# Practice M6: Nagios

## Part 1

### Preparation

To save some time, we can use the archive **M6-Practice-Nagios (code).zip** provided with the practice

We can create a working folder

Extract the archive there

Navigate to the folder

Finally, create a new **Vagrantfile** by executing the following command

**cp Vagrantfile-1 Vagrantfile**

Alternatively, we can create manually a new **Vagrantfile** with the following content

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.ssh.insert\_key = false

  config.vm.define "master" do |master|

    master.vm.box="shekeriev/centos-8-minimal"

    master.vm.hostname = "master.dob.lab"

    master.vm.network "private\_network", ip: "192.168.99.100"

    master.vm.network "forwarded\_port", guest: 80, host: 8000

    master.vm.provision "shell", path: "add\_hosts.sh"

  end

end

We will need one more file - **add\_hosts.sh** with the following content

#!/bin/bash

echo "192.168.99.100 master.dob.lab master" >> /etc/hosts

echo "192.168.99.101 node1.dob.lab node1" >> /etc/hosts

echo "192.168.99.102 node2.dob.lab node2" >> /etc/hosts

Let us run the environment with the usual command

**vagrant up**

And then create SSH session to the **master** machine

**vagrant ssh master**

### Nagios Installation

To download and install Nagios, we must execute the following

**sudo dnf install epel-release**

Then enable the **PowerTools** repository

**sudo dnf config-manager --set-enabled powertools**

And finally, install **nagios** plus the required packages

**sudo dnf install nagios nagios-common nagios-selinux**

Now, that we have it installed, we must enable and start both **httpd** and **nagios** services

**sudo systemctl enable --now httpd**

**sudo systemctl enable --now nagios**

Then adjust the firewall

**sudo firewall-cmd --add-service={http,https} --permanent**

**sudo firewall-cmd --reload**

And then open a browser tab on the host and navigate to http://<m1-ip-address>/nagios

We will be asked for user credentials which we currently do not have

Let us correct this but first, examine the web configuration file for Nagios

**cat /etc/httpd/conf.d/nagios.conf**

We can see that a **passwd** file is required to exist in **/etc/nagios** folder

If we check, we will see that the file exists

**ls -l /etc/nagios/**

And if we ask for its content

**sudo cat /etc/nagios/passwd**

We will see that there is a user **nagiosadmin** defined, but we do not know its password

Let us initialize its password with

**sudo htpasswd /etc/nagios/passwd nagiosadmin**

Restart the web server

**sudo systemctl restart httpd**

Now, we have the required credentials, so let us return to the web browser and try again

We can explore different views or sections - **Tactical Overview**, **Map**, **Hosts**, **Host Groups**, **Services** and **Service Groups**

If we notice errors with any of the services, it could be due to missing plugins

We can correct this by installing a particular set of plugins with

**sudo dnf install nagios-plugins-{ping,disk,users,procs,load,swap,ssh,http}**

Instead, we can install all plugins by executing this

**sudo dnf install nagios-plugins-all**

Depending on our distribution and its version the above can result to an error

Usually the error is resolved by excluding some of the plugins, for example

**sudo dnf install nagios-plugins-\* --exclude nagios-plugins-all,nagios-plugins-ssl\_validity**

Now, we must restart the **nagios** service

**sudo systemctl restart nagios**

Check its status

**systemctl status nagios**

Return to the browser tab and check if everything is working now

Now, return on the command line

Install the **tree** command if missing

**sudo dnf install tree**

Explore the configuration folders hierarchy of Nagios

**sudo tree /etc/nagios**

Check the main configuration file

**cat /etc/nagios/nagios.cfg**

Now, let us explore the plugins folder

**cd /usr/lib64/nagios/plugins/**

See the list of plugins

**ls -l**

We can explore how they work

Let us check the **check\_ping** plugin help

**./check\_ping -h**

Now, let us see it in action

**./check\_ping -H 8.8.8.8 -w 2,20% -c 5,50%**

We can experiment with different set of parameters and/or with other plugins

Return to the browser tab and navigate to **Services**

There is still a warning for the **HTTP** service

If we pay attention to the message and check the document root folder for the default site, we will notice that the **index.html** file is missing

Let us create a simple **index.html** file in the **/var/www/html** folder

**echo '<h1>This is a demo site</h1>' | sudo tee /var/www/html/index.html**

Now, if return in the browser tab, after a while we will see that the warning will go away

Should we do not want to wait until next check, we can force it

Click on the **HTTP** check

Then click on the **Re-schedule the next check of this service** in the right part of the screen

Confirm with **Commit**

And finaly click on **Done**

Now, we can check again the state of the service

It should be green

### Change Nagios Look and Feel

So far, we saw that we can add plugins

In the similar way we can add or change the interface of **Nagios**

We can browse them here: <https://exchange.nagios.org/directory/Addons/Frontends-%28GUIs-and-CLIs%29/Web-Interfaces/Themes-and-Skins>

Let us download a new theme

**wget https://github.com/ynlamy/vautour-style/releases/latest/download/vautour\_style.zip -O vautour\_style.zip**

Download and install **unzip** if you do not have it already

**sudo dnf install unzip**

Next, we must extract the archive’s content to the following folder **/usr/share/nagios/html**

Please note that this operation will overwrite the folder’s contents. So, it may be a good idea to do a backup first

**sudo unzip vatutor\_style.zip -d /usr/share/nagios/html**

Now, we can return in the browser tab and refresh the page

A whole new look and feel for **Nagios**

## Part 2

### Working with Nagios

Exit the SSH session from the **master** machine with

**exit**

Now, either change the **Vagrantfile** with another one from the archive

**cp Vagrantfile-2 Vagrantfile**

Or make sure that your **Vagrantfile** is with the following content

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.ssh.insert\_key = false

  config.vm.define "master" do |master|

    master.vm.box="shekeriev/centos-8-minimal"

    master.vm.hostname = "master.dob.lab"

    master.vm.network "private\_network", ip: "192.168.99.100"

    master.vm.network "forwarded\_port", guest: 80, host: 8000

    master.vm.provision "shell", path: "add\_hosts.sh"

  end

  config.vm.define "node1" do |node1|

    node1.vm.box="shekeriev/centos-8-minimal"

    node1.vm.hostname = "node1.dob.lab"

    node1.vm.network "private\_network", ip: "192.168.99.101"

    node1.vm.network "forwarded\_port", guest: 80, host: 8001

    node1.vm.provision "shell", path: "add\_hosts.sh"

  end

end

Bring up the new environment with

**vagrant up**

Now, we should have not one but two machines - **master** and **node1**

Let us install configure a web server on the new machine

Open a new session

**vagrant ssh node1**

Now, we must install the web server

**sudo dnf install httpd**

Then enable and start it with

**sudo systemctl enable --now httpd**

And finally, open a port in the firewall

**sudo firewall-cmd --add-service={http,https} --permanent**

**sudo firewall-cmd --reload**

As a very last step, we can create a custom **index.html** file in the **/var/www/html** folder with

**echo '<h1>Remote host: Demo site.</h1>' | sudo tee /var/www/html/index.html**

Now, we can open a browser tab on our host and navigate to

[**http://localhost:8001**](http://localhost:8001)

We should see the simple page, we created earlier

Now, exit the **node1** machine

**exit**

And return to the **master** machine

**vagrant ssh master**

It is time to start extending the configuration of **Nagios** to include our new host

First, let us create a folder to store our new set of files

**sudo mkdir /etc/nagios/objects/dob-files**

Now, let us create a file named **host-node1.cfg** to store the host object

**sudo vi /etc/nagios/objects/dob-files/host-node1.cfg**

It must have the following content

define host {

        use                     linux-server

        host\_name               node1.dob.lab

        alias                   Slave Host Node 1

        address                 192.168.99.101

        icon\_image              linux40.png

        statusmap\_image         linux40.png

}

Save and close the file

Let us now create a template for a service

Create a new file named **template-service-remote.cfg**

**sudo vi /etc/nagios/objects/dob-files/template-service-remote.cfg**

It should have the following content

define service {

        name                            remote-service

        use                             generic-service

        max\_check\_attempts              5

        normal\_check\_interval           2

        retry\_check\_interval            1

        register                        0

}

Save and close the file

Now, let us create a service definition file that will reuse the template we created

Create a new file named **service-node1-http.cfg**

**sudo vi /etc/nagios/objects/dob-files/service-node1-http.cfg**

It should have the following content

define service {

        use                             remote-service

        host\_name                       node1.dob.lab

        service\_description             HTTP

        check\_command                   check\_http!$HOSTADDRESS$

        notifications\_enabled           0

}

Save and close the file

In order **Nagios** to be able to read our files and include them, we must change its configuration

Open the main configuration file for editing

**sudo vi /etc/nagios/nagios.cfg**

Add the following row where you like (for example on ***row 50*** or around the other **cfg\_dir** directives)

**cfg\_dir=/etc/nagios/objects/dob-files**

Save and close the file

It is a good practice to check the configuration before attempting restart of a service

For **Nagios**, this can be done with

**sudo nagios -v /etc/nagios/nagios.cfg**

Depending on the distribution in use and the installed **Nagios** version, you may see a couple of warnings about **normal\_check\_interval** and **retry\_check\_interval**

If this is the case, go and correct them to **check\_interval** and **retry\_interval**

Check the configuration again

If everything is okay, we must restart the **Nagios** service

**sudo systemctl restart nagios**

Now, return to the browser tab and refresh the **Nagios** portal

We should see the changes

Once we are done exploring, we can return on the command line

Next, we can extend the default **Linux Servers** (**linux-servers**) host group

Open the **/etc/nagios/objects/localhost.cfg** file for editing

**sudo vi /etc/nagios/objects/localhost.cfg**

Locate the **hostgroup** block (it should be around ***row 41***) and change it to look like (you can leave the comments)

define hostgroup {

        hostgroup\_name  linux-servers

        alias           Linux Servers

        members         localhost,node1.dob.lab

}

Save and close the file

Now, let us create our own host group

Create a new file named **group-host-dob.cfg**

**sudo vi /etc/nagios/objects/dob-files/group-host-dob.cfg**

With the following content

define hostgroup {

        hostgroup\_name  dob-hosts

        alias           DOB Demo Servers

        members         node1.dob.lab

}

Save and close the file

Now, let us check the configuration

**sudo nagios -v /etc/nagios/nagios.cfg**

If everything is okay, restart the **Nagios** service

**sudo systemctl restart nagios**

Return to the browser tab with the **Nagios** portal open and check the changes

Once done exploring, return on the command line

It is time to experiment with service groups

Let us add one service group with **group-service-http.cfg** configuration file

**sudo vi /etc/nagios/objects/dob-files/group-service-http.cfg**

With the following content

define servicegroup {

        servicegroup\_name       group-service-http

        alias                   HTTP Services

        members                 localhost, HTTP, node1.dob.lab, HTTP

}

Save and close the file

Now, let us create one more service group stored in **group-service-infra.cfg** configuration file

**sudo vi /etc/nagios/objects/dob-files/group-service-infra.cfg**

With the following content

define servicegroup {

        servicegroup\_name       group-service-infra

        alias                   Infrastructure Services

        members                 localhost, Current Load

}

Save and close the file

Check the **Nagios** configuration for errors

**sudo nagios -v /etc/nagios/nagios.cfg**

If no errors detected, restart the service

**sudo systemctl restart nagios**

Return in the browser tab with the **Nagios** portal open

Refresh the page and explore the changes

## Part 3

Make sure that you exited all sessions to the virtual machines and that you are working on the host

Now, either change the **Vagrantfile** with another one from the archive

**cp Vagrantfile-3 Vagrantfile**

Or make sure that your **Vagrantfile** is with the following content

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  config.ssh.insert\_key = false

  config.vm.define "master" do |master|

    master.vm.box="shekeriev/centos-8-minimal"

    master.vm.hostname = "master.dob.lab"

    master.vm.network "private\_network", ip: "192.168.99.100"

    master.vm.network "forwarded\_port", guest: 80, host: 8000

    master.vm.provision "shell", path: "add\_hosts.sh"

  end

  config.vm.define "node1" do |node1|

    node1.vm.box="shekeriev/centos-8-minimal"

    node1.vm.hostname = "node1.dob.lab"

    node1.vm.network "private\_network", ip: "192.168.99.101"

    node1.vm.network "forwarded\_port", guest: 80, host: 8001

    node1.vm.provision "shell", path: "add\_hosts.sh"

  end

  config.vm.define "node2" do |node2|

    node2.vm.box="shekeriev/centos-8-minimal"

    node2.vm.hostname = "node2.dob.lab"

    node2.vm.network "private\_network", ip: "192.168.99.102"

    node2.vm.provision "shell", path: "add\_hosts.sh"

  end

end

Bring up the new environment with

**vagrant up**

Now, we must have three machines in total - **master**, **node1** and **node2**

### Install NRPE

Let us enter the new machine

**vagrant ssh node2**

First, install the **EPEL** repository

**sudo dnf install epel-release**

Enable the **PowerTools** repository

**sudo dnf config-manager --set-enabled powertools**

Install the **NRPE** component and all related packages with

**sudo dnf install nrpe nrpe-selinux nagios-plugins-nrpe nagios-plugins-all**

Remember, should any plugin cause dependency conflict, skip it

Now, that we have all components installed, we can move on to configure the **NRPE**

Open the main configuration file for editing

**sudo vi /etc/nagios/nrpe.cfg**

Find the **allowed\_hosts** parameter (***row 106***) and change it to

**allowed\_hosts=127.0.0.1,::1,192.168.99.0/24**

Then find the **dont\_blame\_nrpe** parameter (***row 122***) and change it to

**dont\_blame\_nrpe=1**

Find the **allow\_bash\_command\_substitution** parameter (***row 140***) and change it to

**allow\_bash\_command\_substitution=1**

Save and close the file

Enable the appropriate port in the firewall

**sudo firewall-cmd --add-port=5666/tcp --permanent**

**sudo firewall-cmd --reload**

Enable and start the service

**sudo systemctl enable --now nrpe**

Check its status

**systemctl status nrpe**

Using the **NRPE** plugin, we can check if we can connect locally both via **localhost** and the IP address of the machine

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H localhost**

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.102**

Everything should be working as expected

Now exit the session

**exit**

And open a session to the **master** machine

**vagrant ssh master**

Install the **NRPE** plugin here as well

**sudo dnf install nagios-plugins-nrpe**

Now, we can check if we can connect to the remote **NRPE** with

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.102**

We should continue only if everything is okay

If there is no communication or some other type of error, we must return on the **node2** machine and check for example, if the service is working, or the firewall is configured

Once, we are fine, we can create a command definition

Open the file that stores other commands definitions for editing

**sudo vi /etc/nagios/objects/commands.cfg**

Go to the end of the file and add the following block

define command {

        command\_name check-nrpe

        command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$

}

Save and close the file

Next, we will create an object and a configuration file for the second host

**sudo vi /etc/nagios/objects/dob-files/host-node2.cfg**

It should contain the following

define host {

        use                     linux-server

        host\_name               node2.dob.lab

        alias                   Slave Host Node 2

        address                 192.168.99.102

        icon\_image              linux40.png

        statusmap\_image         linux40.png

}

Save and close the file

Now, let us create a service definition

**sudo vi /etc/nagios/objects/dob-files/service-node2-cpu.cfg**

With the following content

define service {

        use                     remote-service

        host\_name               node2.dob.lab

        service\_description     CPU Load

        check\_command           check-nrpe!check\_load

}

Save and close the file

Next, let us add **node2** to the two existing host groups

First, open the **localhost.cfg** file for editing

**sudo vi /etc/nagios/objects/localhost.cfg**

Locate the **hostgroup** block (it should be around row 41) and change it to look like (you can leave the comments)

define hostgroup {

        hostgroup\_name  linux-servers

        alias           Linux Servers

        members         localhost, node1.dob.lab, node2.dob.lab

}

Save and close the file

Then open the file **/etc/nagios/objects/dob-files/group-host-dob.cfg** for editing

**sudo vi /etc/nagios/objects/dob-files/group-host-dob.cfg**

Modify it to match the following

define hostgroup {

        hostgroup\_name  dob-hosts

        alias           DOB Demo Servers

        members         node1.dob.lab, node2.dob.lab

}

Save and close the file

Finally, add the service we defined earlier to the file with infrastructure service group

**sudo vi /etc/nagios/objects/dob-files/group-service-infra.cfg**

The **servicegroup** block should look like

define servicegroup {

        servicegroup\_name       group-service-infra

        alias                   Infrastructure Services

        members                 localhost, Current Load, node2.dob.lab, CPU Load

}

Save and close the file

Let us check the **Nagios** configuration before we attempt restarting the service

**sudo nagios -v /etc/nagios/nagios.cfg**

Finally, restart the service

**sudo systemctl restart nagios**

Go to the browser tab opened earlier, refresh the page, and check the result

### Monitoring MariaDB

Let us experiment with something more complex than what we did so far

For example, assume that we want to monitor a database

This in our case will be **MariaDB**

Exit any existing sessions

**exit**

And open a session to **node2** machine

**vagrant ssh node2**

Install the **MariaDB** database

**sudo dnf install mariadb mariadb-server**

Enable and start the service

**sudo systemctl enable --now mariadb**

Establish a session to the database

**mysql -u root -p**

The **root** password should be empty, so just hit **Enter**

Create a user to be used by **Nagios**

**create user nagios@192.168.99.100 identified by 'Password1';**

Close the session to the database

**quit**

Open the appropriate port in the firewall and reload its configuration

**sudo firewall-cmd --add-port=3306/tcp --permanent**

**sudo firewall-cmd --reload**

Close the session to **node2**

**exit**

Open a session to the **master** machine

**vagrant ssh master**

Let us check if we have a connection to the **MariaDB** database from here

**sudo /usr/lib64/nagios/plugins/check\_mysql -H 192.168.99.102 -u nagios -p Password1**

Everything should be fine, if not, we must track and check the possible reasons

Now, let us create a file with our own **MariaDB** related commands

**sudo vi /etc/nagios/objects/dob-files/command-mysql.cfg**

Make sure that the content matches the following

# 'check\_mysql' command definition

define command {

        command\_name    check-mysql

        command\_line    $USER1$/check\_mysql -H '$HOSTADDRESS$'

}

# 'check\_mysql\_cmdlinecred' command definition

define command {

        command\_name    check-mysql-cmdlinecred

        command\_line    $USER1$/check\_mysql -H '$HOSTADDRESS$' -u '$ARG1$' -p '$ARG2$'

}

# 'check\_mysql\_database' command definition

define command {

        command\_name    check-mysql-database

        command\_line    $USER1$/check\_mysql -d '$ARG3$' -H '$HOSTADDRESS$' -u '$ARG1$'

-p '$ARG2$'

}

# 'check\_mysql\_query' command definition

define command {

        command\_name    check-mysql-query

        command\_line    $USER1$/check\_mysql\_query -q '$ARG4$' -d '$ARG3$'

-H '$HOSTADDRESS$' -u '$ARG1$' -p '$ARG2$'

}

Save and close the file

Now, we can create a new service for the **MariaDB** database running on the **node2** machine

**sudo vi /etc/nagios/objects/dob-files/service-node2-mysql.cfg**

Make sure that you type the following text

define service {

        use                     remote-service

        host\_name               node2.dob.lab

        service\_description     MYSQL

        check\_command           check-mysql-cmdlinecred!nagios!Password1

        notification\_interval   2

}

Save and close the file

Check the configuration of **Nagios**

**sudo nagios -v /etc/nagios/nagios.cfg**

Restart the **Nagios** service

**sudo systemctl restart nagios**

Return to the browser tab and refresh the page

Explore what has changed

### Monitoring Docker

It is time to extend our monitoring towards **Docker**

Exit any currently established session

**exit**

Now, while on our host, create a **docker\_setup.sh** file with the following content

#!/bin/bash

# ... comments are available in the attached file

# Add the Docker repository

dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

# Install Docker

dnf install -y docker-ce docker-ce-cli

# Add the user to Docker group

usermod -aG docker vagrant

usermod -aG docker nrpe

# Start and enable Docker

systemctl enable --now docker

# Print some information

systemctl status docker

docker version

docker system info

Alternatively, use the one provided with the practice

Let us go with the second approach - use the provided file

Check which port is used to communicate over **SSH** with **node2** by executing

**vagrant ssh-config**

Let us assume that the port is **2201**

To copy the locally available **docker\_setup.sh** file, we must execute the following command

**scp -P 2201 docker\_setup.sh vagrant@localhost:.**

Please note that the password is also **vagrant**

Establish a session to the **node2** machine

**vagrant ssh node2**

Set the executable bit of the script

**chmod +x docker\_setup.sh**

Execute the script to install **Docker**

**sudo ./docker\_setup.sh**

Once the installation is complete, add the **nrpe** user to the **sudoers** list

**sudo visudo**

Enter the following line

**nrpe ALL=(ALL) NOPASSWD: ALL**

Save and close the file

Restart the **node2** machine

**sudo reboot**

Once the machine is up, establish a new session to **node2**

**vagrant ssh node2**

Start a new container with the following command

**docker container run -d --name dob-con alpine sleep 1d**

Now, we will need a routine to check our **Docker** container

For this, we can either create a file named **check\_docker\_container.sh** with the following content

#!/bin/bash

# ... comments are available in the attached file

CONTAINER=$1

if [ "x${CONTAINER}" == "x" ]; then

  echo "UNKNOWN - Container ID or Friendly Name Required"

  exit 3

fi

if [ "x$(which docker)" == "x" ]; then

  echo "UNKNOWN - Missing docker binary"

  exit 3

fi

docker info > /dev/null 2>&1

if [ $? -ne 0 ]; then

  echo "UNKNOWN - Unable to talk to the docker daemon"

  exit 3

fi

RUNNING=$(docker inspect --format="{{.State.Running}}" $CONTAINER 2> /dev/null)

if [ $? -eq 1 ]; then

  echo "UNKNOWN - $CONTAINER does not exist."

  exit 3

fi

if [ "$RUNNING" == "false" ]; then

  echo "CRITICAL - $CONTAINER is not running."

  exit 2

fi

RESTARTING=$(docker inspect --format="{{.State.Restarting}}" $CONTAINER)

if [ "$RESTARTING" == "true" ]; then

  echo "WARNING - $CONTAINER state is restarting."

  exit 1

fi

STARTED=$(docker inspect --format="{{.State.StartedAt}}" $CONTAINER)

NETWORK=$(docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" $CONTAINER)

echo "OK - $CONTAINER is running. IP: $NETWORK, StartedAt: $STARTED"

Or use the provided one

Let us follow the second approach again

Exit the session

**exit**

Copy the locally available file to the **node2** machine

**scp -P 2201 check\_docker\_container.sh vagrant@127.0.0.1:.**

Remember that the password is also **vagrant**

Establish again session to the **node2** machine

**vagrant ssh node2**

Make the file we just copied executable

**chmod +x check\_docker\_container.sh**

And move it to the **/usr/lib64/nagios/plugins** folder

**sudo mv check\_docker\_container.sh /usr/lib64/nagios/plugins/**

Next, we must change the configuration of the **Nagios** agent

**sudo vi /etc/nagios/nrpe.cfg**

Add the following line at the end of the file

**command[check-docker-container]=/usr/lib64/nagios/plugins/check\_docker\_container.sh $ARG1$**

Save and close the file

Restart the **Nagios** agent

**sudo systemctl restart nrpe**

Check that we can query the container with

**/usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.102 -c check-docker-container -a dob-con**

If everything goes according to the plan, exit the session to **node2**

**exit**

In case of any errors, especially ***NRPE: Unable to read output*** or ***UNKNOWN - Missing docker binary***, we can switch the **SELinux** to **permissive** mode

The real problem can be seen if we explore the plugin folder with

**ls -alZ /usr/lib64/nagios/plugins/**

It is better however to execute the following (please note that each command is a single line) set of commands to adjust the ownership, permissions, and context (usually, it is enough to deal just with ownership and context):

**sudo chown --reference /usr/lib64/nagios/plugins/check\_users /usr/lib64/nagios/plugins/check\_docker\_container.sh**

**sudo chmod --reference /usr/lib64/nagios/plugins/check\_users /usr/lib64/nagios/plugins/check\_docker\_container.sh**

**sudo chcon --reference /usr/lib64/nagios/plugins/check\_users /usr/lib64/nagios/plugins/check\_docker\_container.sh**

Now, establish a session to the **master** machine

**vagrant ssh master**

Test again with

**/usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.102 -c check-docker-container -a dob-con**

Let us add a new command in the main **commands.cfg** file

**sudo vi /etc/nagios/objects/commands.cfg**

We must add the following block

define command {

        command\_name check-nrpe-arg

        command\_line $USER1$/check\_nrpe -H $HOSTADDRESS$ -c $ARG1$ -a $ARG2$

}

Save and close the file

Now, we can add one more service for **node2**

**sudo vi /etc/nagios/objects/dob-files/service-node2-docker.cfg**

It must have the following definition

define service {

        use                     remote-service

        host\_name               node2.dob.lab

        service\_description     Container Running

        check\_command           check-nrpe-arg!check-docker-container!dob-con

}

Save and close the file

Check the **Nagios** configuration

**sudo nagios -v /etc/nagios/nagios.cfg**

Restart the service

**sudo systemctl restart nagios**

Return to the browser tab and explore the changes