# SNMP Set-up with Logstash

## From the pfsense webinterface unable SNMP

- Services → SNMP
- Ckeck up Enable the SNMP Daemon and its controls
- Set Polling Port → 161
- Set Read Community String ⟶ Public
- Set Bind Interface ⟶ WAN

## Add Firewall to allow SNMP traffic to the server machine

You may need to add a firewall rule on the pfSense box to allow UDP traffic on port 161 from the management network or monitoring system IP address.

- **Firewall** > **Rules**.
- Select the Appropriate Interface → WAN
- Add a New Rule → Click the **Add** button to create a new firewall rule
- Action → Pass
- Interface → WAN
- Address Family → IPv4
- Protocol → UDP
- Source → Address or Alias ⟶ Server machine IP address
- Destination → Address or Alias → SNMP server vIP
- Destination Port Range → From SNMP 161 To SNMP 161
- Save

Under the logstash: on the values.yaml add the plugin to install

```
logstash:
  image: "docker.elastic.co/logstash/logstash"
  imageTag: "8.15.0"
  plugins:
    - logstash-input-snmp
```

Under the input section of the pipline-config.yaml add snmp part

```
input {
      beats {
        port => 5044
      }
      snmp {
        get => ["1.3.6.1.2.1.1.3.0"]
        hosts => [{host => "udp:10.8.9.184/161" community =>
"public"}]
      }
    }
```

Apply Changes

- helm uninstall logstash -n elasticsearch
- helm dep build logstash-parent/ -n elasticsearch
- helm install logstash logstash-parent/ -n elasticsearch
- kubectl get pod -n elasticsearch -w

Troubleshooting

```
error invoking `get` operation: timeout sending snmp get requ
est to target 10.8.9.184/161, ignoring.
```

ping the SNMP server if no response check firewall prevention

## Understand The Data of SNMP

1- To visualize the data of kibana

- Create Data View for SNMP from Kibana → Management → Stack Management → Kibana → Data View
- Click Create Data View type the index patten snmp-data*
- Go to Discover from the top left under data view select snmp-data*
- The fields is now displayed on the left which you can select to display on the table
- @timestamp: This is date and time of when the data is being collected and ingested
- @version: default 1 refer to a version of log
- host.ip: refer the device in which the data was collected
- **iso.org.dod.internet.mgmt.mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifAlias.1 : This refer a manual description or label of the device network ports or interfaces**
  - The fields starting with `iso.org.dod.internet.mgmt.mib-2` refer to the structure of SNMP data, as per the **Management Information Base (MIB)** hierarchy. The MIB is a collection of information organized hierarchically and used by SNMP for managing network devices.
  - `ifAlias.1` refers to the alias for the first network interface.
- iso.org.dod.internet.mgmt.mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCInOctets.1
  - `ifHCInOctets`: This is an SNMP object from the `ifXTable` (Extended Interface Table) that stores the total number of **octets (bytes) received** on a particular network interface, including both unicast and non-unicast packets.
  - The `.1` at the end refers to the first network interface (port).
  - This field tells you the total amount of data (in octets) that has been received on the **first** network interface since the interface started operating.
- **iso.org.dod.internet.mgmt.mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifHCOutOctets.1**

    - `ifHCOutOctets`: This is an SNMP object from the `ifXTable` (Extended Interface Table) that stores the total number of **octets (bytes) sent** on a particular network interface, including both unicast and non-unicast packets.
- **iso.org.dod.internet.mgmt.mib-2.ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.1**
    - `ifName`: This SNMP object provides the **human-readable name** assigned to a network interface.
- iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifAdminStatus.1
    - `ifAdminStatus.1` tells you the administrative status of the first network interface, whether it's enabled, disabled, or in a testing mode.
    - **1**: Interface is **up** (enabled by the administrator).
    - **2**: Interface is **down** (disabled by the administrator).
    - **3**: Interface is in a **testing** state.
- iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr.1
    - provides a **description** of the first network interface (or port). This description typically includes the name or type of the interface, which can help identify what the interface is used for in your network.

  For example, it might return values like:

    - **"GigabitEthernet0/1"** for a Gigabit Ethernet interface.
    - **"FastEthernet0/1"** for a Fast Ethernet interface.
    - **"Loopback"** for a loopback interface.
- iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOperStatus.22
    - indicates the **operational status** of the 22nd network interface on the device. The operational status can help you determine if the interface is functioning correctly or if there are issues.

  The possible values for this field typically include:

    - **1 (up)**: The interface is operational and able to send and receive data.

    - **2 (down)**: The interface is not operational (i.e., it is down and cannot send or receive data).
    - **3 (testing)**: The interface is in a testing state.
- iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifType.1
    - indicates the **type** of the 1st network interface on the device. The interface type provides information about the nature of the network interface and its functionality.
    - Common values for the `ifType` field include:
    - **1 (other)**: Represents an unknown interface type.
    - **2 (regular 1822)**: Represents a 1822 protocol.
    - **6 (ethernetCsmacd)**: Represents an Ethernet interface (IEEE 802.3).
    - **24 (pointToPointSerial)**: Represents a point-to-point serial interface.
    - **32 (hdlc)**: Represents a High-Level Data Link Control (HDLC) interface.
    - **137 (frameRelay)**: Represents a Frame Relay interface.
- **iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0**
    - provides a description of the system of the specific network device from which the SNMP (Simple Network Management Protocol) data is being collected. This could be any device that participates in a network, including:
    - **Routers**: Such as Cisco, Juniper, or other manufacturers' devices.
    - **Switches**: Network switches that manage data traffic within a local area network.
    - **Servers**: Physical or virtual machines running various operating systems.
    - **Firewalls**: Security devices that monitor and control incoming and outgoing network traffic.
    - **Networked printers**: Devices that can be accessed over a network for printing.

  - The description provided will characterize the specific type of the device (like its brand, model, and OS version)
- **iso.org.dod.internet.mgmt.mib-2.system.sysName.0**
  - Represents the **system name** of the network device. This name is often a human-readable identifier assigned to the device, which can be used for various management and monitoring purposes. Here's what it signifies:
- **iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance**
  - This field indicates the amount of time (in hundredths of a second) that the device has been running since its last reboot. A value of `0` indicates the device has just been initialized.
- To Verify of the device is working or not we got to check
  - `ifOperStatus` and `ifAdminStatus`

Filtering the SNMP data Using Logstash Pipeline-config.yaml filter

```
  main_02_filter.conf: |
    filter {
      if "pfsense-snmp" in [tags] {
        ruby {
          code => '
            interfaces = {}
            event.to_hash.each do |k, v|
              if k.start_with?("iso.org.dod.internet.mgmt.mib
-2")
                oid_parts = k.split(".")
                interface_index = oid_parts.last
                metric_name = oid_parts[-2]

                interfaces[interface_index] ||= {"InterfaceN
o" => interface_index}
                case metric_name
                when "ifAlias"
```

```
                  interfaces[interface_index]["ifAlias"] = v
                when "ifHCInOctets"
                  interfaces[interface_index]["ifHCInOctets"]
= v
                when "ifHCOutOctets"
                  interfaces[interface_index]["ifHCOutOctet
s"] = v
                when "ifAdminStatus"
                  interfaces[interface_index]["ifAdminStatu
s"] = v
                when "ifType"
                  interfaces[interface_index]["ifType"] = v
                when "ifDescr"
                  interfaces[interface_index]["ifDescr"] = v
                when "ifName"
                  interfaces[interface_index]["ifName"] = v
                when "ifOperStatus"
                  interfaces[interface_index]["ifOperStatus"]
= v
                end
              end
            end

            interfaces.each do |_, interface|
              if interface["ifAdminStatus"] == 1 &&
                interface["ifOperStatus"] == 2 &&
                interface["ifAlias"] && !interface["ifAlia
s"].empty? && interface["ifAlias"] != "--"
                interface["ifDeviceStatus"] = "Down"
              else
                interface["ifDeviceStatus"] = ""
              end
            end
            event.set("interfaces", interfaces.values)
          '
        }
        split {
          field => "interfaces"
        }

        mutate {
          rename => { "[interfaces][InterfaceNo]" => "InterfaceNo" }
          rename => { "[interfaces][ifAlias]" => "ifAlias" }
          rename => { "[interfaces][ifHCInOctets]" => "ifHCInOctets" }
          rename => { "[interfaces][ifHCOutOctets]" => "ifHCOutOctets" }
          rename => { "[interfaces][ifAdminStatus]" => "ifAdminStatus" }
          rename => { "[interfaces][ifType]" => "ifType" }
          rename => { "[interfaces][ifDescr]" => "ifDescr" }
          rename => { "[interfaces][ifName]" => "ifName" }
          rename => { "[interfaces][ifOperStatus]" => "ifOperStatus" }
          rename => { "[interfaces][ifDeviceStatus]" => "ifDeviceStatus" }
          remove_field => ["interfaces"]
        }
      }
    }
```