**SNMP monitoring and easing it with Prometheus.**

What is **SNMP**: **SNMP**abbreviates to Simple Network Management Protocol, which explains what it does.

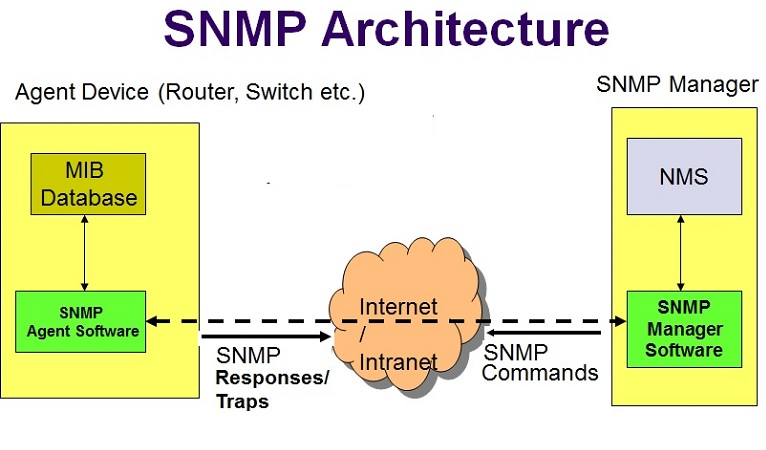
SNMP is used to manage network devices(mostly called as Managed object) by setting value for certain attribute and monitor network devices by polling necessary metrics from the device.



***How it works:***

SNMP comprises simple Client-Server Architecture. The SNMP client running on your Network management solution will be responsible for polling data or setting data. And the SNMP server running on your actual device will respond to SNMP client’s call.

SNMP Agent will not be turned on in network devices by default. The network admin has to enable SNMP if needed.



It is necessary to understand SNMP Mib and SNMP OIDs to use SNMP and poll the metrics that we need.

**Understanding SNMP Mib and OIDs:**

**MIB** stands for **Management Information Base** and is a collection of definitions that define the properties of the managed object within the device to be managed. MIB files are written in an independent format and the object information they contain is organized hierarchically. The various pieces of information can be accessed by SNMP.

**OIDs** or **Object Identifiers** uniquely identify managed objects in the MIB.

Generally, an OID is a long sequence of numbers, coding the nodes, separated by dots. Here is a sample structure of an OID:

eg: To get system up time of a managed device , you can poll this

OID -1.3.6.1.6.3.10.2.1.3 and it will return a the number of seconds since the SNMP engine last.

So OID is to uniquely identify a certain metric and MIB contains tree of OIDs based on the feature and organization of the manufacturer.

**SNMP Versions:**

We can cover the security aspects of SNMP , since it has evolved a lot and to make sure you chose the right one.

SNMP V1 -Anyone with access to the network can poll the device data (Weaker Security)

SNMP V2 — Includes improvements in the areas of performance, security, confidentiality, and manager-to-manager communications.

SNMP V3- Makes data encryption possible. It also allows admins to specify different authentication requirements on a granular basis for managers and agents. This prevents unauthorized authentication and can optionally be used to require encryption for data transfers. So you will be able to set authentication and privacy parameters, so the data will be polled only by authenticated SNMP server and data will be encrypted in the way.

**SNMP Operations:**

*Pulling the data from Managed devices:*

**GetRequest**- To get values for particular OID

**SetRequest**- To set values on particular OID

**GetNextRequest**- To get values from next OID

**GetBulkRequest**- To get values of the MIB tree in bulk

*Pushing data from Managed devices to SNMP server:*

Traps- Event traps Will be sent from a network device to trap server in case any event occurred in Network device ie: Interface down, VPN down and etc. The trap server location and credentials have to be configured in each network devices supposed to be monitored.

For more details SNMP operation, please check [here](https://docstore.mik.ua/orelly/networking_2ndEd/snmp/ch02_06.htm)

**Prometheus and What it does:**

Prometheus is a Time series Database, where the information changing as time moves on can be stored efficiently, queried in a tailored manner, and retrieved quickly than ever.

**Prime features of Prometheus :**

* A multi-dimensional [data model](https://prometheus.io/docs/concepts/data_model/) with time series data identified by metric name and key/value pairs
* A [flexible query language](https://prometheus.io/docs/prometheus/latest/querying/basics/) to leverage this dimensionality
* Time series collection happens via a pull model over HTTP
* Multiple modes of graphing and exposed API to get Time series data.

Let us start with Prometheus installing and we will cover few of advantage using Prometheus TSDB(Time Series Database)

**Installing and running Prometheus:**

Prometheus is an Opensource solution, you can easily download/build [here](https://prometheus.io/download) and

tar xvfz prometheus-\*.tar.gz  
cd prometheus-\*

Prometheus comes with default configuration and you can start you Prometheus server straight away.

./prometheus or in windows prometheus.exe

The Prometheus server will be using the default configuration, if needed you can also point to your own configuration.

./prometheus --config.file=prometheus.yml

Check [here](https://prometheus.io/docs/introduction/first_steps/) for more details

Now Prometheus server is up and running, it is time to for some SNMP Monitoring action

**Installing and running SNMP Exporter:**

***What is an exporter:*** An exporter is a library, that collects data from a source and transforms it into a format that will be accepted by my Prometheus server

***What is an SNMP Exporter: An SNMP Exporter is a tool which collects data from the managed device and exposes it in a format that will be accepted by Prometheus server.***

SNMP Exporter is Opensource and you can get it from [here](https://github.com/prometheus/snmp_exporter/releases) and run it by

./snmp\_exporter

The SNMP Exporter reads a config file “snmp.yml” by default and configuration contains the OIDs to walk/get from device and credentials to use in case if it is SNMP v2 or SNMP v3.

The snmp.yml configuration file is not intended to be handwritten, as there will be a large number of OIDs be specified in the configuration and it is complex to name and labeling the metrics. So we can use a generator to generate the snmp..yml configuration.

This config generator uses NetSNMP to parse MIBs, and generates configs for the snmp\_exporter using them.

**Installing and using snmp.yml generator:**

*Due to the dynamic dependency on NetSNMP, you must build the generator yourself.*

sudo apt-get install build-essential libsnmp-dev snmp-mibs-downloader # Debian-based distros  
go get github.com/prometheus/snmp\_exporter/generator  
cd ${GOPATH-$HOME/go}/src/github.com/prometheus/snmp\_exporter/generator  
go build

Easy for Docker users:

docker build -t snmp-generator .  
docker run -ti \  
 -v $HOME/.snmp/mibs:/root/.snmp/mibs \  
 -v $PWD/generator.yml:/opt/generator.yml:ro \  
 -v $PWD/out/:/opt/ \  
 snmp-generator generate

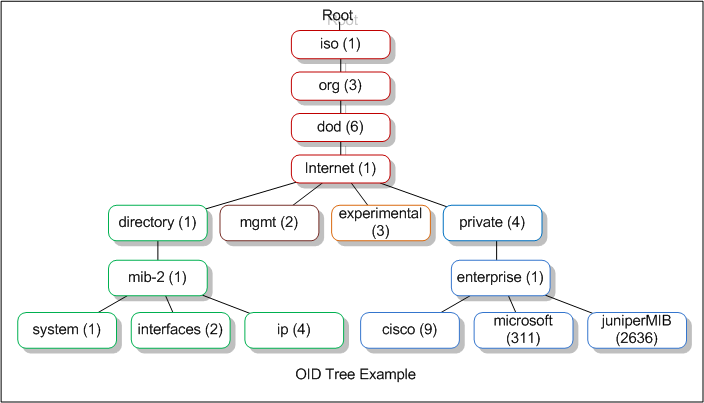
The SNMP Mibs has to be placed in the folder `$HOME/.snmp/mibs`, so NetSNMP can use it.

For easier example, we will create the snmp.yml on our own for a Cisco router.

Cisco:  
 version: 3  
 auth:  
 username: snmpUser  
 password: yourPassword  
 auth\_protocol: SHA  
 priv\_protocol: DES  
 security\_level: authPriv  
 priv\_password: privacyPassword  
 walk:  
 - 1.3.6.1.2.1.1 # sysInfo  
 - 1.3.6.1.2.1.2.2 # ifTable  
 - 1.3.6.1.2.1.31.1.1 # ifXTable  
 metrics:  
 #sysInfo  
 - name: sysUpTime  
 oid: 1.3.6.1.2.1.1.3  
 type: counter  
 lookups:  
 - labels:  
 labelname: sysDescr  
 oid: 1.3.6.1.2.1.1.1.0  
 type: DisplayString  
 - labels:  
 labelname: sysName  
 oid: 1.3.6.1.2.1.1.5.0  
 type: DisplayString  
 - labels:  
 labelname: sysLocation  
 oid: 1.3.6.1.2.1.1.6.0  
 type: DisplayString  
 - labels:  
 labelname: sysContact  
 oid: 1.3.6.1.2.1.1.4.0  
 type: DisplayString  
 #Interfaces  
 #Interface ifIndex  
 - name: ifIndex  
 oid: 1.3.6.1.2.1.2.2.1.1  
 type: gauge  
 indexes:  
 - labelname: ifIndex  
 type: Integer  
 lookups:  
 - labels:  
 - ifIndex  
 labelname: ifDescr  
 oid: 1.3.6.1.2.1.2.2.1.2  
 type: DisplayString  
 - labels:  
 - ifIndex  
 labelname: ifName  
 oid: 1.3.6.1.2.1.31.1.1.1.1  
 type: DisplayString  
 - labels:  
 - ifIndex  
 labelname: ifAlias  
 oid: 1.3.6.1.2.1.31.1.1.1.18  
 type: DisplayString  
 #Interface Type  
 - name: ifType  
 oid: 1.3.6.1.2.1.2.2.1.3  
 type: gauge  
 indexes:  
 - labelname: ifIndex  
 type: Integer  
 lookups:  
 - labels:  
 - ifIndex  
 labelname: ifDescr  
 oid: 1.3.6.1.2.1.2.2.1.2  
 type: DisplayString  
 - labels:  
 - ifIndex  
 labelname: ifName  
 oid: 1.3.6.1.2.1.31.1.1.1.1  
 type: DisplayString  
 - labels:  
 - ifIndex  
 labelname: ifAlias  
 oid: 1.3.6.1.2.1.31.1.1.1.18  
 type: DisplayString

The above example has

1. SNMP module “Cisco”, you can have any number of modules you want.
2. The modules define the SNMP version to use .ie: version: 3
3. The auth: block contains all the SNMP user name and password for communicating with the managed device.
4. The walk: block contains all the OIDs to walk on. As we know the SNMP Mibs are in the tree format, you can select any node of the tree to walk on. So SNMP Exporter will make snmpwalk on particular OID and collect all children nodes on the particular node.



Tree structure of Mib

In the example we walk through 1.3.6.1.2.1.1 because we needed metrics from that tree ie: 1.3.6.1.2.1.1.3(sysUpTime) , 1.3.6.1.2.1.1.1.0(sysDescr), 1.3.6.1.2.1.1.5.0(sysName), 1.3.6.1.2.1.1.6.0(sysLocation), 1.3.6.1.2.1.1.4.0(sysContact)

Instead of getting each and every node separately, we can walk in the parent node and get all metric values from the walk output. It is faster and efficient.

But be cautious that you are not walking on a tree with higher depth or more top-level node, as it would make the SNMP walk much longer time and you won’t be needing that much of data.

5. The metrics: block defines what are the metrics to be collected, type and what lookups should be applied after collected.  
eg:

#Interface Speed  
 - name: ifSpeed  
 oid: 1.3.6.1.2.1.2.2.1.5  
 type: gauge  
 indexes:  
 - labelname: ifIndex  
 type: Integer  
 lookups:  
 - labels:  
 - ifIndex  
 labelname: ifDescr  
 oid: 1.3.6.1.2.1.2.2.1.2  
 type: DisplayString  
 - labels:  
 - ifIndex  
 labelname: ifName  
 oid: 1.3.6.1.2.1.31.1.1.1.1  
 type: DisplayString

A router may consist of multiple interfaces and each interface will be respective speed. Let's say if the particular Cisco router has 5 interfaces

(*if1, if2, if 3, if4, etho1*).

So to collect interface speed of the interfaces, we have to check the walk on the parent node of interface OID 1.3.6.1.2.1.2.2.1.5 and pick the results based on the interface index. That is what specified in indexes block.

We will be having snmpwalk results like:

IF-MIB::ifSpeed.1 = Gauge32: 10000000  
IF-MIB::ifSpeed.2 = Gauge32: 100000000  
IF-MIB::ifSpeed.3 = Gauge32: 100000000  
IF-MIB::ifSpeed.4 = Gauge32: 0  
IF-MIB::ifSpeed.5 = Gauge32: 0

where .1, .2, .3 are the interface indexes.

lookups block specifies what values to be added to the labeled dimensions. These lookup values should not be a frequently changing values since that might create different time series data whenever a change in any one of the label values.

*Prometheus fundamentally stores all data as*[time series](http://en.wikipedia.org/wiki/Time_series)*: streams of timestamped values belonging to the same metric and the same set of labeled dimensions. Besides stored time series, Prometheus may generate temporary derived time series as the result of queries.*

So, this is how we should model the snmp.yml file.

Now that we have SNMP Exporter configuration ready, we can check with an example how it works using

[***http://<SNMPExporterIP>:9116/snmp?target=<(IP)1.2.3.4***](http://localhost:9116/snmp?target=1.2.3.4)***>&module=<SNMPModule>***

Calling the above URL should provide you with all metric values with the timestamp, which we can use in Prometheus.

**Make Prometheus collect data:**

Now that we have utility to collect SNMP data let’s create a Job in Prometheus to use the utility (SNMP Exporter) to collect data and store the values in Prometheus Time Series Database.

***Configuring Prometheus:***

Prometheus configuration has two important parts for basic SNMP monitoring.

1. global
2. scrape\_configs

The default prometheus.yml file will be used when starting the server and let’s take a look at it.

global:  
 scrape\_interval: 15s  
 evaluation\_interval: 15s  
  
rule\_files:  
 # - "first.rules"  
 # - "second.rules"  
  
scrape\_configs:  
 - job\_name: prometheus  
 static\_configs:  
 - targets: ['localhost:9090']

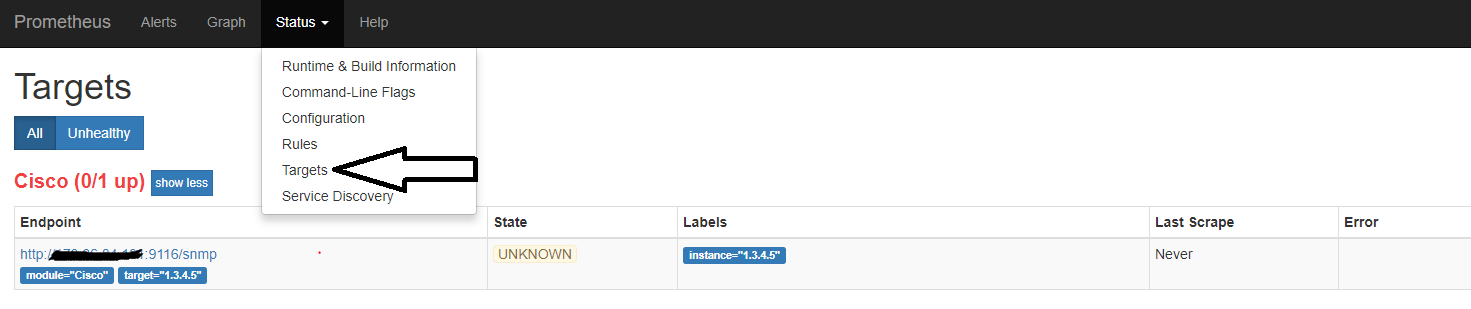
The global section describes the poll interval (scrape\_interval) of Prometheus server and The evaluation\_interval option controls how often Prometheus will evaluate rules.

The scrape\_configs consists of what devices should be monitored by Prometheus. You can create any number of jobs (just to isolate and use different configurations(scrape interval, modules..etc) for different devices).  
Each job section consists of targets(devices) to be polled and scrape interval, scrape timeout and modules to be used. So let us take look at a modified Prometheus configuration file.

# Sample config for Prometheus.  
global:  
 scrape\_interval: 5m  
 scrape\_timeout: 10s  
 evaluation\_interval: 1m  
  
# A scrape configuration containing exactly one endpoint to scrape:  
scrape\_configs:  
 # Cisco  
 - job\_name: 'Cisco'  
 scrape\_interval: 120s  
 scrape\_timeout: 120s  
 file\_sd\_configs:  
 - files :  
 - /etc/prometheus/targetCisco.yml  
 # SNMP device.  
 metrics\_path: /snmp  
 params:  
 module: [Cisco] #which OID's we will be querying in  
 relabel\_configs:  
 - source\_labels: [\_\_address\_\_]  
 target\_label: \_\_param\_target  
 - target\_label: \_\_address\_\_  
 replacement: 127.0.0.1:9116 # The SNMP exporter's real hostname:port.

So running the Prometheus server now would run a Job named Cisco to poll the devices specified in the scrape\_configs(static\_configs or file\_sd\_configs ) and collect data to store in TSDB.

We can check it in Prometheus server UI -> Status -> Targets



Once the status is up it means the Prometheus server was able to use SNMP Exporter to collect data from the device.

We should we able to see data in Prometheus using Query and visualize the data in Graph or plain console.

Thanks for reading this far !!! :-)

I will post more blogs on other features and best practices of Prometheus and SNMP monitoring in upcoming blogs.