under these various configurations is an interesting and important area for further research.

6.1 HTTPS Settings

SSL (TLS) (HTTPS) can be configured to protect data transfer between the Hadoop Web User Interface (consoles) and clients. This is recommended but not required to configure Hadoop security Kerberos.

In hdfs-site.xml, set the `dfs.http.policy` property to either `HTTPS_ONLY` or `HTTP_AND_HTTPS` to enable SSL for the web console of the HDFS daemons. This does not affect KMS nor HttpFS as they are implemented on top of Tomcat and do not react/respond to this parameter. See Hadoop KMS for instructions on enabling KMS over HTTPS. See Hadoop HDFS over HTTP-Server Setup for instructions on HttpFS over HTTPS.

To enable SSL for Web Console of YARN daemons set `yarn.http.policy` to `HTTPS_ONLY` in yarn.site.xml

To enable SSL for Web Console of MapReduce JobHistory server set `mapreduce.jobhistory.http.policy` to `HTTPS_ONLY` in mapred-site.xml

In summary, add the following properties to the identified configuration files:

hdfs-site.xml

```
<property>
     <name>dfs.http.policy</name>
     <value>HTTP_AND_HTTPS</value>
     <description>both HTTP and HTTPS supported</description>
</property>
<property>
     <name>dfs.namenode.https-address</name>
     <value>babar.sehadoop.tes:50470</value>
      <description>non-HA and non-Federation mode</description>
</property>
<property>
     <name>dfs.namenode.secondary.https-address</name>
     <value>basil.sehadoop.tes:50091</value>
     <description>web UI for 2nd namenode</description>
</property>
```

yarn.site.xml

```
<property>
     <name>yarn.http.policy</name>
     <value>HTTPS_ONLY</value>
      <description>only HTTPS supported</description>
</property>
```

mapred-site.xml

```
<property>
     <name>dfs.http.policy</name>
     <value>HTTP_AND_HTTPS</value>
      <description>both HTTP and HTTPS supported</description>
</property>
```

core-site.xml

```
<property>
     <name>hadoop.rpc.protection</name>
     <value>authentication</value>
     <description>default value</description>
</property>
```

6.2 Kerberos

Kerberos provides strong identification/authentication services in Apache Hadoop; however, it is complex to configure.

https://www.ibm.com/docs/en/spectrum-scale-bda?topic=hdfs-enabling-kerberos-opensource-apache-hadoop-ces

https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/authentication-with-kerberos/content/enabling_kerberos_authentication_using_ambari.html

core-site.xml

```
<property>
     <name>hadoop.security.auth_to_local</name>
     <value></value>
     <description>expressions supporting Kerberos</description>
</property>
```

6.3 Stale Configuration Page in Cloudera Cluster

Stale configuration parameters apply only to Cloudera Manager implementations. This feature is not in available in Apache Hadoop configurations.

6.4 Speculative Execution

Speculative execution backs up slow tasks on alternative machines, to manage performance impacts. Speculative Execution is turned on by default. Tasks which are taking a long time to finish, e.g., due to memory unavailability, are copied to another DataNode. If the job finishes at this other location, it is killed in the original DataNode.

The following properties when added to mapred-site.xml file can be used to manage speculative execution. The default values are true and need to set to false to disable this service:

```
<property>
     <name>mapreduce.map.tasks.speculative.execution</name>
     <value>true</value>
</property>
<property>
     <name>mapreduce.reduce.tasks.speculative.execution</name>
     <value>true</value>
```

```
</property>
```

6.5 Data Node Commissioning and Decommissioning

These actions address adding and removing DataNodes to replace failed or overloaded DataNodes.

This process involves the following steps:

(1) Adding the new DataNode to the slaves file (Section 4.8)

(2) Updating yarn-site.xml with the following properties

```
<property>
    <name>yarn.resourcemanager.node.include-path</name>
    <value>/usr/lib/hadoop/etc/hadoop/yarn.include</value>
</property>
<property>
    <name>yarn.resourcemanager.nodes.exclude-path</name>
    <value>/apache/hadoop/etc/hadoop/yarn.exclude</value>
</property>
```

(3) Updating hdfs-site.xml with the following properties

```
<property>
    <name> dfs.hosts</name>
<value>/usr/lib/hadoop/etc/hadoop/dfs.include</value>
    </property>
<property>
    <name> dfs.hosts.exclude</name>
    <value>/apache/hadoop/etc/hadoop/dfs.exclude</value>
</property>
```

(4) Update these referenced files in the folders indicated with the network addresses of the DataNodes to be added in dfs.include and yarn.include files and the DataNodes to be removed in dfs.exclude and yarn.exclude files (Note that these file names can be changed to something different, if appropriate)

(5) Refresh HDFS and YARN

```
hdfs dfsadmin -refreshNodes

yarn rmadmin -refreshNodes
```

(6) Confirm DataNodes are live

```
hdfs dfsadmin -report
```

(7) Stop the datanode and nodemanager on the decommissioned node

```
$HADOOP_HOME/sbin/hadoop-daemon.sh stop datanode

$HADOOP_HOME/sbin/yarn-daemon.sh stop nodemanager
```

(8) Remove the decommissioned node from the slaves file

(9) Balance the data across the cluster, by executing the following command

```
hdfs balancer
```

6.6 Trash Functionality

Apache Hadoop provides a trash feature. The Trash feature is helpful for Hadoop Administrators in case of accidental deletion of files and directories.

By default the trash functionality is not enabled. Properties have to be configured to the core-site.xml file to enable trash functionality.

The trash folder is titled ".Trash" and located in the user's HDFS home directory with the

If trash is enabled and a file or directory is deleted, the file is moved to the .Trash directory in the user's home directory instead of being deleted.

Deleted files are stored on HDFS in the following location:

/user/<user name>/.Trash/Current/<deleted file/folder name>

This is enabled by adding the following property to core-site.xml

```
<property>
     <name>fs.trash.interval</name>
     <value>30</value>
</property>
```

where 30 is the time interval in minutes after which the files will be deleted from .Trash

To recover the file use the "hdfs dfs -mv" command to move it to the desired / previous location

The core-site.xml on the NameNode has to be update to support this feature. Copies on the DataNode are not required to enable this feature, however, useful for consistency.

6.7 Rack Awareness

For AWS since virtualized shared servers are used, this is not relevant. However, in environments where the datacenter configuration can be provided, that is which computers are in which racks, this service can be applied. To enable rack awareness execute the following steps.

(1) Create a topology file, this topology file needs to contain information about the data node to rack assignments - titled topo.data (text file)

(2) Create a shell script that reads the topology file and crates a map on the location of the data nodes in the racks, i.e., rack placement

(3) update the parameter in hdfs-site.xml file "net.topology.script.file.name" with the name and location of the shell script.

(4) Restart the NameNode

(5) Running command following commands will describe the awareness of racks:

```
hdfs fsck -racks
```

```
hadoop balancer
hdfs dfsadmin -report
```

6.8 High Availability

This reduces the risk that the NameNode is a single point of failure.

This is a complex configuration that involves multiple NameNodes and Zookeeper.

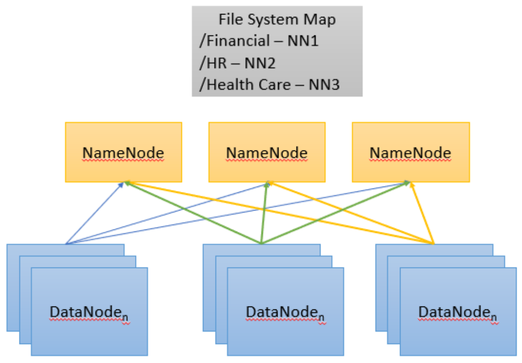
https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html#Deployment

6.9 Cluster Federation Architecture

This configuration, if implemented, allows for multiple name nodes to be deployed. The DataNodes send heartbeat and communicate to multiple data nodes. This is different than having multiple clusters where the NameNodes communicate with each other. In this configuration reduces the amount of meta data the NameNode has to store in its memory. Failure of a single NameNode allows other NameNodes to continue serve it's assigned DataNode storage information (metadata).

This provides the illusion of having separate clusters for each work groups.

Advantages are isolation, namespace scalability, performance, fault tolerance, and backup. This is more expensive in that the NameNodes are enterprise grade machines, so by increasing the number of NameNodes, (and standby NameNodes), the cost increases significantly.



7 Additional Hadoop Ecosystem Component Configurations

This section describes the configuration of components necessary for configuration of the cluster, in addition to the Hadoop-core components.

7.1 Apache Spark

This section describes the installation of Apache Spark which is used as the programming engine. Java, Python3, and Scala are prerequisites for Spark, as described in Section 2.

There are two variants of Spark binary distributions you can download.

(1) One is pre-built with a certain version of Apache Hadoop; this Spark distribution contains built-in Hadoop runtime, so we call it with-hadoop Spark distribution. For with-hadoop Spark distribution, since it contains a built-in Hadoop runtime already, by default, when a job is submitted to Hadoop Yarn cluster, to prevent jar conflict, it will not populate Yarn's classpath into Spark. To override this behavior, you can set spark.yarn.populateHadoopClasspath=true.

(2) The other one is pre-built with user-provided Hadoop; since this Spark distribution doesn't contain a built-in Hadoop runtime, it's smaller, but users have to provide a Hadoop installation separately. We call this variant no-hadoop Spark distribution. For no-hadoop Spark distribution, Spark will populate Yarn's classpath by default in order to get Hadoop runtime. Ensure that HADOOP_CONF_DIR or YARN_CONF_DIR points to the directory which contains the (client side) configuration files for the Hadoop cluster (check - these values/variables point to /home/hdfs/hadoop/etc/hadoop - which contains on the configuration files - e.g., hdfs-site.xml). These configs are used to write to HDFS and connect to the YARN ResourceManager. The configuration contained in this directory will be distributed to the YARN cluster so that all containers used by the application use the same configuration.

Spark jobs can run on YARN in two modes: cluster mode and client mode. Understanding the difference between the two modes is important for choosing an appropriate memory allocation configuration, and to submit jobs as expected.

A Spark job consists of two parts: Spark Executors that run the actual tasks, and a Spark Driver that schedules the Executors.

- Cluster mode: everything runs inside the cluster. You can start a job from your laptop and the job will continue running even if you close your computer. In this mode, the Spark Driver is encapsulated inside the YARN Application Master.
- Client mode the Spark driver runs on a client, such as your laptop. If the client is shut down, the job fails. Spark Executors still run on the cluster, and to schedule everything, a small YARN Application Master is created.

Client mode is well suited for interactive jobs, but applications will fail if the client stops. For long running jobs, cluster mode is more appropriate.

Plan is to install Spark in the Cluster mode

Apache Spark is installed on the same instance as the YARN ResourceManager, celeste.

It is not necessary to install Spark on all the nodes in the Hadoop/YARN cluster. Since Spark runs on top of YARN, it utilizes YARN for the execution of its commands over the cluster's nodes. So, you just have to install Spark on one EC2 instance.

Spark version 3.1.2 was installed using the hdfs account in the directory /apache/spark through the execution of the following steps.

7.1.1 Download Spark

```
cd /apache
wget https://dlcdn.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-without-hadoop.tgz
tar xzf spark-3.1.2-bin-without-hadoop.tgz
```

```
ln -s spark-3.1.2-bin-without-hadoop spark
```

7.1.2 Configure Spark Environment Variables

This configures the settings for spark in the file /etc/profile.d/spark.sh

```
su root
vi /etc/profile.d/spark.sh
```

put the following in this file:

```
export SPARK_HOME=/apache/spark
export PYSPARK_PYTHON=/usr/local/bin/python3.9
export HADOOP_CONF_DIR=/apache/hadoop/etc/hadoop
export YARN_CONF_DIR=/apache/hadoop/etc/hadoop
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

wq! - exit and save the file just created

```
chmod +x /etc/profile.d/spark.sh
source /etc/profile.d/spark.sh
```

7.1.3 (Optional) Updated the Spark log level in log4j

```
cp /apache/spark/conf/log4j.properties.template /apache/spark/conf/log4j.properties
vi /apache/spark/conf/log4j.properties
```

change the log level from Info to Error to reduce the number of messages displaced when programs are run

change the line "`log4j.rootCategory=INFO, console`" to "`log4j.rootCategory=ERROR, console`"

wq! - exit and save this file

7.1.4 Modify the Workers File

```
cp /apache/spark/conf/workers.template /apache/spark/conf/workers
vi /apache/spark/conf/workers
```

remove the listed node (localhost) and add the datanodes

```
pom.sehadoop.test
flora.sehadoop.test
alex.sehadoop.test
```

(this file (workers) does not need to be executable)

7.1.5 Update spark-env.sh

Configurations needed to /apache/spark/conf/spark-env.sh so that Spark is configured to use Hadoop YARN

```
cp /apache/spark/conf/spark-env.sh.template /apache/spark/conf/spark-env.sh
vi /apache/spark/conf/spark-env.sh
```

add the following to this file

```
SPARK_LOCAL_IP=192.168.15.120
SPARK_PUBLIC_DNS=celeste.sehadoop.test
SPARK_LOCAL_DIRS=/apache/spark/spark-temp

SPARK_CONF_DIR=/apache/spark/conf
SPARK_EXECUTOR_CORES=1
SPARK_EXECUTOR_MEMORY=2g
SPARK_DRIVER_MEMORY=2g

SPARK_DIST_CLASSPATH=$(/apache/hadoop/bin/hadoop classpath)

LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
```

wq! - exit and save this file, then make it executable

```
chmod +x /apache/spark/conf/spark-env.sh
```

7.1.6 Update spark-defaults.conf

```
cp /apache/spark/conf/spark-defaults.conf.tempate /apache/spark/conf/spark-
defaults.conf


vi /apache/spark/conf/spark-defaults.conf
```

add the following to the bottom of this file

```
spark.master yarn
spark.dynamicAllocation.enabled true
spark.dynamicAllocation.shuffleTracking.enabled true
spark.eventLog.enabled true
spark.eventLog.dir hdfs:///user/hdfs/spark-logs
spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.logDirectory hdfs:///user/hdfs/spark-logs
spark.history.fs.update.interval 10s
spark.yarn.historyServer.address = celeste.sehadoop.test:18080
spark.history.ui.port = 18080
```

7.1.7 Incorporated Configurations for Dynamic Resource Allocation

https://spark.apache.org/docs/latest/job-scheduling.html#resource-allocation-policy

https://gist.github.com/oza/e0df39adc3aac081c4bf

Made this configuration to support External Shuffle - Added the file /apache/spark/yarn/spark-3.1.2-yarn-shuffle.jar to the /apache/hadoop/share/hadoop/yarn/lib/ folder on pom, flora, alex, babar, basil

```
cp /apache/spark/yarn/spark-3.1.2-yarn-shuffle.jar
/apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar

scp /apache/spark/yarn/spark-3.1.2-yarn-shuffle.jar
babar:/apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar
```

on babar for each computer:

```
scp /apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar
pom:/apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar
```

Or as a short cut:

on babar (`ssh babar`) in the folder /apache/hadoop, created a text file with all the computers names, except babar and celelste titled computers-2

```
for i in $(cat computers-2); do scp
/apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar $i:
/apache/hadoop/share/hadoop/yarn/lib/spark-3.1.2-yarn-shuffle.jar; done
```

7.1.8 Update yarn-site.xml

On all the computers as hdfs, made changes to the /apache/hadoop/etc/hadoop/yarn-site.xml file

Add the following to yarn-site.xml

```
   <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>spark_shuffle</value>
   </property>
   <property>
      <name> yarn.nodemanager.aux-services.spark_shuffle.class</name>
      <value> org.apache.spark.network.yarn.YarnShuffleService</value>
   </property>
   <property>
      <name>yarn.log-aggregation-enable</name>
      <value>true</value>
   </property>
   <property>
       <name>yarn.nodemanager.log-aggregation.roll-monitoring-interval-
seconds</name>
       <value>3600</value>
   </property>
```

Copy this updated file, yarn-site.xml on celeste, to all the computers in the cluster.

```
scp /apache/hadoop/etc/hadoop/yarn-site.xml
babar:/apache/hadoop/etc/hadoop/yarn-site.xml

ssh babar
```

then on babar:

```
cd /apache/hadoop

for i in $(cat computers-2); do scp /apache/hadoop/etc/hadoop/yarn-
site.xml $i:/apache/hadoop/etc/hadoop/yarn-site.xml; done
```

7.1.9 Change YARN Heap Size

Change the YARN Heap Size in the /apache/hadoop/etc/hadoop/yarn-env.sh file

YARN_HEAPSIZE=2000

Then copy this file to all the computers in the cluster, first on celeste copy the file to babar and then on babary copy the file to the rest of the cluster

```
scp /apache/hadoop/etc/hadoop/yarn-env.sh
babar:/apache/hadoop/etc/hadoop/yarn-env.sh

ssh babar
```

on babar

```
cd /apache/hadoop

for i in $(cat computers-2); do scp /apache/hadoop/etc/hadoop/yarn-env.sh
$i:/apache/hadoop/etc/hadoop/yarn-env.sh; done
```

7.1.10 Verify the Spark Shell

Start the HDFS, YARN, and the Hadoop History Server

on babar as hdfs

```
start-dfs.sh

start-yarn.sh

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

Verify programs are running on babar in the folder /apache/hadoop, execute the following

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

(Note: To Stop HDFS, YARN and History Server, execute the following on babar as hdfs:)

```
stop-dfs.sh

stop-yarn.sh

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

Make the log folder on HDFS for spark

```
hdfs dfs -mkdir /user/hdfs/spark-logs
```

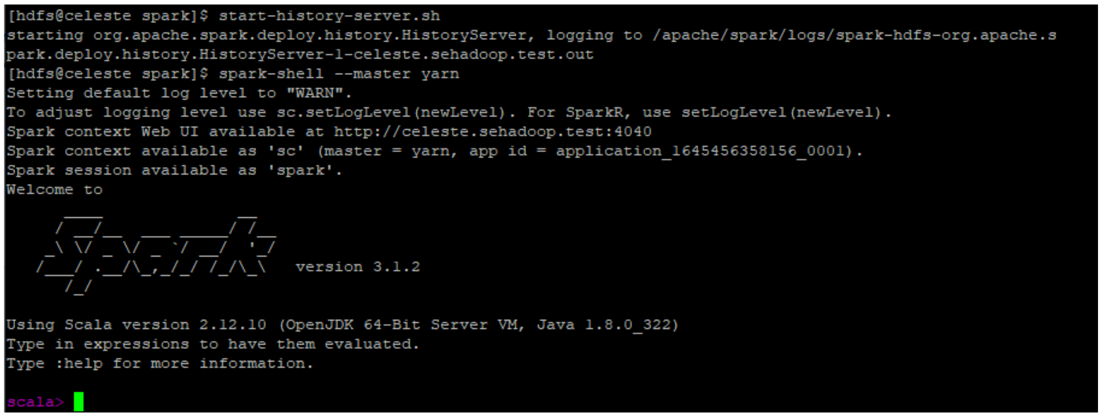
Start the Spark History Server on celeste as hdfs

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

Test spark-shell (Scala)

```
spark-shell --master yarn
```

exit the shell by typing `:quit` or ctrl-D



```
[hdfs@celeste spark]$ start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /apache/spark/logs/spark-hdfs-org.apache.s
park.deploy.history.HistoryServer-1-celeste.sehadoop.test.out
[hdfs@celeste spark]$ spark-shell --master yarn
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://celeste.sehadoop.test:4040
Spark context available as 'sc' (master = yarn, app id = application_1645456358156_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.2
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_322)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

7.1.11 Test Spark by Running an Example Program

```
./bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn
--deploy-mode cluster /apache/spark/examples/jars/spark-examples_2.12-
3.1.2.jar 10
```

This is the same as the following:

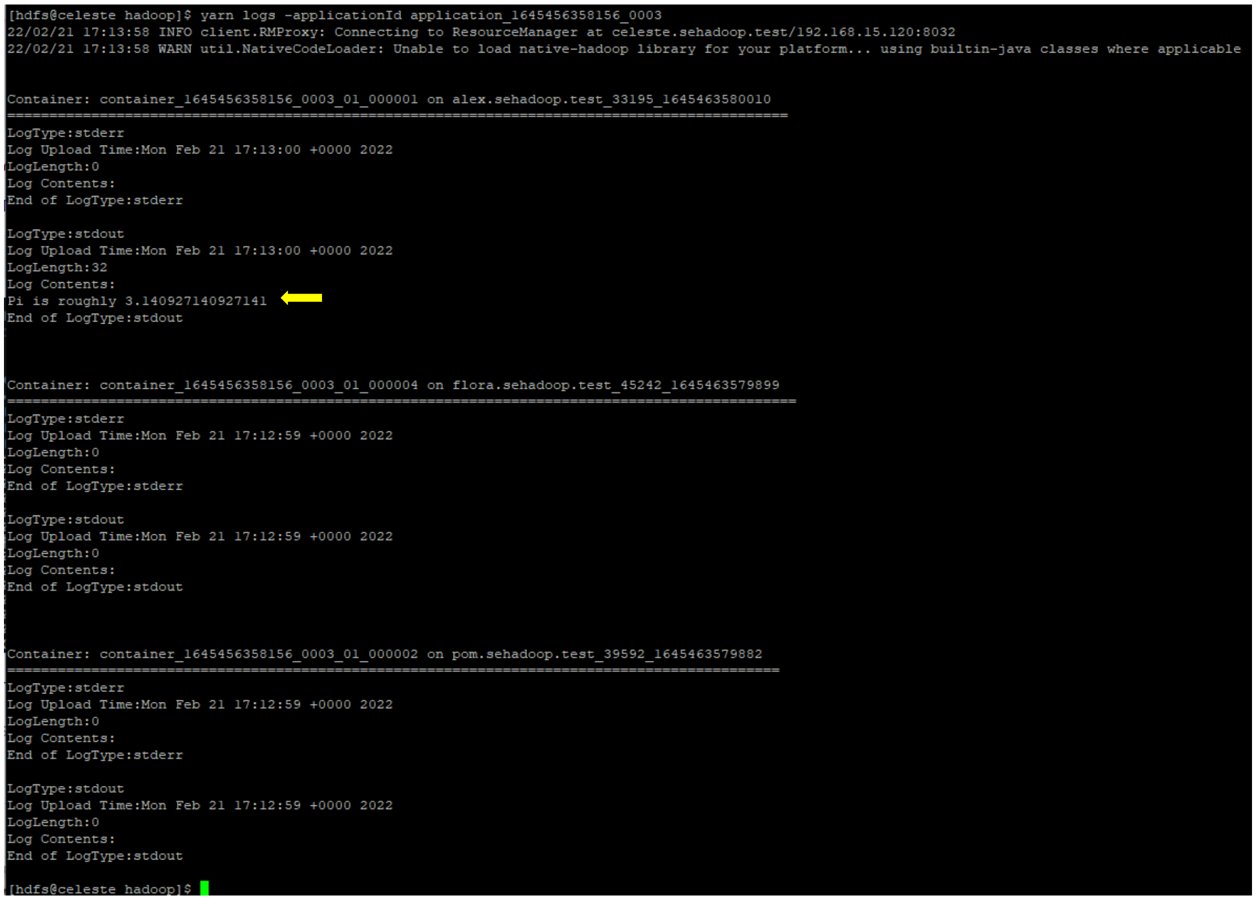
```
./bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master yarn \
 --deploy-mode cluster \
 --driver-memory 1g \
/apache/spark/examples/jars/spark-examples_2.12-3.1.2.jar \
 10
```

The results of running this program are contained in YARN log files which can be viewed with the following command – `yarn logs -applicationId` *`application ID`*

for example

```
yarn logs -applicationId application_1645456358156_0003
```

The result is shown in the middle of the output

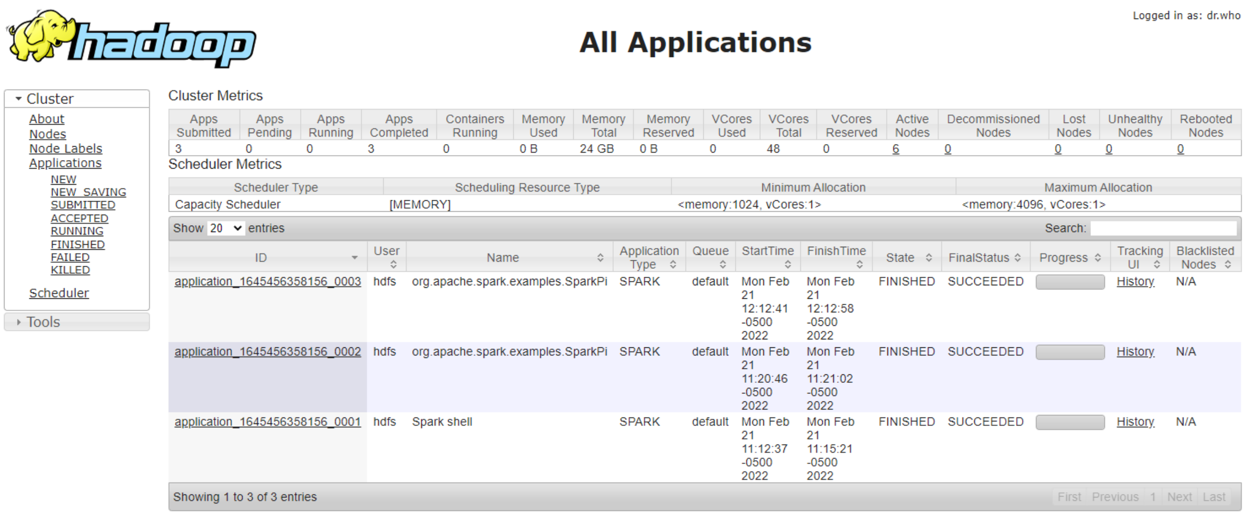


```
[hdfs@celeste hadoop]$ yarn logs -applicationId application_1645456358156_0003
22/02/21 17:13:58 INFO client.RMProxy: Connecting to ResourceManager at celeste.sehadoop.test/192.168.15.120:8032
22/02/21 17:13:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable


Container: container_1645456358156_0003_01_000001 on alex.sehadoop.test_33195_1645463580010
==============================================================================================
LogType:stderr
Log Upload Time:Mon Feb 21 17:13:00 +0000 2022
LogLength:0
Log Contents:
End of LogType:stderr

LogType:stdout
Log Upload Time:Mon Feb 21 17:13:00 +0000 2022
LogLength:32
Log Contents:
Pi is roughly 3.140927140927141
End of LogType:stdout



Container: container_1645456358156_0003_01_000004 on flora.sehadoop.test_45242_1645463579899
===============================================================================================
LogType:stderr
Log Upload Time:Mon Feb 21 17:12:59 +0000 2022
LogLength:0
Log Contents:
End of LogType:stderr

LogType:stdout
Log Upload Time:Mon Feb 21 17:12:59 +0000 2022
LogLength:0
Log Contents:
End of LogType:stdout



Container: container_1645456358156_0003_01_000002 on pom.sehadoop.test_39592_1645463579882
=============================================================================================
LogType:stderr
Log Upload Time:Mon Feb 21 17:12:59 +0000 2022
LogLength:0
Log Contents:
End of LogType:stderr

LogType:stdout
Log Upload Time:Mon Feb 21 17:12:59 +0000 2022
LogLength:0
Log Contents:
End of LogType:stdout

[hdfs@celeste hadoop]$
```

7.1.12 Check the ResourceManager Web UI

From my laptop browser, go to the cluster resource page webUI:

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



Logged in as: dr.who

All Applications

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

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
3	0	0	3	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:

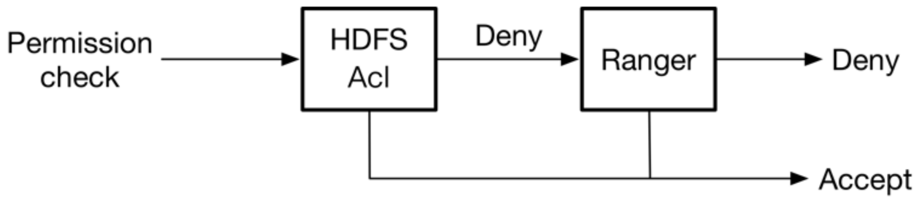
ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1645456358156_0003	hdfs	org.apache.spark.examples.SparkPi	SPARK	default	Mon Feb 21 12:12:41 -0500 2022	Mon Feb 21 12:12:58 -0500 2022	FINISHED	SUCCEEDED		History	N/A
application_1645456358156_0002	hdfs	org.apache.spark.examples.SparkPi	SPARK	default	Mon Feb 21 11:20:46 -0500 2022	Mon Feb 21 11:21:02 -0500 2022	FINISHED	SUCCEEDED		History	N/A
application_1645456358156_0001	hdfs	Spark shell	SPARK	default	Mon Feb 21 11:12:37 -0500 2022	Mon Feb 21 11:15:21 -0500 2022	FINISHED	SUCCEEDED		History	N/A

Showing 1 to 3 of 3 entries First Previous 1 Next Last

7.2 Apache Ranger

Apache Ranger provides security policy management across the cluster.

HDFS File permissions dominates Apache Ranger policies



If a user has HDFS permissions, then access is granted. If a user does not have HDFS permissions, then Ranger is checked, if Ranger has policies that permit access to the user, then access is granted.

Apache Ranger source code is available for download from this site: https://ranger.apache.org/

Version 2.1.0 was installed on pompadour.sehadoop.test as ranger

7.2.1 Create the ranger account and environment variables

```
su root
useradd -m ranger
passwd ranger
usermod -aG hadoop ranger
usermod -aG wheel ranger
exit
su ranger
```

Create environment variables, as root

```
su root
vi /etc/profile.d/ranger.sh
```

put the following in this file

```
export HADOOP_CONF=/apache/hadoop/etc/hadoop
export PATH=$PATH:/apache/ranger/ranger-admin:/apache/ranger/ranger-
tagsync:/apache/ranger/ranger-usersync
```

wq! - exit and save the file just created

```
chmod +x /etc/profile.d/ranger.sh
source /etc/profile.d/ranger.sh
exit
```

```
su ranger
```

7.2.2 Download Ranger

For this experimental configuration, I use a compiled version from the GitHub site identified below

The following steps were executed in the ranger account on pompadour:

```
chmod -R g+rwx /apache

cd /apache

mkdir ranger

cd ranger

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-admin.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-hdfs-plugin.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-atlas-plugin.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-kms.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-knox-plugin.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-ranger-tools.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-schema-registry-plugin.jar

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-solr-plugin.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-solr_audit_conf.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-src.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-tagsync.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-usersync.tar.gz

wget https://github.com/zer0beat/apache-ranger-
compiled/releases/download/2.1.0/ranger-2.1.0-yarn-plugin.tar.gz
```

Untar the main components of Ranger by executing the following command

```
tar zxvf ranger-2.1.0-admin.tar.gz

tar zxvf ranger-2.1.0-tagsync.tar.gz

tar zxvf ranger-2.1.0-usersync.tar.gz
```

Create a symbolic link

```
ln -s ranger-2.1.0-admin/ ranger-admin
ln -s ranger-2.1.0-tagsync/ ranger-tagsync
ln -s ranger-2.1.0-usersync/ ranger-usersync
```

7.2.3 Install MySQL

https://cwiki.apache.org/confluence/display/RANGER/Apache+Ranger+0.5.0+Installation#ApacheRanger0.5.0Installation-Overview

https://www.agiratech.com/apache-ranger-0-7-1-installation

Ranger works with this version of 5.6 or 5.7 of MySQL

```
wget https://dev.mysql.com/get/mysql80-community-release-el7-5.noarch.rpm
sudo rpm -Uvh mysql80-community-release-el7-5.noarch.rpm
yum repolist all |grep mysql
yum repolist enabled | grep mysql
sudo yum-config-manager --disable mysql80-community
sudo yum-config-manager --enable mysql57-community
sudo yum install mysql-community-server
service mysqld start
```

Get the temporary password

```
sudo grep 'temporary password' /var/log/mysqld.log
```

2022-02-26T21:07:42.065492Z 1 [Note] A temporary password is generated for root@localhost: Go>iKADsw3w*

Change the default settings

```
mysql_secure_installation
```

New root password = <xxx replace with database password xxx>

Remove anonymous users? y

Disallow root login remotely? y

Remove test database and access to it? No

```
mysql --version
```

mysql Ver 14.14 Distrib 5.7.37, for Linux (x86_64) using EditLine wrapper

Executed the following commands at the command line to set up the mysql database

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

entered the root password (listed above)

```
CREATE USER 'rangeradmin'@'localhost' IDENTIFIED BY '<xxx replace with database
password xxx>';

CREATE DATABASE ranger;

GRANT ALL PRIVILEGES ON *.* TO 'rangeradmin'@'localhost' IDENTIFIED BY '<xxx
replace with database password xxx> ';

FLUSH PRIVILEGES;

quit
```

Download the MySQL JDBC

```
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-8.0.28-
1.el7.noarch.rpm

sudo rpm -Uvh mysql-connector-java-8.0.28-1.el7.noarch.rpm

yum repolist all |grep mysql

ls /usr/share/java
```

The mysql-connector-java.jar file is in the /usr/share/java folder

7.2.4 Install and Configure Solr for Ranger Audits

https://cwiki.apache.org/confluence/display/RANGER/Install+and+Configure+Solr+for+Ranger+Audits+-+Apache+Ranger+0.5

The Ranger script setup.sh will automatically download, install and configure Solr if the following properties are set in install.properties in the ranger-admin folder (in addition to the items listed below)

```
SOLR_INSTALL true
SOLR_DOWNLOAD_URL
http://archive.apache.org/dist/lucene/solr/5.2.1/solr-5.2.1.tgz
SOLR_INSTALL_FOLDER /opt/solr
```

7.2.5 Configure Ranger Install.Properties Files

The install.properties for each Ranger component needs to be configured.

Before making these changes, made a backup copy of each of the install.properties files in each folder and then changed the permissions of the files

```
cp install.properties install.properties-DEFAULT

chmod u+w install.properties
```

7.2.5.1 ranger-admin install.properties

In the ranger-admin folder needs, the install.properties file needs to be update to include the following settings

```
db_root_user=root
db_root_password=<xxx replace with database password xxx>
#
# DB UserId used for the Ranger schema
#
db_name=ranger
db_user=rangeradmin
db_password=<xxx replace with database password xxx>

rangerAdmin_password=<xxx replace with password xxx>
rangerTagsync_password<xxx replace with password xxx>
rangerUsersync_password<xxx replace with password xxx>

#Source for Audit Store. Currently solr and elasticsearch are supported.
# * audit_store is solr
audit_store=solr
# Added SOLR values
SOLR_INSTALL true
SOLR_DOWNLOAD_URL http://archive.apache.org/dist/lucene/solr/5.2.1/solr-
5.2.1.tgz
SOLR_INSTALL_FOLDER /opt/solr
SOLR_HOST_URL=http://pompadour.sehadoop.test:8983
SOLR_RANGER_PORT=8983

# * audit_solr_url URL to Solr. E.g. http://<solr_host>:6083/solr/ranger_audits
audit_solr_urls=http://localhost:8983/solr/ranger_audits
audit_solr_user=solr
audit_solr_password=
audit_solr_zookeepers=


#
# ------- UNIX User CONFIG ----------------
#
unix_user=ranger
unix_user_pwd=<xxx replace with password xxx>
unix_group=hadoop
#
# ------- UNIX User CONFIG  - END ----------------
#
#
```

7.2.5.2 ranger-tagsync install.properties

Many of these properties depend upon the installation of Apache Atlas.

```
TAG_SOURCE_ATLAS_KAFKA_BOOSTRAP_SERVERS=troubadour.sehadoop.test:9092
TAG_SOURCE_ATLAS_KAFKA_ZOOKEEPER_CONNECT=troubadour.sehadoop.test:3000
```

```
TAG_SOURCE_ATLASREST_ENDPOINT=http://troubadour.sehadoop.test:21000
TAG_SOURCE_ATLASREST_USERNAME=admin
TAG_SOURCE_ATLASREST_PASSWORD=admin
TAG_SOURCE_FILE_FILENAME=/apache/ranger/data/tags.json
TAGSYNC_ATLAS_TO_RANGER_SERVICE_MAPPING=sehadoop,hdfs,sehadoop
```

#this setting is a mapping between the atlas cluster name, the service, and the service name in ranger, alternatively this could be `atlas_hive_cluster,hive,hivedev`

```
unix_group=hadoop
rangerTagsync_password=<xxx replace with password xxx>
hadoop_conf=/apache/hadoop/etc/hadoop
```

for our use case we are mapping between atlas and ranger for the HDFS files in the sehadoop cluster

7.2.5.3 ranger-usersync install.properties

We configured usersync for unix authentication, the following values were used in this file

```
ranger_base_dir = /apache/ranger
POLICY_MGR_URL=http://pomadour.sehadoop.test:6080
unix_group=hadoop
rangerUsersync_password=<xxx replace with password xxx>
```

This can also be configured to synchronize with the user permissions in the LDAP directory. This configuration was not confirmed during setup.

7.2.6 Execute Ranger Setup

7.2.6.1 ranger-admin setup

To execute ranger-admin setup, execute the following

```
cd /apache/ranger/ranger-admin
./setup.sh
```

The core-site.xml file in /apache/hadoop/etc/hadoop needs to be copied to /apache/ranger/conf/core-site.xml

```
cp /apache/hadoop/etc/hadoop/core-site.xml /apache/ranger/ranger-admin/conf/core-
site.xml
```

7.2.6.2 ranger-tagsync setup

To execute ranger-tagsync setup, execute the following

```
cd /apache/ranger/ranger-tagsync
./setup.sh
```

7.2.6.3 ranger-usersync setup

To execute ranger-usersync setup, executed the following

```
cd /apache/ranger/ranger-usersync
./setup.sh
```

7.2.7 Start Ranger

```
/apache/ranger/ranger-admin/ranger-admin start
/apahce/ranger/ranger-tagsync/ranger-tagsync-services.sh start
/apache/ranger/ranger-usersync start
```

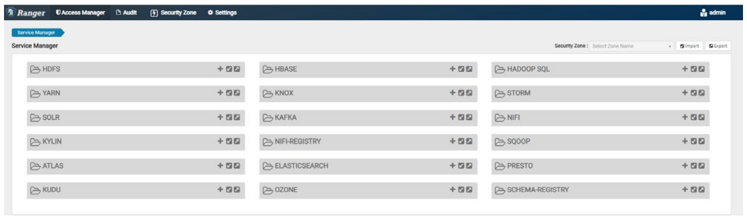
7.2.8 Login to the Ranger WebUI

login to the Ranger Admin Web based Graphical User Interface (WebUI), using a laptop browser, at

http://<ip address for pomapdour>:6080

username = admin

initial password = admin, changed the password to `<xxx replace with password xxx>`



7.2.9 Stop Ranger

```
/apache/ranger/ranger-admin/ranger-admin stop
/apache/ranger/ranger-tagsync/ranger-tagsync-services.sh stop
/apache/ranger/ranger-usersync/ranger-usersync stop
```

7.3 Apache Atas

Apache Atlas provide meta data management in the cluster.

In our sehadoop cluster we are installing Atlas on troubadour.sehadoop.test

7.3.1 Create the atlas account

```
su root
useradd -m atlas
passwd atlas
usermod -aG hadoop atlas
usermod -aG wheel atlas
chmod -R g+rwx /apache
exit
su atlas
```

7.3.2 Install maven

```
cd /apache

wget https://dlcdn.apache.org/maven/maven-3/3.8.4/binaries/apache-maven-
3.8.4-bin.tar.gz

tar xzvf apache-maven-3.8.4-bin.tar.gz
```

7.3.3 Download Atlas

The source files are available from atlas.apache.org. A built version is available through Udemy.

```
cd /apache

wget https://apache-ranger-udemy.s3.amazonaws.com/jars/atlas/apache-atlas-
2.1.0-server.tar.gz

tar xzvf apache-atlas-2.1.0-server.tar.gz
```

7.3.4 Configure and Start Atlas

The following local environment variables set in /etc/profile.d/atlas.sh:

```
export MANAGE_LOCAL_HBASE=true

export MANAGE_LOCAL_SOLR=true

export $PATH:/apache/atlas/bin
```

Created a sym link and started atlas

```
ln -s /apache/apache-atlas-2.1.0 /apache/atlas

sh /apache/atlas/conf/atlas-env.sh

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

7.3.5 Login to the Atlas WebUI

login to the Atlas Web based Graphical User Interface (WebUI), using a laptop browser, at

http://<ip address for troubadour>:21000

username = admin

initial password = admin, changed the password to `<xxx replace with password xxx>`

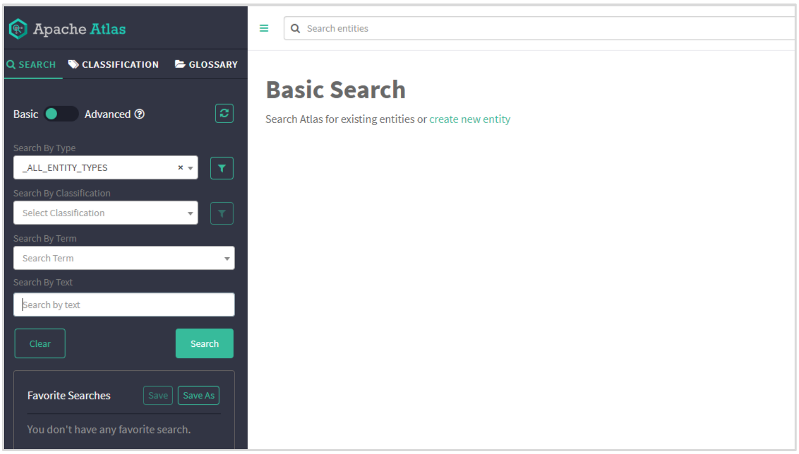
Note: Apache Atlas takes about 5 minutes to start up, review the logs to verify there are no issues. Zookeeper may take awhile to start.

The logs for Apache Atlas are in /apache/atlas/logs/application.log

execute the following command to review the latest logs

```
tail -f /apache/atlas/logs/application.log
```

This is the default WebUI, after login



7.3.6 Stop Atlas

```
/apache/atlas/bin/atlas_stop.py
```

7.4 Directory Services - FreeIPA LDAP

FreeIPA was set up and configured on belle to manage accounts on all the computers in the cluster. There are several web sites and even YouTube videos that provide details on FreeIPA installation. A few are listed below. The subsections below summarize the action taken to set up an instance with FreeIPA to support experimenting with LDAP and Kerberos with this cluster.

https://www.freeipa.org/

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

7.4.1 Install FreeIPA Server

From root terminal, run:

```
#yum install freeipa-server
```

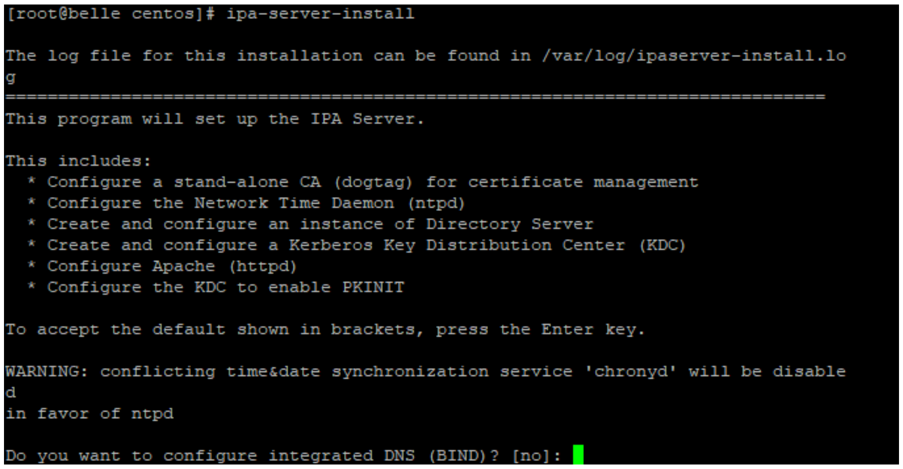
Note that the installed package just contains all the bits that FreeIPA uses, it does not configure the actual server.

7.4.2 Configure FreeIPA Server

The command can take command arguments or can be run in the interactive mode. You can get more details with `man ipa-server-install`. To start the interactive installation, run:

```
ipa-server-install
```

The command will at first gather all required information and then configure all required services.



```
[root@belle centos]# ipa-server-install

The log file for this installation can be found in /var/log/ipaserver-install.lo
g
==============================================================================
This program will set up the IPA Server.

This includes:
  * Configure a stand-alone CA (dogtag) for certificate management
  * Configure the Network Time Daemon (ntpd)
  * Create and configure an instance of Directory Server
  * Create and configure a Kerberos Key Distribution Center (KDC)
  * Configure Apache (httpd)
  * Configure the KDC to enable PKINIT

To accept the default shown in brackets, press the Enter key.

WARNING: conflicting time&date synchronization service 'chronyd' will be disable
d
in favor of ntpd

Do you want to configure integrated DNS (BIND)? [no]:
```

type in yes

7.4.3 Log Into the WebUI

To log into the FreeIPA WebUI, I had to add the external IP address and FQDN for the host to the hosts file on my Windows laptop.

This was done through the following steps:

open Sublime Text Editor as administrator (i.e., run as administrator) (right click on the icon and "run as administrator", enter the laptop admin password to open and save changes to this file)

add the AWS IP address and FQDN hostname to the `hosts` file which is located in the following folder on location on my Windows laptop PC

`c:\Windows\System32\drivers\etc\hosts`

save the hosts file

open a browser and go to the following URL http://belle.sehadoop.test/ipa/ui

login with user id `admin` and the password used before

(note that the credentials entered on the pop up window doesn't work, however entering the credentials on the page on the browser does)

7.4.4 Verify FreeIPA is Running

The following command shows the services running by FreeIPA

```
#ipactl status
```

7.4.5 Install the FreeIPA Clients

on all the other computers (clients) install the FreeIPA client software. From root terminal, run:

```
#yum install ipa-client*
```

Note that the installed package just contains all the bits that FreeIPA uses, it does not configure the actual server.

7.4.6 Configure the FreeIPA Clients

```
ipa-client-install --hostname=`hostname -f` --mkhomedir --
server=belle.sehadoop.test --domain sehadoop.test --realm SEHADOOP.TEST --
force-ntpd
```

type 'y' (yes) twice, when prompted to provide "`User authorized to enroll computers:`"
type 'admin' and admin's password when prompted

8 Footnotes and Tips

This section contains a few notes and tips associated with recovering from errors during the installation process.

8.1 Deleting EC2 Instances

Deleting an EC2 Instance is a two step process. First the instance has to be terminated and then the EBS volume (e.g., the virtual hard drive) has to be deleted. Since AWS charges based upon storage, this second step is important to reduce the amount of unneeded data stored on AWS.

(1) Click on he instance, then Select "Terminate Instance" from the AWS dropdown menu. For example, I deleted the following two instances:

i-0b8bdf3f4ae058f4f (henry)

i-0590677b4abc944e7 (erec)

(2) Delete the EBS volume associated with the instance. For example, these were the two volumes associate with the instances deleted in step 1.

https://us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#Volumes:sort=desc:createTime

vol-023482ee5422274b1

vol-0067dfce0c0eec4c9

8.2 SSH Operations

Recommend relooking the SSH configuration for an operational environment to further lockdown the passwordless-SSH connections between the systems. This initial configuration was based upon a best effort to get things up and running.

8.3 Operating System Scripts

A few Linux operating script that were written to expedite certain actions. For example,

Shutting down all the EC2 Instances

Made a list of the EC2 instances I'm shutting down in the file computers-nn

Create a script titled "down_all" with the following contents

```
for i in $(cat computers-nn); do ssh $i "shutdown now; echo -e '\n'"; done
```

Make the script executable

```
chmod +x down_all.sh
```

run the script down_all.sh as root on babar, this will require entering the root password for each computer when prompted since root ssh is not enabled

8.4 HDFS DFS Warning Messages

The following warning message is displayed when executing hdfs dfs commands and submitting jobs:

```
WARN util.NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
```

This message is displayed because, by default, the Hadoop libraries are pre-built for a 32 bit operating system. So, when you run any Hadoop command, including submitting jobs, this error is displayed.

A work-around is to add the following two lines to the end of the end of the `hadoop-env.sh` file to suppress the warning messages,

```
export HADOOP_HOME_WARN_SUPPRESS=1
export HADOOP_ROOT_LOGGER="WARN,DRFA"
```

The Hadoop 2.10 and prior binaries posted on the Apache Hadoop site are the 32-bit version. You have to build them to have a 64-bit version. The 32-bit binary version that can be downloaded directly, will work for experimentation but it will not use memory greater than 4GB

8.5 Group Settings /apache

Change the group ownership of /apache and all subfolders by executing the following command

```
su root
chgrp -R hadoop /apache
```