Name: Khelvin P. Nicolas	**Date Performed: 14/10/2024**
Course/Section: CPE 212 - CPE31S2	**Date Submitted: 14/10/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY: 1st Sem - 3rd Yr**

Activity 8: Install, Configure, and Manage Availability Monitoring Tools

1. Objectives

Create and design a workflow that installs, configure and manage enterprise monitoring tools using Ansible as an Infrastructure as Code (IaC) tool.

2. Discussion

Availability monitoring is a type of monitoring tool that we use if the certain workload is up or reachable on our end. Site downtime can lead to loss of revenue, reputational damage and severe distress. Availability monitoring prevents adverse situations by checking the uptime of infrastructure components such as servers and apps and notifying the webmaster of problems before they impact on business.

3. Tasks

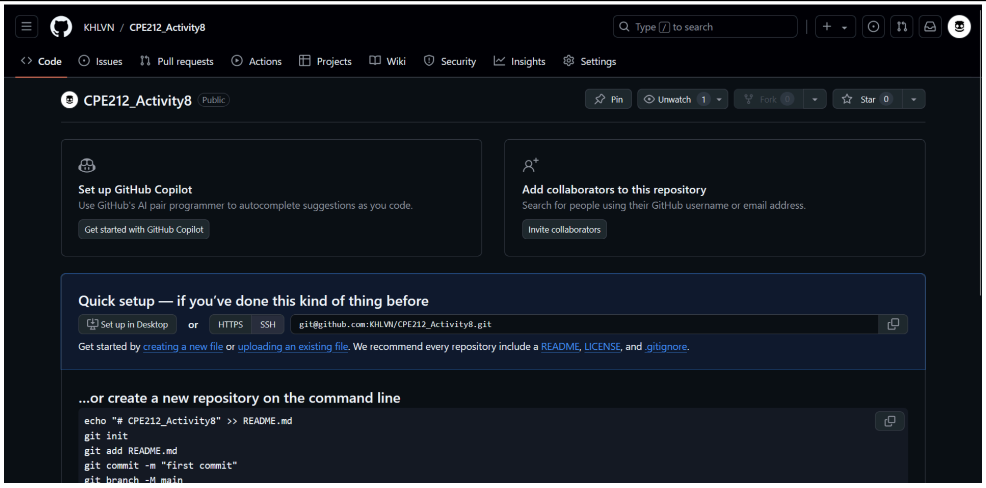
1. Create a playbook that installs Nagios in both Ubuntu and CentOS. Apply the concept of creating roles.
2. Describe how you did step 1. (Provide screenshots and explanations in your report. Make your report detailed such that it will look like a manual.)
3. Show an output of the installed Nagios for both Ubuntu and CentOS.
4. Make sure to create a new repository in GitHub for this activity.

4. Output (screenshots and explanations)

Phase 1: Setting up working environment for Nagios installation

To install Nagios in our managed nodes, we must first have a working environment which we can use to install any application that we might use, whether it is a real corporate environment or not. In this activity, we created a GitHub repository to save all the configurations and playbooks that will be used, it also includes the inventory file.

GitHub Repository for Activity 8: https://github.com/KHLVN/CPE212_Activity8



Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Add collaborators to this repository
Search for people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

```
git@github.com:KHLVN/CPE212_Activity8.git
```

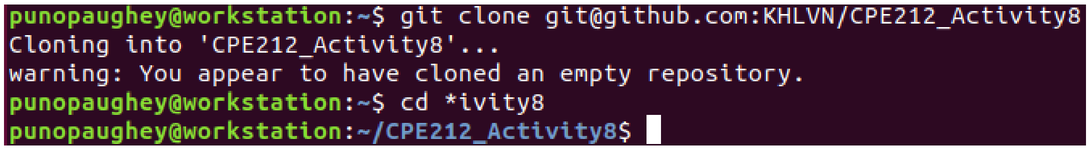
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# CPE212_Activity8" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
```

Figure 1.1: Create GitHub repository

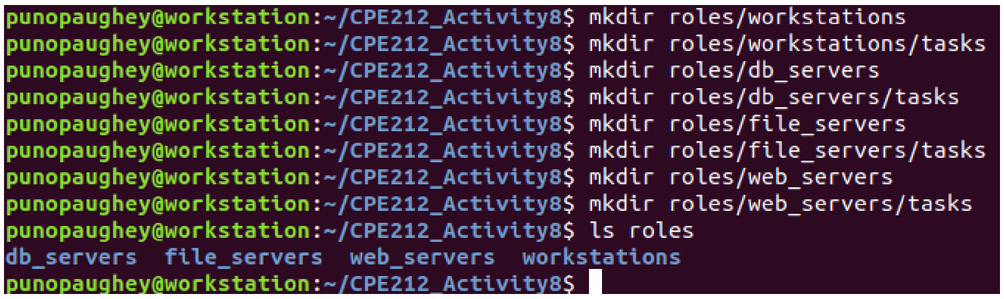
Inside the workstation, we cloned our newly created repository. We will perform the activity inside of this directory so we can commit and push it easily in the remote repository later.



```
punopaughey@workstation:~$ git clone git@github.com:KHLVN/CPE212_Activity8
Cloning into 'CPE212_Activity8'...
warning: You appear to have cloned an empty repository.
punopaughey@workstation:~$ cd *ivity8
punopaughey@workstation:~/CPE212_Activity8$ 
```

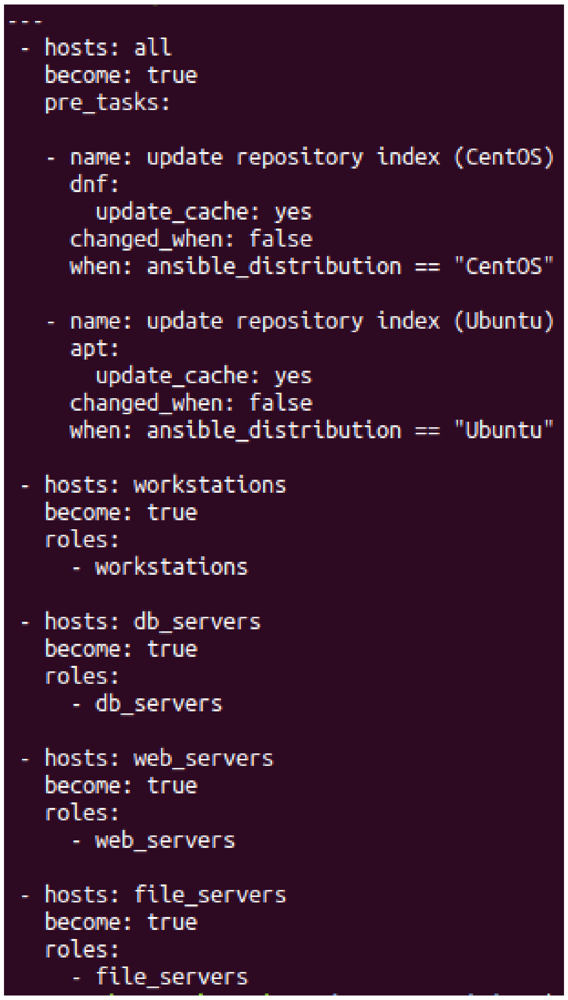
Figure 1.2: Cloning created repo in control node

We created directories for specific roles that we want to assign to our managed nodes, including `**db_servers**` for database management, `**file_servers**` for file sharing and FTP, and `**web_servers**` for web hosting. Each directory contains a `**tasks**` subdirectory where we can place a playbook to perform tasks for the corresponding role.



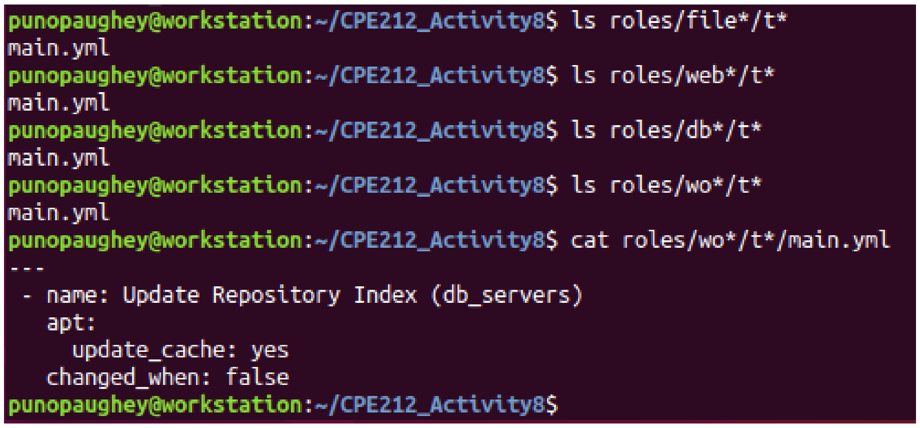
```
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/workstations
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/workstations/tasks
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/db_servers
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/db_servers/tasks
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/file_servers
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/file_servers/tasks
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/web_servers
punopaughey@workstation:~/CPE212_Activity8$ mkdir roles/web_servers/tasks
punopaughey@workstation:~/CPE212_Activity8$ ls roles
db_servers  file_servers  web_servers  workstations
punopaughey@workstation:~/CPE212_Activity8$ 
```

Figure 1.3: Creating directories for specific roles



```
---
- hosts: all
  become: true
  pre_tasks:

  - name: update repository index (CentOS)
    dnf:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "CentOS"

  - name: update repository index (Ubuntu)
    apt:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "Ubuntu"

- hosts: workstations
  become: true
  roles:
    - workstations

- hosts: db_servers
  become: true
  roles:
    - db_servers

- hosts: web_servers
  become: true
  roles:
    - web_servers

- hosts: file_servers
  become: true
  roles:
    - file_servers
```

Figure 1.4: Creating initial routine.yml file



```
punopaughey@workstation:~/CPE212_Activity8$ ls roles/file*/t*
main.yml
punopaughey@workstation:~/CPE212_Activity8$ ls roles/web*/t*
main.yml
punopaughey@workstation:~/CPE212_Activity8$ ls roles/db*/t*
main.yml
punopaughey@workstation:~/CPE212_Activity8$ ls roles/wo*/t*
main.yml
punopaughey@workstation:~/CPE212_Activity8$ cat roles/wo*/t*/main.yml
---
 - name: Update Repository Index (db_servers)
   apt:
     update_cache: yes
   changed_when: false
punopaughey@workstation:~/CPE212_Activity8$
```

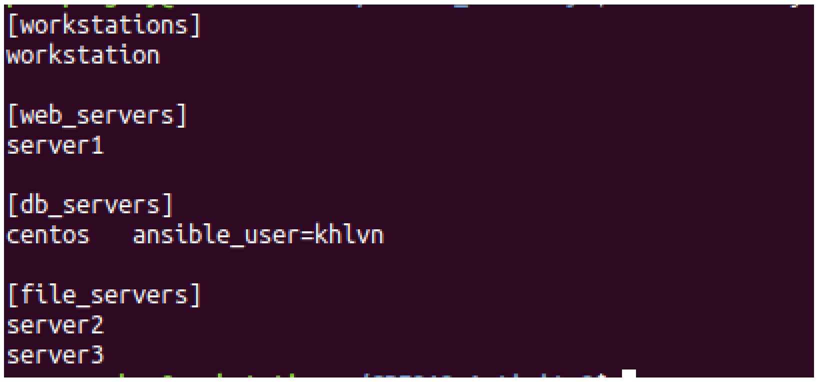
Figure 1.5: Creating initial main.yml file inside all roles’ directories

By mapping our network environment, we can see the following nodes:

- **workstation (control node)**
- **server1 (managed node)**
- **server2 (managed node)**
- **server3 (managed node)**
- **centos (managed node)**

Servers 1, 2, and 3 uses Ubuntu distribution while **centos** uses CentOS distribution

For this inventory file, we've assigned **server1** as web_server and **centos** as db_server, leaving the rest of the managed nodes as file_servers.



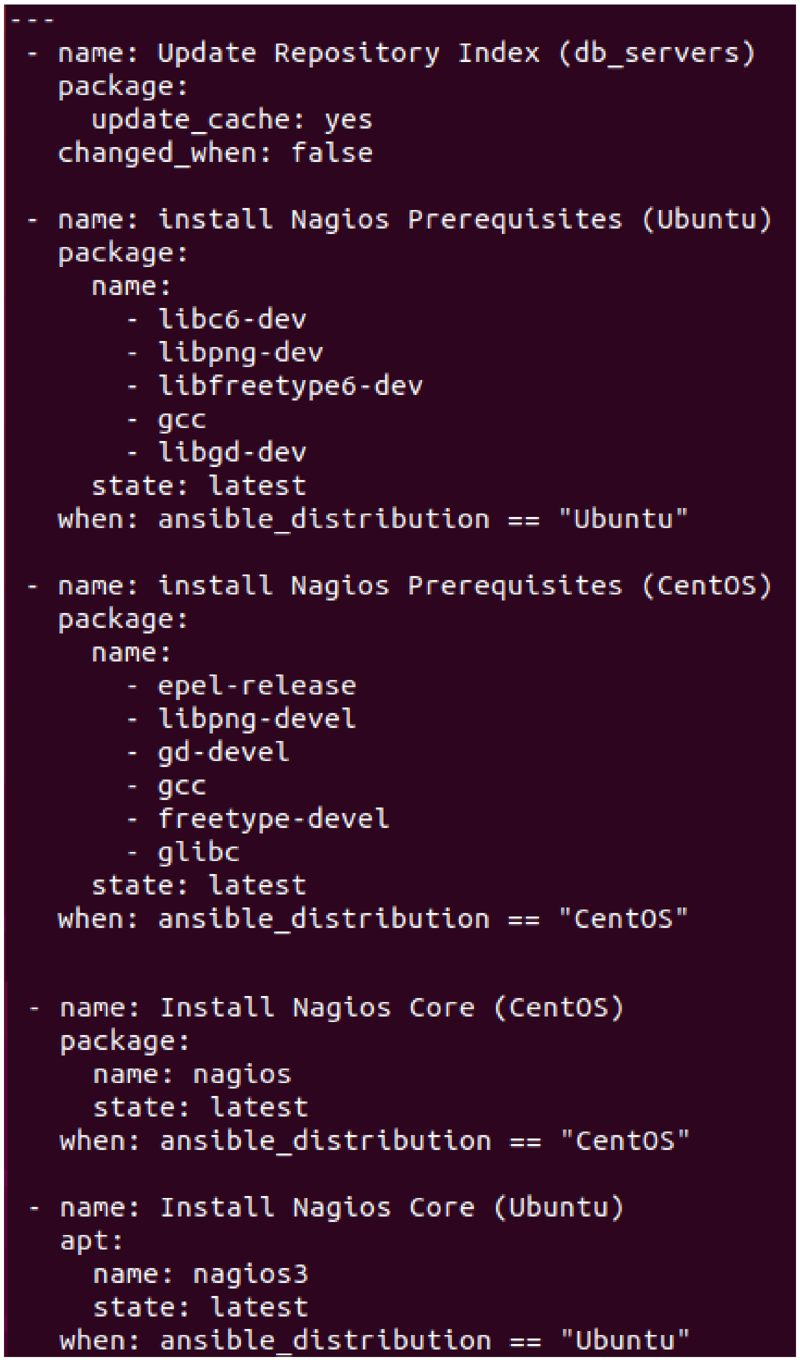
```
[workstations]
workstation

[web_servers]
server1

[db_servers]
centos   ansible_user=khlvn

[file_servers]
server2
server3
```

Figure 1.6: setting roles for each control node in inventory

After setting up the environment, we are now ready to proceed to the next step which is to install Nagios availability monitoring tool in our remote servers.

Phase 2: Installing Nagios using Ansible Playbook

In the previous tasks, we set up our sample network of managed nodes to mimic an enterprise network. Now, we are tasked with installing an availability monitoring tool called Nagios on our managed nodes or remote servers. Create a playbook that will install the Nagios core on both distributions (CentOS and Ubuntu). Make sure to install the required dependencies first. After the installation of Nagios is complete, we should also include another task in the playbook that enables the Nagios service for use. Refer to the sample code below to properly install Nagios on our systems.

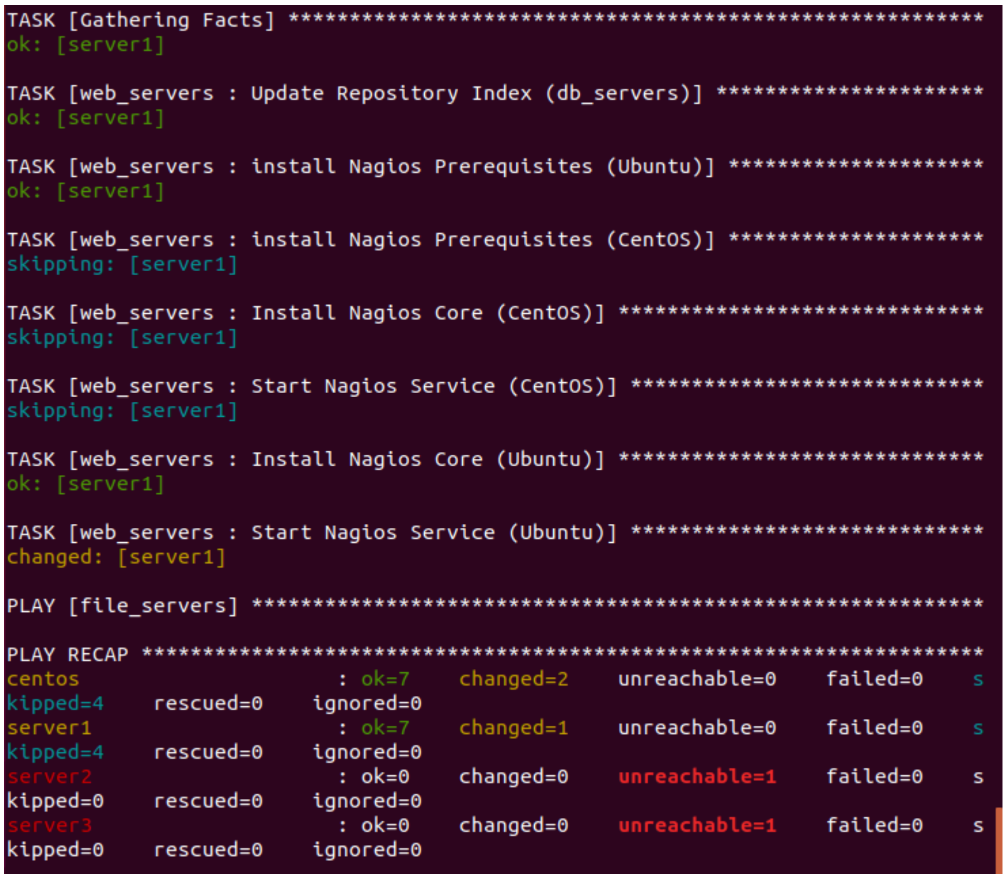


```
---
 - name: Update Repository Index (db_servers)
   package:
     update_cache: yes
   changed_when: false

 - name: install Nagios Prerequisites (Ubuntu)
   package:
     name:
       - libc6-dev
       - libpng-dev
       - libfreetype6-dev
       - gcc
       - libgd-dev
     state: latest
   when: ansible_distribution == "Ubuntu"

 - name: install Nagios Prerequisites (CentOS)
   package:
     name:
       - epel-release
       - libpng-devel
       - gd-devel
       - gcc
       - freetype-devel
       - glibc
     state: latest
   when: ansible_distribution == "CentOS"


 - name: Install Nagios Core (CentOS)
   package:
     name: nagios
     state: latest
   when: ansible_distribution == "CentOS"

 - name: Install Nagios Core (Ubuntu)
   apt:
     name: nagios3
     state: latest
   when: ansible_distribution == "Ubuntu"
```

Figure 1.7: Create main.yml file to install Nagios in both Ubuntu and CentOS

After writing the code above and saving it as a YAML file, we should now run the playbook using the command ***ansible-playbook --ask-become-pass <yourplaybookname>.yml***.

After running the playbook, as shown in the figure below, we can see that Nagios was successfully installed along with its required dependencies, using only the Ansible playbook.

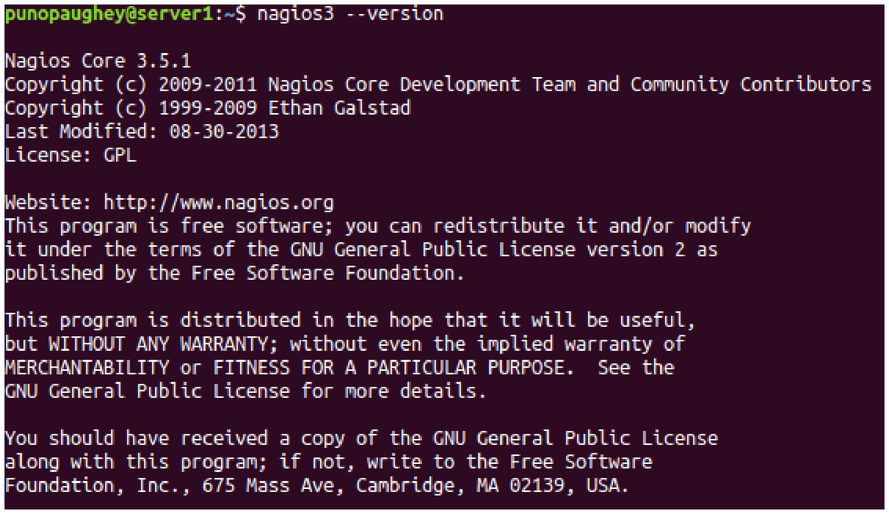


```
TASK [Gathering Facts] *****************************************************************
ok: [server1]

TASK [web_servers : Update Repository Index (db_servers)] **********************
ok: [server1]

TASK [web_servers : install Nagios Prerequisites (Ubuntu)] *********************
ok: [server1]

TASK [web_servers : install Nagios Prerequisites (CentOS)] *********************
skipping: [server1]

TASK [web_servers : Install Nagios Core (CentOS)] ******************************
skipping: [server1]

TASK [web_servers : Start Nagios Service (CentOS)] *****************************
skipping: [server1]

TASK [web_servers : Install Nagios Core (Ubuntu)] ******************************
ok: [server1]

TASK [web_servers : Start Nagios Service (Ubuntu)] *****************************
changed: [server1]

PLAY [file_servers] ************************************************************

PLAY RECAP *********************************************************************
centos                     : ok=7    changed=2    unreachable=0    failed=0    s
kipped=4    rescued=0    ignored=0
server1                    : ok=7    changed=1    unreachable=0    failed=0    s
kipped=4    rescued=0    ignored=0
server2                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
server3                    : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figure 1.8: Running site.yml playbook

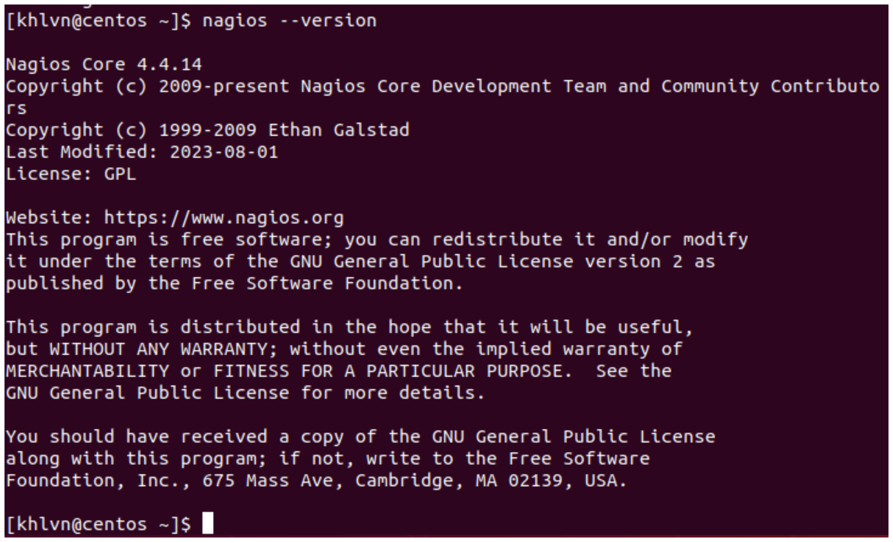
Ignore the unreachable error displayed for server2 and server3 since we will not use them for this activity, but you may do so.

To ensure that Nagios is successfully installed on the managed nodes, run the command ***nagios --version***. Sometimes, the version number needs to be included in the command, depending on the Nagios version installed on the system. For example, the Ubuntu distribution uses Nagios 3, so you should use ***nagios3 --version*** to check the Nagios version; otherwise, an error will be displayed.



```
punopaughey@server1:~$ nagios3 --version

Nagios Core 3.5.1
Copyright (c) 2009-2011 Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-30-2013
License: GPL

Website: http://www.nagios.org
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 2 as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Figure 1.9.1: Nagios Core installed in Ubuntu



```
[khlvn@centos ~]$ nagios --version

Nagios Core 4.4.14
Copyright (c) 2009-present Nagios Core Development Team and Community Contributo
rs
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2023-08-01
License: GPL

Website: https://www.nagios.org
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 2 as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

[khlvn@centos ~]$
```

Figure 1.9.2: Nagios Core installed in CentOS

Now that we have successfully installed Nagios on our servers, we can proceed with configuring and setting it up to fully utilize its monitoring capabilities. This will allow us to actively monitor our systems, network, and infrastructure, track the performance of critical services, detect potential issues, and receive alerts when problems arise. By configuring Nagios to oversee key resources, we can ensure better visibility, improved uptime, and quicker response times in managing our environment.

Reflections:

Answer the following:

1. **What are the benefits of having an availability monitoring tool?**
   - Availability monitoring tools like Nagios in Systems Administration offer numerous benefits that are vital for maintaining the health and performance of the remote servers. One of the primary advantages is constant monitoring, which allows systems administrators to keep an eye on systems, applications, and services 24/7, enabling the detection of issues before they escalate into severe problems, and also minimizes downtime.

Conclusions:

- We have implemented the concept of roles that we used in the previous activity when installing the availability monitoring tool Nagios on our remote servers. Each distribution has different methods for installing Nagios, as they require various prerequisites that serve the same purpose. Additionally, we have manually started the Nagios service manually in our remote servers but it is possible to implement it in the playbook.