TSDR

Release master

Sep 10, 2018

Contents

This page provides pointers to the TSDR installation and user documentation.

CHAPTER 1

Installation Guides

1.1 TSDR Install Guide

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

1.1.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

> https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
> https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

1.1.2 TSDR Installation Guide

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to gather and store network metrics from SDN protocols, traditional network protocols as well as SDN and network environment data. This data is stored in a common format using one of several databases and is accessible by a REST northbound interface, Grafana interface (beta) and by the ODL API.

This document details instructions for installing TSDR, it's features and datastores.

Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight SDN controller. Please refer to the User Guide for the detailed description of the functionality of the project and how to use the corresponding features provided in TSDR. TSDR supports HSQLDB, HBase, or Cassandra Data Stores

Pre Requisites for Installing TSDR

The software requirements for TSDR HBase Data Store are as follows:

- In the case when the user chooses HBase or Cassandra data store, besides the software that ODL requires, we also require HBase and Cassandra database running in single node deployment scenario.

No additional software is required for the HSQLDB Data Stores.

Preparing for Installation

- When using HBase data store, download HBase from the following website:

  http://archive.apache.org/dist/hbase/hbase-0.94.15/

- When using Cassandra data store, download Cassandra from the following website:

  http://www.eu.apache.org/dist/cassandra/2.1.10/

- No additional steps are required to install the TSDR HSQL Data Store.

Installing HSQLDB Data Store

Once OpenDaylight distribution is up, from Karaf console install the any collector. The HSQLDB data store is installed by default

```
feature:install odl-tsdr-openflow-statistics-collector
```

This will install hsqldb related dependency features (and can take sometime) as well as OpenFlow statistics collector before returning control to the console.

Installing HBase Data Store

Installing TSDR HBase Data Store contains two steps:

1. Installing HBase server, and
2. Installing TSDR HBase Data Store features from ODL Karaf console.

In this release, we only support HBase single node running together on the same machine as OpenDaylight. Therefore, follow the steps to download and install HBase server onto the same machine as where OpenDaylight is running:

1. Create a folder in Linux operating system for the HBase server. For example, create an hbase directory under `/usr/lib`:

```
mkdir /usr/lib/hbase
```

2. Unzip the downloaded HBase server tar file.

   Run the following command to unzip the installation package:

```
tar xvf <hbase-installer-name>  /usr/lib/hbase
```

3. Make proper changes in hbase-site.xml

   (a) Under <hbase-install-directory>/conf/, there is a hbase-site.xml. Although it is not recommended, an experienced user with HBase can modify the data directory for hbase server to store the data.

   (b) Modify the value of the property with name "hbase.rootdir" in the file to reflect the desired file directory for storing hbase data.

   The following is an example of the file:

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>file:///usr/lib/hbase/data</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/usr/lib/hbase/zookeeper</value>
  </property>
</configuration>
```

4. start hbase server:

```
cd <hbase-installation-directory>
./start-hbase.sh
```

5. start hbase shell:

```
cd <hbase-insatllation-directory>
./hbase shell
```

6. start Karaf console
7. install hbase data store feature from Karaf console:

```
feature:install odl-tsdr-hbase
```

Installing Cassandra Data Store

Installing TSDR Cassandra Data Store contains two steps:

1. Installing Cassandra server, and
2. Installing TSDR Cassandra Data Store features from ODL Karaf console.

In this release, we only support Cassadra single node running together on the same machine as OpenDaylight. Therefore, follow these steps to download and install Cassandra server onto the same machine as where OpenDaylight is running:

1. Install Cassandra (latest stable version) by downloading the zip file and untar the tar ball to cassandra/ directory on the testing machine:

```
mkdir cassandra
wget http://www.eu.apache.org/dist/cassandra/2.1.10/apache-cassandra-2.1.10-bin.
↪tar.gz[2.1.10 is current stable version, it can vary]
mv apache-cassandra-2.1.10-bin.tar.gz cassandra/
cd cassandra
tar -xvzf apache-cassandra-2.1.10-bin.tar.gz
```

2. Start Cassandra from cassandra directory by running:

```
./apache-cassandra-2.1.10/bin/cassandra
```

3. Start cassandra shell by running:

```
./apache-cassandra-2.1.10/bin/cqlsh
```

4. Start Karaf according to the instructions above.
5. Install Cassandra data store feature from Karaf console:

```
feature:install odl-tsdr-cassandra
```

Verifying your Installation

After the TSDR data store is installed, no matter whether it is HBase data store, Cassandra data store, or HSQLDB data store, the user can verify the installation with the following steps.

1. Verify if the following two TSDR commands are available from Karaf console:

```
tsdr:list
tsdr:purgeAll
```

2. Verify if OpenFlow statistics data can be received successfully:

   Run "feature:install odl-tsdr-openflow-statistics-collector" from Karaf.

   Run mininet to connect to ODL controller. For example, use the following command to start a three node topology:

```
mn --topo single,3  --controller 'remote,ip=172.17.252.210,port=6653' --switch␣
↪ovsk,protocols=OpenFlow13
```

   From Karaf console, the user should be able to retrieve the statistics data of OpenFlow statistics data from the console:

```
tsdr:list FLOWSTATS
```

Troubleshooting

Check the `../data/log/karaf.log` for any exception related to TSDR features.

Post Installation Configuration

Post Installation Configuration for HSQLDB Data Store

The feature installation takes care of automated configuration of the datasource by installing a file in <install folder>/etc named org.ops4j.datasource-metric.cfg. This contains the default location of <install folder>/tsdr where the HSQLDB datastore files are stored. If you want to change the default location of the datastore files to some other location update the last portion of the url property in the org.ops4j.datasource-metric.cfg and then restart the Karaf container.

Post Installation Configuration for HBase Data Store

Please refer to HBase Data Store User Guide.

Post Installation Configuration for Cassandra Data Store

There is no post configuration for TSDR Cassandra data store.

Upgrading From a Previous Release

The HBase data store was supported in the previous release as well as in this release. However, we do not support data store upgrade for HBase data store. The user needs to reinstall TSDR and start to collect data in TSDR HBase datastore after the installation.

HSQLDB and Cassandra are new data stores introduced in this release. Therefore, upgrading from previous release does not apply in these two data store scenarios.

Uninstalling TSDR Data Stores

To uninstall TSDR HSQLDB data store

To uninstall the TSDR functionality with the default store, you need to do the following from karaf console:

```
feature:uninstall odl-tsdr-hsqldb-all
feature:uninstall odl-tsdr-core
feature:uninstall odl-tsdr-hsqldb
feature:uninstall odl-tsdr-openflow-statistics-collector
```

It is recommended to restart the Karaf container after the uninstallation of the TSDR functionality with the default store.

To uninstall TSDR HBase Data Store

To uninstall the TSDR functionality with the HBase data store,

- Uninstall HBase data store related features from karaf console:

```
feature:uninstall odl-tsdr-hbase
feature:uninstall odl-tsdr-core
```

- stop hbase server:

```
cd <hbase-installation-directory>
./stop-hbase.sh
```

- remove the file directory that contains the HBase server installation:

```
rm -r <hbase-installation-directory>
```

It is recommended to restart the Karaf container after the uninstallation of the TSDR data store.

To uninstall TSDR Cassandra Data Store

To uninstall the TSDR functionality with the Cassandra store,

- uninstall cassandra data store related features following from karaf console:

```
feature:uninstall odl-tsdr-cassandra
feature:uninstall odl-tsdr-core
```

- stop cassandra database:

```
ps auwx | grep cassandra
sudo kill pid
```

- remove the cassandra installation files:

```
rm <cassandra-installation-directory>
```

It is recommended to restart the Karaf container after uninstallation of the TSDR data store.

1.2 TSDR HSQLDB Data Store Installation Guide

This document is for the user to install the artifacts that are needed for using the HSQLDB Data Store in Time Series Data Repository.

1.2.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP

- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

> https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
> https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The

Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

1.2.2 HSQLDB Overview

Time series data records of OpenFlow statistics are collected periodically and stored in a persistent store. For non-production usage, the bundled default datastore (HSQLDB) is loaded with any collector. The TSDR records get persisted in HSQLDB store in <install folder>/tsdr/ folder by default.

1.2.3 Installing TSDR with default HSQLDB datastore

Once OpenDaylight distribution is up, from the Karaf console install the TSDR feature with default datastore (HSQLDB store used) can be installed by

```
feature:install odl-tsdr-openflow-statistics-collector
```

This will install all dependent features and the HQSLDB datastore.

1.2.4 Verifying your Installation

If the feature install was successful you should be able to see the following TSDR commands added.

```
tsdr:list
```

1.2.5 Troubleshooting

Check the ../data/log/karaf.log for any exception related to TSDR or HSQLDB related features.

1.2.6 Post Installation Configuration

The feature installation takes care of automated configuration of the datasource by installing a file in <install folder>/etc named org.ops4j.datasource-metric.cfg. This contains the default location of <install folder>/tsdr where the HSQLDB datastore files are stored. If you want to change the default location of the datastore files to some other location update the last portion of the url property in the org.ops4j.datasource-metric.cfg and then restart the Karaf container.

1.2.7 Uninstalling TSDR with default HSQLDB datastore

To uninstall the TSDR functionality with the default store, you need to do the following from karaf console.

```
feature:uninstall odl-tsdr-core feature:uninstall odl-tsdr-hsqldb
```

You can also restart the Karaf container with the “clean” keyword.

1.3 TSDR HBase Data Store Installation Guide

This document is for the user to install the artifacts that are needed for using the HBase Data Store in Time Series Data Repository.

1.3.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector

  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

1.3.2 Prerequisites for Installing TSDR HBase Data Store

The hardware requirements are the same as those for standard ODL controller installation.

The supported operating system for TSDR HBase Data Store is Unix.

Preparing for Installation

Download HBase from the following web site.

> http://archive.apache.org/dist/hbase/hbase-0.94.15/

Other versions of HBase work, but the later versions may not.

Upgrading to the latest HBase is on the fast track to be implemented.

1.3.3 Installing TSDR HBase Data Store

Installing TSDR HBase Data Store contains two steps: Installing HBase server and installing TSDR HBase Data Store features from ODL Karaf console.

This installation guide will only cover the first step. For installing TSDR HBase Data Store features, please refer to TSDR HBase Data Store User Guide.

TSDR supports HBase single node running together on the same machine as ODL controller. Therefore, follow the steps to download and install HBase server onto the same box as where ODL controller is running:

Create a folder in Linux operating system for the HBase server.

For example, create an hbase directory under /usr/lib:

mkdir /usr/lib/hbase

Unzip the downloaded HBase server tar file.

Run the following command to unzip the installation package:

tar xvf <hbase-installer-name> /usr/lib/hbase

Make proper changes in hbase-site.xml

The following is an example of the file:

```
<configuration>
   <property>
      <name>hbase.rootdir</name>
      <value>file:///usr/lib/hbase/data</value>
   </property>
   <property>
      <name>hbase.zookeeper.property.dataDir</name>
      <value>/usr/lib/hbase/zookeeper</value>
   </property>
</configuration>
```

1.3.4 Verifying your Installation

After the HBase server is properly installed, start hbase server and hbase shell.

start hbase server cd <hbase-installation-directory> ./start-hbase.sh

start hbase shell cd <hbase-insatllation-directory> ./hbase shell

1.3.5 Post Installation Configuration

Please refer to HBase Data Store User Guide.

1.3.6 Uninstalling HBase Data Store

To uninstall TSDR HBase Data Store, stop hbase server cd <hbase-installation-directory> ./stop-hbase.sh

To remove the file directory that contains the HBase server installation. rm -r <hbase-installation-directory>

CHAPTER 2

User Guides

2.1 TSDR User Guide

2.1.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

> https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
> https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

2.1.2 Configuring TSDR Data Stores

For detailed instruction, see the individual datastore installation guides.

Configure HSQLDB Data Store

The HSQLDB files are stored automatically in <karaf install folder>/tsdr/ directory. If you want to change the default storage location, the configuration file to change can be found in <karaf install folder>/etc directory. The filename is org.ops4j.datasource-metric.cfg. Change the last portion of the url=jdbc:hsqldb:./tsdr/metric to point to different directory.

After installing HBase Server on the same machine as OpenDaylight, if the user accepts the default configuration of the HBase Data Store, the user can directly proceed with the installation of HBase Data Store from Karaf console.

Optionally, the user can configure TSDR HBase Data Store following HBase Data Store Configuration Procedure.

- HBase Data Store Configuration Steps
  - Open the file etc/tsdr-persistence-hbase.peroperties under karaf distribution directory.
  - Edit the following parameters:
    - HBase server name
    - HBase server port
    - HBase client connection pool size
    - HBase client write buffer size

After the configuration of HBase Data Store is complete, proceed with the installation of HBase Data Store from Karaf console.

- HBase Data Store Installation Steps
  - Start Karaf Console
  - Run the following commands from Karaf Console: feature:install odl-tsdr-hbase

To Configure Cassandra Data Store

Currently, there's no configuration needed for Cassandra Data Store. The user can use Cassandra data store directly after installing the feature from Karaf console.

Additionally separate commands have been implemented to install various data collectors.

Administering or Managing TSDR Data Stores

To Administer HSQLDB Data Store

Once the TSDR default datastore feature (odl-tsdr-hsqldb-all) is enabled, the TSDR captured OpenFlow statistics metrics can be accessed from Karaf Console by executing the command

```
tsdr:list <metric-category> <starttimestamp> <endtimestamp>
```

wherein

- <metric-category> = any one of the following categories FlowGroupStats, FlowMeterStats, FlowStats, FlowTableStats, PortStats, QueueStats

- <starttimestamp> = to filter the list of metrics starting this timestamp
- <endtimestamp> = to filter the list of metrics ending this timestamp
- <starttimestamp> and <endtimestamp> are optional.
- Maximum 1000 records will be displayed.

To Administer HBase Data Store

Using Karaf Command to retrieve data from HBase Data Store

The user first need to install hbase data store from karaf console:

feature:install odl-tsdr-hbase

The user can retrieve the data from HBase data store using the following commands from Karaf console:

tsdr:list tsdr:list <CategoryName> <StartTime> <EndTime>

Typing tab will get the context prompt of the arguments when typeing the command in Karaf console.

To Administer Cassandra Data Store

The user first needs to install Cassandra data store from Karaf console:

feature:install odl-tsdr-cassandra

Then the user can retrieve the data from Cassandra data store using the following commands from Karaf console:

tsdr:list tsdr:list <CategoryName> <StartTime> <EndTime>

Typing tab will get the context prompt of the arguments when typeing the command in Karaf console.

Installing TSDR Data Collectors

When the user uses HSQLDB data store and installed "odl-tsdr-hsqldb-all" feature from Karaf console, besides the HSQLDB data store, OpenFlow data collector is also installed with this command. However, if the user needs to use other collectors, such as NetFlow Collector, Syslog Collector, SNMP Collector, and Controller Metrics Collector, the user needs to install them with separate commands. If the user uses HBase or Cassandra data store, no collectors will be installed when the data store is installed. Instead, the user needs to install each collector separately using feature install command from Karaf console.

The following is the list of supported TSDR data collectors with the associated feature install commands:

- OpenFlow Data Collector

```
feature:install odl-tsdr-openflow-statistics-collector
```

- NetFlow Data Collector

```
feature:install odl-tsdr-netflow-statistics-collector
```

- Syslog Data Collector

```
feature:install odl-tsdr-syslog-collector
```

- Controller Metrics Collector

```
feature:install odl-tsdr-controller-metrics-collector
```

- Web Activity Collector

```
feature:install odl-tsdr-restconf-collector
```

- sFlow Data Collector (experimental)

```
feature:install odl-tsdr-sflow-statistics-colletor
```

- SNMP Data Collector (experimental)

```
feature:install odl-tsdr-snmp-data-collector
```

In order to use controller metrics collector, the user needs to install Sigar library.

The following is the instructions for installing Sigar library on Ubuntu:

- Install back end library by "sudo apt-get install libhyperic-sigar-java"
- Execute "export LD_LIBRARY_PATH=/usr/lib/jni/:/usr/lib:/usr/local/lib" to set the path of the JNI (you can add this to the ".bashrc" in your home directory)
- Download the file "sigar-1.6.4.jar". It might be also in your ".m2" directory under "~/.m2/resources/org/fusesource/sigar/1.6.4"
- Create the directory "org/fusesource/sigar/1.6.4" under the "system" directory in your controller home directory and place the "sigar-1.6.4.jar" there

Querying TSDR from REST APIs

TSDR provides two REST APIs for querying data stored in TSDR data stores.

- Query of TSDR Metrics
  - URL: http://localhost:8181/tsdr/metrics/query
  - Verb: GET
  - Parameters:
    - tsdrkey=[NID=][DC=][MN=][RK=]

```
The TSDRKey format indicates the NodeID(NID), DataCategory(DC),␣
↪MetricName(MN), and RecordKey(RK) of the monitored objects.
For example, the following is a valid tsdrkey:
[NID=openflow:1][DC=FLOWSTATS][MN=PacketCount][RK=Node:openflow:1,Table:0,
↪Flow:3]
The following is also a valid tsdrkey:
tsdrkey=[NID=][DC=FLOWSTATS][MN=][RK=]
In the case when the sections in the tsdrkey is empty, the query will␣
↪return all the records in the TSDR data store that matches the filled␣
↪tsdrkey. In the above example, the query will return all the data in␣
↪FLOWSTATS data category.
The query will return only the first 1000 records that match the query␣
↪criteria.
```

    - from=<time_in_seconds>
    - until=<time_in_seconds>

The following is an example curl command for querying metric data from TSDR data store:

curl -G -v -H "Accept: application/json" -H "Content-Type: application/json" "http://localhost:8181/tsdr/metrics/query" –data-urlencode "tsdrkey=[NID=][DC=FLOWSTATS][MN=][RK=]" –data-urlencode "from=0" –data-urlencode "until=240000000000"lmore

- Query of TSDR Log type of data
  - URL:http://localhost:8181/tsdr/logs/query
  - Verb: GET
  - Parameters:
    - tsdrkey=tsdrkey=[NID=][DC=][RK=]

```
The TSDRKey format indicates the NodeID(NID), DataCategory(DC), and␣
↪RecordKey(RK) of the monitored objects.
For example, the following is a valid tsdrkey:
[NID=openflow:1][DC=NETFLOW][RK]
The query will return only the first 1000 records that match the query␣
↪criteria.
```

    - from=<time_in_seconds>
    - until=<time_in_seconds>

The following is an example curl command for querying log type of data from TSDR data store:

curl -G -v -H "Accept: application/json" -H "Content-Type: application/json" "http://localhost:8181/tsdr/logs/query" –data-urlencode "tsdrkey=[NID=][DC=NETFLOW][RK=]" –data-urlencode "from=0" –data-urlencode "until=240000000000"lmore

Grafana integration with TSDR

TSDR provides northbound integration with Grafana time series data visualization tool. All the metric type of data stored in TSDR data store can be visualized using Grafana.

For the detailed instruction about how to install and configure Grafana to work with TSDR, please refer to the following link:

https://wiki.opendaylight.org/view/Grafana_Integration_with_TSDR_Step-by-Step

Configuring TSDR Data Collectors

SNMP Data Collector Device Credential Configuration (experimental)

After installing SNMP Data Collector, a configuration file under etc/ directory of ODL distribution is generated: etc/tsdr.snmp.cfg is created.

The following is a sample tsdr.snmp.cfg file:

credentials=[192.168.0.2,public],[192.168.0.3,public]

The above credentials indicate that TSDR SNMP Collector is going to connect to two devices. The IPAddress and Read community string of these two devices are (192.168.0.2, public), and (192.168.0.3) respectively.

The user can make changes to this configuration file any time during runtime. The configuration will be picked up by TSDR in the next cycle of data collection.

Polling interval configuration for SNMP Collector and OpenFlow Stats Collector

The default polling interval of SNMP Collector and OpenFlow Stats Collector is 30 seconds and 15 seconds respectively. The user can change the polling interval through restconf APIs at any time. The new polling interval will be picked up by TSDR in the next collection cycle.

- Retrieve Polling Interval API for SNMP Collector
  - URL: http://localhost:8181/restconf/config/tsdr-snmp-data-collector:TSDRSnmpDataCollectorConfig
  - Verb: GET
- Update Polling Interval API for SNMP Collector
  - URL: http://localhost:8181/restconf/operations/tsdr-snmp-data-collector:setPollingInterval
  - Verb: POST
  - Content Type: application/json
  - Input Payload:

```
{
   "input": {
       "interval": "15000"
   }
}
```

- Retrieve Polling Interval API for OpenFlowStats Collector
  - URL: http://localhost:8181/restconf/config/tsdr-openflow-statistics-collector:TSDROSCConfig
  - Verb: GET
- Update Polling Interval API for OpenFlowStats Collector
  - URL: http://localhost:8181/restconf/operations/tsdr-openflow-statistics-collector:setPollingInterval
  - Verb: POST
  - Content Type: application/json
  - Input Payload:

```
{
   "input": {
       "interval": "15000"
   }
}
```

Purging Service configuration

After the data stores are installed from Karaf console, the purging service will be installed as well. A configuration file called tsdr.data.purge.cfg will be generated under etc/ directory of ODL distribution.

The following is the sample default content of the tsdr.data.purge.cfg file:

host=127.0.0.1 data_purge_enabled=true data_purge_time=23:59:59 data_purge_interval_in_minutes=1440 retention_time_in_hours=168

The host indicates the IPAddress of the data store. In the case when the data store is together with ODL controller, 127.0.0.1 should be the right value for the host IP. The other attributes are self-explained. The user can change those attributes at any time. The configuration change will be picked up right away by TSDR Purging service at runtime.

How to use TSDR to collect, store, and view OpenFlow Interface Statistics

Overview

This tutorial describes an example of using TSDR to collect, store, and view one type of time series data in OpenDaylight environment.

Prerequisites

You would need to have the following as prerequisits:

- One or multiple OpenFlow enabled switches. Alternatively, you can use mininet to simulate such a switch.
- Successfully installed OpenDaylight Controller.
- Successfully installed HBase Data Store following TSDR HBase Data Store Installation Guide.
- Connect the OpenFlow enabled switch(es) to OpenDaylight Controller.

Target Environment

HBase data store is only supported in Linux operation system.

Instructions

- Start OpenDaylight.
- Connect OpenFlow enabled switch(es) to the controller.
  - If using mininet, run the following commands from mininet command line:
    - mn –topo single,3 –controller *remote,ip=172.17.252.210,port=6653* –switch ovsk,protocols=OpenFlow13
- Install TSDR hbase feature from Karaf:
  - feature:install odl-tsdr-hbase
- Install OpenFlow Statistics Collector from Karaf:
  - feature:install odl-tsdr-openflow-statistics-collector
- run the following command from Karaf console:
  - tsdr:list PORTSTATS

You should be able to see the interface statistics of the switch(es) from the HBase Data Store. If there are too many rows, you can use "tsdr:list InterfaceStats|more" to view it page by page.

By tabbing after "tsdr:list", you will see all the supported data categories. For example, "tsdr:list FlowStats" will output the Flow statistics data collected from the switch(es).

Troubleshooting

Karaf logs

All TSDR features and components write logging information including information messages, warnings, errors and debug messages into karaf.log.

HBase and Cassandra logs

For HBase and Cassandra data stores, the database level logs are written into HBase log and Cassandra logs.

- HBase log
  - HBase log is under <HBase-installation-directory>/logs/.
- Cassandra log
  - Cassandra log is under {cassandra.logdir}/system.log. The default {cassandra.logdir} is /var/log/cassandra/.

Security

TSDR gets the data from a variety of sources, which can be secured in different ways.

- OpenFlow Security
  - The OpenFlow data can be configured with Transport Layer Security (TLS) since the OpenFlow Plugin that TSDR depends on provides this security support.
- NetFlow Security
  - NetFlow, which cannot be configured with security so we recommend making sure it flows only over a secured management network.
- Syslog Security
  - Syslog, which cannot be configured with security so we recommend making sure it flows only over a secured management network.
- SNMP Security
  - The SNMP version3 has security support. However, since ODL SNMP Plugin that TSDR depends on does not support version 3, we (TSDR) will not have security support at this moment.
- sFlow Security
  - The sflow has security support.

Support multiple data stores simultaneously at runtime

TSDR supports running multiple data stores simultaneously at runtim. For example, it is possible to configure TSDR to push log type of data into Cassandra data store, while pushing metrics type of data into HBase.

When you install one TSDR data store from karaf console, such as using feature:install odl-tsdr-hsqldb, a properties file will be generated under <Karaf-distribution-directory>/etc/. For example, when you install hsqldb, a file called tsdr-persistence-hsqldb.properties is generated under that directory.

By default, all the types of data are supported in the data store. For example, the default content of tsdr-persistence-hsqldb.properties is as follows:

```
metric-persistency=true
log-persistency=true
binary-persistency=true
```

When the user would like to use different data stores to support different types of data, he/she could enable or disable a particular type of data persistence in the data stores by configuring the properties file accordingly.

For example, if the user would like to store the log type of data in HBase, and store the metric and binary type of data in Cassandra, he/she needs to install both hbase and cassandra data stores from Karaf console. Then the user needs to modify the properties file under <Karaf-distribution-directory>/etc as follows:

- tsdr-persistence-hbase.properties

```
metric-persistency=false
log-persistency=true
binary-persistency=true
```

- tsdr-persistence-cassandra.properties

```
metric-psersistency=true
log-persistency=false
binary-persistency=false
```

2.2 TSDR HSQLDB Datastore User Guide

This document describes how to use the embedded datastore HSQLDB, which is the default datastore (recommended for non-production use case) introduced as part of OpenDaylight Time Series Data Respository (TSDR) project. This store captures the time series data. This document contains configuration, administration, management, using, and troubleshooting sections for the feature.

2.3 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

2.3.1 SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics

- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default

datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

2.4 HSQLDB Overview

2.4.1 Administering or Managing TSDR with default datastore HSQLDB

Once the TSDR default datastore feature (odl-tsdr-all) is enabled, the TSDR captured OpenFlow statistics metrics can be accessed from Karaf Console by executing the command

> tsdr:list <metric-category> <starttimestamp> <endtimestamp>

wherein

> <metric-category> = any one of the following categories FlowGroupStats, FlowMeterStats, FlowStats, FlowTableStats, PortStats, QueueStats <starttimestamp> = to filter the list of metrics starting this timestamp <endtimestamp> = to filter the list of metrics ending this timestamp

If either of <starttimestamp> or <endtimestamp> is not specified, this command displays the latest 1000 metrics captured.

2.5 TSDR HBase User Guide

2.5.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series

data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

> https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
> https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for

ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

2.5.2 TSDR HBase Data Store User Guide

This document describes how to use HBase to store time series data generated from the TSDR persistence framework data collectors. This document contains configuration, administration, management, usage and troubleshooting sections for the HBase data store feature.

Configuring TSDR with HBase Data Store

After installing HBase Server on the same VM as the OpenDaylight Controller, if the user accepts the default configuration of the HBase Data Store, the user can directly proceed with the installation of HBase Data Store from Karaf console. Optionally, the user can configure TSDR HBase Data Store following HBase Data Store Configuration Procedure.

HBase Data Store Configuration Steps

- Open the file etc/tsdr-persistence-hbase.properties under Karaf distribution directory.
- **Edit the following parameters**
  - HBase server name
  - HBase server port
  - HBase client connection pool size
  - HBase client write buffer size

After the configuration of HBase Data Store is complete, proceed with the installation of HBase Data Store from Karaf console.

- **HBase Data Store Installation Steps**
  - Start Karaf Console
  - **Run the following commands from Karaf Console:**
    - feature:install odl-tsdr-hbase

Administering or Managing TSDR HBase Data Store

Using Karaf Command to retrieve data from HBase Data Store

The user can retrieve the data from HBase data store using the following commands from Karaf console:

- tsdr:list
- tsdr:list <CategoryName> <StartTime> <EndTime>

Typing tab will get the context prompt of the arguments when typing the command in Karaf console.

Troubleshooting issues with log files

- Karaf logs

Similar to other OpenDaylight components and features, TSDR HBase Data Store writes logging information to Karaf logs. All the information messages, warnings, error messages, and debug messages are written to Karaf logs.

- HBase logs

For HBase system level logs, the user can check standard HBase server logs, which is under:

- <HBase-installation-directory>/logs.

2.5.3 Tutorials

How to use TSDR to collect, store, and view OpenFlow Interface Statistics

Overview

This tutorial describes an example of using TSDR to collect, store, and view one type of time series data in OpenDaylight environment.

Prerequisites

You would need to have the following as prerequisites: - One or multiple OpenFlow enabled switches. Alternatively, you can use mininet to simulate such a switch. - Successfully installed OpenDaylight Controller. - Successfully installed HBase Data Store following TSDR HBase Data Store Installation Guide. - Connect the OpenFlow enabled switch(es) to OpenDaylight Controller.

Target Environment

HBase data store is only supported on the Linux operating system.

Instructions

- Start OpenDaylight controller.
- Connect OpenFlow enabled switch(es) to the controller. If using mininet, run the following commands from mininet command line:

  mn –topo single,3 –controller 'remote,ip=172.17.252.210,port=6653' –switch ovsk,protocols=OpenFlow13
- If using real switch(es), the OpenDaylight controller should be able to discover the network topology containing the switches.
- Install a collector
- Install TSDR hbase feature from Karaf:

  feature:install odl-tsdr-hbase
- run the following command from Karaf console:

  tsdr:list InterfaceStats

You should be able to see the interface statistics of the switch(es) from the HBase Data Store. If there are too many rows, you can use "tsdr:list InterfaceStats|more" to view it page by page.

Tab (auto-complete) after "tsdr:list", you will see all the supported data categories.

For example, "tsdr:list FlowStats" will output the Flow statistics data collected from the switch(es).

2.6 ElasticSearch User Guide

2.6.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

> https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
> https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector

  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

2.6.2 Setting Up the environment

To setup and run the TSDR data store ElasticSearch feature, you need to have an ElasticSearch node (or a cluster of such nodes) running. You can use a customized ElasticSearch docker image for this purpose.

Your ElasticSearch (ES) setup must have the "Delete By Query Plugin" installed. Without this, some of the ES functionality won't work properly.

2.6.3 Creating a custom ElasticSearch docker image

(You can skip this section if you already have an instance of ElasticSearch running)

Run the following set of commands:

```
cat << EOF > Dockerfile
FROM elasticsearch:2
RUN /usr/share/elasticsearch/bin/plugin install --batch delete-by-query
EOF
```

To build the image, run the following command in the directory where the Dockerfile was created:

```
docker build . -t elasticsearch-dd
```

You can check whether the image was properly created by running:

```
docker images
```

This should print all your container images including the elasticsearch-dd.

Now we can create and run a container from our image by typing:

```
docker run -d -p 9200:9200 -p 9300:9300 --name elasticsearch-dd elasticsearch-dd
```

To see whether the container is running, run the following command:

```
docker ps
```

The output should include a row with elasticsearch-dd in the NAMES column. To check the std out of this container use

```
docker logs elasticsearch-dd
```

2.6.4 Running the ElasticSearch feature

Once the features have been installed, you can change some of its properties. For example, to setup the URL where your ElasticSearch installation runs, change the *serverUrl* parameter in tsdr/persistence-elasticsearch/src/main/resources/configuration/initial/:

```
tsdr-persistence-elasticsearch.properties
```

All the data are stored into the TSDR index under a type. The metric data are stored under the metric type and the log data are store under the log type. You can modify the files in tsdr/persistence-elasticsearch/src/main/resources/configuration/initial/:

```
tsdr-persistence-elasticsearch_metric_mapping.json
tsdr-persistence-elasticsearch_log_mapping.json
```

to change or tune the mapping for those types. The changes in those files will be promoted after the feature is reloaded or the distribution is restarted.

2.6.5 Testing the setup

We can now test whether the setup is correct by downloading and installing mininet, which we use to send some data to the running ElasticSearch instance.

Installing the necessary features:

```
start OpenDaylight
feature:install odl-restconf odl-l2switch-switch odl-tsdr-openflow-statistics-
↪collector
feature:install odl-tsdr-elasticsearch
```

We can check whether the distribution is now listening on port 6653:

```
netstat -an | grep 6653
```

Run mininet

```
sudo mn --topo single,3 --controller 'remote,ip=distro_ip,port=6653' --switch ovsk,
↪protocols=OpenFlow13
```

where the distro_ip is the IP address of the machine where the OpenDaylight distribution is running. This command will create three hosts connected to one OpenFlow capable switch.

We can check if data was stored by ElasticSearch in TSDR by running the following command:

```
tsdr:list FLOWTABLESTATS
```

The output should look similar to the following:

```
[NID=openflow:1][DC=FLOWTABLESTATS][MN=ActiveFlows][RK=Node:openflow:1,
↪Table:50][TS=1473427383598][3]
[NID=openflow:1][DC=FLOWTABLESTATS][MN=PacketMatch][RK=Node:openflow:1,
↪Table:50][TS=1473427383598][12]
[NID=openflow:1][DC=FLOWTABLESTATS][MN=PacketLookup][RK=Node:openflow:1,
↪Table:50][TS=1473427383598][12]
[NID=openflow:1][DC=FLOWTABLESTATS][MN=ActiveFlows][RK=Node:openflow:1,
↪Table:80][TS=1473427383598][3]
[NID=openflow:1][DC=FLOWTABLESTATS][MN=PacketMatch][RK=Node:openflow:1,
↪Table:80][TS=1473427383598][17]
[NID=openflow:1][DC=FLOWTABLESTATS][MN=PacketMatch][RK=Node:openflow:1,
↪Table:246][TS=1473427383598][19]
...
```

Or you can query your ElasticSearch instance:

```
curl -XPOST "http://elasticseach_ip:9200/_search?pretty" -d'{ "from": 0, "size":␣
↪10000, "query": { "match_all": {} } }'
```

The elasticseach_ip is the IP address of the server where the ElasticSearch is running.

2.6.6 Web Activity Collector

The Web Activity Collector records the meaningful REST requests made through the OpenDaylight RESTCONF interface.

How to test the RestConf Collector

- Issue a restconf command that uses either POST,PUT or DELETE. For example, you could call the register-filter RPC of tsdr-syslog-collector.
- Look up data in TSDR database from Karaf.

```
tsdr:list restconf
```

- You should see the request that you have sent, along with its information (URL, HTTP method, requesting IP address and request body)
- Try to send a GET request, then check again, your request should not be registered, because the collector does not register GET requests by default.
- Open the file: "etc/tsdr.restconf.collector.cfg", and add GET to the list of METHODS_TO_LOG, so that it becomes:

```
METHODS_TO_LOG=POST,PUT,DELETE,GET
```

  - Try again to issue your GET request, and check if it was recorded this time, it should be recorder.
  - Try manipulating the other properties (PATHS_TO_LOG (which URLs do we want to log from), REMOTE_ADDRESSES_TO_LOG (which requesting IP addresses do we want to log from) and CONTENT_TO_LOG (what should be in the request's body in order to log it)), and see if the requests are getting logged.
  - Try providing invalid properties (unknown methods for the METHODS_TO_LOG parameter, or the same method repeated multiple times, and invalid regular expressions for the other parameters), then check karaf's log using "log:display". It should tell you that the value is invalid, and that it will use the default value instead.

2.7 REST Web Activity (RestConf collector) User Guide

2.7.1 TSDR Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) is an extendible collector framework used to collect and store network metrics from SDN protocols, traditional network protocols as well as SDN controller and environment data. This data is stored in a common format using one of several datastores and is accessible by a REST interface, Grafana interface (beta) and by the ODL provided API.

SDN, Environment and Traditional Network Data Collected

- OpenFlow
- NetFlow
- sFlow
- REST
- SNMP
- SysLog
- Controller Metrics
- **TSDR Features URL:** https://git.opendaylight.org/gerrit/gitweb?p=tsdr.git;a=blob;f=features/features-tsdr/pom.xml

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight infrastructure. TSDR provides the framework for plugging in various data collectors to collect OpenFlow, sFlow, NetFlow, SysLog and Controller metrics time series data in a common data model and generic TSDR data persistence API. The user can choose which data stores to

be plugged into the TSDR Persistence framework. Three data stores are currently supported: HSQLDB (loaded by default), HBase and Cassandra.

With the capabilities of data collection, storage, query, aggregation and purging provided by TSDR, network administrators could leverage various data driven applications built on top of TSDR for security risk detection, performance analysis, operational configuration optimization, traffic engineering and network analytics with automated intelligence.

TSDR Model/Architecture

https://wiki.opendaylight.org/view/TSDR_Data_Storage_Service_and_Persistence_with_HBase_Plugin
https://wiki.opendaylight.org/view/TSDR_Data_Collection_Service

TSDR has the following major components:

- Data Collection Service
  - OpenFlow Data Collector
  - RestConf Data Collector
  - NetFlow Data Collector
  - SNMP Data Collector
  - sFlow Data Collector
  - SysLog Data Collector
  - Controller Metrics Data Collector
- Data Storage Service
  - HSQLDB Datastore (loaded by default)
  - HBase Datastore
  - Cassandra Datastore
- Data Query Service
- ElasticSearch Service
- Grafana integration for time series data visualization
- Data Aggregation Service
- Data Purging Service

The Data Collection Service handles the collection of time series data into TSDR and hands it over to the Data Storage Service. The Data Storage Service stores the data into TSDR through the TSDR Persistence Layer. The TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. The Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. The Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

TSDR provides component-based services on a common data model. These services include the data collection service, data storage service and data query service. The TSDR data storage service supports HSQLDB (the default datastore), HBase and Cassandra datastores. Between these services and components, time series data is communicated using a common TSDR data model. This data model is designed around the abstraction of time series data commonalities. With these services, TSDR is able to collect the data from the data sources and store them into one of the TSDR data stores; HSQLDB, HBase and Cassandra datastores. Data can be retrieved with the Data Query service using the default OpenDaylight RestConf interface or its ODL API interface. TSDR also has integrated support for

ElasticSearch capabilities. TSDR data can also be viewed directly with Grafana (beta) for time series visualization or various chart formats.

2.7.2 RestConf Collector

The RestConf Web Activity Collector records the meaningful REST requests made through the OpenDaylight RESTCONF interface.

2.7.3 How to test the RestConf Collector

- Issue a restconf command that uses either POST,PUT or DELETE. For example, you could call the register-filter RPC of tsdr-syslog-collector.
- Look up data in TSDR database from Karaf.

```
tsdr:list restconf
```

- You should see the request that you have sent, along with its information (URL, HTTP method, requesting IP address and request body)
- Try to send a GET request, then check again, your request should not be registered, because the collector does not register GET requests by default.
- Open the file: "etc/tsdr.restconf.collector.cfg", and add GET to the list of METHODS_TO_LOG, so that it becomes:

```
METHODS_TO_LOG=POST,PUT,DELETE,GET
```

  - Try again to issue your GET request, and check if it was recorded this time, it should be recorder.
  - Try manipulating the other properties (PATHS_TO_LOG (which URLs do we want to log from), REMOTE_ADDRESSES_TO_LOG (which requesting IP addresses do we want to log from) and CONTENT_TO_LOG (what should be in the request's body in order to log it)), and see if the requests are getting logged.
  - Try providing invalid properties (unknown methods for the METHODS_TO_LOG parameter, or the same method repeated multiple times, and invalid regular expressions for the other parameters), then check karaf's log using "log:display". It should tell you that the value is invalid, and that it will use the default value instead.