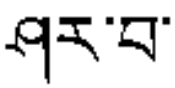
Open Source Whitepaper
November 5, 2012

SHERPASURFING – Cloudera Enterprise Real Time Query (RTQ) Impala Cyber Security Mission Platform



"No one remembers who climbed Mount Everest the second time."

sherpasurfing@gmail.com Twitter @WayneWheeles

About the Author

My name is Wayne Wheeles and I have spent over a decade in the Cyber Security Mission Space working for a variety of firms and customers. I work primarily in the area of network forensics, enrichment, frameworks, analytic development and mission support services. For detailed information on my career, I would refer the reader to my linkedin.com profile.

Note: I do not work for Cloudera; I am a partner and I have no financial interest at all with Cloudera. I consider my evaluation to be independent, honest and objective. This being said, I have nothing but positive to say about my partnership and work with Cloudera.

Recognition

I would be remiss if I did not take the moment to thank two of the best engineers in the world who worked with me on this effort. Hanh Le (SW Engineer, Analytics, BIGDATA) and Robert Webb (Infrastructure, Provisioning). Surrounding yourself with talent like them cannot result in anything but success.

Introduction

On October 24, 2012 Cloudera announced the release of an Impala Enterprise Real Time Query (RTQ) built on top of the Hadoop (HDFS/HBASE) stack. At the STRATA Conference, a series of demonstrations were provided for attendees comparing queries being executed on HIVE (as batch/map reduce) and Impala (RTQ) side by side. The demonstration was impressive and after some discussion, it was decided to port the objects of SherpaSurfing over to run on top of Impala. This document will walk the reader through the provisioning, installation, configuration and instantiation of the SherpaSurfing Cyber Security objects.

Two weeks from the first time that I watched the demonstration, I actually performed a demonstration of SherpaSurfing running on top of Impala at FedCyber 2012. This document will provide the backstory, details and lessons learned from that activity. This document is intended to work in tandem with the elements out on Github: configuration guide, object definitions, and some sample sanitized data sets.

Where to Start

Please note that I could have used the Impala VM that is provided but decided that I wanted to set up a simulated mission stack receiving: netflow (silk), intrusion detection system (snort), backlists, whitelist and geo location enrichment. In considering the types of use cases to be executed, four nodes seemed to be the ideal cluster for the demonstration.

Provisioning

The table provided below contains the basic information on the provisioning of the Virtual Machines (VMs) that were used in the demonstration. The VMs were delivered bare metal and the ISOs were downloaded for the installation of 64bit Cent OS 6.2. An appendix is provided at the end of this document that contains each package installed on each machine.

Servers/Provisioning			
Server Name	**Processors**	**Memory/Disk**	**Operating System**
Sherpa1	4 x Processor	16GB/70GB	64bit Cent OS 6.2
Sherpa2	4 x Processor	16GB/70GB	64bit Cent OS 6.2
Sherpa3	4 x Processor	16GB/70GB	64bit Cent OS 6.2
Sherpa4	4 x Processor	8GB/30GB	64bit Cent OS 6.2

Allocation

The allocation of the servers for the demonstration was intended to be very simple: One Cloudera Manager/App Tier machine and three Data Nodes. The initial planned allocation is provided below:

Allocation	
Server Name	**Allocation**
Sherpa1	Data Node, Impala, Hive, Oozie, Hbase,
Sherpa2	Data Node, Impala, Hive, Oozie, Hbase
Sherpa3	Data Node, Impala, Hive, Oozie, Zookeeper, Hbase, Hue
Sherpa4	Cloudera Manager, Tomcat

The allocation is important to ensure that the correct components are installed on each machine. Equally important, some machines will require a desktop environment; in this scenario: Sherpa3 and Sherpa4 both have desktop environments installed.

Initial Step

Following the download of the 64 bit CentOS 6.2 ISO, each node was booted with the ISO and the installation of the OS was completed in roughly two hours.

Sherpa3 and Sherpa4 Desktop Installation

When installing and intending to use Cloudera Manager and HUE, a desktop will be required. The step that was used to complete this task on sherpa3 and sherpa4 was:

1.) Open a terminal window as root
2.) yum install gnome-desktop
3.) vi /etc/inittab (see below)
4.) change the bootlevel from 3 to 5 in /etc/inittab

More /etc/inittab

```
# inittab is only used by upstart for the default runlevel.
#
# ADDING OTHER CONFIGURATION HERE WILL HAVE NO EFFECT ON YOUR SYSTEM.
#
# System initialization is started by /etc/init/rcS.conf
#
# Individual runlevels are started by /etc/init/rc.conf
#
# Ctrl-Alt-Delete is handled by /etc/init/control-alt-delete.conf
#
# Terminal gettys are handled by /etc/init/tty.conf and /etc/init/serial.conf,
# with configuration in /etc/sysconfig/init.
#
# For information on how to write upstart event handlers, or how
# upstart works, see init(5), init(8), and initctl(8).
#
# Default runlevel. The runlevels used are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
```

5.) Save the file and reboot the machine

Networking

In order to deploy the cluster and maximize the power/utility of Cloudera Manager some recommended networking configurations are recommended. **These recommended changes should be made on each machine in the cluster; it will simplify your experience.**

Configure /etc/hosts

1.) Open a console window as root
2.) vi /etc/hosts
3.) Add in each host in the cluster to the /etc/hosts file (example below from sherpa3)

Reference below # more /etc/hosts

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
[ipaddress]     sherpa3
[ipaddress]     sherpa1
[ipaddress]     sherpa2
[ipaddress]     sherpa4
```

4.) Save file

Firewall Configuration

This configuration is recommended for each machine in the cluster

1.) Open console window on as root
2.) Execute the commands in the console window:

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

Disable SELinux

This configuration is recommended for each machine in the cluster

1.) Open console window as root
2.) vi /etc/selinux/config (example below)

[root@sherpa3 etc]# more /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted

3.) update to SELINUX=disabled
4.) save file

Setting up Cloudera Manager 4.1

Having worked with Hadoop since 1.0, one of the items that really have been improved is deploying and configuring the cluster. Cloudera Manager is an incredible step in the right direction, on the demonstration mission platform; the entire cluster was deployed in less than four hours. Download the Cloudera Manager (4.x):

https://ccp.cloudera.com/display/SUPPORT/Cloudera+Manager+Downloads

Move the (cloudera-manager-installer.bin) file onto the machine in the cluster which will act as a provisioning server. The Cloudera Manager is installed on one machine in the cluster. In this case, sherpa4 was identified as the application server and Cloudera Manager/Provisioner.

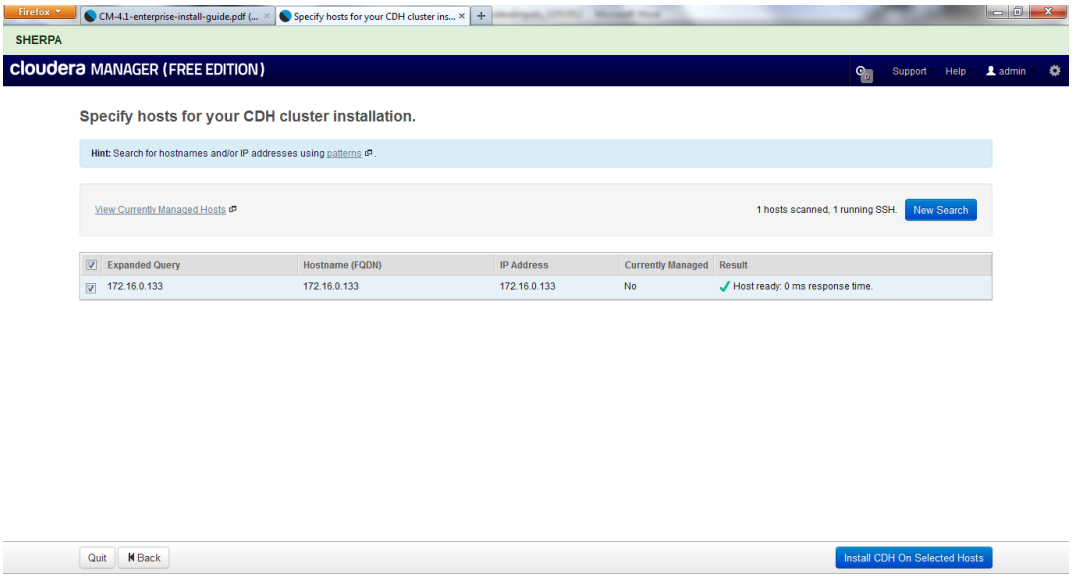
For the installation of Cloudera Manager; the embedded PostGreSQL database was used. In the documentation this is considered to be Path A; just wanted to keep the excitement to a minimum. So here are the steps:

1. Move the Cloudera-manager-installer.bin to the machine in the cluster where it will be resident.
2. Change permissions on the file for make it executable: chmod u+x cloudera-manager-installer.bin
3. From a terminal window as user root execute the command ./cloudera-manager-installer.bin
4. There will be a series of prompts for licenses and agreements to accept.
5. When completed the Cloudera Manager by default will be running on port 7180
6. In order to access the console, launch browser http://sherpa4:7180

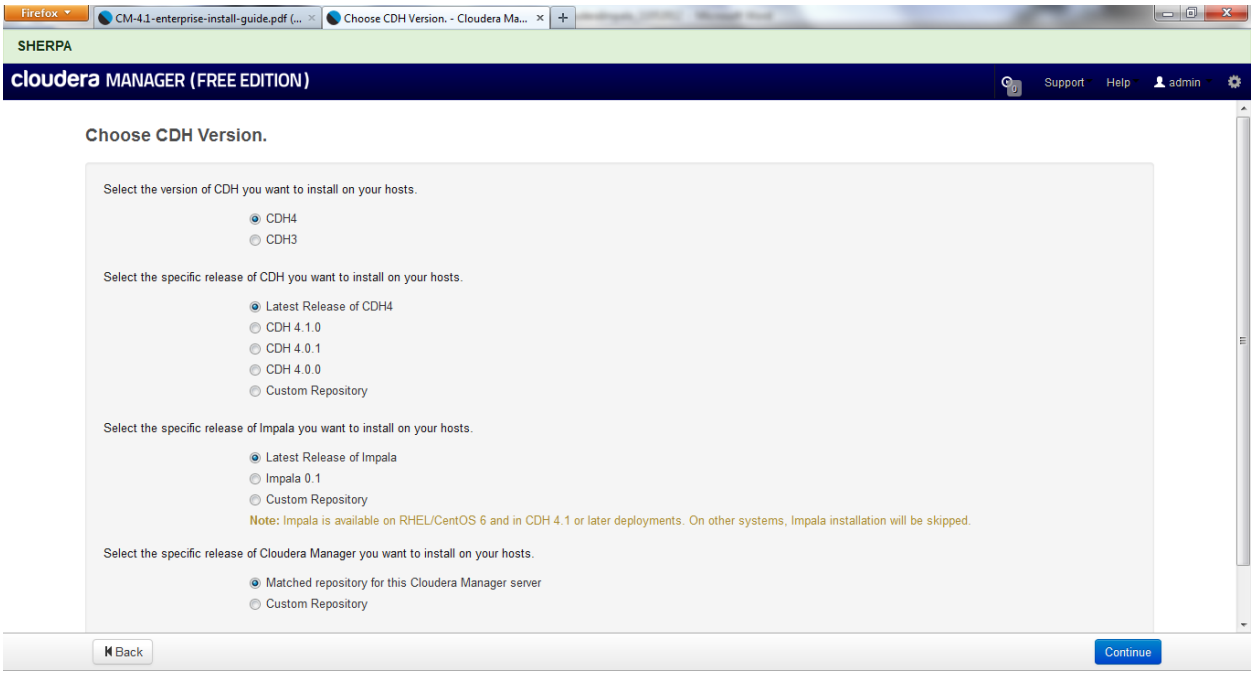
At this point the Cloudera Manager is installed and everything is in place to deploy the entire cluster.

Deploying the cluster using Cloudera Manager

1. Log into the Cloudera Manager using the default (admin, admin) credentials which will need to be changed.
2. Select “Host” from the toolbar
3. Select “Add Hosts”
4. Enter the names or addresses of the machines (sherpa1, sherpa2, sherpa3, sherpa4)
5. Select “search” when entering hosts is completed
6. The servers that were entered now will appear and need to be selected

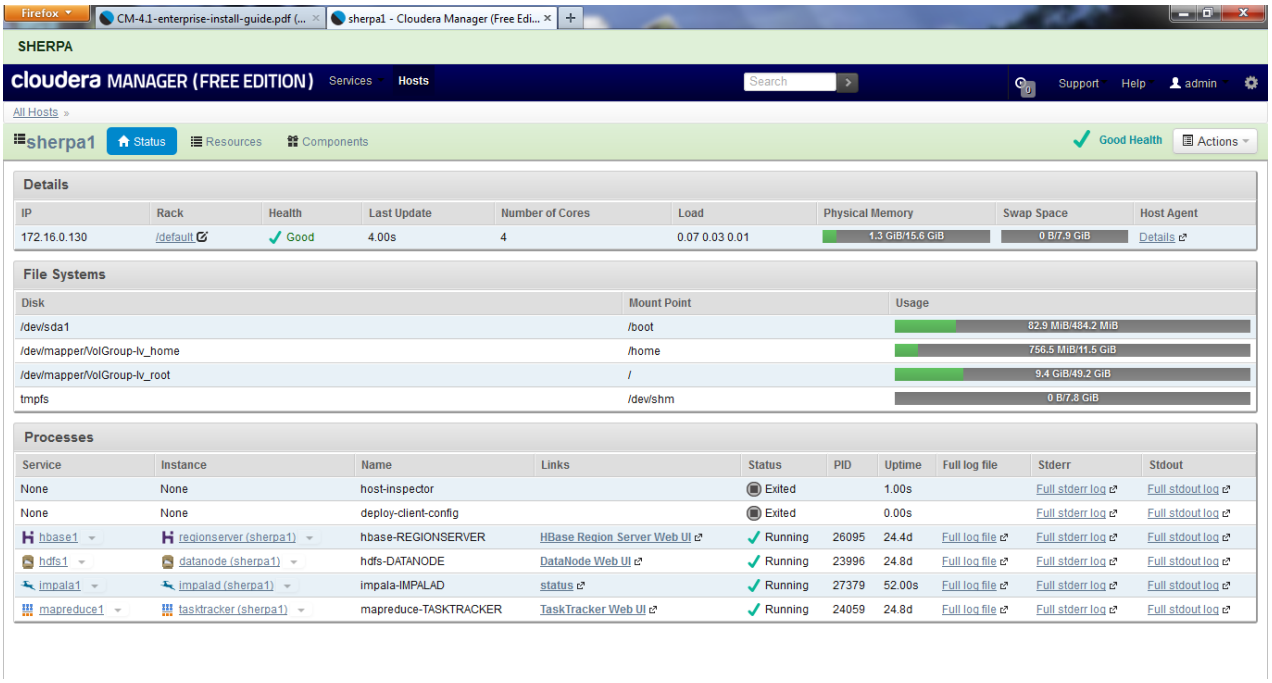


7. Down on the bottom of the page on the right hand side push the button labeled “Install CDH on Selected Hosts”.
8. When the “Install CDH on Selected Hosts” a menu will be provided such as below



9. For the machines in the Cluster, the default installation pattern was selected by pushing the button labeled “continue”.

Screen shot from Sherpa 1 – showing host management, health and utilization at the top; the bottom shows the services deployed on the node.



At this point, the deployment of the cluster is completed.

Deployment and Configuration of the IMPALA and HIVE services

Prior to the deployment and configuration of Impala/Hive first a few assignments:

Sherpa3 – State Store, Impalad
Sherpa2 – Impalad
Sherpa1 - Impalad

Impala

1. Log into Cloudera Manager Console and click on services from the toolbar.
2. Click on “Add a Service”
3. The result will be a table of different services available to be added:

Select the type of service you want to add.

	Service Type	Description
○	HDFS	Apache Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks and distributes them on compute hosts throughout a cluster to enable reliable, extremely rapid computations.
○	MapReduce	Apache Hadoop MapReduce supports distributed computing on large data sets across your cluster (requires HDFS).
○	YARN	Apache Hadoop MapReduce 2.0 (MRv2), or YARN, is a data computation framework that supports MapReduce applications (requires HDFS). The current upstream MRv2 release is not yet considered stable and should not be considered production-ready at this time.
○	ZooKeeper	Apache ZooKeeper is a centralized service for maintaining and syncronizing configuration data.
○	HBase	Apache HBase provides random, real-time, read/write access to large data sets (requires HDFS and ZooKeeper).
○	Oozie	Oozie is a workflow coordination service to manage data processing jobs on your cluster.
○	Hue	Hue is a graphical user interface to work with Cloudera's Distribution Including Apache Hadoop (requires HDFS and MapReduce).
○	Flume	Flume collects and aggregates data from almost any source into a persistent store such as HDFS.
○	Impala	Impala provides a real-time SQL query interface for data stored in HDFS and HBase (requires Hive's metastore). The current Impala release is beta software and not recommended for use in production.

4. Select Impala from the list
5. Click the “Continue” button on the bottom of the page.
6. Select the nodes to deploy Impala on:
    a. Sherpa3 – state store, impalad
    b. Sherpa2 – Impalad
    c. Sherpa1 – Impalad
7. Click on finish to complete the installation of Impala.

Hive Configuration

Hive will require some limited reconfiguration in order to ensure that it is pointing to the state store used by Impala.

1. For the servers that Impala is being installed on perform the following steps as root
2. Open a console window
3. cd /etc/hive/conf.dist/
4. vi hive-site.xml
5. Change the javax.jdo.option.ConnectionURL to the following (please replace metastoreserver with the name of your server)

```
<property>

 <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://[metastoreserver]:3306/hivemetastoredb?createDatabaseIfNotExist=true</value>

 <description>JDBC connect string for a JDBC metastore</description>

</property>
```

1. Now from the console window su – impala
2. Hive [return]
3. Use Sherpa; [return]
4. Create table test(test int);
5. Quit; [return]
6. Impala-shell[return]
7. Connect sherpa3:21000 [return]
8. Use Sherpa; [return]
9. Refresh; [return]
10. Show tables; [return]
11. Verify that test table is present.

Creation of Hive Data Structures and loading Development Data

The DDL provided in the SherpaSurfing baseline was used to create the HIVE data structures.

Sherpa 3 - hive-site.xml

Following post out on the Impala Users Group, many people have asked questions about the configuration of the hive-site.xml so I made sure and included it in the document.

```
<?xml version="1.0"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<!-- Hive Configuration can either be stored in this file or in the hadoop configuration files  -->
<!-- that are implied by Hadoop setup variables.                                                -->
<!-- Aside from Hadoop setup variables - this file is provided as a convenience so that Hive    -->
```

<!-- users do not have to edit hadoop configuration files (that may be managed as a centralized -->

<!-- resource). -->

<!-- Hive Execution Parameters -->

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://sherpa3:3306/hivemetastoredb?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hive</value>

<description>username TOUSE against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>hive</value>

<description>password TOUSE against metastore database</description>

</property>

</configuration>

Sherpa1 Layout

OS: CentOS 6.3
Kernel: 2.6.32-279
Memory: 16Gb
Disk: 75.2Gb
Partition layout:
/dev/sda1 * 1 64 512000 83 Linux
/dev/sda2 64 9138 72887296 8e Linux LVM
Disk /dev/mapper/VolGroup-lv_root: 53.7 GB, 53687091200 bytes
Disk /dev/mapper/VolGroup-lv_swap: 8438 MB, 8438939648 bytes
Disk /dev/mapper/VolGroup-lv_home: 12.5 GB, 12507414528 bytes

Sherpa2 Layout

OS: CentOS 6.3
Kernel: 2.6.32-279
Memory: 16Gb

Disk /dev/sda: 75.2 GB, 75161927680 bytes

/dev/sda1 * 1 64 512000 83 Linux
/dev/sda2 64 9138 72887296 8e Linux LVM
Disk /dev/mapper/VolGroup-lv_root: 53.7 GB, 53687091200 bytes
Disk /dev/mapper/VolGroup-lv_swap: 8438 MB, 8438939648 bytes
Disk /dev/mapper/VolGroup-lv_home: 12.5 GB, 12507414528 bytes

Sherpa3 Layout

OS: CentOS 6.3
Kernel: 2.6.32-279
Memory: 16Gb

Disk /dev/sda: 75.2 GB, 75161927680 bytes

/dev/sda1 * 1 64 512000 83 Linux
/dev/sda2 64 9138 72887296 8e Linux LVM
Disk /dev/mapper/VolGroup-lv_root: 53.7 GB, 53687091200 bytes
Disk /dev/mapper/VolGroup-lv_swap: 8438 MB, 8438939648 bytes
Disk /dev/mapper/VolGroup-lv_home: 12.5 GB, 12507414528 bytes