Hortonworks Cybersecurity Package

Installation

(April 13, 2017)

docs.cloudera.com

Hortonworks Cybersecurity Package: Installation

Hortonworks Cybersecurity Package (HCP) is a modern data application based on Apache Metron, powered by Apache Hadoop, Apache Storm, and related technologies.

HCP provides a framework and tools to enable greater efficiency in Security Operation Centers (SOCs) along with better and faster threat detection in real-time at massive scale. It provides ingestion, parsing and normalization of fully enriched, contextualized data, threat intelligence feeds, triage and machine learning based detection. It also provides end user near real-time dashboards.

Based on a strong foundation in the Hortonworks Data Platform (HDP) and Hortonworks DataFlow (HDF) stacks, HCP provides an integrated advanced platform for security analytics.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Figures

List of Tables

1. Hortonworks Cybersecurity Package Information Roadmap

This roadmap provides links to the information resources that are available for Hortonworks Cybersecurity Package (HCP) powered by Apache Metron.

Table 1.1. HCP Information Roadmap

Information type	**Resources**
Overview	• Apache Metron Website (Source: Apache wiki)
Installing	• Ambari Install Guide (Source: Hortonworks) • Command Line Install Guide (Source: Hortonworks) • Ambari Upgrade Guide (Source: Hortonworks) • Command Line Upgrade Guide (Source: Hortonworks)
Administering	• Apache Metron Documentation (Source: Apache wiki)
Developing	• Community Resources (Source: Apache wiki)
Reference	• About Metron (Source: Apache wiki)
Resources for contributors	• How to Contribute (Source: Apache wiki)
Hortonworks Community Connection	• Hortonworks Community Connection for Metron (Source: Hortonworks)

2. Preparing to Install

This chapter describes how to prepare to install Hortonworks Cybersecurity Package (HCP) powered by Apache Metron. HCP is a cybersecurity application framework that provides the ability to parse diverse security data feeds, enrich, triage, and store the data at scale, and detect cybersecurity anomalies. For more information about the HCP architecture and capabilities, see HCP Architecture.

The chapter contains the following sections:

2.1. Minimum System Requirements

HCP has the following system requirements:

2.1.1. Operating System Requirements

HCP currently supports CentOS v7.x.

2.1.2. Browser Requirements

The Ambari Install Wizard runs as a browser-based Web application. You must have a machine capable of running a graphical browser to use this tool. The minimum required browser versions are:

- Windows (7, 8)
  - Internet Explorer 10
  - Firefox 18
  - Google Chrome 26
- Mac OS x (10.6 or later)
  - Firefox 18
  - Safari 5

  - Google Chrome 26
- Linux (CentOS)
  - Firefox 18
  - Google Chrome 26

On any platform, we recommend updating your browser to the latest, stable version.

2.1.3. Infrastructure Requirements

This section provides the indicative specifications for your physical nodes.

Table 2.1. Physical Nodes

Role	Indicative Specifications
PCAP Collector Card	Ethernet—Adapter—X520—DA2 or DPDK compatible card 20 GB/Sec
PCAP Collector Server	• CPUs: 2 x 8 Core Processors • Memory: 128 GB RAM • Disk Storage: 10 x 2 TB SATA Drives • Network: 2 x 10 GB NIC
NiFi Server	• CPUs: 2 x 8 Core Processors • Memory: 128 GB RAM • Disk Storage: 10 x 2 TB SATA Drives • Network: 2 x 10 GB NIC
Apache Kafka / Storm Server	• CPUs: 2 x 8 Core Processors • Memory: 128 GB RAM • Disk Storage: 10 x 2 TB SATA Drives • Network: 2 through 10 GB NIC
Metron Master Nodes	• CPUs: 2 x 8 Core Processors • Memory: 128 GB RAM • Disk Storage: 10 x 2 TB SATA Drives • Network: 2 x 10 GB NIC
HCP Worker Nodes— Balanced	• CPUs: 2 x 8 Core Processors • Memory: 128 GB RAM • Disk Storage: 10—2 TB SATA Drives • Network: 2—10 GB NIC

2.1.4. Software Requirements

The host that you choose to use to deploy Apache Metron must have the following software tools installed:

- Hadoop (HDP 2.5 recommended)
  - Apache Hadoop 2.7.3
  - Apache Storm 1.0.1
  - Apache Kafka 0.10.0.1
  - Apache HBase 1.1.2
  - Apache ZooKeeper 3.4.6
- **Note**

  Supervisor, Kafka Broker, and the HBase client must be installed on the Metron Install Host.
- Installable during the Ambari installation of HCP

  The following software is required for HCP, but this software can be installed manually or during the HCP Ambari installation. Hortonworks recommends that you wait to install this software until the Ambari installation of HCP.
  - Elasticsearch 2.3.3
  - Kibana 4.5.1

2.1.5. Memory Requirements

For memory requirements, see Memory Requirements in the [Apache Ambari Installation] guide.

2.1.6. Maximum Open File Descriptors

The recommended maximum number of open file descriptors is 50,000, or more. To check the current value set for the maximum number of open file descriptors, execute the following shell commands on each host:

```
ulimit -Sn

ulimit -Hn
```

If the output is not greater than 50,000, run the following command to set it to a suitable default:

```
ulimit -n 50000
```

3. Installing HCP

You can choose any of the following three methods to install Hortonworks Cybersecurity Package (HCP) powered by Apache Metron:

- Installing HCP on an Ambari-Managed Cluster Using Ambari [5]
- Manually Installing HCP [22]

3.1. Installing HCP on an Ambari-Managed Cluster Using Ambari

The following sections provide instructions on how to install HCP using Ambari on an Ambari-managed HDP 2.5 cluster that meets the prerequisites listed next.

Note

The instructions in this section apply to a cluster running CentOS. For information on installing on a cluster running Ubuntu, see Manually Installing Apache Metron on Ubuntu 14.04.

Installing HCP using Ambari onto an Ambari-managed cluster requires the following major steps:

- Installing HCP on an Ambari Cluster [7]
- Installing, Configuring, and Deploying a HDP Cluster with HCP [8]
- Importing Zeppelin Notebook Using Ambari [14]
- Streaming Data into HCP [14]
- Verifying That HCP Deployed Successfully [17]

3.1.1. Prerequisites for an Existing Cluster

You can install HCP on an Ambari-managed cluster running HDP 2.5 and Ambari 2.4.2. However, the cluster must meet the requirements listed in the following sections:

- Specifications for Hadoop Cluster [5]
- Specifications for Metron Nodes [6]

3.1.1.1. Specifications for Hadoop Cluster

All Hadoop-related nodes must meet the following specifications:

- All cluster nodes must be running CentOS 6.x, CentOS 7.x, or Ubuntu 14.04
- The cluster must be running HDP 2.5 managed by Ambari 2.4.2

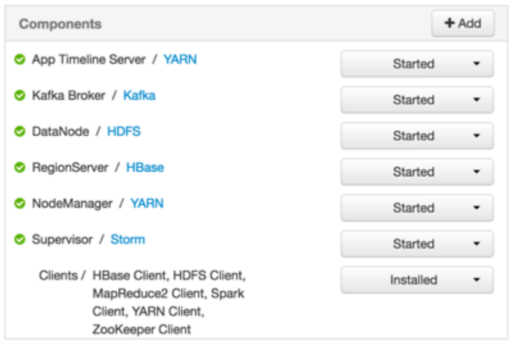
- The cluster must have a minimum of the following nodes:
  - Two Hadoop master nodes
  - Four Hadoop slaves nodes
  - One node for Ambari
- Each of the Hadoop Slave and Master nodes must meet the following minimum specifications: See Minimum System Requirements.
- The following services must be installed across the Hadoop Master and Slave nodes:
  - HDFS
  - HBase
  - ZooKeeper
  - Kafka
  - Storm
  - YARN

  To determine the supported version for each service, refer to Ambari, and choose **`Admin > Stacks and Versions`**.
- Each of the following components must be installed on at least four slave nodes:

Note

For security reasons, no other workloads should be running on the cluster.

Figure 3.1. Ambari Component



Components	
App Timeline Server / YARN	Started
Kafka Broker / Kafka	Started
DataNode / HDFS	Started
RegionServer / HBase	Started
NodeManager / YARN	Started
Supervisor / Storm	Started
Clients / HBase Client, HDFS Client, MapReduce2 Client, Spark Client, YARN Client, ZooKeeper Client	Installed

3.1.1.2. Specifications for Metron Nodes

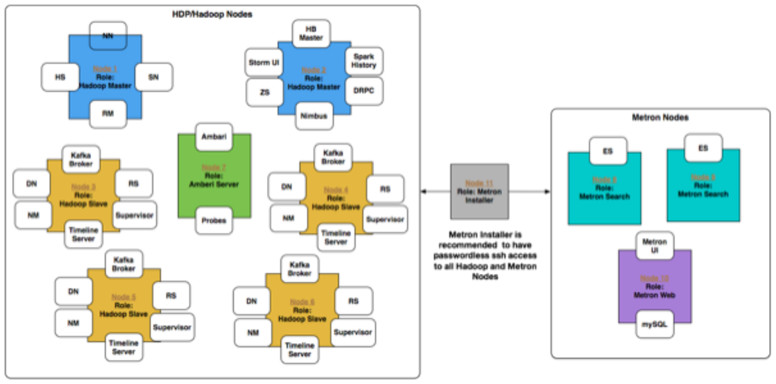
The following specifications must be meet for the Metron nodes:

- At least three nodes must be dedicated for Metron—specific components.
- You must have root access on all Metron nodes.

- The Metron Installer node must have Docker installed.

  The following figure illustrates a sample deployment architecture based on the previous specifications:

  Figure 3.2. Sample Deployment Architecture

3.1.2. Installing HCP on an Ambari Cluster

Prior to installing the HCP Ambari management pack, you must complete the following:

- Meet all of the cluster specifications listed in Specifications for Hadoop Cluster.
- Meet all of the metron node specifications listed in Specifications for Metron Nodes.
- Download and install Ambari. See Installing Ambari CentOS 7.
- Set up the Ambari server. See Set up the Ambari Server.
- Downloading the HCP Management Module [18]

Installing the HCP management pack on an Ambari cluster requires the following major steps:

- Installing HCP Ambari Management Pack [7]
- Starting the Ambari Server [8]

3.1.2.1. Installing HCP Ambari Management Pack

Prerequisites

An HCP Ambari management pack bundles service definitions, stack definitions, and stack add-on service definitions so they do not need to be included with the Ambari core functionality and can be updated in between major releases. The HCP management pack includes Metron, plus the parser topologies, indexing topologies, and enrichment topologies.

1. Download the HCP management pack tar file from the HCP repo location:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HCP/centos7/1.x/updates/1.1.0.
   0/tars/metron/hcp-ambari-mpack-1.1.0.0-71.tar.gz
   ```

2. Install the HCP management pack:

   ```
   ambari-server install-mpack --mpack=/${MPACK_DOWNLOAD_DIRECTORY}/hcp-ambari-
   mpack-1.1.0.0-71.tar.gz --verbose
   ```

   You should see a message saying that the management pack completed successfully.

3.1.2.2. Starting the Ambari Server

After you install the HCP Ambari management pack, you need to start or restart the Ambari server, depending on whether your are installing HCP on a new or existing cluster.

To start the Ambari server, enter the following:

```
ambari-server start
```

To restart the Ambari server, enter the following:

```
ambari-server restart
```

3.1.3. Installing, Configuring, and Deploying a HDP Cluster with HCP

Use the Ambari Install wizard running in your browser to install, configure, and deploy your cluster.

1. Open Ambari Web using a web browser.
   a. Point your browser to `http://<your.ambari.server>:8080`, where `<your.ambari.server>` is the name of your ambari server host. For example, a default Ambari server host is located at `http://c6401.ambari.apache.org:8080`.
   b. Log in to the Ambari Server using the default user name/password: **`admin/admin`**. You can change these credentials later.

      For a new cluster, the Ambari install wizard displays a Welcome page from which you launch the Ambari Install wizard.
2. For an existing cluster, choose **Choose Services** from the **Actions/Add Service Wizard** menu and skip to Step 7.
3. From the Ambari Welcome page, choose **Launch Install Wizard**.
4. In **Name your cluster**, type a name for the cluster you want to create, and then choose **Next**.

   Use no white spaces or special characters in the name.
5. Select the HDP stack you want to run.

   HCP supports HDP 2.5.
6. Enter the set up information for which the install wizard prompts you.

You need to supply the FQDN of each of your hosts. The wizard also needs to access the private key file you created in Set Up Password-less SSH. Using the host names and key file information, the wizard can locate, access, and interact securely with all hosts in the cluster.

a. Use the **Target Hosts** text box to enter your list of host names, one per line.

   You can use ranges inside brackets to indicate larger sets of hosts. For example, for host01.domain through host10.domain use host[01-10].domain

   Note

   If you are deploying on EC2, use the internal Private DNS host names.

b. If you want to let Ambari automatically install the Ambari Agent on all your hosts using SSH, select **Provide your SSH Private Key** and either use the **Choose File** button in the **Host Registration Information** section to find the private key file that matches the public key you installed earlier on all your hosts or cut and paste the key into the text box manually.

   Note

   If you are using IE 9, the `Choose File` button may not appear. Use the text box to cut and paste your private key manually.

   Fill in the user name for the SSH key you have selected. If you do not want to use root , you must provide the user name for an account that can execute sudo without entering a password.

c. Choose **Register** and **Confirm** to continue.

   Ambari displays the **Choose Services** dialog box that lists the hosts that Ambari has located for your cluster.

7. Check the hosts listed in the **Choose Services** dialog box to make sure they are the correct hosts and that they have the correct directories, packages, and processes required to continue the install.

   If any hosts were selected in error, you can remove them by selecting the appropriate check boxes and clicking the grey **Remove Selected** button. To remove a single host, click the small white **Remove** button in the **Action** column.

   At the bottom of the screen, you might notice a yellow box that indicates some warnings were encountered during the check process. For example, your host might already have a copy of `wget` or `curl`. Choose **Click here** to see a list of what was checked and what caused the warning. The warnings page also provides access to a python script that can help you clear any issues you may encounter and let you run **Rerun Checks**.

8. Choose the services to install into the cluster, and then click **Next**.

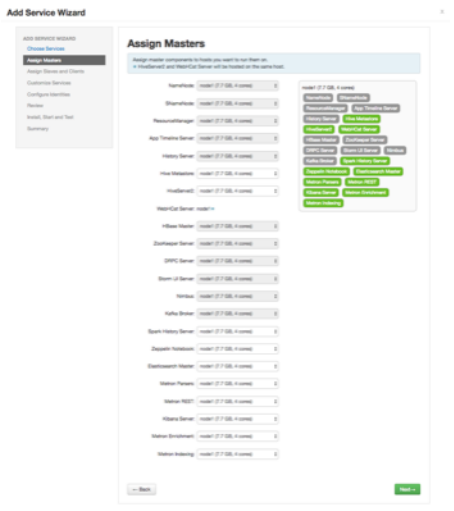
   HCP requires the following services:

- HDFS
- HBase
- ZooKeeper
- Storm
- Kafka
- Flume
- Ambari Metric Service
- Metron
- Elasticsearch (Can be installed either manually or by Ambari. Hortonworks recommends installing Elasticsearch by Ambari.)
- Kibana (Can be installed either manually or by Ambari. Hortonworks recommends installing Kibana by Ambari.
- Zeppelin Notebook
- Spark
- Hive

Ambari displays the **Assign Masters** window.

9. Verify that the Ambari install wizard has assigned the master components for selected services to appropriate hosts in your cluster.

Figure 3.3. Assign Masters Window



Assign Masters

Assign master components to hosts you want to run them on.
* HiveServer2 and WebHCat Server will be hosted on the same host.

NameNode:	node1 (7.7 GB, 4 cores)
SNameNode:	node1 (7.7 GB, 4 cores)
ResourceManager:	node1 (7.7 GB, 4 cores)
App Timeline Server:	node1 (7.7 GB, 4 cores)
History Server:	node1 (7.7 GB, 4 cores)
Hive Metastore:	node1 (7.7 GB, 4 cores)
HiveServer2:	node1 (7.7 GB, 4 cores)
WebHCat Server:	node1
HBase Master:	node1 (7.7 GB, 4 cores)
ZooKeeper Server:	node1 (7.7 GB, 4 cores)
DRPC Server:	node1 (7.7 GB, 4 cores)
Storm UI Server:	node1 (7.7 GB, 4 cores)
Nimbus:	node1 (7.7 GB, 4 cores)
Kafka Broker:	node1 (7.7 GB, 4 cores)
Spark History Server:	node1 (7.7 GB, 4 cores)
Zeppelin Notebook:	node1 (7.7 GB, 4 cores)
Elasticsearch Master:	node1 (7.7 GB, 4 cores)
Metron Parsers:	node1 (7.7 GB, 4 cores)
Metron REST:	node1 (7.7 GB, 4 cores)
Kibana Server:	node1 (7.7 GB, 4 cores)
Metron Enrichment:	node1 (7.7 GB, 4 cores)
Metron Indexing:	node1 (7.7 GB, 4 cores)

node1 (7.7 GB, 4 cores)

If Ambari detects any errors in your master component assignments, it will indicate the error in red.

a. To change the host assignment for a service, select a host name from the drop-down menu for that service.

b. To remove a ZooKeeper instance, click the green minus icon next to the host address you want to remove.

c. When you are satisfied with the assignments, click **Next**.

10. Verify that the Ambari install wizard has assigned the slave components (DataNodes, NodeManagers, and RegionServers) to appropriate hosts in your cluster.

a. Use all or none to select all of the hosts in the column or none of the hosts, respectively.

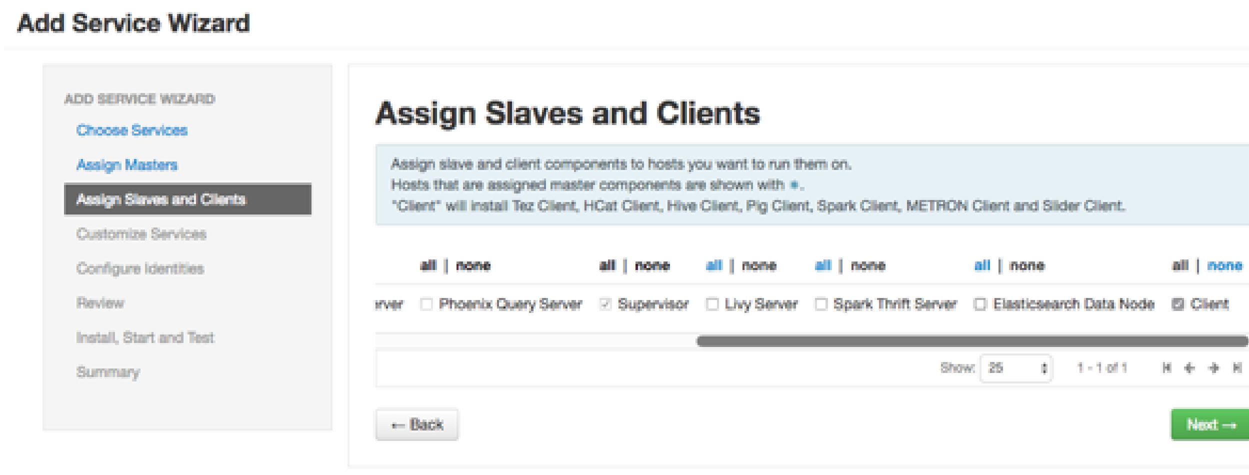
   If a host has an asterisk next to it, that host is also running one or more master components. Hover your mouse over the asterisk to see which master components are on that host.

b. Select a minimum of one Elasticsearch data node. The data node **cannot** be on same host as the master.

c. Fine-tune your selections by using the check boxes next to specific hosts.

d. Check the **Client** checkbox for any components that have the **Supervisor** checkbox checked.

Figure 3.4. ambari_assign_slaves_clients.png


Add Service Wizard

Assign Slaves and Clients

Assign slave and client components to hosts you want to run them on.
Hosts that are assigned master components are shown with ✱.
"Client" will install Tez Client, HCat Client, Hive Client, Pig Client, Spark Client, METRON Client and Slider Client.

e. When you are satisfied with your assignments, click **Next**.

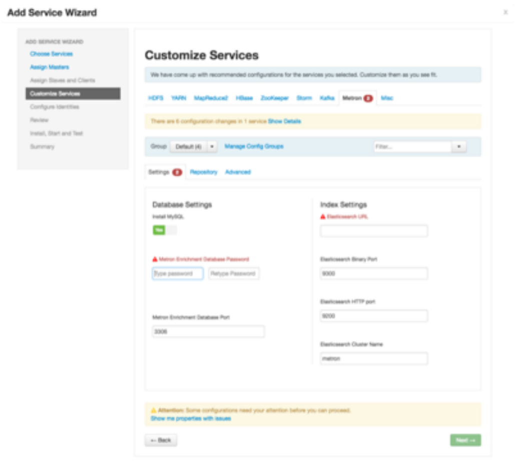
11. Review each service tab in the **Customize Services** dialog box and modify your HDP cluster setup if appropriate.

a. Browse through each service tab. By hovering your cursor over each of the properties, you can see a brief description of what the property does.

   The number of service tabs shown depends on the services you decided to install in your cluster. Any tab that requires input displays a red badge with the number of properties that need attention. Select each service tab that displays a red badge number and enter the appropriate information.

Directories	The choice of directories where HDP will store information is critical. Ambari will attempt to choose reasonable defaults based on the mount points available in your environment but you are strongly encouraged to review the default directory settings recommended by Ambari. In particular, confirm directories such as /tmp and /var are not being used for HDFS NameNode directories and DataNode directories under the HDFS tab.
Elasticsearch zen_discovery_ping_unicast_hosts	A comma separated list of Elasticsearch data nodes that you identified in Step 10.
kibana_es_url	Set to the fully-qualified url for the Elasticsearch master: http://es-master-host:9200.
Metron	The Metron tab contains a few tabs that contain information that is critical to HCP set up.

Figure 3.5. Metron Settings Tab



Customize Services

We have come up with recommended configurations for the services you selected. Customize them as you see fit.

There are 6 configuration changes in 1 service Show Details

Group Default (4) Manage Config Groups Filter...

Database Settings

Install MySQL: Yes

Metron Enrichment Database Password: Type password / Retype Password

Metron Enrichment Database Port: 3306

Index Settings

Elasticsearch URL:

Elasticsearch Binary Port: 9300

Elasticsearch HTTP port: 9200

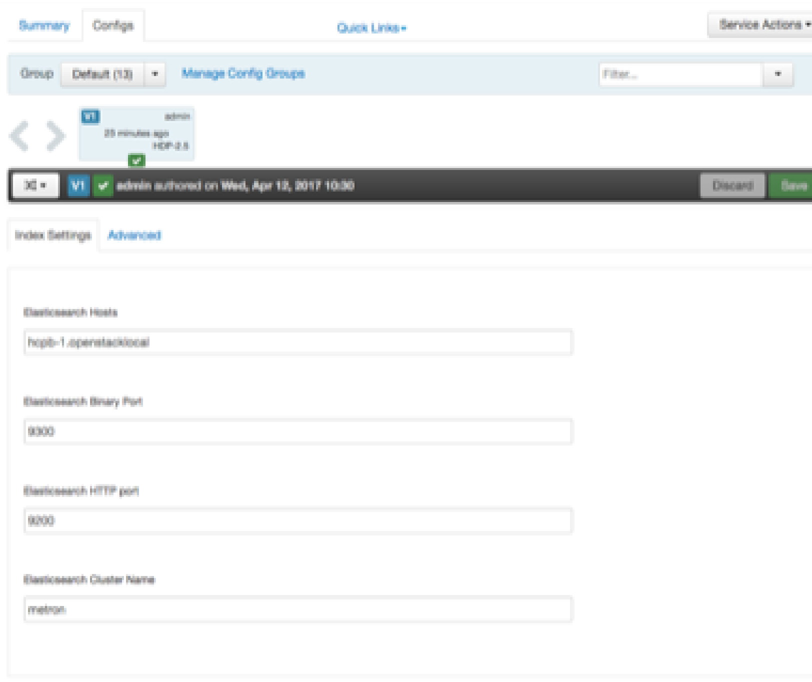
Elasticsearch Cluster Name: metron

Attention: Some configurations need your attention before you can proceed.
Show me properties with issues

Customize Services	This tab allows you to choose your repository location.

> **Important**
>
> This tab is pre-populated and you should not modify its contents unless you are going to use a local repository.

Figure 3.6. Customize Services.png



Elasticsearch Hosts: hcpb-1.openstacklocal

Elasticsearch Binary Port: 9300

Elasticsearch HTTP port: 9200

Elasticsearch Cluster Name: metron

Advanced Tab (Metron)	Most of the fields in the Advanced tab are auto populated and should not be modified. You can modify the following field with caution:
Service Account Users and Groups	The service account users and groups are available under the **Misc** tab. These are the operating system accounts the service components will run as. If these users do not exist on your hosts, Ambari will automatically create the users and groups locally on the hosts. If these users already exist, Ambari will use those accounts. Depending on how your environment is configured, you might not allow groupmod or usermod operations. If this is the case, you must be sure all users and groups are already created and be sure to select the **Skip group modifications** option on the **Misc** tab. This tells Ambari to not modify group membership for the service users.

12. Check the assignments displayed by Ambari to ensure that everything is correct, and then click **Deploy**.

    If you need to make changes, use the left navigation bar to return to the appropriate screen.

    The progress of the install displays on the screen. Ambari installs, starts, and runs a simple test on each component. Overall status of the process displays in a progress bar at the top of the screen and host-by-host status displays in the main section. Do not refresh your browser during this process. Refreshing the browser might interrupt the progress indicators.

    To see specific information on what tasks have been completed per host, click the link in the **Message** column for the appropriate host. In the **Tasks** pop-up, click the individual task to see the related log files. You can select filter conditions by using the **Show** drop-down list. To see a larger version of the log contents, click the **Open** icon or, to copy the contents to the clipboard, use the **Copy** icon.

13. When `Successfully installed and started the services` appears, click **Next**.

3.1.4. Importing Zeppelin Notebook Using Ambari

If you would like to install Zeppelin, complete the following steps after you have successfully installed HCP.

For more information about Zeppelin, see Analyzing Enriched Data Using Apache Zeppelin.

1. Login to Ambari at `http://$AMBARI_HOST:8080`.
2. In Ambari, click **Metron>Service Actions>Zeppelin Notebook Import**.

   Ambari imports the Zeppelin Notebook.
3. Login to Zeppelin at `http://$ZEPPELIN_HOST:9995`.
4. Search for the notebook named **Metron - YAF Telemetry**.

3.1.5. Streaming Data into HCP

To prepare for HCP to ingest data source data into HCP, you must stream each raw event stream from the telemetry data source into its own individual Kafka topic. This applies to the telemetry data sources for which HCP includes parsers (for example, Bro, Snort, and YAF). Even though HCP includes parsers for these data sources, HCP does not install these data sources or ingest the raw data. This is something that you must do.

Note

When you install and configure Snort, you must configure Snort to include the year in the timestamp by modifying the `snort.conf` file as follows:

```
# Configure Snort to show year in timestamps
config show_year
```

Depending on the type of data you are streaming into HCP, you can use one of the following methods:

NiFi	This type of streaming method works for most types of data sources. For information on installing NiFi, see NiFi Installation Guide. For information on using NiFi to ingest data sources into HCP, see Building a DataFlow. **Note** Ensure that the NiFi web application is using port 8089.
Performant network ingestion probes	This type of streaming method is ideal for streaming high volume packet data. See Setting Up pcap to View Your Raw Data for more information.
Real-time and batch threat intelligence feed loaders	This type of streaming method is used for real-time and batch threat intelligence feed loaders. For more information see Using Threat Intel Feed Sources.

3.1.5.1. Creating a NiFi Flow to Stream Events to HCP

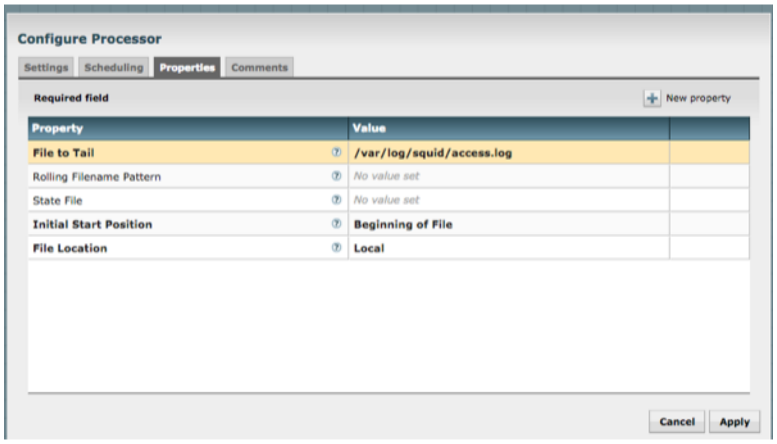
You can use NiFi to create a flow to capture events from the new data source and push them into HCP.

Important

The following task is an example using the Squid data source. Prior to creating a NiFi flow to stream Squid events to HCP, you would need to install Squid and create parsers for the data source.

1. Drag the first icon on the toolbar (the processor icon) to your workspace.
2. Select the TailFile type of processor and click **Add**.
3. Right-click the processor icon and select **Configure** to display the **Configure Processor** dialog box.
   - In the **Settings** tab, change the name to `Ingest $DATASOURCE Events`.
   - In the **Properties** tab, configure the following:

Figure 3.7. NiFi Configure Processor Dialog Box EC2 Dashboard



Configure Processor

Property	Value
File to Tail	/var/log/squid/access.log
Rolling Filename Pattern	No value set
State File	No value set
Initial Start Position	Beginning of File
File Location	Local

4. Repeat Step 1.

5. Select the PutKafka type of processor and click **Add**.

6. Right-click the processor and select **Configure**.

7. In the **Settings** tab, change the name to `Stream to Metron` and then click the relationship check boxes for failure and success.

8. In the Properties tab, set the following three properties:

   - Known Brokers: $KAFKA_HOST:6667

   - Topic Name: $DATAPROCESSOR

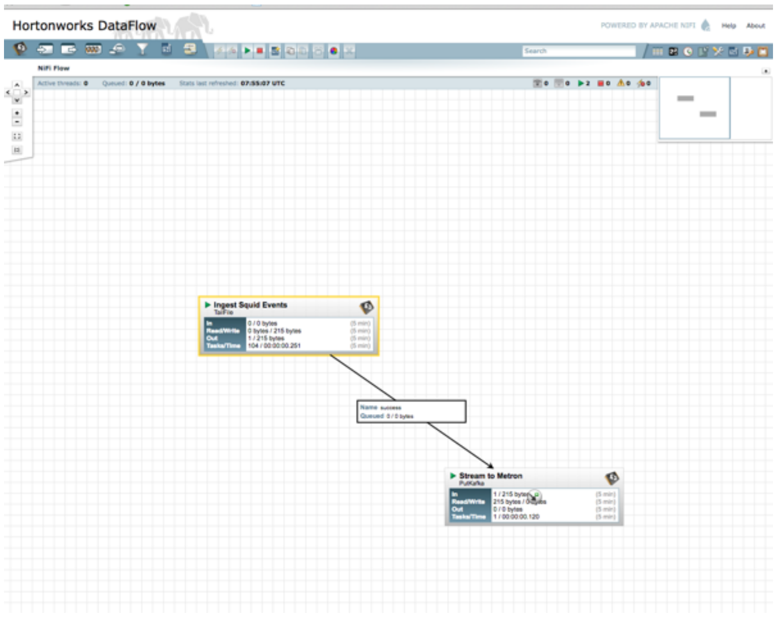
   - Client Name: nifi-$DATAPROCESSOR

9. Create a connection by dragging the arrow from the Ingest $DATAPROCESSOR Events processor to the Stream to Metron processor.

10. Press the Shift key and draw a box around both parsers to select the entire flow; then click the play button (green arrow).

    You should see all of the processor icons turn into green arrows:

Figure 3.8. NiFi Configure Processor Dialog Box EC2 Dashboard



11. Generate some data using the new data processor client.

    You should see metrics on the processor of data being pushed into Metron.

12. Look at the Storm UI for the parser topology and you should see tuples coming in.

13. After about five minutes, you should see a new Elastic Search index called $DATAPROCESSOR_index* in the Elastic Admin UI.

For more information about creating a NiFi data flow, see the NiFi documentation.

3.1.6. Verifying That HCP Deployed Successfully

After you install HCP, verify that your services are displayed in Ambari and that you can access the Metron Dashboard.

1. Verify that the topologies bundled with HCP are deployed.

   From Ambari, navigate to **Storm** > **Quick Links** > **Storm UI**.

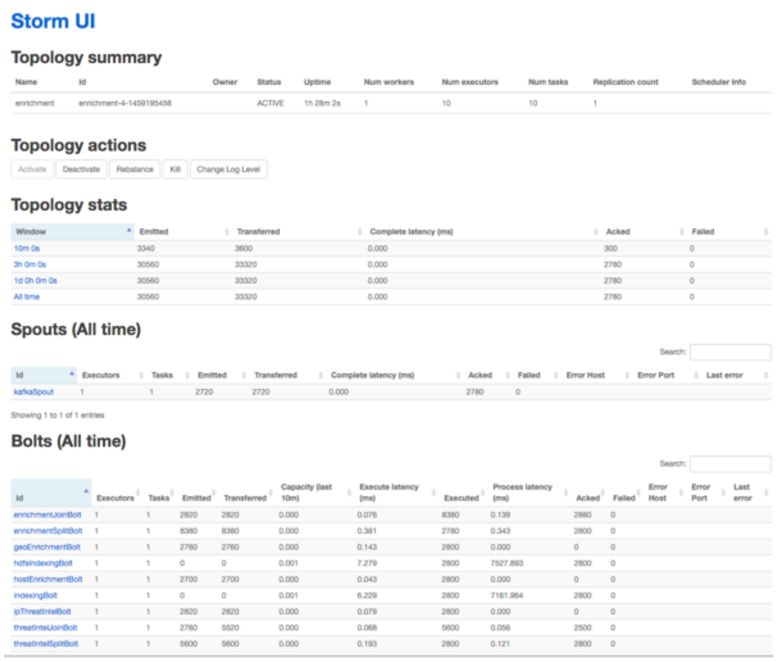
   You should see the following topologies listed :

   - Snort
   - pcap
   - YAF (Yet Another Flowmeter)
   - Bro Network Security Monitor
   - Indexing topology

2. Check that the enrichment topology has emitted some data.

   This could take a few minutes to show up in the Storm UI. The Storm enrichment topology UI should look something like the following:

   Figure 3.9. Storm UI with Enrichment Details



Storm UI

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Scheduler Info
enrichment	enrichment-4-1459195458		ACTIVE	1h 28m 2s	1	10	10	1	

Topology actions

Activate | Deactivate | Rebalance | Kill | Change Log Level

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	3340	3600	0.000	300	0
3h 0m 0s	30560	33320	0.000	2780	0
1d 0h 0m 0s	30560	33320	0.000	2780	0
All time	30560	33320	0.000	2780	0

Spouts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error
kafkaSpout	1	1	2720	2720	0.000	2780	0			

Showing 1 to 1 of 1 entries

Bolts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error
enrichmentJoinBolt	1	1	2820	2820	0.000	0.076	8380	0.139	2880	0			
enrichmentSplitBolt	1	1	8380	8380	0.000	0.381	2780	0.343	2800	0			
geoEnrichmentBolt	1	1	2760	2760	0.000	0.143	2800	0.000	0	0			
hdfsIndexingBolt	1	1	0	0	0.001	7.279	2800	7527.893	2800	0			
hostEnrichmentBolt	1	1	2700	2700	0.000	0.043	2800	0.000	0	0			
indexingBolt	1	1	0	0	0.001	6.229	2800	7161.964	2800	0			
ipThreatIntelBolt	1	1	2820	2820	0.000	0.079	2800	0.000	0	0			
threatIntelJoinBolt	1	1	2760	5520	0.000	0.068	5600	0.056	2500	0			
threatIntelSplitBolt	1	1	5600	5600	0.000	0.193	2800	0.121	2800	0			

3. Ensure that the HCP user interface is available and receiving data by displaying the HCP UI at $METRON_UI_HOST:5000.

   Check to ensure that the indexing is done correctly and the data is visualized.

4. Check to ensure that some data is written into HDFS at /apps/metron for at least one of the data sources.

To customize HCP to meet your own needs, see the following sections in the *HCP Administration Guide* for instructions on how to configure HCP:

- Adding New Telemetry Data Sources
- Enriching Telemetry Events
- Using Threat Intel Feeds
- Prioritizing Threat Intelligence

3.1.7. Downloading the HCP Management Module

The HCP Management Module is not bundled with the HCP 1.1.0 bits. You must download and install the Management module.

Before beginning the download and installation, you must set up the following:

- A running Metron cluster
- Environment variables for:
  - METRON_HOME
  - METRON_VERSION
  - ZOOKEEPER
  - BROKERLIST
  - HDFS_URL
  - STORM_REST_URL

  The zookeeper and kafka broker list should be a comma-delimited list of host:port values for the nodes in the quorums. For example:

  ```
  ip-10-0-0-25.us-west-1.compute.internal:2181,ip-10-0-0-95.us-west-1.compute.
  internal:2181,ip-10-0-0-107.us-west-1.compute.internal:2181,ip-10-0-0-125.
  us-west-1.compute.internal:2181
  ip-10-0-0-25.us-west-1.compute.internal:6667,ip-10-0-0-95.us-west-1.compute.
  internal:6667,ip-10-0-0-107.us-west-1.compute.internal:6667,ip-10-0-0-125.
  us-west-1.compute.internal:6667
  ```

  See Installation Variables for more information.

Perform the following tasks to download and install the HCP Management module:

- Install REST Application [19]
- Install Management Module User Interface [22]

3.1.7.1. Install REST Application

To install a REST API to interact with Metron, complete the following steps:

1. Create a file `/etc/yum.repos.d/HCP.repo` containing the following information. Make sure the information matches your installation, including the version number, etc.

   ```
   #VERSION_NUMBER=1.1.0.0-71
   [HCP-1.1.0.0-71]
   name=HCP Version - HCP-1.1.0.0-71
   baseurl=http://s3.amazonaws.com/dev.hortonworks.com/HCP/centos6/1.x/BUILDS/
   1.1.0.0-71
   gpgcheck=1
   gpgkey=http://s3.amazonaws.com/dev.hortonworks.com/HCP/centos6/1.x/BUILDS/1.
   1.0.0-71/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
   enabled=1
   priority=1
   ```

2. Install the REST package:

   - CentOS 6 and 7

     ```
     yum install metron-rest
     ```

- Ubuntu

  ```
  apt-get install metron-rest
  ```

3. Connect to MySQL and create a Metron REST database:

   ```
   mysql -uroot -p
   CREATE DATABASE IF NOT EXISTS metronrest;
   ```

4. Create a Metron user in MySQL with a password, then apply database access permission to the Metron user:

   ```
   CREATE USER 'metron'@'REST_HOST' IDENTIFIED BY 'Myp@ssw0rd';
   GRANT ALL PRIVILEGES ON metronrest.* TO 'metron'@'REST_HOST';
   ```

5. Create user and authorities tables:

   ```
   use metronrest;
   create table if not exists users(
     username varchar(50) not null primary key,
     password varchar(50) not null,
     enabled boolean not null
   );
   create table authorities (
     username varchar(50) not null,
     authority varchar(50) not null,
     constraint fk_authorities_users foreign key(username) references
    users(username)
   );
   create unique index ix_auth_username on authorities (username,authority);
   ```

6. Add one or more users to the REST application:

   ```
   use metronrest;
   insert into users (username, password, enabled) values ('your_username',
   'your_password',1);
   insert into authorities (username, authority) values ('your_username',
    'ROLE_USER');
   ```

7. Exit MySQL:

   ```
   quit
   ```

8. Create the `$METRON_HOME/config/application.yml` file containing the following:

   ```
   server:
     port: 8082

   spring:
     datasource:
         driverClassName: org.h2.Driver
         url: jdbc:h2:file:./metrondb
         username: root
         password: root
         platform: h2
     jpa:
       hibernate:
         ddl-auto: update

   zookeeper:
   ```

```
  url:  ${ZOOKEEPER}

kafka:
  broker:
    url: ${BROKERLIST}

hdfs:
  namenode:
    url: ${HDFS_URL}

grok:
  path:
    temp: ./patterns/temp
    default: /apps/metron/patterns

storm:
  ui:
    url:   ${STORM_REST_URL}
  parser:
    script.path: ${METRON_HOME}/bin/start_parser_topology.sh
  enrichment:
    script.path: ${METRON_HOME}/bin/start_enrichment_topology.sh
  indexing:
    script.path: ${METRON_HOME}/bin/start_elasticsearch_topology.sh
```

9. Replace the H2 connection information in the `$METRON_HOME/config/application.yml` file with MySQL connection information:

```
spring:
  datasource:
        driverClassName: com.mysql.jdbc.Driver
        url: jdbc:mysql://mysql_host:3306/metronrest
        username: metron_rest_user
        password: metron_rest_password
```

10. Install the appropriate MySQL client library for your version of MySQL. For example:

```
cd $METRON_HOME/lib
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.
1.41.tar.gz
tar xf mysql-connector-java-5.1.41.tar.gz
```

11. Start the REST application:

```
nohup java -Dloader.path=$METRON_HOME/lib/mysql-connector-java-5.1.
41/mysql-connector-java-5.1.41-bin.jar -jar $METRON_HOME/lib/metron-
rest-$METRON_VERSION.jar --spring.config.location=$METRON_HOME/config/
application.yml > rest-api.log 2>&1 &
```

where:

nohup	No hangup. The `nohup` command is a POSIX command to ignore the HUP (hangup) signal. The HUP signal is used, by convention, by a terminal to warn dependent processes of logout. Output that would normally go to the terminal goes to a file called `nohup.out` if it has not already been redirected.
2>&1	Redirect all `stderr` output to `stdout`. File descriptor 1 is `stdout`, file descriptor 2 is `stderr`. The ampersand instructs the shell to interpret the 1 as a file descriptor instead of as a file named "1".

&   Background the process

12. To add additional users:

```
use metronrest;
insert into users (username, password, enabled) values ('your_username',
'your_password',1);
insert into authorities (username, authority) values ('your_username',
 'ROLE_USER');
commit;
```

13. In a cluster with Kerberos enabled, update the security settings in `/etc/sysconfig/metron`.

```
METRON_SPRING_PROFILES_ACTIVE="vagrant,dev"
METRON_JVMFLAGS="Djava.security.auth.login.config=$METRON_HOME/client_jaas.
confg"
METRON_SPRING_OPTIONS="--kerberos.enables=true"
```

14. Make sure you can access the Swagger UI at http://host:port/swagger-ui.html#/.

    The exposed REST endpoints can be accessed with the Swagger UI at http://host:port/swagger-ui.html#/. The default port is 8082 but can be changed in the `$METRON_HOME/config/application.yml` file by setting "server.port" to the desired port.

3.1.7.2. Install Management Module User Interface

After you install the REST application, you need to install the Management module user interface. To install the Management module UI, complete the following steps:

1. Install the Management module application:

```
yum install metron-config
```

2. Install the `nodejs`:

```
curl --silent --location https://rpm.nodesource.com/setup_6.x | bash - &&
 yum install -y nodejs
```

3. Navigate to the `/usr/hcp/current/metron/` directory:

```
cd /usr/hcp/current/metron/
```

4. Start the Management module:

```
./bin/start_management_ui.sh -p 4200 -r http://$REST_HOST:8082
```

5. Launch the Management module in a browser window at http://host:4200.

   The default credentials for username/password are **`user/password`**.

3.2. Manually Installing HCP

The following sections provide instructions on how to manually install HCP on a virtual machine (VM) or an existing cluster. You can perform this manual installation on any cluster if it meets the prerequisites listed next. Installation on a VM is not recommended for production deployment but rather for development and testing environments.

Note

The instructions in this section apply to a cluster running CentOS. For information on installing on a cluster running Ubuntu, see Manually Installing Apache Metron on Ubuntu 14.04.

Note

Prior to starting this installation, you must set up Elasticsearch. For information on setting up Elasticsearch, see Elasticsearch documentation.

Manually installing HCP requires the following major steps:

- Preparing the Environment [24]
- Installing HCP [24]
- Verifying that HCP Deployed Successfully [33]

3.2.1. Installation Variables

During a manual HCP cluster installation, you must insert your own values into the following list.

METRON_HOME	The location of the Metron home; usually `/usr/hcp/HCP_RELEASE/metron`.
KAFKA_HOST	The host on which a Kafka broker is installed.
ZOOKEEPER_HOST	The host on which a ZooKeeper server is installed.
PROBE_HOST	The host on which your sensor, probes are installed. If you do not have any sensors installed, pick the host where an Apache Storm supervisor is running.
$DATASOURCE_HOST	The host on which you want to install your data source. If you do not care, install the data source on the PROBE_HOST.
NIFI_HOST	The host on which you will install NiFi. This should be the same host on which you installed $DATASOURCE.
HOST_WITH_ENRICHMENT_TAG	The host in your inventory hosts file that you listed under the group "enrichment."
SEARCH_HOST	The host where you have Elasticsearch or Apache Solr running.

	This is the host in your inventory hosts file that you listed under the group "search." Choose one of the search hosts.
SEARCH_HOST_PORT	The port of the search host where indexing is configured (for example, 9300).
METRON_UI_HOST	The host on which your Apache Metron UI web application is running. This is the host in your inventory hosts file that you put under the group "web."
METRON_VERSION	The release of the Metron binaries you are working with (for example, 0.2.0BETA—RC2).

3.2.2. Preparing the Environment

Installing HCP on an Ambari-managed cluster requires that you meet the following specifications on the cluster:

- Specifications for Hadoop Cluster [5]
- Specifications for Metron Nodes [6]

3.2.3. Installing HCP

Manually installing HCP involves several steps. This section details each of these steps, including the following:

- Setting Environment Variables [25]
- Creating a Repository [25]
- Installing HCP [26]
- Creating Kafka Topics [26]
- Creating HBase Tables [27]
- Creating an HCP Global.json File [27]
- Setting up the Metron Enrichment [28]
- Setting Up Indexing [28]
- Pushing the Configuration Changes to ZooKeeper [29]
- Streaming Data into HCP [29]
- Starting Your Parsers [32]
- Starting Your Enrichments [32]
- Starting Indexing [32]

- Downloading the HCP Management Module [35]
- Importing the Apache Zeppelin Notebook Manually [33]

3.2.3.1. Setting Environment Variables

Before beginning the installation instructions, you must set the following environmental variables, using values specific to your deployment.

From the Metron install host, enter the following to define your environment variables:

Note

The Metron install host must be running Supervisor, Kafka Broker, and HBase client.

```
mysqlrootuser=root
mysqldbuser=metron – always specify the DB user as metron
mysqldbhost={{ metron install host }} – Where you will be doing the metron
 install
mysqldbpasswd={{ metron user mysql password }}
mysql -u $mysqlrootuser -e "CREATE USER '$mysqldbuser'@'$mysqldbhost'
 IDENTIFIED BY '$mysqldbpasswd';"
mysql -u root -e "CREATE USER '$mysqldbuser'@'localhost' IDENTIFIED BY
 '$mysqldbpasswd';"
mysql -u root -e "GRANT ALL PRIVILEGES ON *.* TO
 '$mysqldbuser'@'$mysqldbhost';"
mysql -u root -e "GRANT ALL PRIVILEGES ON *.* TO '$mysqldbuser'@'localhost';"
mysql -u root -e "GRANT ALL PRIVILEGES ON *.* TO '$mysqldbuser'@'%' IDENTIFIED
 BY '$mysqldbpasswd';"
mysql -u root -e "flush privileges;"
```

3.2.3.2. Creating a Repository

Complete the following steps to create a HCP repository:

1. Create a file called `/etc/yum.repos.d/hcp.repo`.

   ```
   vi /etc/yum/repos.d/hcp.repo
   ```

2. Populate the file with the latest repository information.

   For example:

   ```
   #VERSION_NUMBER=1.1.0.0
   [HCP-1.0.0.0-52]
   name=HCP Version - HCP-1.1.0.0
   baseurl=http://public-repo-1.hortonworks.com/HCP/centos7/1.x/updates/1.1.0.
   0/hcp.repo
   gpgcheck=1
   gpgkey=http://public-repo-1.hortonworks.com/HCP/centos7/1.x/updates/1.1.0.0/
   hcp.repo/BUILDS/1.1.0.0/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
   enabled=1
   priority=1


   [HDP-UTILS-2.5.3.0]
   ```

```
name=HDP-UTILS Version - HDP-UTILS-2.5.3.0
baseurl=http://s3.amazonaws.com/dev.hortonworks.com/HDP-UTILS-2.5.3.0/repos/
centos7
gpgcheck=1
gpgkey=http://s3.amazonaws.com/dev.hortonworks.com/HCP/centos7/1.x/updates/
2.5.3.0/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

3.2.3.3. Installing HCP

1. If you are on a cluster, log in to the Metron Installer node.

   You should have specified the Metron Installer host during your cluster installation. See Installation Variables for more information.

2. Install HCP:

   ```
   yum install -y metron*
   ```

   HCP will display the size of the download and ask you if it is okay to proceed. Enter **Y** and press Enter.

   The installation will take several minutes.

3.2.3.4. Creating Kafka Topics

You need to create a Kafka topic for each supported data source.

1. Assign the ZooKeeper host to the zookeeper variable

   ```
   zookeeper={{zookeeper_quorum}}
   ```

2. From the `kafka-bin` directory (`/usr/hdp/current/kafka-broker/bin`), enter the following commands:

   Because you assigned the zookeeper variable in Step 1, you can copy and paste the following commands.

   ```
   #bro
   ./kafka-topics.sh \
   --zookeeper ${zookeeper} \
   --create --partitions 1 --replication-factor 1 \
       --config retention.bytes=10737418240 \
   --topic bro

   #yaf
   ./kafka-topics.sh \
   --zookeeper ${zookeeper} \
   --create --partitions 1 --replication-factor 1 \
       --config retention.bytes=10737418240 \
   --topic yaf

   #snort
   ./kafka-topics.sh \
   --zookeeper ${zookeeper} \
   --create --partitions 1 --replication-factor 1 \
       --config retention.bytes=10737418240 \
   ```

```
--topic snort

#Parser Invalid
./kafka-topics.sh \
--zookeeper ${zookeeper} \
--create --partitions 1 --replication-factor 1 \
    --config retention.bytes=10737418240 \
--topic parser_invalid

#Parser Error
./kafka-topics.sh \
--zookeeper ${zookeeper} \
--create --partitions 1 --replication-factor 1 \
    --config retention.bytes=10737418240 \
--topic parser_error

#enrichments
./kafka-topics.sh \
--zookeeper ${zookeeper} \
--create --partitions 1 --replication-factor 1 \
    --config retention.bytes=10737418240 \
--topic enrichments

#indexing
./kafka-topics.sh \
--zookeeper ${zookeeper} \
--create --partitions 1 --replication-factor 1 \
    --config retention.bytes=10737418240 \
--topic indexing

./kafka-topics.sh --zookeeper ${zookeeper} --list
```

The final command lists the Kafka topics you just created. You should see the following topics listed:

```
bro
enrichments
indexing
parser_error
parser_invalid
snort
yaf
```

3.2.3.5. Creating HBase Tables

Create the HBase tables by entering the following command from the Metron install host:

```
for i in {pcap,access_tracker,threatintel,enrichment} ; do echo "create
 '${i}','t'" | hbase shell -n; done
```

3.2.3.6. Creating an HCP Global.json File

The `Global.json` file lets Metron know where things are.

To create the `Global.json` file, complete the following steps:

1. Create a file called `$METRON_HOME/config/zookeeper/global.json`.

   ```
   touch /usr/metron/1.0.0.1.0.0.0-52/config/zookeeper/global.json
   ```

2. Populate the file with the following information:

   ```
   {
     "es.clustername": "{{ elasticsearch_cluster_name }}",
     "{{es_master_host}}": "{{ groups.search[0] }}",
     "es.port": "{{ elasticsearch_transport_port }}",
     "es.date.format": "yyyy.MM.dd.HH"
   }
   ```

   where

es.clustername	This should be `metron`.
es_master_host	This can be either an IP address or a host name.
es.port	The Elasticsearch port. Default is 9300.
es.data.format	The data format used by Elasticsearch. You can leave this format as is.

3.2.3.7. Setting up the Metron Enrichment

You need to modify the `enrichment.properties` file to match your HCP configuration.

To modify the enrichment properties file, complete the following steps:

1. Open the `METRON_HOME/config/enrichment.properties` file:

   ```
   vi METRON_HOME/config/enrichment.properties
   ```

2. Modify the following Kafka entries to reflect your configuration:

   ```
   kafka.zk={{zookeeper_quorum}}
   kafka.broker={{kafka_brokers}}
   enrichment.output.topic=indexing
   ```

3. Add the following text above #### Threat Intel ####:

   ```
   ##### Host Enrichment #####

   hbase.provider.impl=org.apache.metron.hbase.HTableProvider
   enrichment.simple.hbase.table=enrichment
   enrichment.simple.hbase.cf=t
   ```

4. Replace the Threat Intel text with the following:

   ```
   threat.intel.tracker.table=access_tracker
   threat.intel.tracker.cf=t
   threat.intel.simple.hbase.table=threatintel
   threat.intel.simple.hbase.cf=t
   threat.intel.ip.table=
   threat.intel.ip.cf=
   ```

3.2.3.8. Setting Up Indexing

1. Modify the `$METRON_HOME/config/elasticsearch.properties` file to reflect your configuration:

```
vi $METRON_HOME/config/elasticsearch.properties
```

2. Modify the number of Storm workers if needed.

```
indexing.workers={{ defaults to 1 start with number of ES hosts }}
indexing.executors=0
```

3. Modify the Kafka text to reflect your configuration:

```
kafka.zk={{zookeeper_quorum}}
kafka.broker={{kafka_brokers}}
kafka.start=UNCOMMITTED_EARLIEST
```

4. Modify the Elasticsearch text to reflect your configuration:

```
{{es_master_host}}={{es_host}}
es.port=9300
es.clustername=metron
```

3.2.3.9. Pushing the Configuration Changes to ZooKeeper

The final step in the installation is to push the configuration changes to ZooKeeper.

Complete the following steps to push the configuration changes to ZooKeeper:

1. Change to the `$METRON_HOME/bin` directory.

```
cd $METRON_HOME/bin
```

2. Because you set the ZooKeeper variable in an earlier step, you can simply run the following command:

```
./zk_load_configs.sh -i ../config/zookeeper -m PUSH -z ${zookeeper}
```

3. To verify that your push was successful, run the following command:

```
./zk_load_configs.sh -i ../config/zookeeper -m DUMP -z ${zookeeper}
```

3.2.3.10. Loading GeoIP Data

Load the GeoIP data by entering the following command from the Metron install host:

```
./geo_enrichment_load.sh -z {zookeeper}
```

3.2.3.11. Streaming Data into HCP

To prepare for HCP to ingest data source data into HCP, you must stream each raw event stream from the telemetry data source into its own individual Kafka topic. This applies to the telemetry data sources for which HCP includes parsers (for example, Bro, Snort, and YAF). Even though HCP includes parsers for these data sources, HCP does not install these data sources or ingest the raw data. This is something that you must do.

Note

When you install and configure Snort, you must configure Snort to include the year in the timestamp by modifying the `snort.conf` file as follows:

```
# Configure Snort to show year in timestamps
config show_year
```

Depending on the type of data you are streaming into HCP, you can use one of the following methods:

NiFi	This type of streaming method works for most types of data sources. For information on installing NiFi, see NiFi Installation Guide. For information on using NiFi to ingest data sources into HCP, see Building a DataFlow. **Note** Ensure that the NiFi web application is using port 8089.
Performant network ingestion probes	This type of streaming method is ideal for streaming high volume packet data. See Setting Up pcap to View Your Raw Data for more information.
Real-time and batch threat intelligence feed loaders	This type of streaming method is used for real-time and batch threat intelligence feed loaders. For more information see Using Threat Intel Feeds.

3.2.3.11.1. Creating a NiFi Flow to Stream Events to HCP

You can use NiFi to create a flow to capture events from the new data source and push them into HCP.

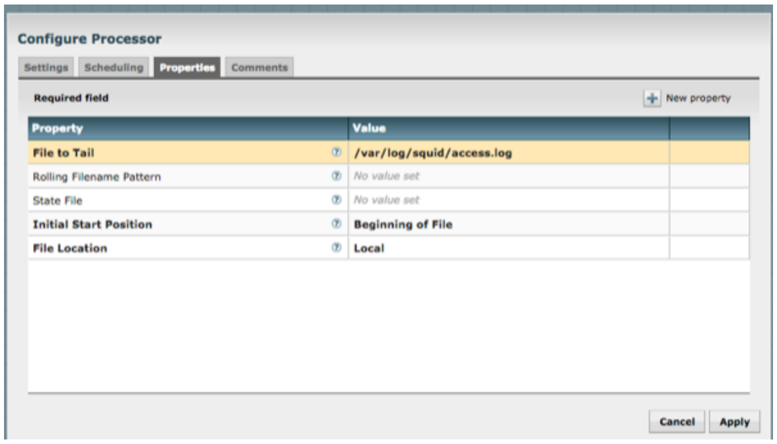
Important

The following task is an example using the Squid data source. Prior to creating a NiFi flow to stream Squid events to HCP, you would need to install Squid and create parsers for the data source.

To perform this task, complete the following steps:

1. Drag the first icon on the toolbar (the processor icon) to your workspace.
2. Select the TailFile type of processor and click **Add**.
3. Right-click the processor icon and select **Configure** to display the **Configure Processor** dialog box.
    - In the **Settings** tab, change the name to **`Ingest $DATASOURCE Events`**.
    - In the **Properties** tab, configure the following:

Figure 3.10. NiFi Configure Processor Dialog Box EC2 Dashboard



Property	Value
File to Tail	/var/log/squid/access.log
Rolling Filename Pattern	No value set
State File	No value set
Initial Start Position	Beginning of File
File Location	Local

4. Repeat Step 1.

5. Select the PutKafka type of processor and click **Add**.

6. Right-click the processor and select **Configure**.

7. In the **Settings** tab, change the name to `Stream to Metron` and then click the relationship check boxes for failure and success.

8. In the **Properties** tab, set the following three properties:

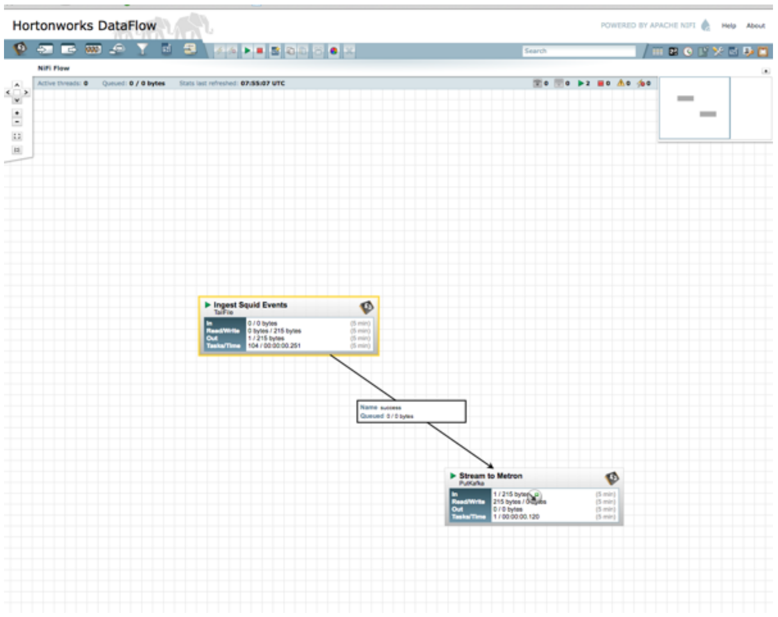
   - Known Brokers: $KAFKA_HOST:6667
   - Topic Name: $DATAPROCESSOR
   - Client Name: nifi-$DATAPROCESSOR

9. Create a connection by dragging the arrow from the Ingest $DATAPROCESSOR Events processor to the Stream to Metron processor.

10. Press the Shift key and draw a box around both parsers to select the entire flow; then click the play button (green arrow).

    You should see all of the processor icons turn into green arrows:

Figure 3.11. NiFi Configure Processor Dialog Box EC2 Dashboard



11.Generate some data using the new data processor client.

You should see metrics on the processor of data being pushed into Metron.

12.Look at the Storm UI for the parser topology and you should see tuples coming in.

13.After about five minutes, you should see a new Elastic Search index called $DATAPROCESSOR_index* in the Elastic Admin UI.

For more information about creating a NiFi data flow, see the NiFi documentation.

3.2.3.12. Starting Your Parsers

Start each of the parsers included in HCP:

```
./start_parser_topology.sh -k ${kafka_brokers} -z ${zookeeper} -s bro
./start_parser_topology.sh -k ${kafka_brokers} -z ${zookeeper} -s snort
./start_parser_topology.sh -k ${kafka_brokers} -z ${zookeeper} -s yaf
```

3.2.3.13. Starting Your Enrichments

Start the enrichments included in HCP:

```
./start_enrichment_topology.sh
```

3.2.3.14. Starting Indexing

Start the indexing:

```
./start_elasticsearch_topology.sh
```

3.2.4. Importing the Apache Zeppelin Notebook Manually

If you would like to import the Apache Zeppelin notebook manually, complete the following steps:

For more information about Apache Zeppelin, see Analyzing Enriched Data Using Apache Zeppelin.

1. Use ssh to navigate to the host where you want to install Zeppelin.

```
ssh $METRON_HOME
```

2. Use the following command to import the "`$METRON_HOME/config/zeppelin/metron-yaf-telemetry.json`" file onto your Zeppelin host.

```
curl -s -XPOST https://github.com/apache/incubator-metron/blob/master/
metron-platform/metron-indexing/src/main/config/zeppelin/metron/metron-yaf-
telemetry.json/api/notebook/import -d @"$METRON_HOME/config/zeppelin/metron-
yaf-telemetry.json"
```

3. Navigate to `http://$ZEPPELIN_HOST:9995`.

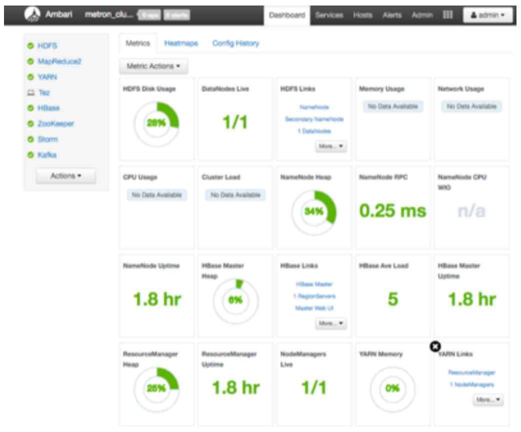
3.2.5. Verifying that HCP Deployed Successfully

After you install HCP, you can verify that your installation was a success by checking the various tools you installed.

1. Check Ambari to verify that all of the services are active by entering $AMBARI_HOST:8080 in a web browser, to display Ambari.

   The Ambari dashboard should look similar to the following:

   Figure 3.12. Ambari Metron Dashboard



2. Verify that the topologies bundled with HCP are deployed.

   From Ambari, navigate to **Storm** > **Quick Links** > **Storm UI**.

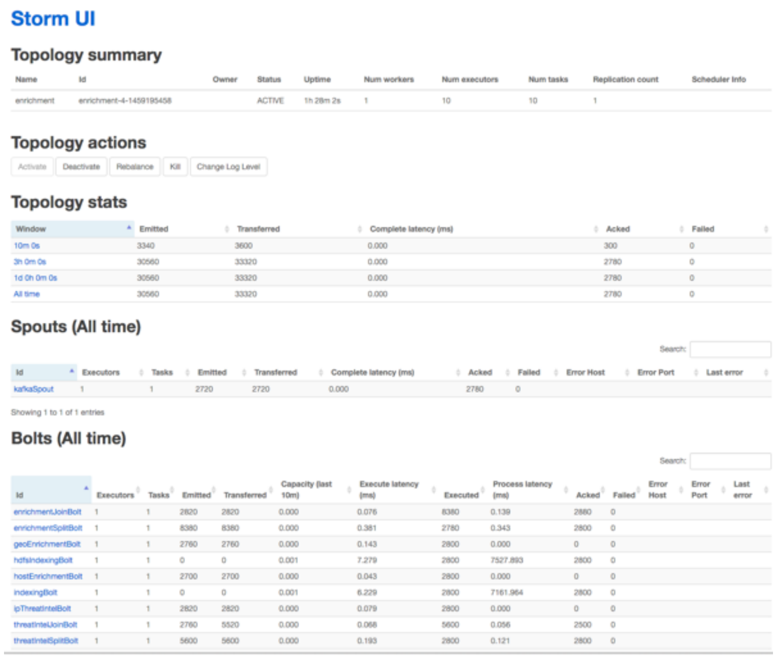
   You should see the following topologies listed :

- Bro
- Snort
- YAF
- Enrichment
- Indexing

3. Check that the enrichment topology has emitted some data.

   This could take a few minutes to show up in the Storm UI. The Storm enrichment topology UI should look something like the following:

   Figure 3.13. Storm UI with Enrichment



Storm UI

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Scheduler Info
enrichment	enrichment-4-1459195458		ACTIVE	1h 28m 2s	1	10	10	1	

Topology actions

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	3340	3600	0.000	300	0
3h 0m 0s	30560	33320	0.000	2780	0
1d 0h 0m 0s	30560	33320	0.000	2780	0
All time	30560	33320	0.000	2780	0

Spouts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error
kafkaSpout	1	1	2720	2720	0.000	2780	0			

Showing 1 to 1 of 1 entries

Bolts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error
enrichmentJoinBolt	1	1	2820	2820	0.000	0.076	8380	0.139	2880	0			
enrichmentSplitBolt	1	1	8380	8380	0.000	0.381	2780	0.343	2800	0			
geoEnrichmentBolt	1	1	2760	2760	0.000	0.143	2800	0.000	0	0			
hdfsIndexingBolt	1	1	0	0	0.001	7.279	2800	7527.893	2800	0			
hostEnrichmentBolt	1	1	2700	2700	0.000	0.043	2800	0.000	0	0			
indexingBolt	1	1	0	0	0.001	6.229	2800	7161.964	2800	0			
ipThreatIntelBolt	1	1	2820	2820	0.000	0.079	2800	0.000	0	0			
threatIntelJoinBolt	1	1	2760	5520	0.000	0.068	5600	0.056	2500	0			
threatIntelSplitBolt	1	1	5600	5600	0.000	0.193	2800	0.121	2800	0			

4. Ensure that the HCP user interface is available and receiving data by displaying the HCP UI at $METRON_UI_HOST:5000.

   Check to verify that the indexing is done correctly and the data is visualized.

To customize HCP to meet your own needs, see the following sections in the *HCP Administration Guide* for instructions on how to configure HCP:

- Adding New Telemetry Data Sources
- Enriching Telemetry Events
- Using Threat Intel Feeds
- Prioritizing Threat Intelligence

3.2.6. Downloading the HCP Management Module

The HCP Management Module is not bundled with the HCP 1.1.0 bits. You must download and install the Management module.

Before beginning the download and installation, you must set up the following:

- A running Metron cluster
- Environment variables for:
  - METRON_HOME
  - ZOOKEEPER
  - BROKERLIST
  - HDFS_URL
  - STORM_REST_URL

  The zookeeper and kafka broker list should be a comma-delimited list of host:port values for the nodes in the quorums. For example:

  ```
  ip-10-0-0-25.us-west-1.compute.internal:2181,ip-10-0-0-95.us-west-1.compute.
  internal:2181,ip-10-0-0-107.us-west-1.compute.internal:2181,ip-10-0-0-125.
  us-west-1.compute.internal:2181
  ip-10-0-0-25.us-west-1.compute.internal:6667,ip-10-0-0-95.us-west-1.compute.
  internal:6667,ip-10-0-0-107.us-west-1.compute.internal:6667,ip-10-0-0-125.
  us-west-1.compute.internal:6667
  ```

  See Installation Variables for more information.

Perform the following tasks to download and install the HCP Management module:

- Install REST Application [35]
- Install Management Module User Interface [38]

3.2.6.1. Install REST Application

To install a REST API to interact with Metron, complete the following steps:

1. Create a file `/etc/yum.repos.d/HCP.repo` containing the following information. Make sure the information matches your installation, including the version number, etc.

   ```
   #VERSION_NUMBER=1.1.0.0-71
   [HCP-1.1.0.0-71]
   name=HCP Version - HCP-1.1.0.0-71
   baseurl=http://s3.amazonaws.com/dev.hortonworks.com/HCP/centos6/1.x/BUILDS/
   1.1.0.0-71
   gpgcheck=1
   gpgkey=http://s3.amazonaws.com/dev.hortonworks.com/HCP/centos6/1.x/BUILDS/1.
   1.0.0-71/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
   enabled=1
   priority=1
   ```

2. Install the REST package:

   - CentOS 6 and 7

     ```
     yum install metron-rest
     ```

   - Ubuntu

     ```
     apt-get install metron-rest
     ```

3. Connect to MySQL and create a Metron REST database:

   ```
   mysql -uroot -p
   CREATE DATABASE IF NOT EXISTS metronrest;
   ```

4. Create a Metron user in MySQL with a password, then apply database access permission to the Metron user:

   ```
   CREATE USER 'metron'@'REST_HOST' IDENTIFIED BY 'Myp@ssw0rd';
   GRANT ALL PRIVILEGES ON metronrest.* TO 'metron'@'REST_HOST';
   ```

5. Create user and authorities tables:

   ```
   use metronrest;
   create table if not exists users(
     username varchar(50) not null primary key,
     password varchar(50) not null,
     enabled boolean not null
   );
   create table authorities (
     username varchar(50) not null,
     authority varchar(50) not null,
     constraint fk_authorities_users foreign key(username) references
    users(username)
   );
   create unique index ix_auth_username on authorities (username,authority);
   ```

6. Add one or more users to the REST application:

   ```
   use metronrest;
   insert into users (username, password, enabled) values ('your_username',
   'your_password',1);
   insert into authorities (username, authority) values ('your_username',
    'ROLE_USER');
   ```

7. Exit MySQL:

   ```
   quit
   ```

8. Create the `$METRON_HOME/config/application.yml` file containing the following:

   ```
   server:
     port: 8082

   spring:
     datasource:
         driverClassName: org.h2.Driver
         url: jdbc:h2:file:./metrondb
         username: root
         password: root
         platform: h2
   ```

```
  jpa:
    hibernate:
      ddl-auto: update

zookeeper:
  url:  ${ZOOKEEPER}

kafka:
  broker:
    url: ${BROKERLIST}

hdfs:
  namenode:
    url: ${HDFS_URL}

grok:
  path:
    temp: ./patterns/temp
    default: /apps/metron/patterns

storm:
  ui:
    url:   ${STORM_REST_URL}
  parser:
    script.path: ${METRON_HOME}/bin/start_parser_topology.sh
  enrichment:
    script.path: ${METRON_HOME}/bin/start_enrichment_topology.sh
  indexing:
    script.path: ${METRON_HOME}/bin/start_elasticsearch_topology.sh
```

9. Replace the H2 connection information in the `$METRON_HOME/config/application.yml` file with MySQL connection information:

```
spring:
  datasource:
        driverClassName: com.mysql.jdbc.Driver
        url: jdbc:mysql://mysql_host:3306/metronrest
        username: metron_rest_user
        password: metron_rest_password
```

10. Install the appropriate MySQL client library for your version of MySQL. For example:

```
cd $METRON_HOME/lib
wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.
1.41.tar.gz
tar xf mysql-connector-java-5.1.41.tar.gz
```

11. Start the REST application:

```
nohup java -Dloader.path=$METRON_HOME/lib/mysql-connector-java-5.1.
41/mysql-connector-java-5.1.41-bin.jar -jar $METRON_HOME/lib/metron-
rest-$METRON_VERSION.jar --spring.config.location=$METRON_HOME/config/
application.yml > rest-api.log 2>&1 &
```

where:

nohup	No hangup. The `nohup` command is a POSIX command to ignore the HUP (hangup) signal. The HUP signal is used, by convention, by a terminal to warn dependent processes of logout. Output that would normally go to the terminal goes to a file called `nohup.out` if it has not already been redirected.

2>&1	Redirect all `stderr` output to `stdout`. File descriptor 1 is `stdout`, file descriptor 2 is `stderr`. The ampersand instructs the shell to interpret the 1 as a file descriptor instead of as a file named "1".
&	Background the process

12. To add additional users:

    ```
    use metronrest;
    insert into users (username, password, enabled) values ('your_username',
    'your_password',1);
    insert into authorities (username, authority) values ('your_username',
     'ROLE_USER');
    commit;
    ```

13. In a cluster with Kerberos enabled, update the security settings in `/etc/sysconfig/metron`.

    ```
    METRON_SPRING_PROFILES_ACTIVE="vagrant,dev"
    METRON_JVMFLAGS="Djava.security.auth.login.config=$METRON_HOME/client_jaas.
    confg"
    METRON_SPRING_OPTIONS="--kerberos.enables=true"
    ```

14. Make sure you can access the Swagger UI at http://host:port/swagger-ui.html#/.

    The exposed REST endpoints can be accessed with the Swagger UI at http://host:port/swagger-ui.html#/. The default port is 8082 but can be changed in the `$METRON_HOME/config/application.yml` file by setting "server.port" to the desired port.

3.2.6.2. Install Management Module User Interface

After you install the REST application, you need to install the Management module user interface. To install the Management module UI, complete the following steps:

1. Install the Management module application:

    ```
    yum install metron-config
    ```

2. Install the `nodejs`:

    ```
    curl --silent --location https://rpm.nodesource.com/setup_6.x | bash - &&
     yum install -y nodejs
    ```

3. Navigate to the `/usr/hcp/current/metron/` directory:

    ```
    cd /usr/hcp/current/metron/
    ```

4. Start the Management module:

    ```
    ./bin/start_management_ui.sh -p 4200 -r http://$REST_HOST:8082
    ```

5. Launch the Management module in a browser window at http://host:4200.

3.3. Optimization Guidelines

In any Storm based platform, there are many parameters that control the system's performance. The values of these parameters vary greatly with differences in cluster size and data velocity. Ensuring that you have a properly tuned index is key to overall system performance. The following Storm parameters may also be tuned to improve overall system performance. See the Storm user guide for detailed discussion.

- num.workers
- num.ackers
- max.spout.pending
- topology.worker.childopts – increase heap size (-XmxNNNNm –XmsNNNNm)
- topology.workers"

4. Enabling Kerberos

How you enable Kerberos for your HCP environment depends on how you installed HCP. If you installed HCP with Ambari, you can use Ambari to enable Kerberos. If you installed HCP manually, you will need to enable Kerberos manually. This chapter contains instructions for enabling Kerberos using each method.

- Using Ambari to Enable Kerberos [40]
- Manually Enabling Kerberos [47]

Note

The Management module is not supported in a Kerberized environment.

4.1. Using Ambari to Enable Kerberos

If you installed HDP using Ambari, then you can use Ambari to enable Kerberos for your HCP environment.

To enable Kerberos on Ambari, complete the following steps:

- Installing and Configuring the KDC [40]
- Installing the JCE [45]
- Enabling Kerberos on Ambari [45]

4.1.1. Installing and Configuring the KDC

Ambari is able to configure Kerberos in the cluster to work with an existing MIT KDC (key distribution center), or existing Active Directory installation. This section describes the steps necessary to prepare for this integration.

Note

If you do not have an existing KDC (MIT or Active Directory), Install a new MIT KDC. Installing a KDC on a cluster host **after** installing the Kerberos client might overwrite the krb5.conf file generated by Ambari.

You may choose to have Ambari connect to the KDC and automatically create the necessary Service and Ambari principals, generate and distribute the keytabs (“Automated Kerberos Setup”). Ambari also provides an advanced option to manually configure Kerberos. If you choose this option, you must create the principals, generate and distribute the keytabs. Ambari will not do this automatically (“Manual Kerberos Setup”).

- Use an Existing MIT KDC [41]
- Use an Existing Active Directory [41]

- Use Manual Kerberos Setup [42]

Use the instructions to (Optional) Install a new MIT KDC if you do not have an existing KDC available.

4.1.1.1. Use an Existing MIT KDC

To use an existing MIT KDC for the cluster, you must prepare the following:

- Ambari Server and cluster hosts have network access to both the KDC and KDC admin hosts.
- KDC administrative credentials are on-hand.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords.

4.1.1.2. Use an Existing Active Directory

To use an existing Active Directory domain for the cluster with Automated Kerberos Setup, you must prepare the following:

- Ambari Server and cluster hosts have network access to, and be able to resolve the DNS names of, the Domain Controllers.
- Active Directory secure LDAP (LDAPS) connectivity has been configured.
- Active Directory User container for principals has been created and is on-hand. For example, "OU=Hadoop,OU=People,dc=apache,dc=org"
- Active Directory administrative credentials with delegated control of “Create, delete, and manage user accounts” on the previously mentioned User container are on-hand.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords.

Note

If Centrify is installed and being used on any of the servers in the cluster, it is critical that you refer to Centrify's integration guide before attempting to enable Kerberos Security on your cluster. The documentation can be found in the Centrify Server Suite documentation library, with a direct link to the Hortonworks specific PDF here.

4.1.1.3. Use Manual Kerberos Setup

To perform Manual Kerberos Setup, you must prepare the following:

- Cluster hosts have network access to the KDC.
- Kerberos client utilities (such as kinit) have been installed on every cluster host.
- The Java Cryptography Extensions (JCE) have been setup on the Ambari Server host and all hosts in the cluster.
- The Service and Ambari Principals will be manually created in the KDC before completing this wizard.
- The keytabs for the Service and Ambari Principals will be manually created and distributed to cluster hosts before completing this wizard.

4.1.1.4. (Optional) Install a new MIT KDC

The following gives a very high level description of the KDC installation process. For more information see specific Operating Systems documentation, such as RHEL documentation or CentOS documentation.

Note

Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

Install the KDC Server

1. Install a new version of the KDC server:

   RHEL/CentOS/Oracle Linux

   ```
   yum install krb5-server krb5-libs krb5-workstation
   ```

   Ubuntu/Debian

   ```
   apt-get install krb5-kdc krb5-admin-server
   ```

2. Using a text editor, open the KDC server configuration file, located by default here:

   ```
   vi /etc/krb5.conf
   ```

3. Change the [realms] section of this file by replacing the default “kerberos.example.com” setting for the kdc and admin_server properties with the Fully Qualified Domain Name of the KDC server host. In the following example, “kerberos.example.com” has been replaced with “my.kdc.server”.

   ```
   [realms]
    EXAMPLE.COM = {
      kdc = my.kdc.server
      admin_server = my.kdc.server
   }
   ```

4. Some components such as HUE require renewable tickets. To configure MIT KDC to support them, ensure the following settings are specified in the `libdefaults` section of the `/etc/krb5.conf` file.

   ```
   renew_lifetime = 7d
   ```

Note

For Ubuntu/Debian, the setup of the default realm for the KDC and KDC Admin hostnames is performed during the KDC server install. You can re-run setup using dpkg-reconfigure krb5-kdc. Therefore, Steps 2 and 3 above are not needed for Ubuntu/Debian.

Create the Kerberos Database

- Use the utility kdb5_util to create the Kerberos database.

  RHEL/CentOS/Oracle Linux

  ```
  kdb5_util create -s
  ```

  Ubuntu/Debian

  ```
  krb5_newrealm
  ```

Start the KDC

- Start the KDC server and the KDC admin server.

  RHEL/CentOS/Oracle Linux 6

  ```
  /etc/rc.d/init.d/krb5kdc start
  ```

  ```
  /etc/rc.d/init.d/kadmin start
  ```

  RHEL/CentOS/Oracle Linux 7

  ```
  systemctl start krb5kdc
  ```

  ```
  systemctl start kadmin
  ```

  Ubuntu/Debian

  ```
  service krb5-kdc restart
  ```

  ```
  service krb5-admin-server restart
  ```

Important

When installing and managing your own MIT KDC, it is **very important** to **set up the KDC server to auto-start on boot**. For example:

RHEL/CentOS/Oracle Linux 6

```
chkconfig krb5kdc on
```

```
chkconfig kadmin on
```

RHEL/CentOS/Oracle Linux 7

```
systemctl enable krb5kdc

systemctl enable kadmin
```

Create a Kerberos Admin

Kerberos principals can be created either on the KDC machine itself or through the network, using an “admin” principal. The following instructions assume you are using the KDC machine and using the `kadmin.local` command line administration utility. Using `kadmin.local` on the KDC machine allows you to create principals without needing to create a separate "admin" principal before you start.

Note

You will need to provide these admin account credentials to Ambari when enabling Kerberos. This allows Ambari to connect to the KDC, create the cluster principals and generate the keytabs.

1. Create a KDC admin by creating an admin principal.

   ```
   kadmin.local -q "addprinc admin/admin"
   ```

2. Confirm that this admin principal has permissions in the KDC ACL. Using a text editor, open the KDC ACL file:

   RHEL/CentOS/Oracle Linux

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

   Ubuntu/Debian

   ```
   vi /etc/krb5kdc/kadm5.acl
   ```

3. Ensure that the KDC ACL file includes an entry so to allow the admin principal to administer the KDC for your specific realm. When using a realm that is different than EXAMPLE.COM, **be sure there is an entry for the realm you are using**. If not present, principal creation will fail. For example, for an admin/admin@HADOOP.COM principal, you should have an entry:

   ```
   */admin@HADOOP.COM *
   ```

4. After editing and saving the kadm5.acl file, you must restart the kadmin process.

   RHEL/CentOS/Oracle Linux 6

   ```
   /etc/rc.d/init.d/kadmin restart
   ```

   RHEL/CentOS/Oracle Linux 7

   ```
   systemctl restart kadmin
   ```

Ubuntu/Debian

```
service krb5-admin-server restart
```

4.1.2. Installing the JCE

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on the Ambari Server and on all hosts in the cluster.

Important

If you are using Oracle JDK, **you must** distribute and install the JCE **on all hosts** in the cluster, including the Ambari Server. **Be sure to restart Ambari Server after installing the JCE**. If you are using OpenJDK, some distributions of the OpenJDK come with unlimited strength JCE automatically and therefore, installation of JCE is not required.

4.1.2.1. Install the JCE

1. On the Ambari Server, obtain the JCE policy file appropriate for the JDK version in your cluster.
   - For Oracle JDK 1.8:

     http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
   - For Oracle JDK 1.7:

     http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html
2. Save the policy file archive in a temporary location.
3. On Ambari Server and on each host in the cluster, add the unlimited security policy JCE jars to `$JAVA_HOME/jre/lib/security/`.

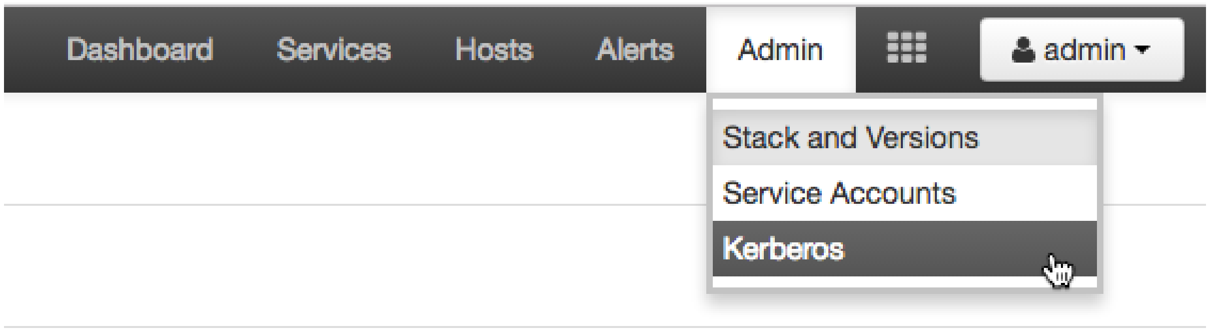
   For example, run the following to extract the policy jars into the JDK installed on your host:

   ```
   unzip -o -j -q jce_policy-8.zip -d /usr/jdk64/jdk1.8.0_60/jre/lib/security/
   ```
4. Restart Ambari Server.

4.1.3. Enabling Kerberos on Ambari

Once you have completed the prerequisites, you are ready to enable Kerberos for Ambari.

1. From the Ambari UI, click **Admin**, and select **Kerberos**.



2. Click **Enable Kerberos** to launch the **Enable Kerberos Wizard**.

3. From the **Get Started** screen, select the type of KDC you want to use.

4. Provide information about the KDC and admin account.

   a. In the **KDC** section, enter the following information:

      - In the **KDC Host** field, the IP address or FQDN for the KDC host. Optionally a port number may be included.

      - In the **Realm name** field, the default realm to use when creating service principals.

      - (Optional) In the **Domains** field, provide a list of patterns to use to map hosts in the cluster to the appropriate realm. For example, if your hosts have a common domain in their FQDN such as host1.hortonworks.local and host2.hortonworks.local, you would set this to:

        ```
        .hortonworks.local,hortonworks.local
        ```

   b. In the **Kadmin** section, enter the following information:

      - In the **Kadmin Host** field, the IP address or FQDN for the KDC administrative host. Optionally a port number may be included.

      - The **Admin principal** and **password** that will be used to create principals and keytabs.

      - (Optional) If you have configured Ambari for encrypted passwords, the **Save Admin Credentials** option will be enabled. With this option, you can have Ambari store the KDC Admin credentials to use when making cluster changes.

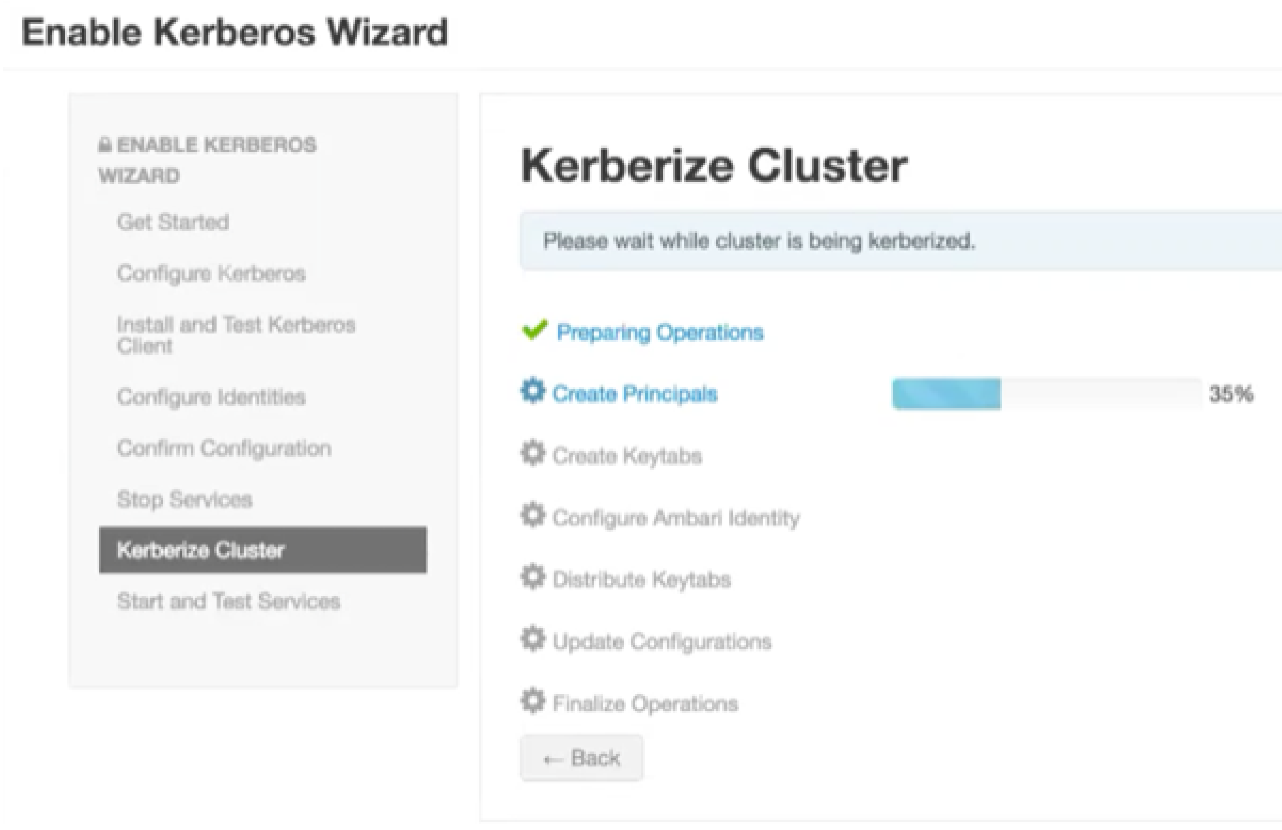
5. From the **Install and Test Kerberos Client** page, proceed with the install. Click **Next** when complete.

6. From the **Configure Identities** page, you can customize the Kerberos identities as needed, and proceed to kerberize the cluster.

   Be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "-${**cluster-name**}" from principal name string.

Click the **Advanced** tab to review the principals and keytabs for each service.

7. Confirm your configurations, and click next to proceed kerberizing your cluster.



Kerberize Cluster

Please wait while cluster is being kerberized.

✓ Preparing Operations
Create Principals 35%
Create Keytabs
Configure Ambari Identity
Distribute Keytabs
Update Configurations
Finalize Operations

4.2. Manually Enabling Kerberos

If you installed HDP manually, then you need to enable Kerberos manually.

To manually enable Kerberos, complete the following steps:

Note: These are manual instructions for Kerberizing Metron Storm topologies from Kafka to Kafka. This does not cover the Ambari MPack, sensor connections, or MAAS.

1. Stop all topologies - you will restart them again once Kerberos has been enabled.

```
for topology in bro snort enrichment indexing; do storm kill $topology; done
```

2. Set up Kerberos:

Note

If you copy/paste this full set of commands, the `kdb5_util` command will not run as expected. So, run the commands individually to ensure they all execute.

Be sure to set 'node1' to the correct host for your kdc.

```
yum -y install krb5-server krb5-libs krb5-workstation
sed -i 's/kerberos.example.com/node1/g' /etc/krb5.conf
cp /etc/krb5.conf /var/lib/ambari-server/resources/scripts
# This step takes a moment. It creates the kerberos database.
kdb5_util create -s
/etc/rc.d/init.d/krb5kdc start
/etc/rc.d/init.d/kadmin start
chkconfig krb5kdc on
chkconfig kadmin on
```

3. Set up the admin and metron user principals.

   You'll kinit as the metron user when running topologies. Make sure to remember the passwords.

   ```
   kadmin.local -q "addprinc admin/admin"
   kadmin.local -q "addprinc metron"
   ```

4. Create the metron user HDFS home directory:

   ```
   sudo -u hdfs hdfs dfs -mkdir /user/metron && \
   sudo -u hdfs hdfs dfs -chown metron:hdfs /user/metron && \
   sudo -u hdfs hdfs dfs -chmod 770 /user/metron
   ```

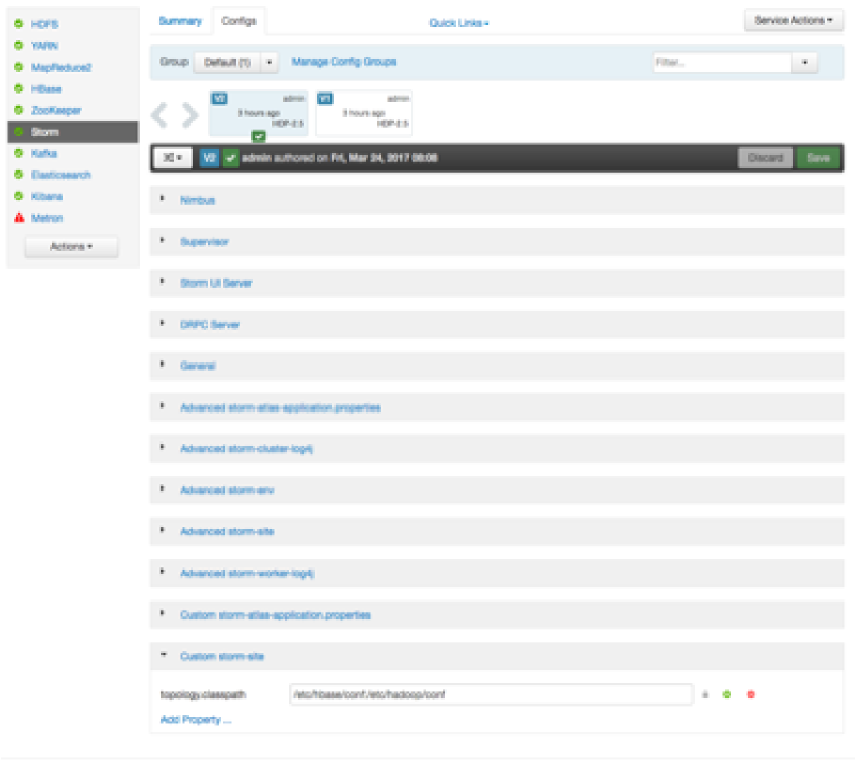
5. In Ambari, set up Storm to run with Kerberos and run worker jobs as the submitting user:

   a. Add the following properties to custom storm-site:

      ```
      topology.auto-credentials=['org.apache.storm.security.auth.kerberos.
      AutoTGT']
      nimbus.credential.renewers.classes=['org.apache.storm.security.auth.
      kerberos.AutoTGT']
      supervisor.run.worker.as.user=true
      ```

   b. In the Storm config section in Ambari, choose **Add Property** under custom storm-site:

Figure 4.1. Ambari Storm Site



V2 admin authored on Fri, Mar 24, 2017 08:08

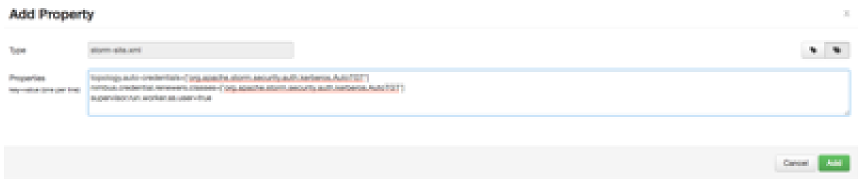
- Nimbus
- Supervisor
- Storm UI Server
- DRPC Server
- General
- Advanced storm-atlas-application.properties
- Advanced storm-cluster-log4j
- Advanced storm-env
- Advanced storm-site
- Advanced storm-worker-log4j
- Custom storm-atlas-application.properties
- Custom storm-site

topology.classpath	/etc/hbase/conf:/etc/hadoop/conf

Add Property ...

c. In the dialog window, choose the **bulk property add mode** toggle button and add the following values:

Figure 4.2. Add Property



Type: storm-site.xml

Properties (key=value (one per line)):

```
topology.auto-credentials=['org.apache.storm.security.auth.kerberos.AutoTGT']
nimbus.credential.renewers.classes=['org.apache.storm.security.auth.kerberos.AutoTGT']
supervisor.run.worker.as.user=true
```

6. Kerberize the cluster via Ambari.

   More detailed documentation can be found in Enabling Kerberos Security in the [HDP Security] guide.

   a. For this exercise, choose existing MIT KDC (this is what you set up and installed in the previous steps.)

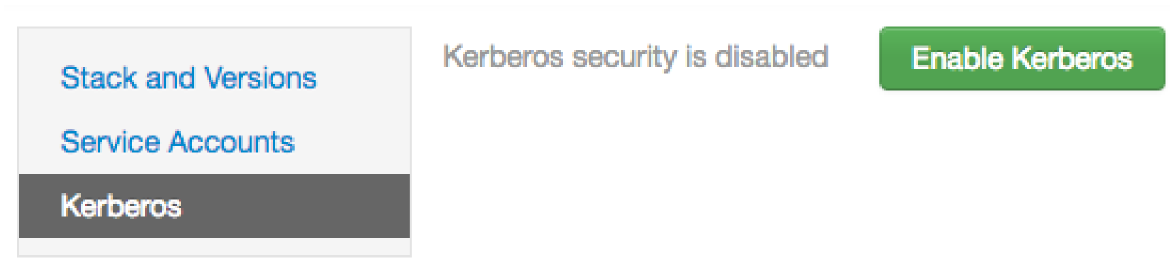
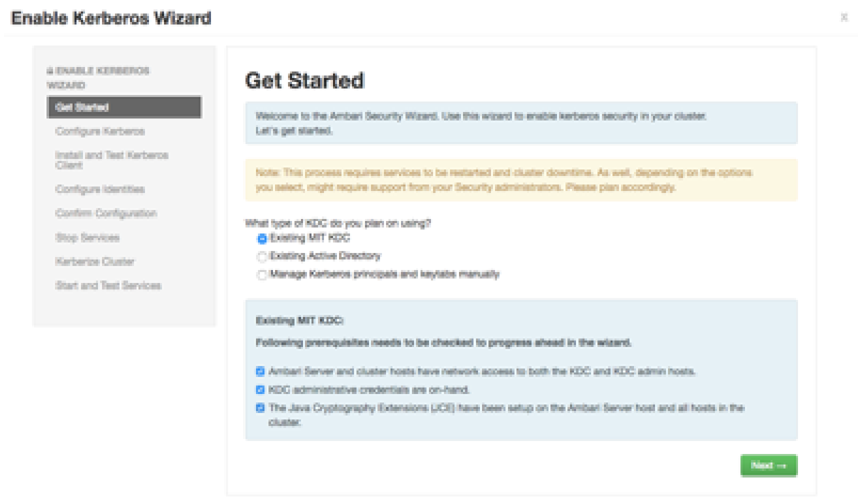


Figure 4.3. Enable Kerberos Wizard



Get Started

Welcome to the Ambari Security Wizard. Use this wizard to enable kerberos security in your cluster. Let's get started.

Note: This process requires services to be restarted and cluster downtime. As well, depending on the options you select, might require support from your Security administrators. Please plan accordingly.

What type of KDC do you plan on using?
- [✓ Existing MIT KDC]
- Existing Active Directory
- Manage Kerberos principals and keytabs manually

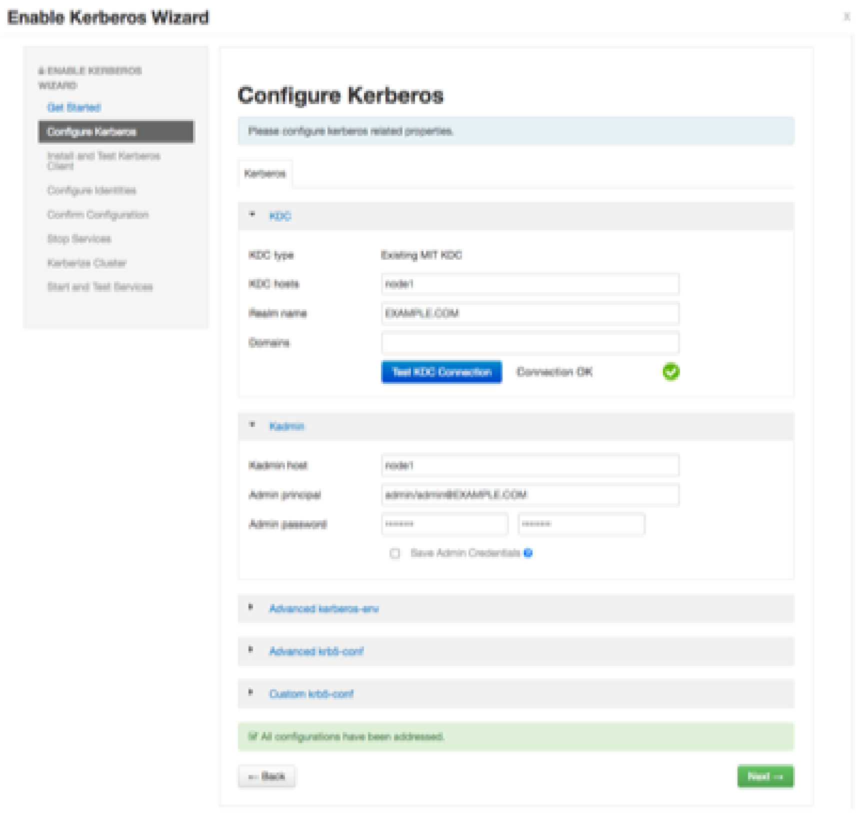
Existing MIT KDC:

Following prerequisites needs to be checked to progress ahead in the wizard.

- ✓ Ambari Server and cluster hosts have network access to both the KDC and KDC admin hosts.
- ✓ KDC administrative credentials are on-hand.
- ✓ The Java Cryptography Extensions (JCE) have been setup on the Ambari Server host and all hosts in the cluster.

b. Set up Kerberos configuration. Realm is EXAMPLE.COM. The admin principal will end up as admin/admin@EXAMPLE.COM when testing the KDC. Use the password you entered during the step to add the admin principal.

Figure 4.4. Enable Kerberos Wizard



Enable Kerberos Wizard

Configure Kerberos

Please configure kerberos related properties.

KDC

KDC type	Existing MIT KDC
KDC hosts	node1
Realm name	EXAMPLE.COM
Domains	

Connection OK

Kadmin

Kadmin host	node1
Admin principal	admin/admin@EXAMPLE.COM
Admin password	****** ******

All configurations have been addressed.

c. Click through to **Start and Test Services**.

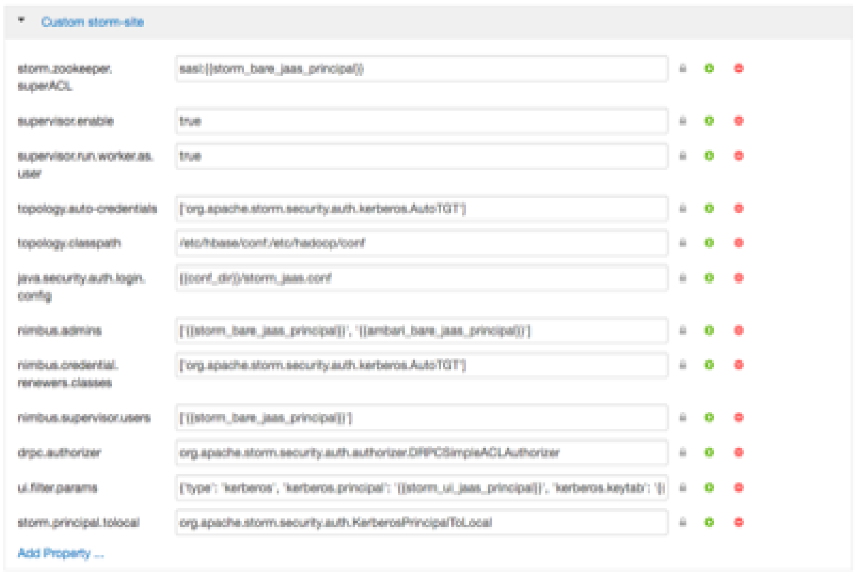
   Let the cluster spin up, but don't worry about starting up Metron via Ambari. You will run the parsers manually against the rest of the Hadoop cluster Kerberized.

   The wizard will fail at starting Metron, but it is okay.

d. Click **continue**.

   When you’re finished, the custom storm-site should look similar to the following:

Figure 4.5. Final Custom Storm-site



storm.zookeeper.superACL	sasl:{{storm_bare_jaas_principal}}
supervisor.enable	true
supervisor.run.worker.as.user	true
topology.auto-credentials	['org.apache.storm.security.auth.kerberos.AutoTGT']
topology.classpath	/etc/hbase/conf:/etc/hadoop/conf
java.security.auth.login.config	{{conf_dir}}/storm_jaas.conf
nimbus.admins	['{{storm_bare_jaas_principal}}', '{{ambari_bare_jaas_principal}}']
nimbus.credential.renewers.classes	['org.apache.storm.security.auth.kerberos.AutoTGT']
nimbus.supervisor.users	['{{storm_bare_jaas_principal}}']
drpc.authorizer	org.apache.storm.security.auth.authorizer.DRPCSimpleACLAuthorizer
ui.filter.params	{'type': 'kerberos', 'kerberos.principal': '{{storm_ui_jaas_principal}}', 'kerberos.keytab': '{{
storm.principal.tolocal	org.apache.storm.security.auth.KerberosPrincipalToLocal

7. Set up Metron keytab:

```
kadmin.local -q "ktadd -k metron.headless.keytab metron@EXAMPLE.COM" && \
cp metron.headless.keytab /etc/security/keytabs && \
chown metron:hadoop /etc/security/keytabs/metron.headless.keytab && \
chmod 440 /etc/security/keytabs/metron.headless.keytab
```

8. Kinit with the metron user:

```
kinit -kt /etc/security/keytabs/metron.headless.keytab metron@EXAMPLE.COM
```

9. First create any additional Kafka topics you will need.

   You need to create the topics before adding the required ACLs. The current full dev installation will deploy bro, snort, enrichments, and indexing only. For example:

```
${HDP_HOME}/kafka-broker/bin/kafka-topics.sh --zookeeper ${ZOOKEEPER}:2181
 --create --topic yaf --partitions 1 --replication-factor 1
```

10. Set up Kafka ACLs for the topics:

```
export KERB_USER=metron;
for topic in bro enrichments indexing snort; do
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --topic
 ${topic};
done;
```

11. Set up Kafka ACLs for the consumer groups:

```
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --group
 bro_parser;
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --group
 snort_parser;
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --group
 yaf_parser;
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --group
 enrichments;
${HDP_HOME}/kafka-broker/bin/kafka-acls.sh --authorizer kafka.security.
auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --group
 indexing;
```

12. Add metron user to the Kafka cluster ACL:

```
/usr/hdp/current/kafka-broker/bin/kafka-acls.sh --authorizer kafka.
security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=
${ZOOKEEPER}:2181 --add --allow-principal User:${KERB_USER} --cluster kafka-
cluster
```

13. You also need to grant permissions to the HBase tables. Kinit as the hbase user and add ACLs for metron:

```
kinit -kt /etc/security/keytabs/hbase.headless.keytab hbase-
metron_cluster@EXAMPLE.COM
echo "grant 'metron', 'RW', 'threatintel'" | hbase shell
echo "grant 'metron', 'RW', 'enrichment'" | hbase shell
```

14. Create a ".storm" directory in the metron user's home directory and switch to that directory.

```
su metron && cd ~/
mkdir .storm
cd .storm
```

15. Create a custom client jaas file.

    This should look identical to the Storm client jaas file located in `/etc/storm/conf/client_jaas.conf` except for the addition of a Client stanza. The Client stanza is used for ZooKeeper. All quotes and semicolons are necessary.

```
[metron@node1 .storm]$ cat client_jaas.conf
StormClient {
 com.sun.security.auth.module.Krb5LoginModule required
 useTicketCache=true
 renewTicket=true
 serviceName="nimbus";
};
Client {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="/etc/security/keytabs/metron.headless.keytab"
 storeKey=true
 useTicketCache=false
 serviceName="zookeeper"
 principal="metron@EXAMPLE.COM";
};
KafkaClient {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="/etc/security/keytabs/metron.headless.keytab"
 storeKey=true
 useTicketCache=false
 serviceName="kafka"
 principal="metron@EXAMPLE.COM";
};
```

16. Create a storm.yaml with jaas file info. Set the array of nimbus hosts accordingly.

```
[metron@node1 .storm]$ cat storm.yaml
nimbus.seeds : ['node1']
java.security.auth.login.config : '/home/metron/.storm/client_jaas.conf'
storm.thrift.transport : 'org.apache.storm.security.auth.kerberos.
KerberosSaslTransportPlugin'
```

17. Create an auxiliary storm configuration json file in the metron user's home directory.

Note

The login config option in the file points to our custom `client_jaas.conf`.

```
cd /home/metron
[metron@node1 ~]$ cat storm-config.json
{
"topology.worker.childopts" : "-Djava.security.auth.login.config=/home/
metron/.storm/client_jaas.conf"
}
```

18. Set up enrichment and indexing:

    a. Modify `enrichment.properties`:

    ```
    ${METRON_HOME}/config/enrichment.propertieskafka.security.protocol=
    PLAINTEXTSASL topology.worker.childopts=-Djava.security.auth.login.
    config=/home/metron/.storm/client_jaas.conf
    ```

    b. Modify `elasticsearch.properties`:

    ```
    ${METRON_HOME}/config/elasticsearch.propertieskafka.security.protocol=
    PLAINTEXTSASL topology.worker.childopts=-Djava.security.auth.login.
    config=/home/metron/.storm/client_jaas.conf
    ```

19. Kinit with the metron user again:

```
kinit -kt /etc/security/keytabs/metron.headless.keytab metron@EXAMPLE.COM
```

20. Restart the parser topologies.

Be sure to pass in the new parameter, “-ksp” or “–kafka_security_protocol.” Run this from the metron home directory.

```
for parser in bro snort; do ${METRON_HOME}/bin/start_parser_topology.sh -z
 ${ZOOKEEPER}:2181 -s ${parser} -ksp SASL_PLAINTEXT -e storm-config.json;
 done
```

21. Now restart the enrichment and indexing topologies:

```
${METRON_HOME}/bin/start_enrichment_topology.sh
${METRON_HOME}/bin/start_elasticsearch_topology.sh
```

22. Push some sample data to one of the parser topics.

    For example, for yaf we took raw data from https://github.com/mmiklavc/incubator-metron/blob/994ba438a3104d2b7431bee79d2fce0257a96fec/metron-platform/metron-integration-test/src/main/sample/data/yaf/raw/YafExampleOutput

```
cat sample-yaf.txt | ${HDP_HOME}/kafka-broker/bin/kafka-console-producer.sh
 --broker-list ${BROKERLIST}:6667 --security-protocol SASL_PLAINTEXT --topic
 yaf
```

23. Wait a few moments for data to flow through the system and then check for data in the Elasticsearch indexes. Replace yaf with whichever parser type you’ve chosen.

```
curl -XGET "${ZOOKEEPER}:9200/yaf*/_search"
curl -XGET "${ZOOKEEPER}:9200/yaf*/_count"
```

    You should have data flowing from the parsers all the way through to the indexes.