APPLICATION NOTE

How To Run SimpleAgentPro® In A Docker Container CLI Version (Non-GUI)

This document describes how to setup and run SimpleAgentPro in a docker environment. This version installs the container as a non-GUI SAPro. The SAPro can be controlled using cli commands.

Prerequisites:

This tutorial is based on Ubuntu 16.04 but Docker should work on any 64-bit Ubuntu version. Additionally, your kernel must be 3.10 at minimum.

Once you got access to your server, run the command below to check your kernel version, to make sure the kernel is at least 3.10 or above.

```
ssoft@ubuntu-1604:~$ uname -r
4.4.0-24-generic
```

Installing Docker on Ubuntu 16.04:

Included in Ubuntu 16.04 Xenial Xerus is version 1.6.1 of Docker but it is not the latest version. Below are the steps to install the latest Docker from the official Docker repository.

To make sure the downloads are valid, add the GPG key for the official Docker repository to your system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

Add the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.
dock er.com/linux/ubuntu $(lsb_release -cs) stable"
```

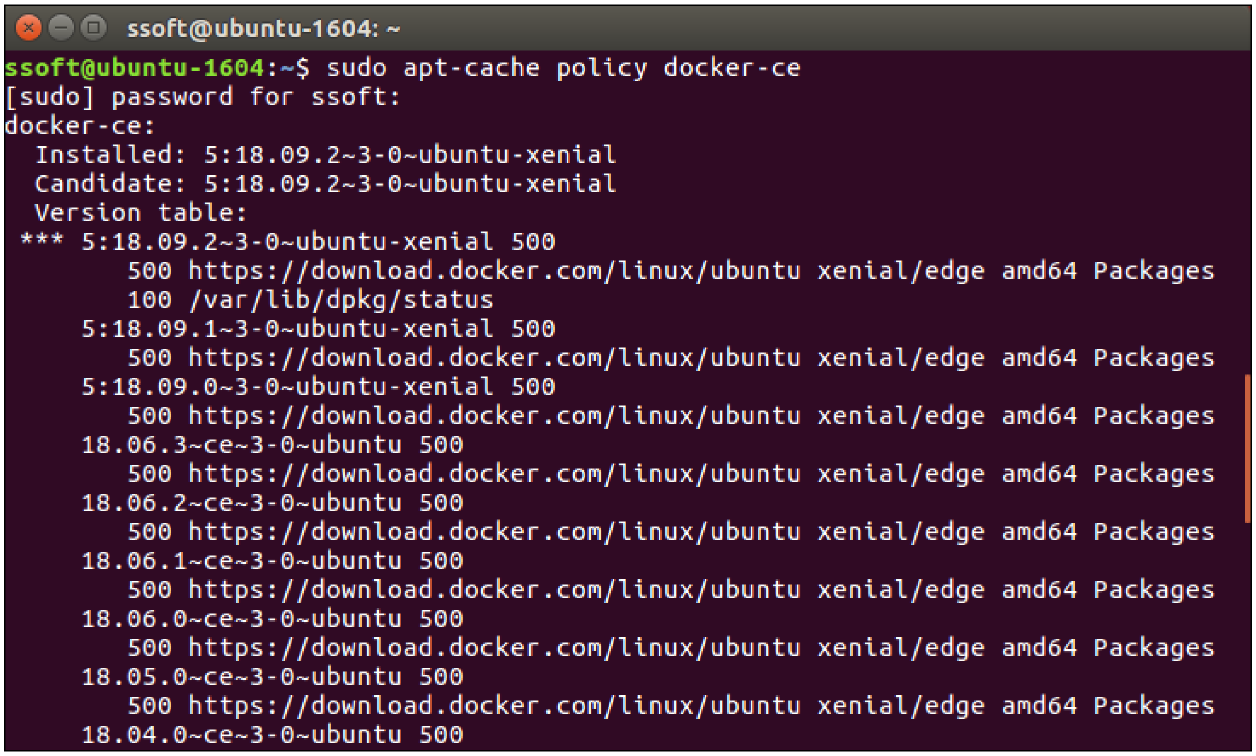
Update the package database with the Docker packages:

```
sudo apt-get update
```

Make sure docker official repositories are added and used instead of default Ubuntu.

```
sudo apt-cache policy docker-ce
```

You should see output similar to the follow:



```
ssoft@ubuntu-1604:~$ sudo apt-cache policy docker-ce
[sudo] password for ssoft:
docker-ce:
  Installed: 5:18.09.2~3-0~ubuntu-xenial
  Candidate: 5:18.09.2~3-0~ubuntu-xenial
  Version table:
 *** 5:18.09.2~3-0~ubuntu-xenial 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
        100 /var/lib/dpkg/status
     5:18.09.1~3-0~ubuntu-xenial 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     5:18.09.0~3-0~ubuntu-xenial 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.06.3~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.06.2~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.06.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.06.0~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.05.0~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu xenial/edge amd64 Packages
     18.04.0~ce~3-0~ubuntu 500
```

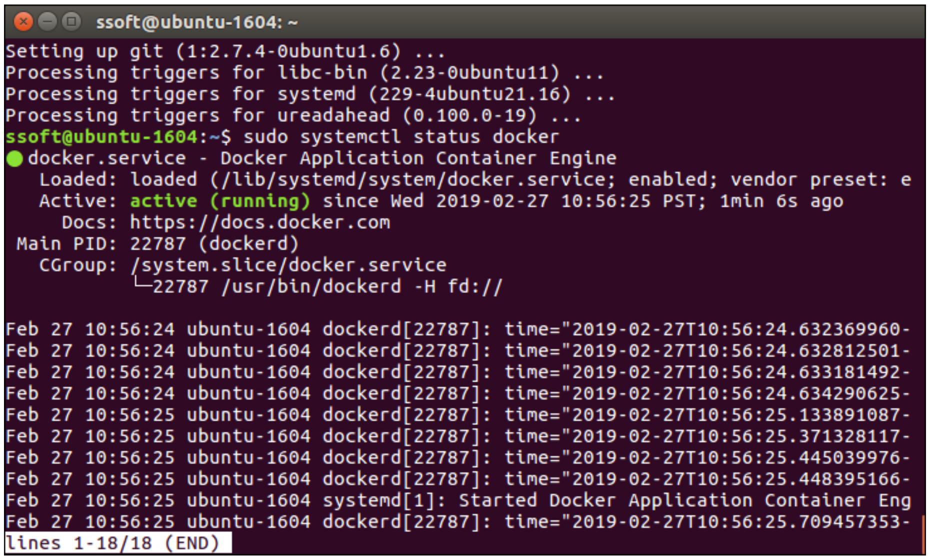
Install Docker:

```
sudo apt-get install -y docker-ce
```

Check to make sure Docker is running:

```
sudo systemctl status docker
```

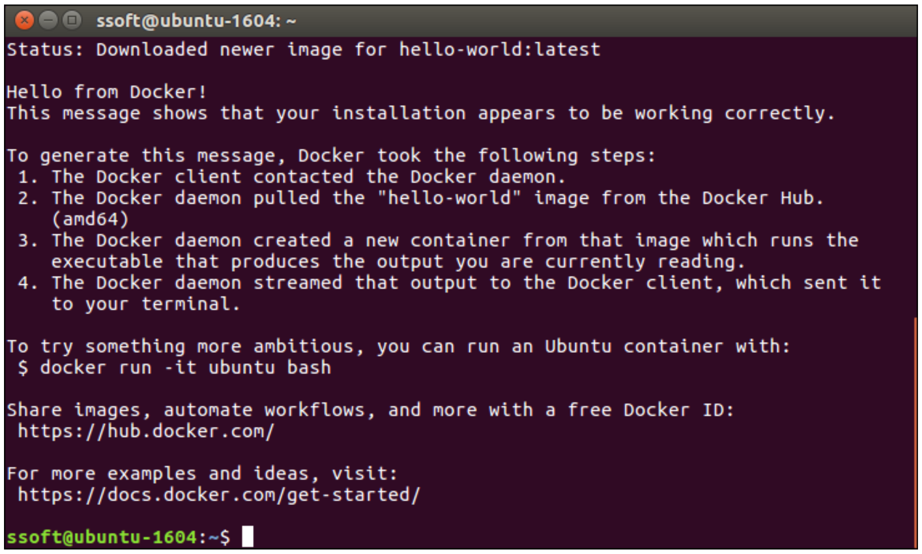
The output should be similar to the following, showing that the service is active and running:



```
Setting up git (1:2.7.4-0ubuntu1.6) ...
Processing triggers for libc-bin (2.23-0ubuntu11) ...
Processing triggers for systemd (229-4ubuntu21.16) ...
Processing triggers for ureadahead (0.100.0-19) ...
ssoft@ubuntu-1604:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
   Active: active (running) since Wed 2019-02-27 10:56:25 PST; 1min 6s ago
     Docs: https://docs.docker.com
 Main PID: 22787 (dockerd)
   CGroup: /system.slice/docker.service
           └─22787 /usr/bin/dockerd -H fd://

Feb 27 10:56:24 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:24.632369960-
Feb 27 10:56:24 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:24.632812501-
Feb 27 10:56:24 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:24.633181492-
Feb 27 10:56:24 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:24.634290625-
Feb 27 10:56:25 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:25.133891087-
Feb 27 10:56:25 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:25.371328117-
Feb 27 10:56:25 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:25.445039976-
Feb 27 10:56:25 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:25.448395166-
Feb 27 10:56:25 ubuntu-1604 systemd[1]: Started Docker Application Container Eng
Feb 27 10:56:25 ubuntu-1604 dockerd[22787]: time="2019-02-27T10:56:25.709457353-
lines 1-18/18 (END)
```

Run the hello-world sample image to verify that ‘docker’ is installed correctly on your system:

```
$ sudo docker run hello-world
```



```
ssoft@ubuntu-1604: ~
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

ssoft@ubuntu-1604:~$
```

Install and Run SimpleAgentPro Docker Image

Note

This tutorial is based on a SimpleAgent Pro Docker Image.

Contact SimpleSoft for information on how to download the SAPro Image, sapro_cli_docker_linux_245_tar.zip.

Download the sapro_cli_docker_linux_245_tar.zip file to the /tmp folder.

Create a new folder under the docker folder.

```
> cd /docker

> mkdir sapro
```

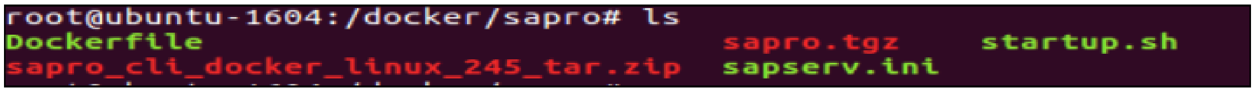
Copy the sapro_cli_docker_linux_245_tar.zip file into the sapro folder.

```
> cd sapro

> cp /tmp/sapro_cli_docker_linux_245_tar.zip .
```

Untar the downloaded file to the sapro folder.

```
> tar xvfz sapro_cli_docker_linux_245_tar.zip
```

You will see the following files in the sapro folder.



```
root@ubuntu-1604:/docker/sapro# ls
Dockerfile                             sapro.tgz      startup.sh
sapro_cli_docker_linux_245_tar.zip     sapserv.ini
```

Update the license server information in sapserv.ini file:

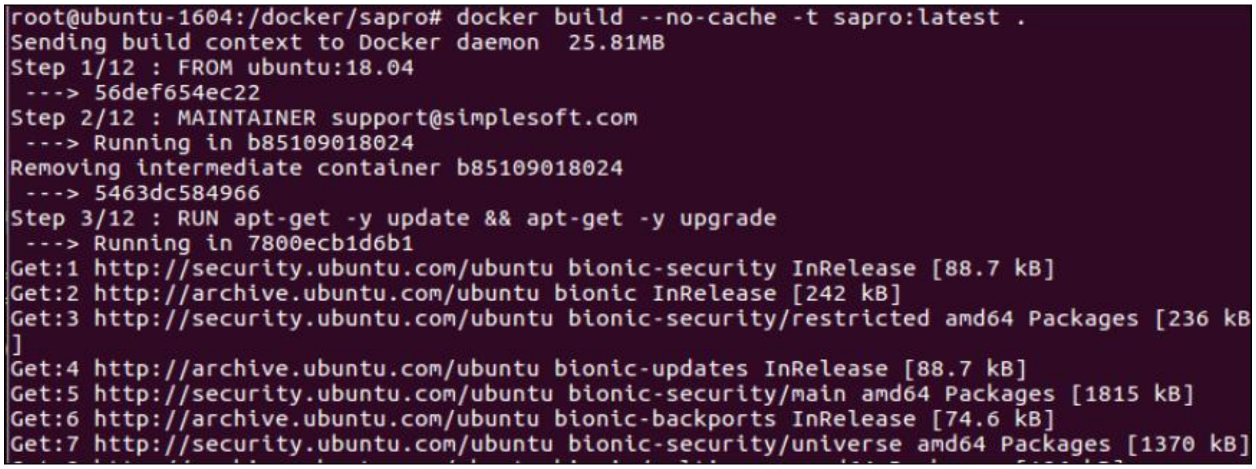
Edit the sapserv.ini file to have correct information for the SAPro license. Below are the default values:

```
[SSMONINFO]
LICSERVERADDR=192.168.0.1
LICSERVERPORT=7878
REGID=98021
DEVICECOUNT=500
```

Build the SAPro image:

Make sure you are still in the sapro directory. The docker build command will build the sapro container based on the information in the Docker file.

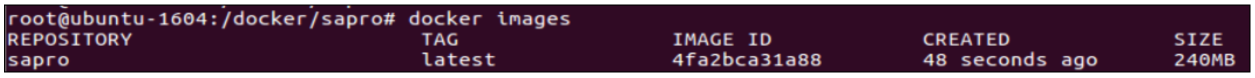
```
> docker build --no-cache -t sapro:latest .
```



```
root@ubuntu-1604:/docker/sapro# docker build --no-cache -t sapro:latest .
Sending build context to Docker daemon  25.81MB
Step 1/12 : FROM ubuntu:18.04
 ---> 56def654ec22
Step 2/12 : MAINTAINER support@simplesoft.com
 ---> Running in b85109018024
Removing intermediate container b85109018024
 ---> 5463dc584966
Step 3/12 : RUN apt-get -y update && apt-get -y upgrade
 ---> Running in 7800ecb1d6b1
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [236 kB
]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1815 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [1370 kB]
```

After the build is completed, we could see the image in the local repository.

```
> docker images
```



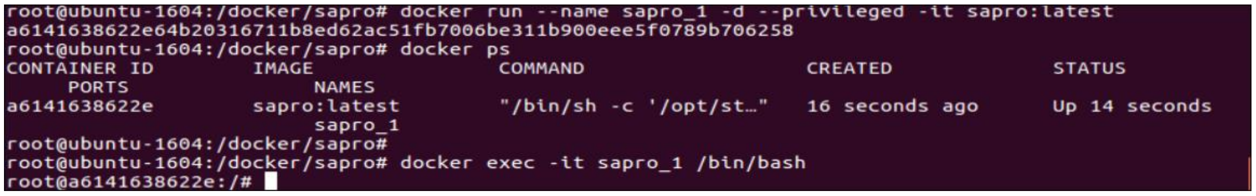
```
root@ubuntu-1604:/docker/sapro# docker images
REPOSITORY                     TAG                 IMAGE ID            CREATED             SIZE
sapro                          latest              4fa2bca31a88        48 seconds ago      240MB
```

Run SAPro Container #1:

```
> docker run --name sapro_1 -d --privileged -it sapro:latest
```

Execute an interactive bash shell:

```
> docker exec –it sapro_1 /bin/bash
```



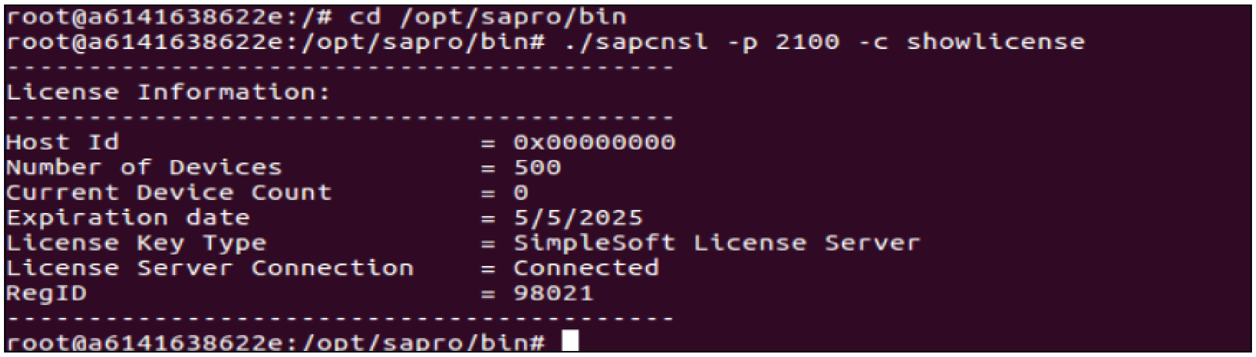
```
root@ubuntu-1604:/docker/sapro# docker run --name sapro_1 -d --privileged -it sapro:latest
a6141638622e64b20316711b8ed62ac51fb7006be311b900eee5f0789b706258
root@ubuntu-1604:/docker/sapro# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
     PORTS               NAMES
a6141638622e        sapro:latest        "/bin/sh -c '/opt/st…"   16 seconds ago      Up 14 seconds
                         sapro_1
root@ubuntu-1604:/docker/sapro#
root@ubuntu-1604:/docker/sapro# docker exec -it sapro_1 /bin/bash
root@a6141638622e:/#
```

--privileged option is required because SAPro will add and delete interfaces which requires elevated privileges.

Test SAPro Container #1:

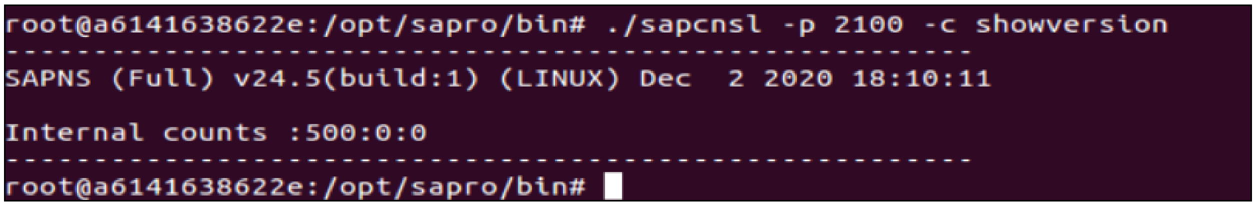
Run the following commands in the bash shell to make sure SAPro #1 is running:

```
> cd /opt/sapro/bin

> ./sapcnsl –p 2100 –c showlicense
```



```
root@a6141638622e:/# cd /opt/sapro/bin
root@a6141638622e:/opt/sapro/bin# ./sapcnsl -p 2100 -c showlicense
----------------------------------------
License Information:
----------------------------------------
Host Id                         = 0x00000000
Number of Devices               = 500
Current Device Count            = 0
Expiration date                 = 5/5/2025
License Key Type                = SimpleSoft License Server
License Server Connection       = Connected
RegID                           = 98021
----------------------------------------
root@a6141638622e:/opt/sapro/bin#
```

```
> ./sapcnsl –p 2100 –c showversion
```

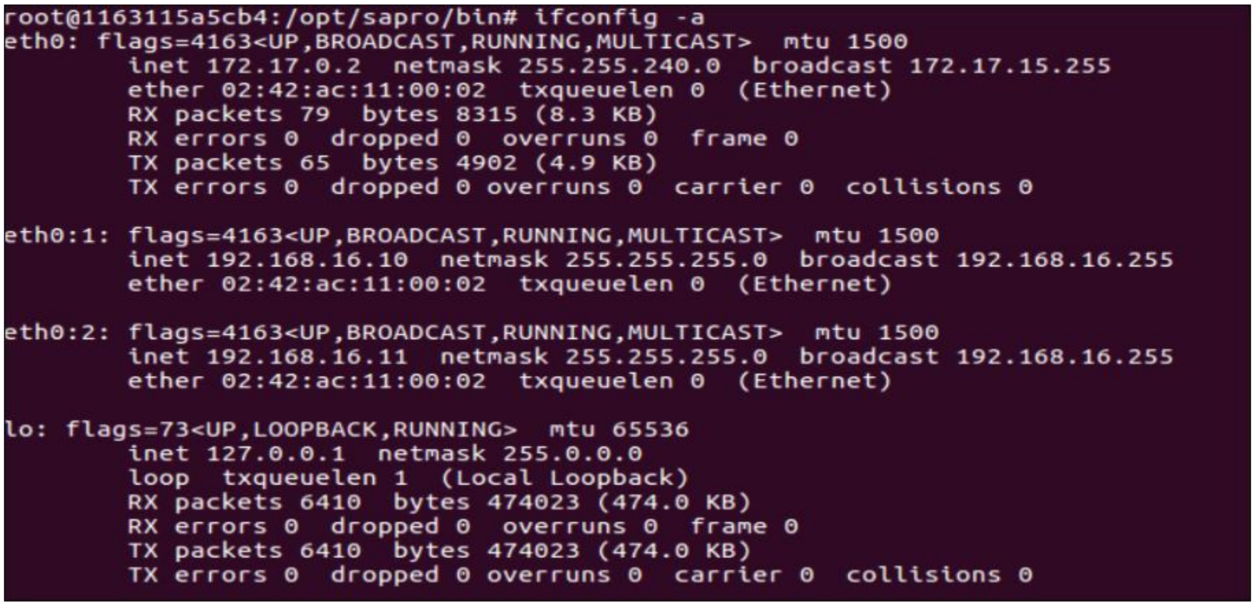


```
root@a6141638622e:/opt/sapro/bin# ./sapcnsl -p 2100 -c showversion
------------------------------------------------------
SAPNS (Full) v24.5(build:1) (LINUX) Dec  2 2020 18:10:11

Internal counts :500:0:0
------------------------------------------------------
root@a6141638622e:/opt/sapro/bin#
```

You can use the following commands to start and test the devices in the sample.map:

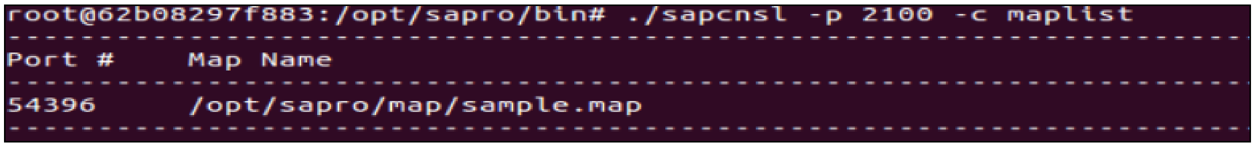
```
> ./sapcnsl –p 2100 –m /opt/sapro/map/sample.map –c start
```

```
> ifconfig
```



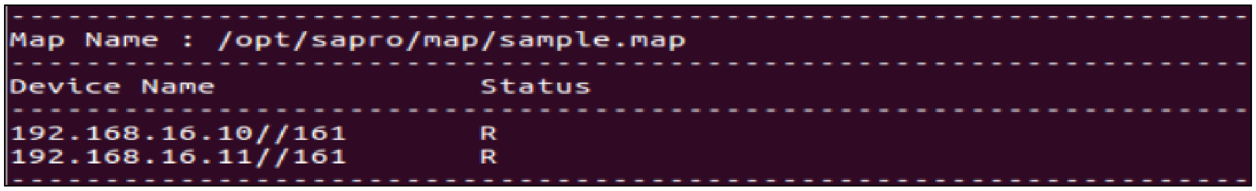
```
root@1163115a5cb4:/opt/sapro/bin# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.240.0  broadcast 172.17.15.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 79  bytes 8315 (8.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 65  bytes 4902 (4.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.16.10  netmask 255.255.255.0  broadcast 192.168.16.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)

eth0:2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.16.11  netmask 255.255.255.0  broadcast 192.168.16.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1  (Local Loopback)
        RX packets 6410  bytes 474023 (474.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6410  bytes 474023 (474.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
> ./sapcnsl –p 2100 –c maplist
```



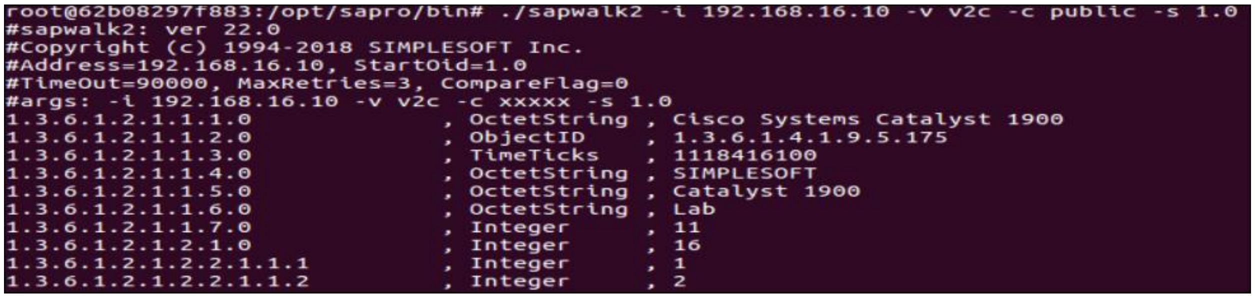
```
root@62b08297f883:/opt/sapro/bin# ./sapcnsl -p 2100 -c maplist
--------------------------------------------------------------------
Port #     Map Name
--------------------------------------------------------------------
54396      /opt/sapro/map/sample.map
--------------------------------------------------------------------
```

```
> ./sapcnsl –p 2100 –c showall
```



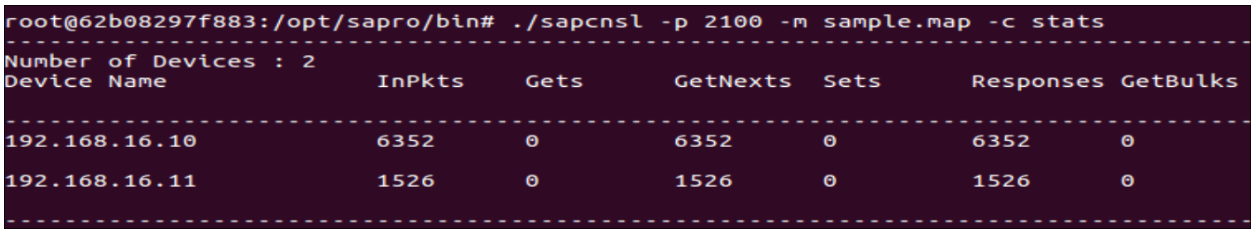
```
--------------------------------------------------------------------
Map Name : /opt/sapro/map/sample.map
--------------------------------------------------------------------
Device Name                   Status
--------------------------------------------------------------------
192.168.16.10//161            R
192.168.16.11//161            R
--------------------------------------------------------------------
```

```
> ./sapwalk2 –i 192.168.16.10 –v v2c –c public –s 1.0
```



```
root@62b08297f883:/opt/sapro/bin# ./sapwalk2 -i 192.168.16.10 -v v2c -c public -s 1.0
#sapwalk2: ver 22.0
#Copyright (c) 1994-2018 SIMPLESOFT Inc.
#Address=192.168.16.10, StartOid=1.0
#TimeOut=90000, MaxRetries=3, CompareFlag=0
#args: -i 192.168.16.10 -v v2c -c xxxxx -s 1.0
1.3.6.1.2.1.1.1.0              , OctetString , Cisco Systems Catalyst 1900
1.3.6.1.2.1.1.2.0              , ObjectID    , 1.3.6.1.4.1.9.5.175
1.3.6.1.2.1.1.3.0              , TimeTicks   , 1118416100
1.3.6.1.2.1.1.4.0              , OctetString , SIMPLESOFT
1.3.6.1.2.1.1.5.0              , OctetString , Catalyst 1900
1.3.6.1.2.1.1.6.0              , OctetString , Lab
1.3.6.1.2.1.1.7.0              , Integer     , 11
1.3.6.1.2.1.2.1.0              , Integer     , 16
1.3.6.1.2.1.2.2.1.1.1          , Integer     , 1
1.3.6.1.2.1.2.2.1.1.2          , Integer     , 2
```

```
> ./sapcnsl -p 2100 -m sample.map -c stats
```



```
root@62b08297f883:/opt/sapro/bin# ./sapcnsl -p 2100 -m sample.map -c stats
------------------------------------------------------------------------------
Number of Devices : 2
Device Name           InPkts    Gets      GetNexts  Sets      Responses GetBulks

------------------------------------------------------------------------------
192.168.16.10         6352      0         6352      0         6352      0

192.168.16.11         1526      0         1526      0         1526      0

------------------------------------------------------------------------------
```

Above are tests run locally on the first SAPro container (Container #1). Below we will start another SAPro container (Container #2) and use this container like an NMS to communicate with Container #1. You could start more SAPro containers using the same procedures shown below.

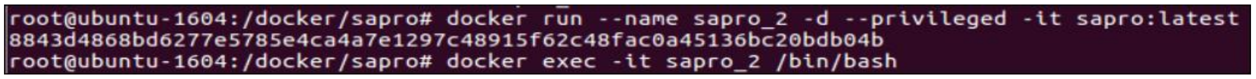
Run SAPro Container #2 as a Poller (i.e., NMS):

Open another terminal on the host and run the following commands.

```
> docker run --name sapro_2 -d --privileged -it sapro:latest
```

Execute an interactive bash shell:

```
> docker exec -it sapro_2 /bin/bash
```

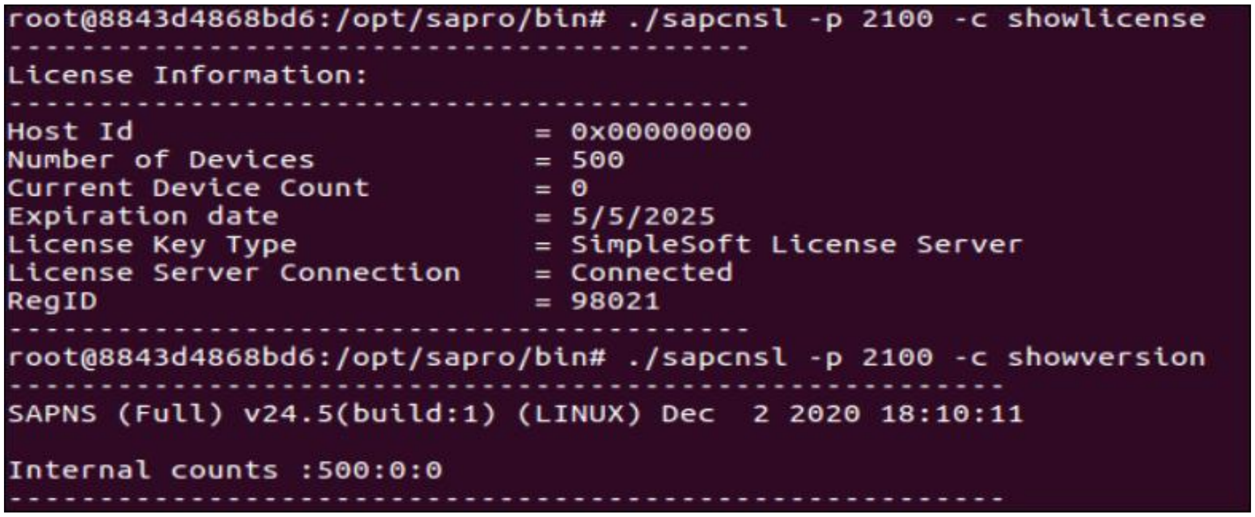


```
root@ubuntu-1604:/docker/sapro# docker run --name sapro_2 -d --privileged -it sapro:latest
8843d4868bd6277e5785e4ca4a7e1297c48915f62c48fac0a45136bc20bdb04b
root@ubuntu-1604:/docker/sapro# docker exec -it sapro_2 /bin/bash
```

Test SAPro Container #2:

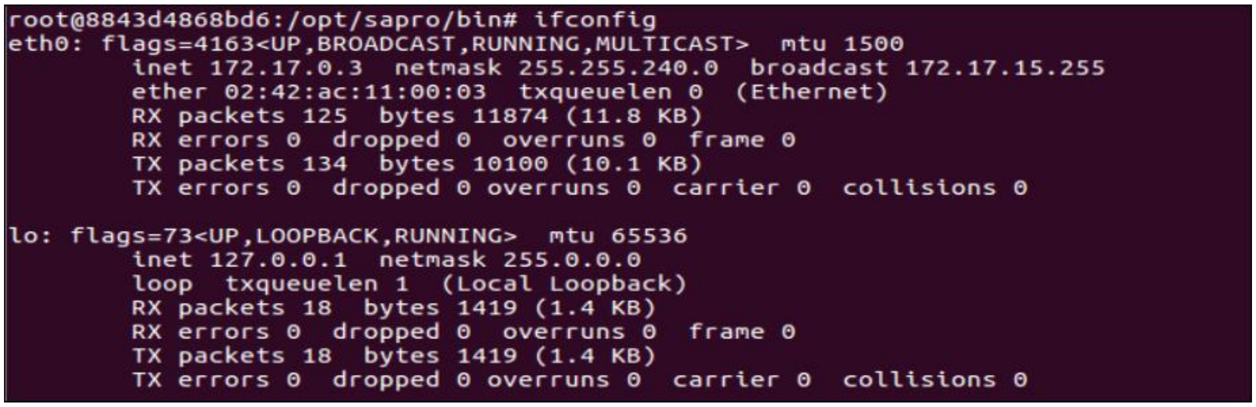
Run the following commands in the bash shell to make sure SAPro #2 is running:

```
> cd /opt/sapro/bin

> ./sapcnsl -p 2100 -c showlicense

> ./sapcnsl -p 2100 -c showversion
```



```
root@8843d4868bd6:/opt/sapro/bin# ./sapcnsl -p 2100 -c showlicense
----------------------------------------
License Information:
----------------------------------------
Host Id                           = 0x00000000
Number of Devices                 = 500
Current Device Count              = 0
Expiration date                   = 5/5/2025
License Key Type                  = SimpleSoft License Server
License Server Connection         = Connected
RegID                             = 98021
----------------------------------------
root@8843d4868bd6:/opt/sapro/bin# ./sapcnsl -p 2100 -c showversion
-------------------------------------------------------
SAPNS (Full) v24.5(build:1) (LINUX) Dec  2 2020 18:10:11

Internal counts :500:0:0
-------------------------------------------------------
```

```
> ifconfig
```



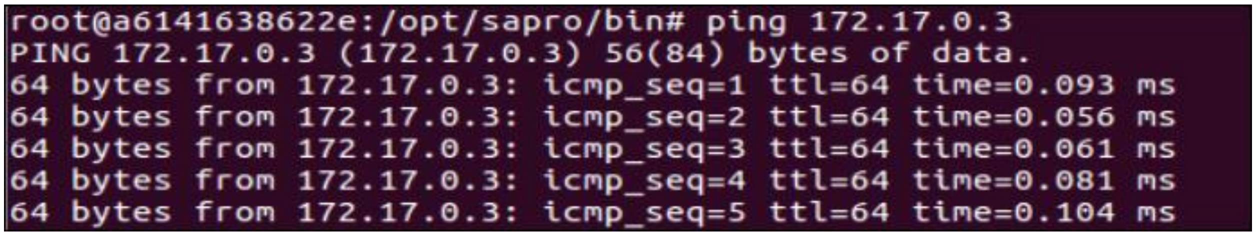
```
root@8843d4868bd6:/opt/sapro/bin# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.240.0  broadcast 172.17.15.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 125  bytes 11874 (11.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 134  bytes 10100 (10.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1  (Local Loopback)
        RX packets 18  bytes 1419 (1.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1419 (1.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note:
IP Address of SAPro Container #1: 172.17.0.2
IP Address of SAPro Container #2: 172.17.0.3

Check Network Connections between the 2 SAPro Containers:

From SAPro Container 1:

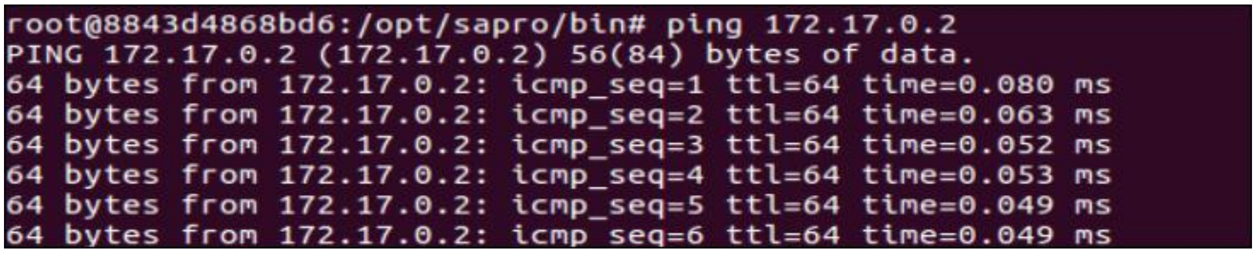
```
> ping 172.17.0.3
```



```
root@a6141638622e:/opt/sapro/bin# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.093 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 172.17.0.3: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 172.17.0.3: icmp_seq=4 ttl=64 time=0.081 ms
64 bytes from 172.17.0.3: icmp_seq=5 ttl=64 time=0.104 ms
```

From SAPro Container 2:

```
> ping 172.17.0.2
```



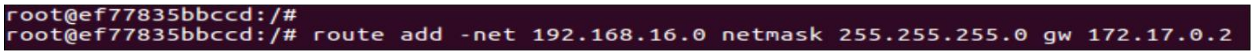
```
root@8843d4868bd6:/opt/sapro/bin# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.053 ms
64 bytes from 172.17.0.2: icmp_seq=5 ttl=64 time=0.049 ms
64 bytes from 172.17.0.2: icmp_seq=6 ttl=64 time=0.049 ms
```

Poll simulated SNMP devices running on SAPro Container #1:

Note:
As shown in the Container #1's setup in previous section, 2 simulated SNMP devices are running with IP addresses: 192.168.16.10 and 192.168.16.11

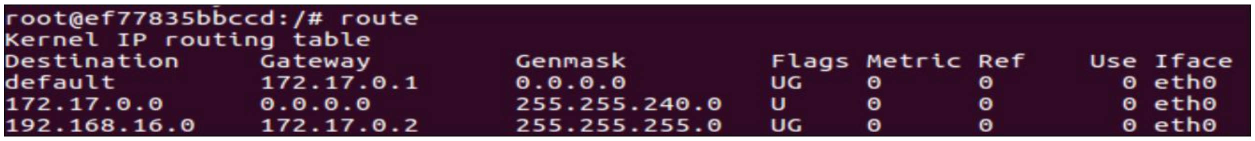
In order to communicate from Container #2 to Container #1, you need to set up the correct route in Container #2 as follows:

```
> route add -net 192.168.16.0 netmask 255.255.255.0 gw 172.17.0.2
```



```
root@ef77835bbccd:/#
root@ef77835bbccd:/# route add -net 192.168.16.0 netmask 255.255.255.0 gw 172.17.0.2
```

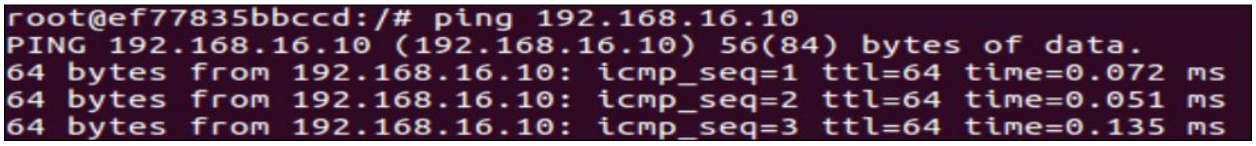
```
> route
```



```
root@ef77835bbccd:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.240.0   U     0      0        0 eth0
192.168.16.0    172.17.0.2      255.255.255.0   UG    0      0        0 eth0
```

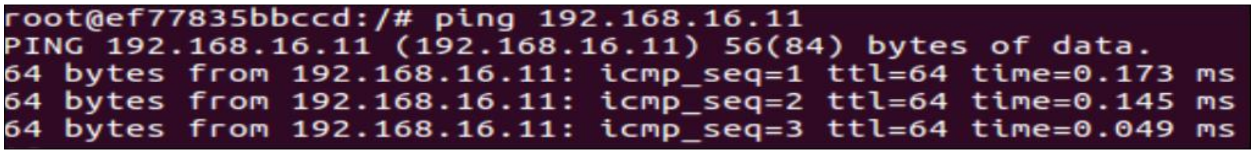
Using ping to test the route and the simulated devices on SAPro Container #1.

```
> ping 192.168.16.10
```



```
root@ef77835bbccd:/# ping 192.168.16.10
PING 192.168.16.10 (192.168.16.10) 56(84) bytes of data.
64 bytes from 192.168.16.10: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from 192.168.16.10: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 192.168.16.10: icmp_seq=3 ttl=64 time=0.135 ms
```

```
> ping 192.168.16.11
```

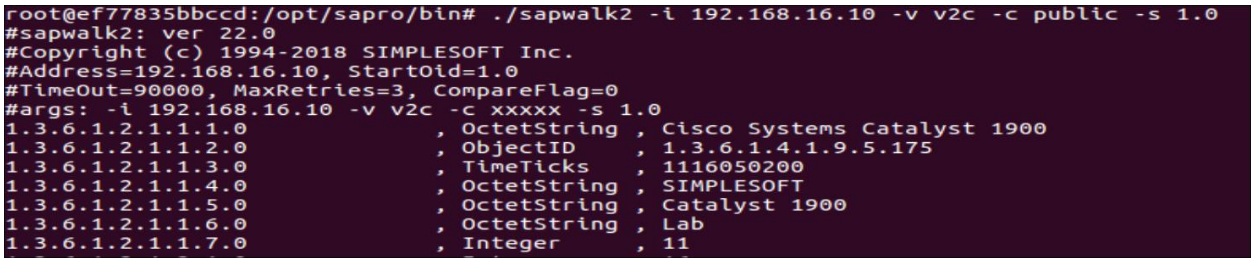


```
root@ef77835bbccd:/# ping 192.168.16.11
PING 192.168.16.11 (192.168.16.11) 56(84) bytes of data.
64 bytes from 192.168.16.11: icmp_seq=1 ttl=64 time=0.173 ms
64 bytes from 192.168.16.11: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 192.168.16.11: icmp_seq=3 ttl=64 time=0.049 ms
```

Using sapwalk2 utility on SAPro Container #2 to poll the 2 simulated devices on SAPro Container #1.

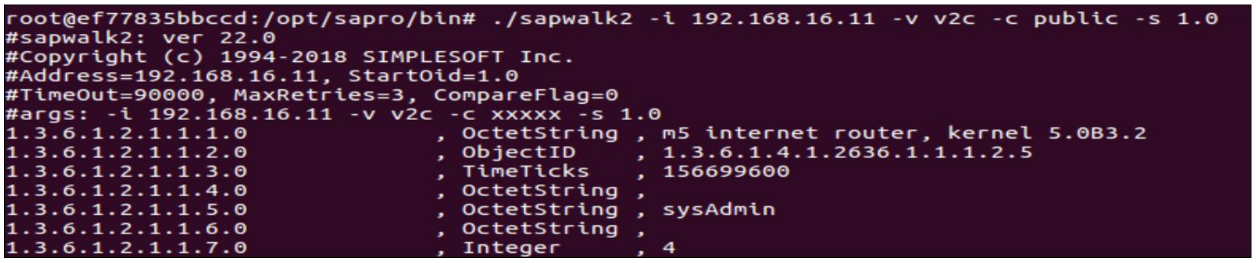
```
> cd /opt/sapro/bin
```

```
> ./sapwalk2 -i 192.168.16.10 -v v2c -c public -s 1.0
```



```
root@ef77835bbccd:/opt/sapro/bin# ./sapwalk2 -i 192.168.16.10 -v v2c -c public -s 1.0
#sapwalk2: ver 22.0
#Copyright (c) 1994-2018 SIMPLESOFT Inc.
#Address=192.168.16.10, StartOid=1.0
#TimeOut=90000, MaxRetries=3, CompareFlag=0
#args: -i 192.168.16.10 -v v2c -c xxxxx -s 1.0
1.3.6.1.2.1.1.1.0              , OctetString , Cisco Systems Catalyst 1900
1.3.6.1.2.1.1.2.0              , ObjectID    , 1.3.6.1.4.1.9.5.175
1.3.6.1.2.1.1.3.0              , TimeTicks   , 1116050200
1.3.6.1.2.1.1.4.0              , OctetString , SIMPLESOFT
1.3.6.1.2.1.1.5.0              , OctetString , Catalyst 1900
1.3.6.1.2.1.1.6.0              , OctetString , Lab
1.3.6.1.2.1.1.7.0              , Integer     , 11
```

```
> ./sapwalk2 -i 192.168.16.11 -v v2c -c public -s 1.0
```



```
root@ef77835bbccd:/opt/sapro/bin# ./sapwalk2 -i 192.168.16.11 -v v2c -c public -s 1.0
#sapwalk2: ver 22.0
#Copyright (c) 1994-2018 SIMPLESOFT Inc.
#Address=192.168.16.11, StartOid=1.0
#TimeOut=90000, MaxRetries=3, CompareFlag=0
#args: -i 192.168.16.11 -v v2c -c xxxxx -s 1.0
1.3.6.1.2.1.1.1.0              , OctetString , m5 internet router, kernel 5.0B3.2
1.3.6.1.2.1.1.2.0              , ObjectID    , 1.3.6.1.4.1.2636.1.1.1.2.5
1.3.6.1.2.1.1.3.0              , TimeTicks   , 156699600
1.3.6.1.2.1.1.4.0              , OctetString ,
1.3.6.1.2.1.1.5.0              , OctetString , sysAdmin
1.3.6.1.2.1.1.6.0              , OctetString ,
1.3.6.1.2.1.1.7.0              , Integer     , 4
```

You can use other commands in sapcnsl to control and view the test results locally or remotely.

Please direct your technical support questions to support@simplesoft.com