Name: Seruelas, Ronn Kristoper H.	**Date Performed:** 10-16-2023
Course/Section: CPE31S4	**Date Submitted:** 10-17-2023
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem 2022-2023

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

Task 1: Preparing a Dedicated repository

1. In your Github account, create a new repository that will be dedicated for the installation of Nagios for both your Ubuntu and CentOS server, and then clone it to your local repository in your workstation.

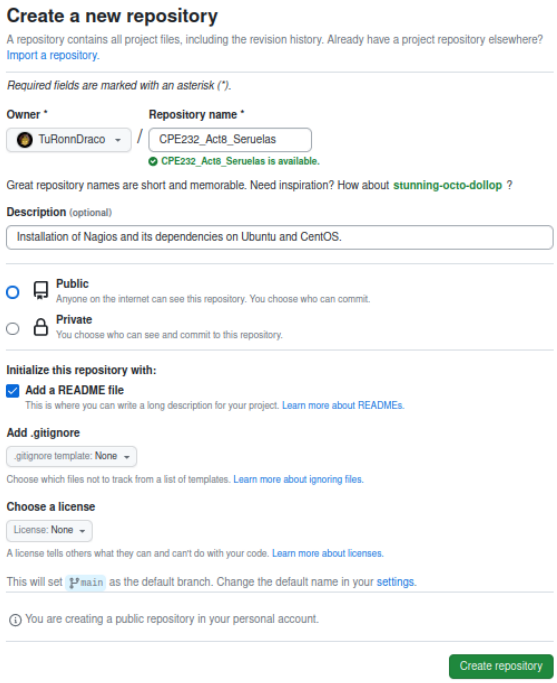

Figure 1.1.1 - Creation of the CPE232_Act8_Seruelas repository.

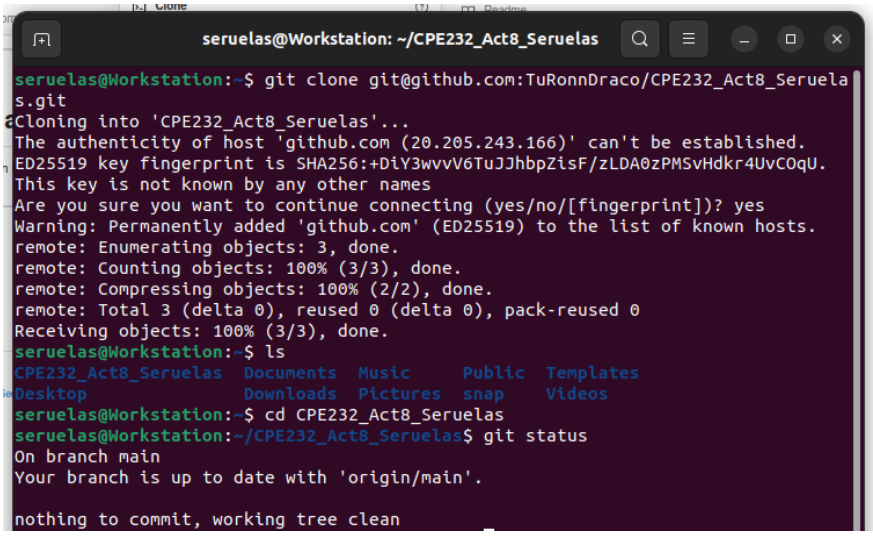

Figure 1.1.2 - Cloning of the github repository unto the local repository of the workstation.

2. In your cloned repository, create the **ansible.cfg** file that will contain the ansible configurations for your repository in order for it to be operable.

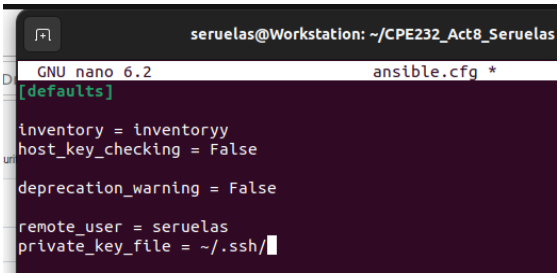

Figure 1.2.1 - Configurations set for ansible.cfg

3. In your cloned repository, set up your **inventory** file by first doing **sudo nano inventory**, then inputting the proper hosts into their remote hosts. You can do this by inputting their respective IP address or their hostnames according to your workstation.

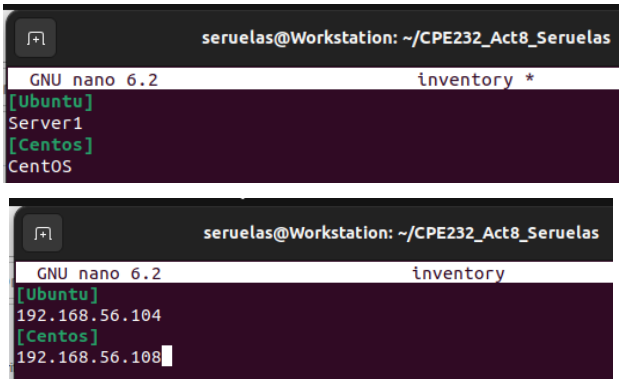

Figure 1.3.1-1.3.2 - Hosts for the inventory file and their two methodologies.

4. After configuring the ansible.cfg and inventory file, verify the connection between the workstation and the remote hosts by executing **ansible all -m ping**.

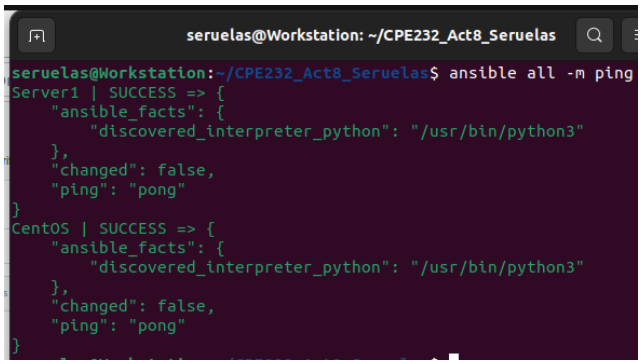

Figure 1.4.1 - Verification of connecting between workstation and the hosts.

Task 2: Preparation of the playbook

1. To start off, first create a yml file using **sudo nano** that will serve as a playbook that will contain all the commands for installing nagios. In this activity, we will create the file, **install_nagios.yml**.

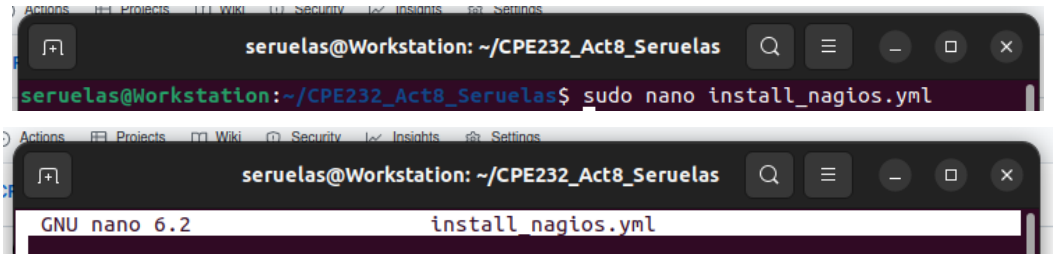

Figure 2.1.1-2.1.2 - Creation of the install_nagios.yml.

2. In the playbook, configure the playbook that it will run for all hosts, and that its designated pre-tasks is to update the repository index of Ubuntu and CentOS.

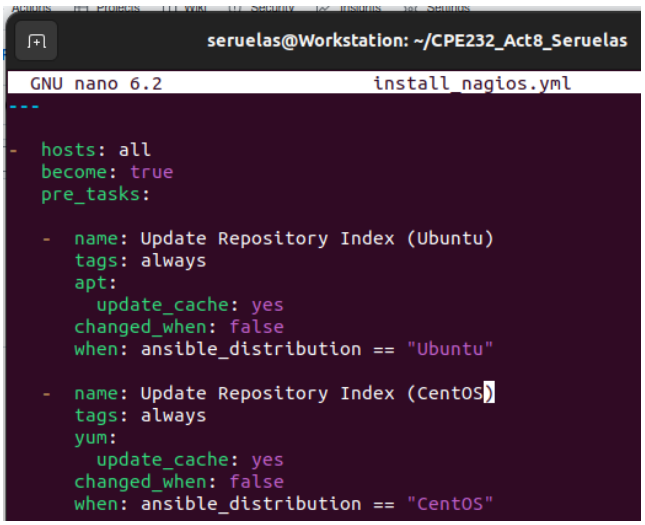

Figure 2.2.1 - Configuration of the playbook that will let it run on all hosts and update the repository indexes of both Ubuntu and CentOS.

3. To create a more efficient and cleaner playbook, we are to divide the tasks of installing nagios into multiple roles. First, we must create a new directory named **roles**, that will contain the different roles that will have their own tasks. Inside of the roles directory will contain the two different roles, Ubuntu and CentOS, within their directories will have another directory named **tasks**, and it will contain the **main.yml**.

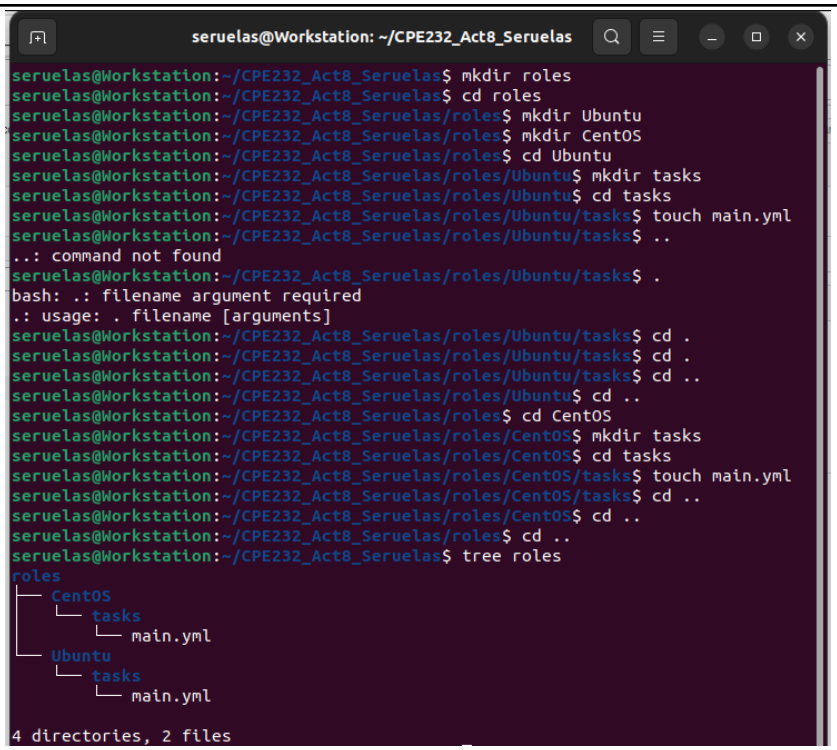

Figure 2.3.1 - Creation of the roles directory, containing different roles with different tasks.

4. After the creation of roles, modify the playbook in which in their own hosts, they will execute only the commands of their own roles.

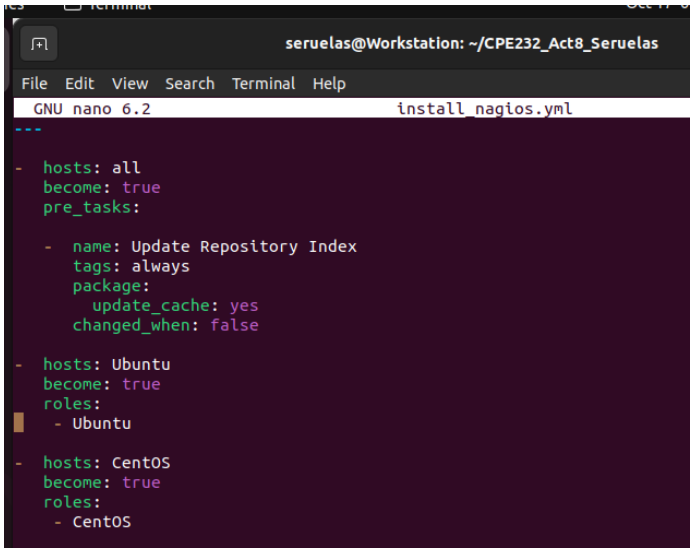

Figure 2.4.1 - Adding the necessary commands to be executed in order to allow the hosts to only execute the commands of their own roles.

5. To verify if the roles are working as intended, modify the **main.yml** of each role to execute a simple command.

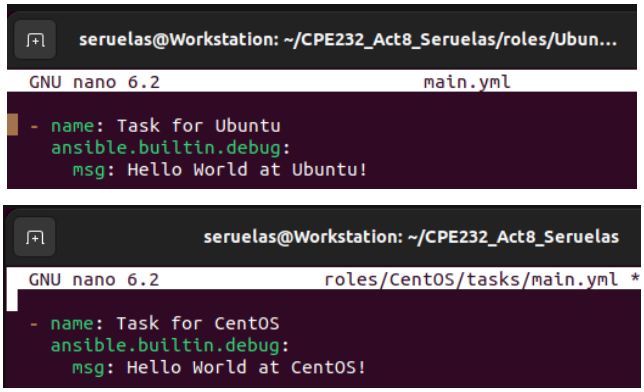

Figure 2.5.1-2.5.2 - Adding simple commands to main.yml of each role.

6. After adding some simple commands to each tasks of the role, execute the playbook by using **ansible-playbook –ask-become-pass <name of playbook>**.

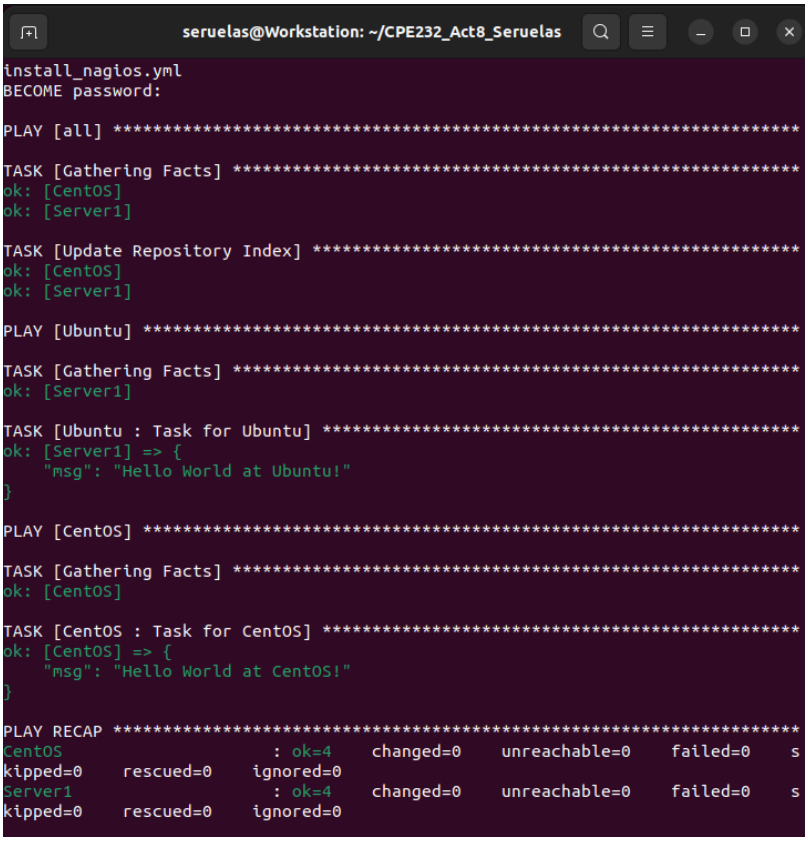

Figure 2.6.1 - Execution of the playbook for testing and verification of setup.

Task 3: Installation of Libraries and Dependencies

1. In Ubuntu, there are a total of 21 packages that can be installed that are the dependencies of Nagios. In the main.yml file of the Ubuntu role, input every package that is considered as a dependency for the Nagios service.

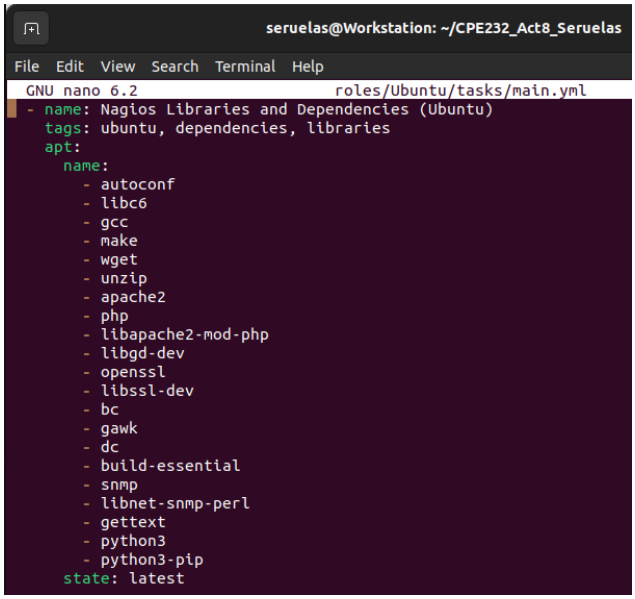

Figure 3.1.1 - Installation of Nagios' libraries and dependencies in Ubuntu.

2. In CentOS, there are a total of 22 packages that are considered as a part of the library or dependency needed by Nagios. In the main.yml of CentOS role, input the installation of every package that is considered as a dependency or library for Nagios.

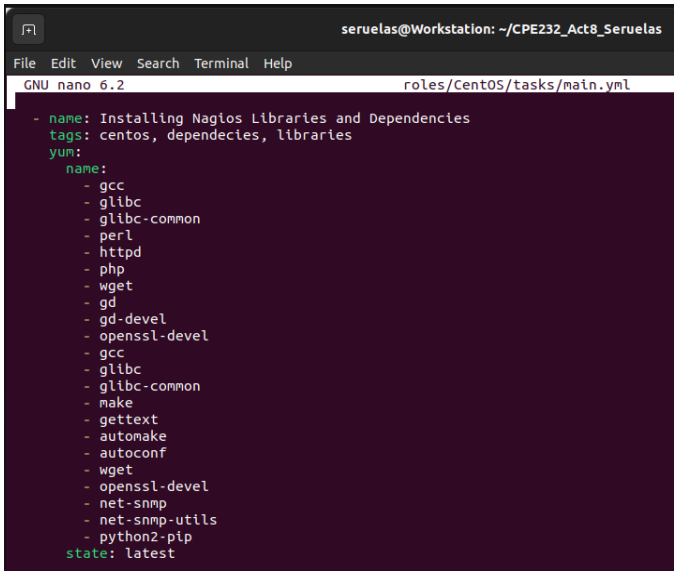

Figure 3.2.1 - Installation of Nagios' libraries and dependencies in CentOS.

Task 4: Installing Nagios4 (Ubuntu Role)

1. Install the **nagios4** and **nagios-plugins** package by including it in the Ubuntu Roles playbook.

```
- name: Install Nagios4
  become: true
  apt:
    name:
      - nagios4
      - nagios-plugins
    state: latest
```

Figure 4.1.1 - Module that will install Nagios4 into the remote host.

2. For Nagios to become operable, we need to configure the apache web server with the **a2enmod**. By using the builtin command **shell** for our playbooks, use the **sudo a2enmod rewrite cgi** to successfully reconfigure our apache server for Nagios4, after that, create a module that will restart the apache web server service.

```
- name: Configure Apache Web Server
  become: true
  shell:
    sudo a2enmod rewrite cgi

- name: Restart Apache Server after Configuration
  service:
    name: apache2
    state: restarted
    enabled: true
```

Figure 4.2.1 - Module that will configure the apache web server for Nagios4, with the module that will restart the Apache2 server after its configuration for Nagios4.

3. Finally, create a module that will enable and start the Nagios4 service.

```
- name: Enable Nagios
  service:
    name: nagios4
    state: restarted
    enabled: true
```

Figure 4.3.1 - Module that enables and starts the Nagios4 service.

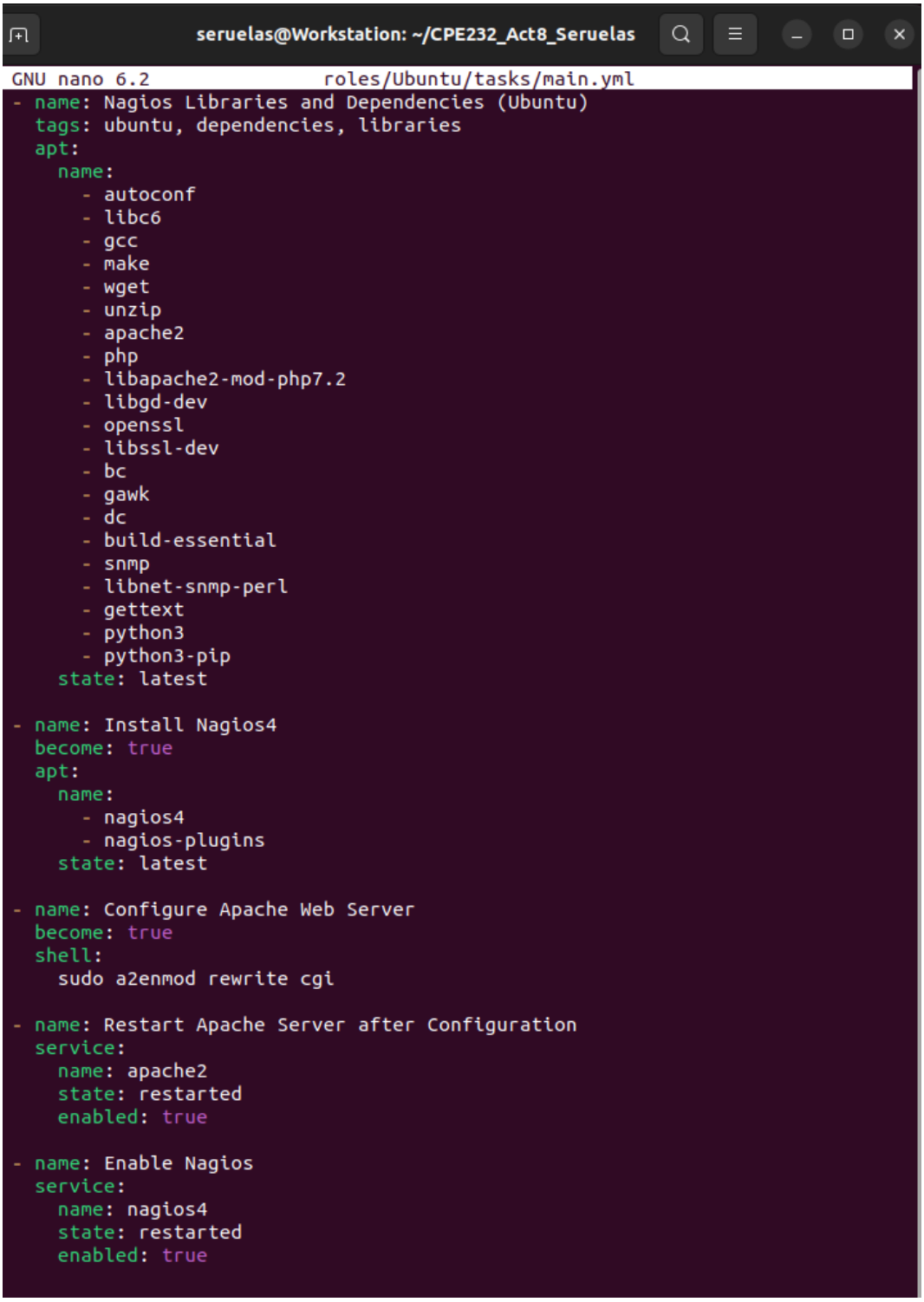

Figure 4.5 - Main.yml of the Ubuntu Root Tasks.

Task 5: Installing Nagios (CentOS Role)

1. Create a directory in which it will be the dedicated directory for Nagios.

```
- name: Creating the Nagios Directory
  file:
    path: ~/nagios
    state: directory
```

Figure 5.1.1 - Module that will create the directory for Nagios.

2. Create a module in which it will download the Nagios file from a repository, and will extract it to the directory.

```
- name: Installation of Nagios
  unarchive:
    src: https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz
    dest: ~/nagios
    remote_src: yes
    mode: 0777
    owner: root
    group: root
```

Figure 5.2.1 - Module responsible for downloading and extracting the files of Nagios to the directory.

3. Create a module in which it will download and extract the plugins for the Nagios service directly to the directory.

```
- name: Installation of Nagios Plugins
  unarchive:
    src: https://github.com/nagios-plugins/nagios-plugins/archive/release-2.3.3.tar.gz
    dest: ~/nagios
    remote_src: yes
    mode: 0777
    owner: root
    group: root
```

Figure 5.3.1 - Module responsible for downloading and extracting the Nagios Plugins.

4. Create a module in which it will configure the Nagios service file in order for it to become operable, in which it will require a user for it to be operable.

```
- name: Compilation, Installation and User Modification
  shell: |
    cd ~/nagios/nagioscore-**
    ./configure
    make all
    make install-groups-users
    usermod -a -G nagios apache
    make install
    make install-daemoninit
    make install-commandmode
    make install-config
    make install-webconf
```

Figure 5.4.1 - Module that is responsible for configuring and creating a user for Nagios.

5. Create a module that will install all the plugins downloaded for Nagios and that will configure them to Nagios.

```
- name: Compilation and Installation of Plugins
  shell: |
    cd ~/nagios/nagios-plugins*
    ./tools/setup
    ./configure
    make
    make install
```

Figure 5.5.1 - Installation and configuration of the plugins for Nagios.

6. Create a module that creates a password for the user that will access the Nagio service from the browser.

```
- name: Adding a Password to the User/s
  community.general.htpasswd:
    path: /usr/local/nagios/etc/htpasswd.users
    name: admin
    password: admin
```

Figure 5.6.1 - Module responsible for setting a password for the administrator.

7. Create a module that will restart and re-enable the HTTPD and the Nagio services.

```
- name: Enabling HTTPD
  service:
    name: httpd
    state: restarted
    enabled: true

- name: Enabling Nagios
  service:
    name: nagios
    state: restarted
    enabled: true
```

Figure 5.7.1 - Module's that will restart and re-enable the HTTPD and the Nagios services.

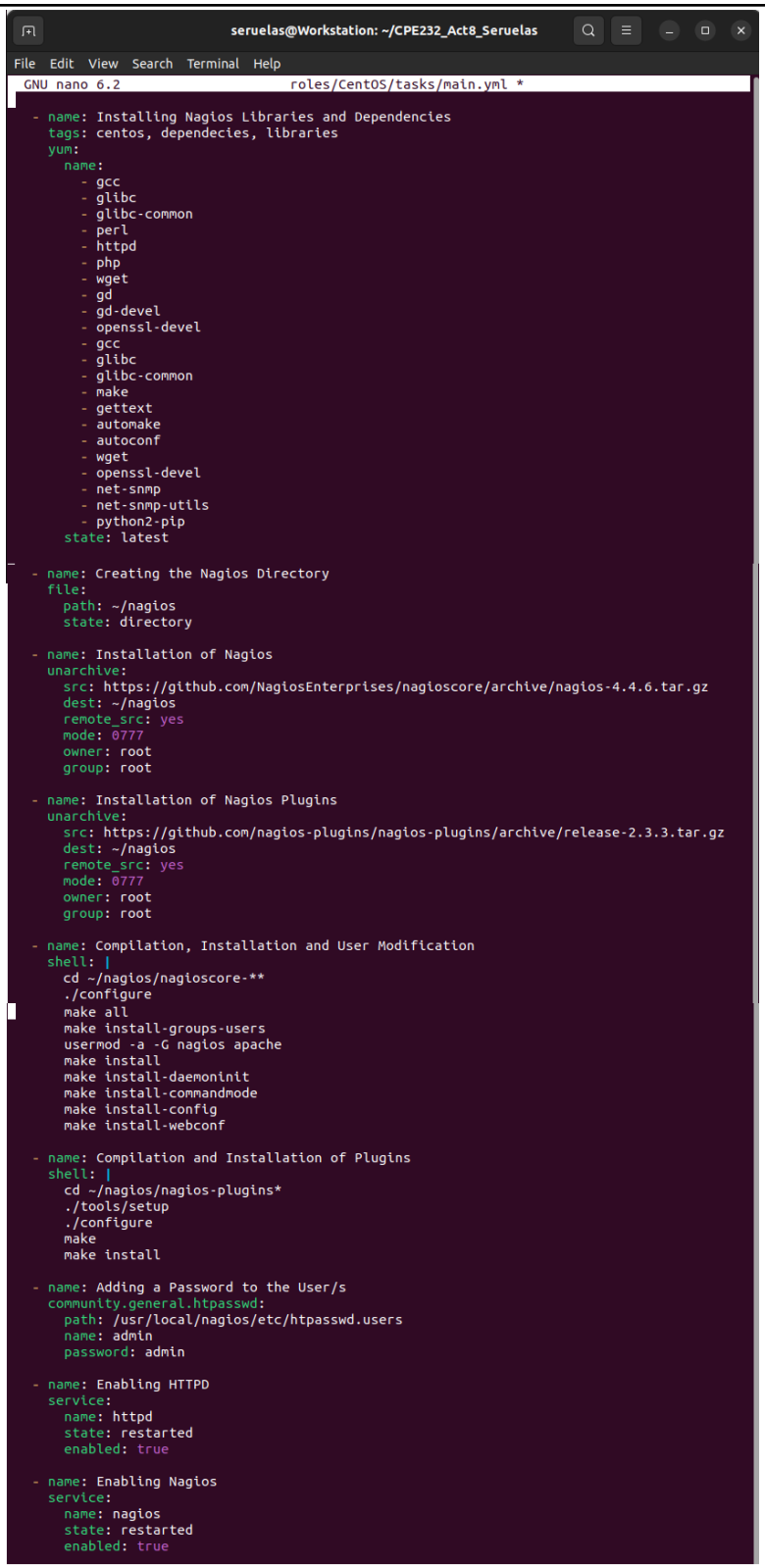

Figure 5.8 - Whole Main.yml of CentOS role.

Task 7: Verification of Playbook

1. Run the playbook and show its output.

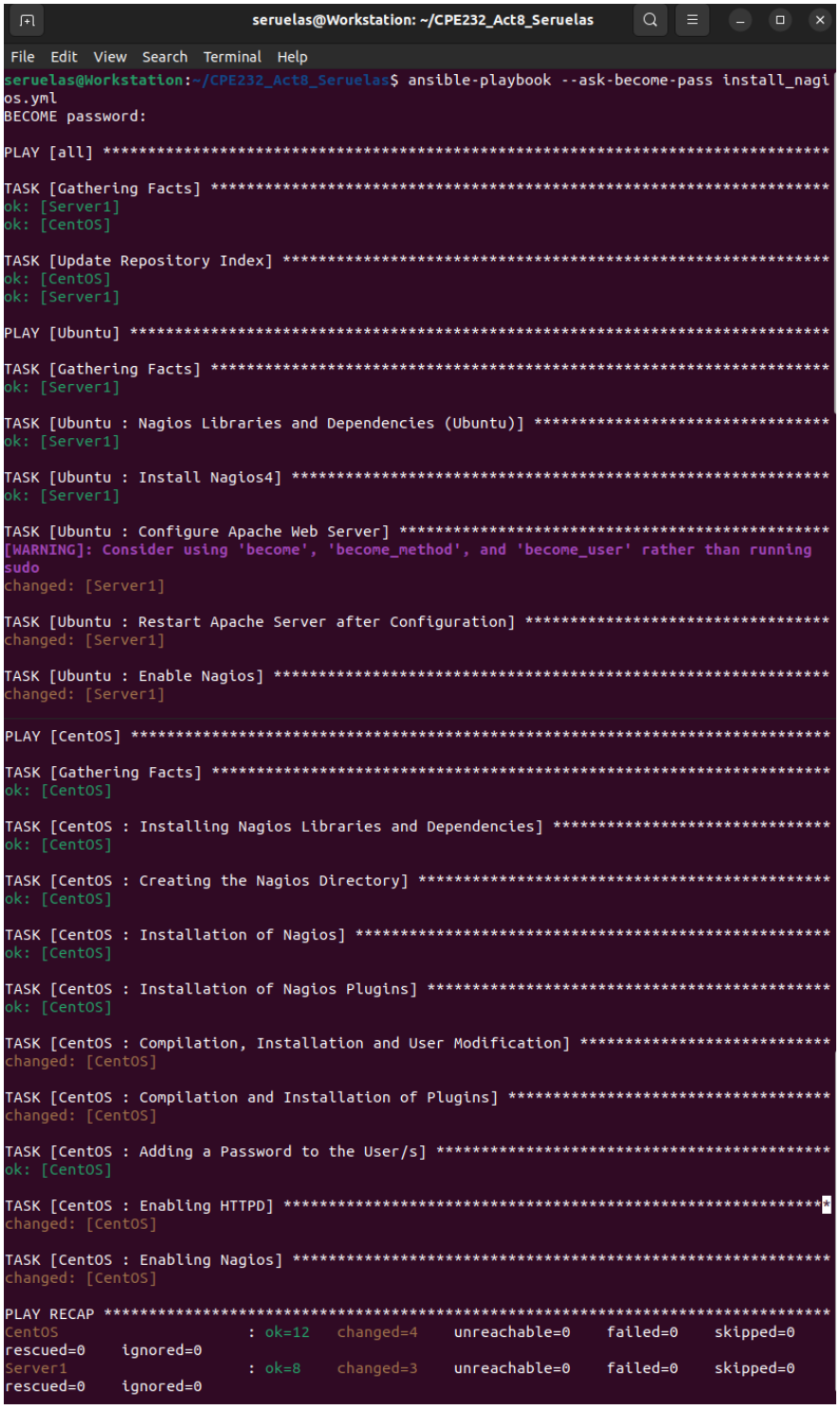

Figure 7.1.1 - Successful run of the install_nagios.yml playbook.

2. Verify the installation of Nagios was successful in the CentOS by executing **systemctl status nagios**, and opening the browser at **<ip address of your host>/nagios/**.

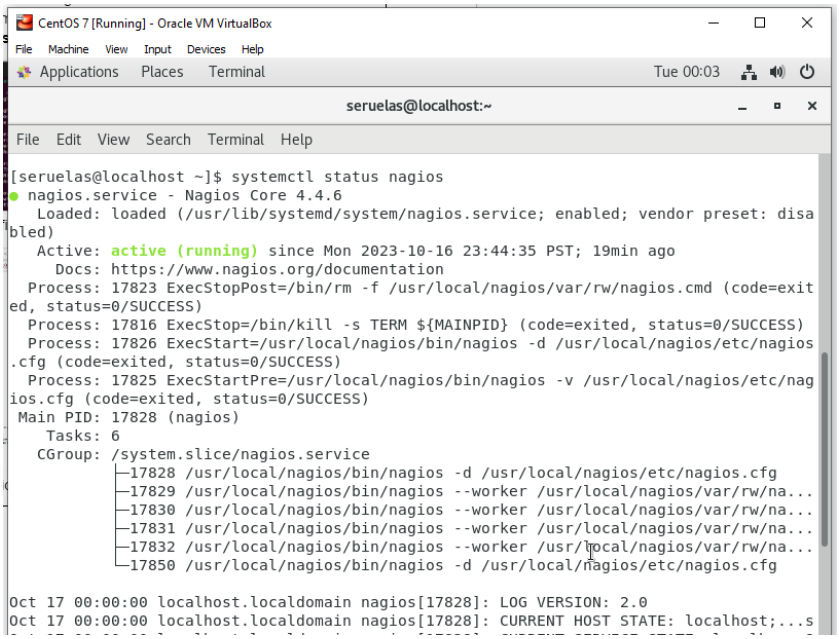

Figure 7.2.1 - Verification of service Nagios running through systemctl.

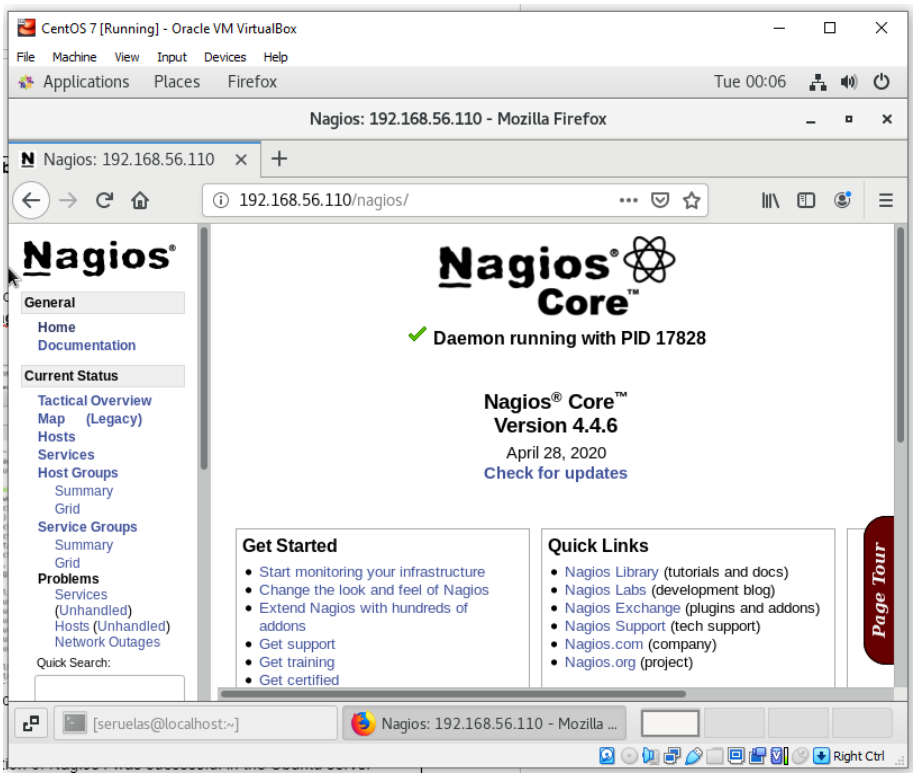

Figure 7.2.2 - Nagios Service opened at CentOS firefox browser.

3. Verify that the installation of Nagios4 was successful in the Ubuntu server by executing the command, **nagios4 –version**, and opening the browser with the url **<ip address of your host>/nagios4/**.

Figure 4.4.1 - Verification of installation of Nagios4 in Ubuntu Server.

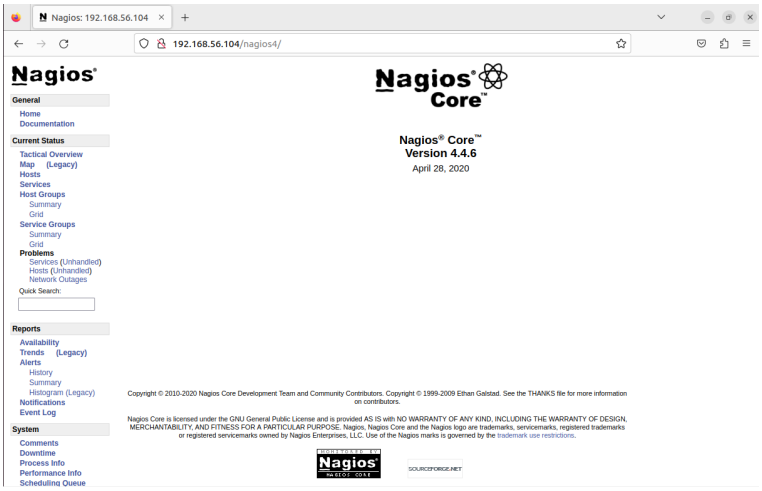

Figure 4.4.2 - Nagios4 Core via browser. (**192.168.56.104/nagios4/**)

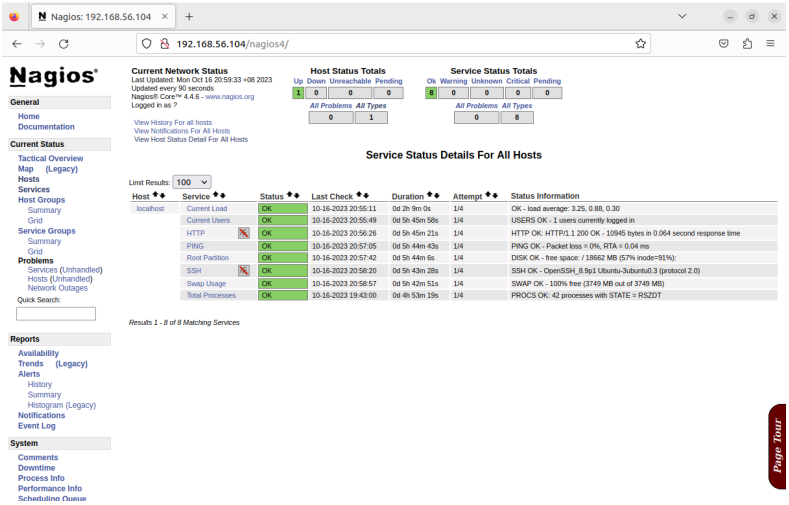

Figure 4.4.3 - Checking the services running tracked by Nagios4

4. Save and Push the local repository to the Github Repository

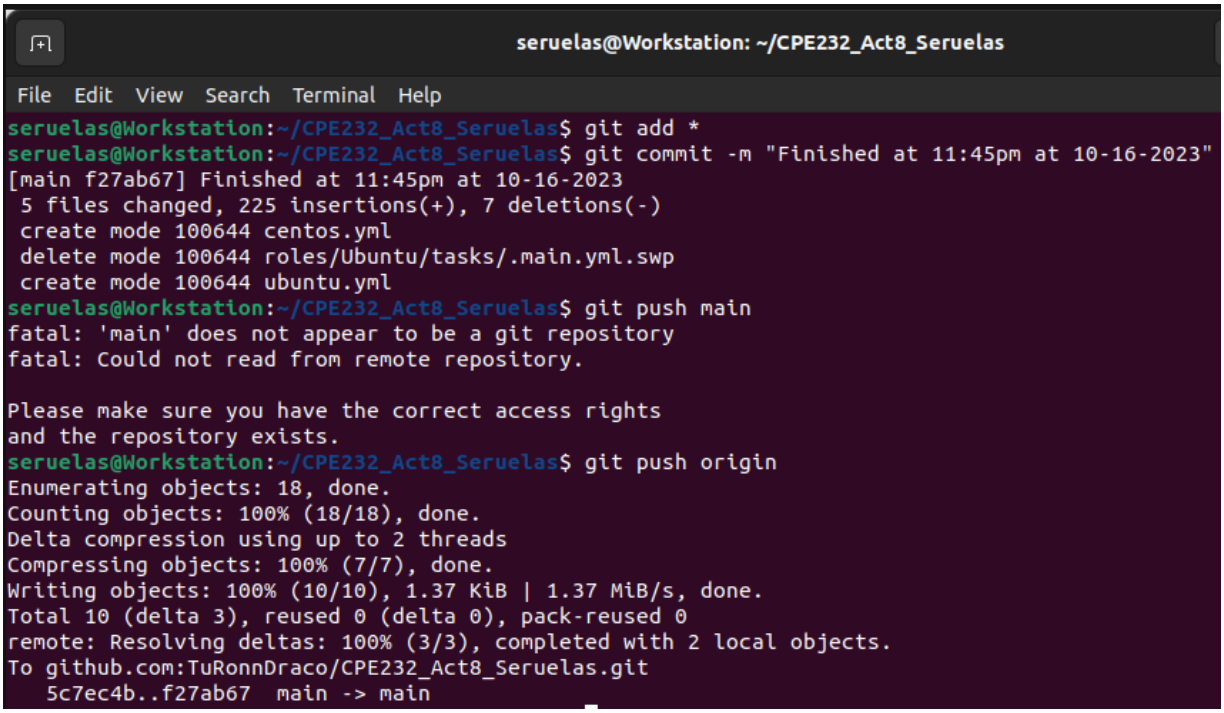

Figure 7.4.1 - Adding, Committing, and Pushing all changes to the Github Repository.

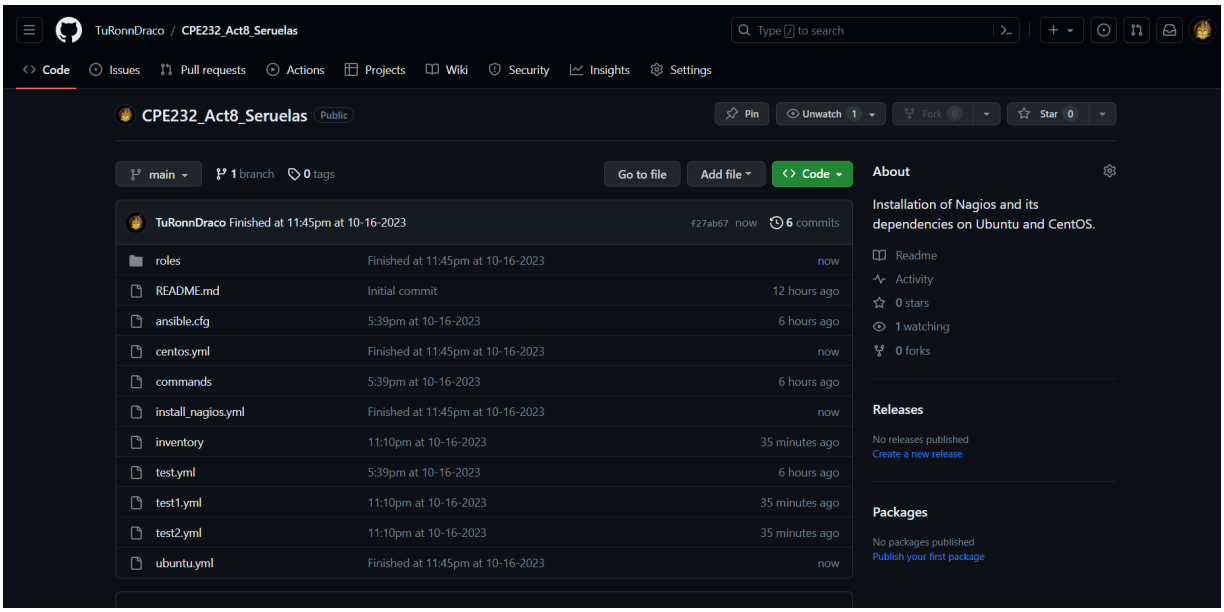

Figure 7.4.2 - Verification of all changes and files added to Github.

https://github.com/TuRonnDraco/CPE232_Act8_Seruelas

Reflections:

Answer the following:

1. What are the benefits of having an availability monitoring tool?
    - The benefits of having an availability monitoring tool is that it allows the administrators to be able to keep track of the services that are currently running in their services, allowing them to know what is running and what services are idle. With this tool, they are able to keep track and they are able to consistently monitor what services are running, unstable and possibly that can lead to stopping any time soon, allowing them to move efficiently.

Conclusions:

In this activity, we were able to learn about the importance of having an availability monitoring tool in system administration and how necessary it is to have one. We have educated ourselves that there are multiple variants of an availability monitoring tool and there are ones that suit best according to different operating systems. In this activity, we were put into a test in which we are able to test our knowledge on creating a playbook with the lessons that we have learned from previous activities. In this activity, we were able to implement the basics of creating a playbook, and also utilizing roles in our playbook in order to create a more efficient and memory saving playbook. In this activity, we were to face a challenge in which we are to install an availability monitoring tool that requires multiple steps in order to make it operable and functionable. In the end of the activity, we are able to conclude that we are able to implement the knowledge that we have learned from our previous activities in creating the playbook in this activity.