Metron 0.4.1 with HDP 2.5 bare-metal install on Centos 7 with MariaDB for Metron REST

Written by Dima Kovalyov <dimdroll@gmail.com>.

Contributed by Laurens Vets <laurens@daemon.be>.

Version 0.4.2 - January 2018

Introduction

We will be installing Metron 0.4.1 with HDP 2.5 on CentOS 7. We will also install MariaDB as a database for Metron REST. Additionally, we'll also install Apache NiFi.
I installed Metron in a test environment with 4 VMs to try it out as well as a single node. I'll try to write this guide so that the necessary steps can easily be adapted for other environments.

Environment

- Single node: 8 CPUs, 32GB RAM.
- Multiple nodes:
  - 4 VMs, 4 CPUs per VM and 16 GB RAM per VM.
  - Hosts:
    10.10.10.1 node1
    10.10.10.2 node2
    10.10.10.3 node3
    10.10.10.4 node4

Prerequisites

- CentOS 7
- Add the epel repository and update your system:

```
yum install epel-release -y
yum update -y
```

- Set up passwordless SSH between our nodes. If passwordless ssh has not yet been set up within the cluster, then in main node generate key:

```
cat /dev/zero | ssh-keygen -q -N "" 2>/dev/null
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
chmod 400 .ssh/authorized_keys
```

  If you're not installing on a single node, add this newly generated key to all the slave nodes:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub <replace_with_node_ip>
```

Side note: You might have to adapt your sshd_config file and add "PermitRootLogin yes" amongst other parameters if you want passwordless root access, but that's outside the scope of this document.

- Increase limits for ElasticSearch and Storm on nodes where you will be installing them (if you don't know, increase it everywhere):

```
echo -e "elasticsearch - memlock unlimited\nstorm - nproc 257597" >> /etc/security/limits.conf
```

- Adjust limits to secure level (link)

```
ulimit -n 32768
ulimit -u 65536
echo -e "* - nofile 32768\n* - nproc 65536" >> /etc/security/limits.conf
```

- Disable IPv6, leaving it enabled may force service to bind to IPv6 addresses only and thus resulting in inability to connect to it (source link):

```
sysctl -w net.ipv6.conf.all.disable_ipv6=1
sysctl -w net.ipv6.conf.default.disable_ipv6=1
echo -e "\n# Disable IPv6\nnet.ipv6.conf.all.disable_ipv6 = 1\nnet.ipv6.conf.default.disable_ipv6 = 1"
>> /etc/sysctl.conf
```

- Disable Transparent Hugepage. Add "transparent_hugepage=never" to the end of the kernel line in "/etc/default/grub" and reboot. (Ambari demands it, do we need to comply?)

```
# Change the line:
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=cl/root rd.lvm.lv=cl/swap rhgb quiet"
# To:
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=cl/root rd.lvm.lv=cl/swap rhgb quiet
transparent_hugepage=never"
# Afterwards, run:
grub2-mkconfig -o /boot/grub2/grub.cfg
```

  After reboot check that changes were applied (make sure that word "never" is selected in square-brackets):

```
cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
```

  Alternatively, if you do not want to mess with kernel parameters, you can create a new systemd service which disables this on each boot. Create the file "/etc/systemd/system/disable-thp.service" with the following content:

```
[Unit]
Description=Disable Transparent Huge Pages (THP)

[Service]
Type=simple
ExecStart=/bin/sh -c "echo 'never' > /sys/kernel/mm/transparent_hugepage/enabled && echo 'never' > /sys
/kernel/mm/transparent_hugepage/defrag"

[Install]
WantedBy=multi-user.target
```

  Restart systemd, start the new service and make sure the new service runs at startup:

```
# systemctl daemon-reload
# systemctl start disable-thp
# systemctl enable disable-thp
```

- Disable SELinux (is a must to install Ambari and build Metron:

```
setenforce 0
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

Metron install pre-preparation

- On all nodes Install pre-requisites for Ambari:

```
yum install git wget curl rpm tar unzip scp bzip2 wget createrepo yum-utils ntp python-pip psutils
python-psutil ntp libffi-devel gcc openssl-devel -y
pip install --upgrade pip
pip install requests
```

- On Metron node install java 1.8 (if you don't know which it is, install it everywhere):

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

- Set path to Java 8 if it does not exist:

```
export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s_/jre/bin/java__")
```

- Save export for future reboots:

```
echo 'export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s_/jre/bin/java__")' > /etc/profile.d/java_18.
sh
chmod +x /etc/profile.d/java_18.sh
source /etc/profile.d/java_18.sh
```

- Download and install Maven 3.3.9:

```
wget https://archive.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
tar -zxf apache-maven-3.3.9-bin.tar.gz
mv apache-maven-3.3.9 /opt
PATH=/opt/apache-maven-3.3.9/bin:$PATH
echo 'export PATH=/opt/apache-maven-3.3.9/bin:$PATH' > /etc/profile.d/maven.sh
chmod +x /etc/profile.d/maven.sh
```

- Check whether Maven works:

```
source /etc/profile.d/maven.sh
mvn -V
```

  You should see something similar to:

```
# mvn -V
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T08:41:47-08:00)
Maven home: /opt/apache-maven-3.3.9
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-514.16.1.el7.x86_64", arch: "amd64", family: "unix"
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.083 s
[INFO] Finished at: 2017-06-06T09:59:03-07:00
[INFO] Final Memory: 13M/479M
[INFO] ------------------------------------------------------------------------
[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal
in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:
<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources,
generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-
sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test,
prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify,
install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoGoalSpecifiedException
```

- On Ambari node install and enable docker (we will need it to build Metron mpack for Ambari):

```
yum install docker-io -y
systemctl start docker
```

- Also on your build box, install npm. This is needed to build metron-config, part of the UI.

```
yum install npm -y
```

- Add "127.0.0.1 localhost" to /etc/hosts if it does not there already
- Install the database we will use for Metron REST on Master node:

```
yum install mariadb-server -y
```

Install JAVA MySQL connector on all nodes:

```
yum install mysql-connector-java -y
```

Configure database for Metron REST

If you haven't run `mysql_secure_installation` after the database installation, do that first:

```
systemctl start mariadb
systemctl enable mariadb
systemctl status mariadb
mysql_secure_installation
```

Should produce following output:

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

Set root password? [Y/n]
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] n
 ... skipping.

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]
 ... Success!
By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
 - Dropping test database...
ERROR 1008 (HY000) at line 1: Can't drop database 'test'; database doesn't exist
 ... Failed!  Not critical, keep moving...
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
 ... Success!

All done!  If you've completed all of the above steps, your MySQL
installation should now be secure.

Thanks for using MySQL!


Cleaning up...
```

Build Metron code

Now we are going to start building Metron code. For advanced users: there seems to be a new way of building mpack and rpms

- Clone Metron repo and switch to 0.4.1 release:

```
git clone https://github.com/apache/metron
cd metron
git checkout Metron_0.4.1
```

- Build Metron with HDP 2.5 profile:

```
cd metron
mvn clean package -DskipTests -T 2C -P HDP-2.5.0.0,mpack
cd metron-deployment/packaging/docker/rpm-docker
mvn clean install -DskipTests -PHDP-2.5.0.0
```

If for some reason, the rpm-docker fails with the message "/bin/bash: ./build.sh: Permission denied", try disabling selinux:

```
setenforce 0
```

And run mvn commands again.

- On all nodes, create a localrepo directory and copy the RPMs from Ambari node there:

```
mkdir /localrepo
cp -rp /root/metron/metron-deployment/packaging/docker/rpm-docker/RPMS/noarch/* /localrepo/
createrepo /localrepo
```

- If you're doing a multi node install, also copy the packages to the other nodes:

```
ssh root@node2 mkdir /localrepo
scp /localrepo/*rpm root@node2:/localrepo/
ssh root@node2 createrepo /localrepo
```

Make sure to do the above on each node.

- Fetch & create logrotate script for Hadoop Services:

```
wget -O /etc/logrotate.d/metron-ambari https://raw.githubusercontent.com/apache/metron/master/metron-
deployment/roles/ambari_common/templates/metron-hadoop-logrotate.yml
sed -i 's/^  {{ hadoop_logrotate_frequency }}.*$/  daily/' /etc/logrotate.d/metron-ambari
sed -i 's/^  rotate {{ hadoop_logrotate_retention }}.*$/  rotate 30/' /etc/logrotate.d/metron-ambari
chmod 0644 /etc/logrotate.d/metron-ambari
```

Ambari 2.4 with HDP 2.5 install

Inspired by: [https://docs.hortonworks.com/HDPDocuments/Ambari-2.4.3.0/bk_ambari-installation/content/ch_Getting_Ready.html]

- Enable time sync, disable firewall and SElinux on every node (I know, but for the sake of simplicity, quickness & testing, I've disabled selinux):

```
systemctl enable ntpd
systemctl start ntpd
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -F
iptables -t mangle -F
iptables -F
iptables -X
iptables-save > /etc/sysconfig/iptables
systemctl stop firewalld
systemctl disable firewalld
setenforce 0
```

Also, if you are using CentOS 7 and Python 2.7.5 and above you will encounter an error during ambari agent install in Ambari UI:

```
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:579)
```

To fix it disable cert check in Python like this (reference link):

```
sed -i 's/verify=platform_default/verify=disable/' /etc/python/cert-verification.cfg
```

- Make sure each node can resolve every other node's hostname or add hostname of each node to `/etc/hosts` on every node. For example add following lines in /etc/hosts of each node:

```
10.10.10.1 node1
10.10.10.2 node2
10.10.10.3 node3
10.10.10.4 node4
```

Where 10.10.10.1, 10.10.10.2, 10.10.10.3 and 10.10.10.4 are the IP addresses of your nodes and "node1", "node2", "node3" and "node4" are their respective hostnames.

- On main node download and setup Ambari repo (you may replace the "2.4.3.0" with a newer Ambari version number):

```
wget -nv http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.4.3.0/ambari.repo -O /etc/yum.
repos.d/ambari.repo
```

- Check that it was added:

```
yum repolist | grep ambari
# Updates-ambari-2.4.3.0   ambari-2.4.3.0 - Updates                              12
```

- Install and setup Ambari server:

```
yum install ambari-server -y
ambari-server setup -s
```

- Add Metron service to Ambari by running mpack command (make sure to specify correct path to mpack in --mpack=):

```
ambari-server install-mpack --mpack=/root/metron/metron-deployment/packaging/ambari/metron-mpack/target
/metron_mpack-0.4.1.0.tar.gz --verbose
```

- Start Ambari:

```
ambari-server start
```

- Access the Ambari UI by going to the following URL in a web browser: `http://<replace_with_master_node_ip>:8080/`. You can use admin/admin as username/password. Start the Install Wizard.

Get Started page: Enter any desired cluster name.

Select Version: Make sure "Public Repository" is checked. You should also see the `/localrepo` directory listed.

Install Options: Specify hostnames of your nodes where Ambari cluster should be installed (all the ones you have specified in /etc/hosts) in "Target Hosts". Copy content of the main node private key (/root/.ssh/id_rsa) in "Host Registration Information". If you receive the warning "The following hostnames are not valid FQDNs", ignore it and click OK.

Choose Services: Select following Services:

HDFS
YARN + MapReduce2
Tez
HBase
Pig
Zookeeper
Storm
Flume
Ambari Metrics
Kafka
Elasticsearch
Kibana
Metron
Slider

OpenTAXII

Pycapa

Zeppelin Notebook

Hive

Assign Masters: Assign "Kafka Broker" on all nodes. Make sure move following components on one common node (Taken from previous guide, is this still necessary?):

Storm UI Server
Metron Indexing
MySQL Server
Kibana Server
Elasticsearch Master
Metron Parsers
Metron Enrichment

Assign Slaves and Clients: select All for:

DataNode
NodeManager
RegionServer
Supervisor
Client

Customize Services: Following is a list of services that need to be configured:

- Set the "NameNode Java heap size" (namenode_heapsize) from the default 1024 MB to at least 4096 MB under HDFS -> Configs.
- For ElasticSearch:
  - Set "zen_discovery_ping_unicast_hosts" to the IP of the node where you assigned ElasticSearch Master on the Assign Master tab.
  - Under "Advanced elastic-site": Change "network_host" to "0.0.0.0". Do not do this if your Metron is exposed to the public internet! Is "[_local_, _site_]" now.
- Kibana:
  - Set "kibana_es_url" to `http://<replace_with_elasticsearch_master_hostname>:9200`. "replace_with_elasticsearch_master_hostname" is the IP of the node where you assigned ElasticSearch Master on the Assign Master tab.
  - Change "kibana_default_application" to "dashboard/Metron-Dashboard"
- Metron: Set "Elasticsearch Hosts" to the IP of the node where you assigned ElasticSearch Master on the Assign Master tab.
- Storm: You might have to increase the number of "supervisor.slots.ports" from the default "[6700, 6701]" to "[6700, 6701, 6702, 6703, 6704]" if you're only installing a single node.
- For metron REST use:

```
Metron JDBC client path: /usr/share/java/mysql-connector-java.jar
Metron JDBC Driver: com.mysql.jdbc.Driver
Metron JDBC password: <DB PASSWORD>
Metron JDBC platform: mysql
Metron JDBC URL: jdbc:mysql://127.0.0.1:3306/<DB NAME>
Metron JDBC username: <DB USERNAME>
```

- Set rest of the configuration values to recommended by Ambari or the ones you desire (like DB passwords) and perform install. In a 3 node cluster, I ended up with:

node1	node2	node3
DataNode	DataNode	DataNode
Elasticsearch Master	App Timeline Server	Elasticsearch Data Node
HBase Client	DRPC Server	Flume
HBase Master	HBase Client	HBase Client
RegionServer	RegionServer	RegionServer
HCat Client	HCat Client	HCat Client
HDFS Client	HDFS Client	HDFS Client
Hive Client	Hive Client	Hive Client
Kafka Broker	History Server	Kafka Broker
Kibana Server	Hive Metastore	MapReduce2 Client
MapReduce2 Client	HiveServer2	Metrics Collector
Grafana	Kafka Broker	Metrics Monitor
Metrics Monitor	MapReduce2 Client	Metron Client
Metron Client	Metrics Monitor	NodeManager
Metron Enrichment	Metron Client	Pig Client
Metron Indexing	MySQL Server	Slider Client
Metron Parsers	Nimbus	Spark Client
Metron REST	NodeManager	Supervisor
NameNode	Pig Client	Tez Client
NodeManager	ResourceManager	YARN Client
Pig Client	NameNode	ZooKeeper Client
Slider Client	Slider Client	ZooKeeper Server
Spark Client	Spark Client	
Spark History Server	Supervisor	
Storm UI Server	Tez Client	
Supervisor	WebHCat Server	
Tez Client	YARN Client	
YARN Client	ZooKeeper Client	
Zeppelin Notebook	ZooKeeper Server	
ZooKeeper Client		
ZooKeeper Server		

- Install everything. Metron REST will probably not work as we still need to add a user and the database to MariaDB.
  At this point, make sure that all the services are up. You might have to manually start a few.
- Configure a user for Metron REST in MySQL. On the node where you installed the Metron REST UI, do:

```
# mysql -u root -p
CREATE USER '<DB USERNAME>'@'localhost' IDENTIFIED BY '<DB PASSWORD>';
CREATE DATABASE IF NOT EXISTS <DB NAME>;
GRANT ALL PRIVILEGES ON <DB NAME>.* TO '<DB USERNAME>'@'localhost';
```

For example:

```
# mysql -u root -p
> CREATE USER 'metron'@'localhost' IDENTIFIED BY 'metron';
> CREATE DATABASE IF NOT EXISTS metronrest;
> GRANT ALL PRIVILEGES ON metronrest.* TO 'metron'@'localhost';
> quit
Bye
#
```

- There's 1 last step we need to do before the metron REST service will run. Due to systemd in Centos 7, doing `service metron-rest start <PASSWORD>` no longer works. Instead we have to edit the configuration file "`/etc/rc.d/init.d/metron-rest`". In this file, change `METRON_JDBC_PASSWORD="$2"` to `METRON_JDBC_PASSWORD="<DB PASSWORD>"` and restart the metron-rest service via the Ambari interface. Make sure that the Metron REST UI is started when moving to the following item.
- Add the Metron REST username and password to the metronrest database:

```
# mysql -u <DB USERNAME> -p
> use <DB NAME>;
> insert into users (username, password, enabled) values ('<USERNAME>','<PASSWORD>',1);
> insert into authorities (username, authority) values ('<USERNAME>', 'ROLE_USER');
> quit
Bye
#
```

- For example, to use the username 'metron' with password 'metron', do the following:

```
# mysql -u metron -p
> use metronrest;
> insert into users (username, password, enabled) values ('metron','metron',1);
> insert into authorities (username, authority) values ('metron', 'ROLE_USER');
> quit
Bye
#
```

Make sure that all the services are up.

- Install metron_pcapservice:

```
# cp /root/metron/metron-platform/metron-api/target/metron-api-0.4.1.jar /usr/metron/0.4.1/lib/
# wget -O /etc/init.d/pcapservice https://raw.githubusercontent.com/apache/metron/master/metron-deployment/roles/metron_pcapservice/templates/pcapservice
# sed -i 's/{{ pcapservice_jar_dst }}/\/usr\/metron\/0.4.1\/lib\/metron-api-0.4.1.jar/' /etc/init.d/pcapservice
# sed -i 's/{{ pcapservice_port }}/8081/' /etc/init.d/pcapservice
# sed -i 's/{{ query_hdfs_path }}/\/tmp/' /etc/init.d/pcapservice
# sed -i 's/{{ pcap_hdfs_path }}/\/apps\/metron\/pcap/' /etc/init.d/pcapservice
# chmod 755 /etc/init.d/pcapservice
# wget -O /etc/logrotate.d/metron-pcapservice https://raw.githubusercontent.com/apache/metron/master/metron-deployment/roles/metron_pcapservice/templates/metron-pcapservice-logrotate.yml
# sed -i 's/^  {{ metron_pcapservice_logrotate_frequency }}.*$/  daily/' /etc/logrotate.d/metron-pcapservice
# sed -i 's/^  rotate {{ metron_pcapservice_logrotate_retention }}.*$/  rotate 30/' /etc/logrotate.d/metron-pcapservice
# chmod 644 /etc/logrotate.d/metron-pcapservice
```

- Install tap interface:

```
# ip tuntap add tap0 mode tap
```

- Bring up tap0 on 10.0.0.100:

```
# ifconfig tap0 10.0.0.100 up
# ip link set tap0 promisc on
```

- Install librdkafka:

```
# yum install cmake make gcc gcc-c++ flex bison libpcap libpcap-devel openssl-devel python-devel
swig zlib-devel perlcyrus-sasl cyrus-sasl-devel cyrus-sasl-gssapi -y
# cd /tmp
# wget -O /tmp/librdkafka-0.9.4.tar.gz https://github.com/edenhill/librdkafka/archive/v0.9.4.tar.
gz
# /bin/gtar --extract -C /tmp -z -f /tmp/librdkafka-0.9.4.tar.gz
# cd /tmp/librdkafka-0.9.4
# ./configure --prefix=/usr/local --enable-sasl
# make
# make install
```

- Install pycapa (I don't think we need the virtualenv anymore in CentOS 7, needs some further investigation):

```
# yum install @Development python-virtualenv libpcap-devel libselinux-python -y
# mkdir /usr/local/pycapa
# cd /usr/local/pycapa
# virtualenv pycapa-venv
# cp -r /root/metron/metron-sensors/pycapa/. /usr/local/pycapa/.
(# /usr/local/pycapa/pycapa-venv/bin/pip install -r requirements.txt)
# cd /usr/local/pycapa
# source pycapa-venv/bin/activate
# pip install -r requirements.txt
# pip install --upgrade pip
# /usr/local/pycapa/pycapa-venv/bin/python setup.py install
# wget -O /etc/init.d/pycapa https://raw.githubusercontent.com/apache/metron/master/metron-
deployment/roles/pycapa/templates/pycapa
# sed -i 's/{{ pycapa_log }}/\/var\/log\/pycapa.log/' /etc/init.d/pycapa
# sed -i 's/{{ pycapa_home }}/\/usr\/local\/pycapa/' /etc/init.d/pycapa
# sed -i 's/{{ python27_home }}/\/opt\/rh\/python27\/root/' /etc/init.d/pycapa
# sed -i 's/{{ pycapa_bin }}/\/usr\/local\/pycapa\/pycapa-venv\/bin/' /etc/init.d/pycapa
# sed -i 's/--kafka {{ kafka_broker_url }}/--kafka-broker <IP:6667>/' /etc/init.d/pycapa
# sed -i 's/--topic {{ pycapa_topic }}/--kafka-topic pcap/' /etc/init.d/pycapa
# sed -i 's/{{ pycapa_sniff_interface }}/tap0/' /etc/init.d/pycapa
# sed -i 's/export LD_LIBRARY_PATH=\/opt\/rh\/python27\/root\/usr\/lib64/export LD_LIBRARY_PATH=\
/usr\/local\/lib/' /etc/init.d/pycapa
# chmod 755 /etc/init.d/pycapa
# yum install @Development libdnet-devel rpm-build libpcap libpcap-devel pcre pcre-devel zlib
zlib-devel glib2-devel -y
# yum install kafka -y
```

- Install bro:

```
# wget -O /tmp/bro-2.4.1.tar.gz https://www.bro.org/downloads/release/bro-2.4.1.tar.gz
# /bin/gtar --extract -C /tmp -z -f /tmp/bro-2.4.1.tar.gz
# cd /tmp/bro-2.4.1
# ./configure --prefix=/usr/local/bro
# make -j4
# make install
```

- Configure bro:

```
# sed -i 's/interface=eth0/interface=tap0/' /usr/local/bro/etc/node.cfg
# /usr/local/bro/bin/broctl install
```

- Edit crontab with `# crontab -e` and add:

```
0-59/5    *    *    *    *    /usr/local/bro/bin/broctl cron
0-59/5    *    *    *    *    rm -rf /usr/local/bro/spool/tmp/*
```

- bro-kafka:

```
# cp -r /root/metron/metron-sensors/bro-plugin-kafka /tmp
# cd /tmp/bro-plugin-kafka
# rm -rf build/
# ./configure --bro-dist=/tmp/bro-2.4.1 --install-root=/usr/local/bro/lib/bro/plugins/ --with-
librdkafka=/usr/local
# make -j4
# make install
```

- Configure bro-kafka plugin:

```
# cat << EOF >> /usr/local/bro/share/bro/site/local.bro
@load Bro/Kafka/logs-to-kafka.bro
redef Kafka::logs_to_send = set(HTTP::LOG, DNS::LOG);
redef Kafka::topic_name = "bro";
redef Kafka::tag_json = T;
redef Kafka::kafka_conf = table( ["metadata.broker.list"] = "<KAFKA_BROKER_IP>:6667" );
EOF
# /usr/local/bro/bin/broctl deploy
# ip link set tap0 promisc on
```

- Install daq:

```
# wget -O /tmp/daq-2.0.6-1.src.rpm https://snort.org/downloads/snort/daq-2.0.6-1.src.rpm
# cd /tmp
# rpmbuild --rebuild daq-2.0.6-1.src.rpm
```

This last command creates the files /root/rpmbuild/RPMS/x86_64/daq-2.0.6-1.x86_64.rpm & /root/rpmbuild/RPMS/x86_64/daq-debuginfo-2.0.6-1.x86_64.rpm. We only need to install the first rpm.

```
# yum install /root/rpmbuild/RPMS/x86_64/daq-2.0.6-1.x86_64.rpm -y
```

- Install snort:

```
# wget -O /tmp/snort-2.9.8.0-1.src.rpm https://snort.org/downloads/archive/snort/snort-2.9.8.0-1.
src.rpm
# cd /tmp
# rpmbuild --rebuild snort-2.9.8.0-1.src.rpm
```

This last command creates the files /root/rpmbuild/RPMS/x86_64/snort-2.9.8.0-1.x86_64.rpm & /root/rpmbuild/RPMS/x86_64/snort-debuginfo-2.9.8.0-1.x86_64.rpm. We only need to install the first rpm.

```
# yum install /root/rpmbuild/RPMS/x86_64/snort-2.9.8.0-1.x86_64.rpm -y
# wget -O /tmp/community-rules.tar.gz https://www.snort.org/downloads/community/community-rules.
tar.gz
# /bin/gtar --extract -C /tmp -z -f /tmp/community-rules.tar.gz
# cp -r community-rules/community.rules /etc/snort/rules
# touch /etc/snort/rules/white_list.rules
# touch /etc/snort/rules/black_list.rules
# touch /var/log/snort/alerts
# chown -R snort:snort /etc/snort
# sed -i 's/^# alert/alert/' /etc/snort/rules/community.rules
# wget -O /tmp/snort.conf https://github.com/apache/metron/raw/master/metron-deployment/roles
/snort/files/snort.conf
# cp snort.conf /etc/snort/snort.conf
# sed -i 's/^ipvar HOME_NET.*$/ipvar HOME_NET any/' /etc/snort/snort.conf
# echo "output alert_csv: /var/log/snort/alert.csv default" >> /etc/snort/snort.conf
# sed -i 's/^ALERTMODE=.*$/ALERTMODE=/' /etc/sysconfig/snort
# sed -i 's/^NO_PACKET_LOG=.*$/NO_PACKET_LOG=1/' /etc/sysconfig/snort
# sed -i 's/^INTERFACE=.*$/INTERFACE=tap0/' /etc/sysconfig/snort
# mkdir /opt/snort-producer
# chmod 755 /opt/snort-producer
# wget -O /opt/snort-producer/start-snort-producer.sh https://github.com/apache/metron/raw/master
/metron-deployment/roles/snort/templates/start-snort-producer.sh
# sed -i 's/{{ snort_alert_csv_path }}/\/var\/log\/snort\/alert.csv/' /opt/snort-producer/start-
snort-producer.sh
# sed -i 's/{{ kafka_prod }}/\/usr\/hdp\/current\/kafka-broker\/bin\/kafka-console-producer.sh/'
/opt/snort-producer/start-snort-producer.sh
# sed -i 's/{{ kafka_broker_url }}/<KAFKA_BROKER_IP>:6667/' /opt/snort-producer/start-snort-
producer.sh
# sed -i 's/{{ snort_topic }}/snort/' /opt/snort-producer/start-snort-producer.sh
```

```
# chmod 755 /opt/snort-producer/start-snort-producer.sh
# wget -O /etc/init.d/snort-producer https://github.com/apache/metron/raw/master/metron-deployment
/roles/snort/templates/snort-producer
# sed -i 's/{{ snort_producer_home }}/\/opt\/snort-producer/' /etc/init.d/snort-producer
# sed -i 's/{{ snort_producer_start }}/\/opt\/snort-producer\/start-snort-producer.sh/' /etc/init.
d/snort-producer
# chmod 755 /etc/init.d/snort-producer
```

- Install yaf:

```
# wget -O /tmp/libfixbuf-1.7.1.tar.gz http://tools.netsa.cert.org/releases/libfixbuf-1.7.1.tar.gz
# /bin/gtar --extract -C /tmp -z -f /tmp/libfixbuf-1.7.1.tar.gz
# cd /tmp/libfixbuf-1.7.1
# ./configure
# make -j4
# make install
# wget -O /tmp/yaf-2.8.0.tar.gz http://tools.netsa.cert.org/releases/yaf-2.8.0.tar.gz
# /bin/gtar --extract -C /tmp -z -f /tmp/yaf-2.8.0.tar.gz
# cd /tmp/yaf-2.8.0
# ./configure --enable-applabel --enable-plugins
# make -j4
# make install
# mkdir /opt/yaf
# chmod 755 /opt/yaf
# wget -O /opt/yaf/start-yaf.sh https://github.com/apache/metron/raw/master/metron-deployment
/roles/yaf/templates/start-yaf.sh
# sed -i 's/{{ yaf_bin }}/\/usr\/local\/bin\/yaf/' /opt/yaf/start-yaf.sh
# sed -i 's/{{ sniff_interface }}/tap0/' /opt/yaf/start-yaf.sh
# sed -i 's/{{ yafscii_bin }}/\/usr\/local\/bin\/yafscii/' /opt/yaf/start-yaf.sh
# sed -i 's/{{ kafka_prod }}/\/usr\/hdp\/current\/kafka-broker\/bin\/kafka-console-producer.sh/'
/opt/yaf/start-yaf.sh
# sed -i 's/{{ kafka_broker_url }}/<BROKER_IP>:6667/' /opt/yaf/start-yaf.sh
# sed -i 's/{{ yaf_topic }}/yaf/' /opt/yaf/start-yaf.sh
# chmod 755 /opt/yaf/start-yaf.sh
# wget -O /etc/init.d/yaf https://github.com/apache/metron/raw/master/metron-deployment/roles/yaf
/templates/yaf
# sed -i 's/{{ yaf_home }}/\/opt\/yaf/' /etc/init.d/yaf
# sed -i 's/{{ yaf_start }}/\/opt\/yaf\/start-yaf.sh/' /etc/init.d/yaf
# sed -i 's/^DAEMONOPTS=\"${@:2}\"$/DAEMONOPTS=\"${@:2} --idle-timeout 0\"/' /etc/init.d/yaf
# chmod 755 /etc/init.d/yaf
```

- Install tcpreplay:

```
# wget -O /tmp/tcpreplay-4.1.1.tar.gz https://github.com/appneta/tcpreplay/releases/download/v4.
1.1/tcpreplay-4.1.1.tar.gz
# /bin/gtar --extract -C /opt -z  -f /tmp/tcpreplay-4.1.1.tar.gz
# cd /opt/tcpreplay-4.1.1/
# ./configure --prefix=/opt
# make -j4
# make install
# mkdir /opt/pcap-replay
# chown root.root /opt/pcap-replay
# chmod 755 /opt/pcap-replay
# cd /opt/pcap-replay
# wget https://github.com/apache/metron/raw/master/metron-deployment/roles/sensor-test-mode/files
/example.pcap
# echo "include \$RULE_PATH/test.rules" >> /etc/snort/snort.conf
# echo "alert tcp any any -> any any (msg:'snort test alert'; sid:999158; )" > /etc/snort/rules
/test.rules
# wget -O /etc/init.d/pcap-replay https://github.com/apache/metron/raw/master/metron-deployment
/roles/pcap_replay/templates/pcap-replay
# sed -i 's/{{ pcap_replay_home }}/\/opt\/pcap-replay/' /etc/init.d/pcap-replay
# sed -i 's/{{ pcap_replay_interface }}/tap0/' /etc/init.d/pcap-replay
# sed -i 's/{{ tcpreplay_prefix }}/\/opt/' /etc/init.d/pcap-replay
# chmod 755 /etc/init.d/pcap-replay
```

- Install monit

```
# yum install monit -y
# wget -O /etc/monitrc https://github.com/apache/metron/raw/master/metron-deployment/roles/monit
/templates/monit/monit.conf
# sed -i 's/{{ inventory_hostname }}/<IP ADDRESS>/' /etc/monitrc
# sed -i 's/{{ monit_user }}/admin/' /etc/monitrc
# sed -i 's/{{ monit_pass }}/monit/' /etc/monitrc
# chmod 600 /etc/monitrc
```

```
# wget -O /etc/monit.d/pcap-replay.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/pcap-replay.monit
# chmod 644 /etc/monit.d/pcap-replay.monit
# wget -O /etc/monit.d/pcap-service.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/pcap-service.monit
# chmod 644 /etc/monit.d/pcap-service.monit
# wget -O /etc/monit.d/pycapa.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/pycapa.monit
# chmod 644 /etc/monit.d/pycapa.monit
# wget -O /etc/monit.d/snort.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/snort.monit
# chmod 644 /etc/monit.d/snort.monit
# wget -O /etc/monit.d/yaf.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/yaf.monit
# chmod 644 /etc/monit.d/yaf.monit
# wget -O /etc/monit.d/bro.monit https://github.com/apache/metron/raw/master/metron-deployment/roles/monit/templates/monit/bro.monit
# sed -i 's/^  with pidfile.*$/  with pidfile \/usr\/local\/bro\/spool\/bro\/\.pid/' /etc/monit.d/bro.monit
# chmod 644 /etc/monit.d/bro.monit
# systemctl enable monit
# systemctl start monit
# systemctl status monit
# monit reload
# monit stop all
# monit start all
# monit summary | tail -n +3 | awk -F"'" '{print $2}'
```

Miscellaneous Issues

- I had a problem with Zeppelin after rebooting this machine and had to manually create the Zeppelin run directory:

```
# mkdir /var/run/zeppelin
# chown zeppelin.hadoop zeppelin/
```

- Additionally, while working with Metron, I've noticed that at some point Zeppelin Notebook started, but immediately stopped again. In the logs, I could see "Address already in use" messages. It turns out that there was still a lingering Zeppelin process on the host. To fix it, stop Zeppelin Notebook in Ambari and then kill the latent process:

```
# ps aux | grep zeppelin
# kill <zeppelin_java_pid>
```

Afterwards, restart Zeppelin Notebook via Ambari.

- I had a couple of issues with Elasticsearch where it wouldn't find a master. This was fixed by doing the following. In Ambari, set the following items:

"masters_also_are_datanodes" to "true"
"expected_data_nodes" = "0"
"gateway_recover_after_data_nodes" = "1"
Restart all services. At this point, I noticed the following in :/etc/elasticsearch/elasticsearch.yml":

```
node:
  data: "true"
  master: "true"
  name: metron1.local
```

After changing this to :

```
node:
  data: true
  master: true
  name: metron
```

and restarting elasticsearch with "service elasticsearch restart", elasticsearch started indexing.

- Another issue with Elasticsearch was that I saw the following error message in Kibana:

```
plugin:elasticsearch     Elasticsearch is still initializing the kibana index.
```

This was fixed by deleting the Kibana index ".kibana": "curl -XDELETE http://localhost:9200/.kibana"

Miscellaneous Services

- Load the correct Elasticsearch template with:
  "# curl -s -w "%{http_code}" -u <USERNAME>:<PASSWORD> -H "X-Requested-By: ambari" -X POST -d '{ "RequestInfo": { "context": "Install ES Template from REST", "command": "ELASTICSEARCH_TEMPLATE_INSTALL"},"Requests/resource_filters": [{"service_name": "METRON","component_name": "METRON_INDEXING","hosts" : "<HOSTNAME>"}]}' http://<AMBARI HOST>:8080/api/v1/clusters/<CLUSTERNAME>/requests"

  For example:

  "# curl -s -w "%{http_code}" -u admin:admin -H "X-Requested-By: ambari" -X POST -d '{ "RequestInfo": { "context": "Install ES Template from REST", "command": "ELASTICSEARCH_TEMPLATE_INSTALL"},"Requests/resource_filters": [{"service_name": "METRON","component_name": "METRON_INDEXING","hosts" : "metron"}]}' http://192.168.10.10:8080/api/v1/clusters/metron/requests"

- Load Kibana Dashboard with:

"# curl -s -w "%{http_code}" -u <USERNAME>:<PASSWORD> -H "X-Requested-By: ambari" -X POST -d '{ "RequestInfo": { "context": "Install Kibana Dashboard from REST", "command": "LOAD_TEMPLATE"},"Requests/resource_filters": [{"service_name": "KIBANA","component_name": "KIBANA_MASTER","hosts" : "<HOSTNAME>"}]}' http://<AMBARI HOST>:8080/api/v1/clusters/<CLUSTERNAME>/requests"

For example:

"# curl -s -w "%{http_code}" -u admin:admin -H "X-Requested-By: ambari" -X POST -d '{ "RequestInfo": { "context": "Install Kibana Dashboard from REST", "command": "LOAD_TEMPLATE"},"Requests/resource_filters": [{"service_name": "KIBANA","component_name": "KIBANA_MASTER","hosts" : "metron"}]}' http://192.168.10.10:8080/api/v1/clusters/metron/requests"

- If you installed Metron on a single node, you might have to increase the number of Storm supervisor slots from the default 2 to 5 or more. This can be done by editing the "supervisor.slots.ports" under Storm in the Ambari UI.

  Change: "supervisor.slots.ports: [6700, 6701]" to "supervisor.slots.ports: [6700, 6701, 6702, 6703, 6704, 6705]"

- Install Apache NiFi in /root (You can pretty much use any directory you want). Download nifi-1.2.0-bin.tar.gz from https://nifi.apache.org/download.html

```
# cd /root
# wget http://apache.mirror.iweb.ca/nifi/1.2.0/nifi-1.2.0-bin.tar.gz
# tar xf nifi-1.2.0-bin.tar.gz
```

Before we run NiFi, we need to change the port as the default port collides with the Ambari port. To do this, we need to change the value "nifi.web.http.port=8080" to "nifi.web.http.port=8089" in the file "nifi-1.1.2/conf/nifi.properties". Install and start NiFi afterwards:

```
# nifi-1.2.0/bin/nifi.sh install
# nifi-1.2.0/bin/nifi.sh start
```

Exposed Interfaces

In the end, you'll end up with a bunch of exposed UIs:
- Ambari: http://node1:8080/
- Kibana: http://node1:5000/
- Sensor Status (monit): http://node1:2812
- Elasticsearch: http://node1:9200/
- Storm UI: http://node1:8744/
- Metron REST interface: http://node1:8082/swagger-ui.html#/
- Management UI: http://node1:4200/ (user/password)
- Apache Nifi: http://node1:8089/nifi/
- Zookeeper: http://node1:2181
- Kafka: http://node1:6667

TROUBLESHOOTING