Name: Jozshua Amiel Alonzo	**Date Performed:** October 22, 2022
Course/Section: CPE31S23	**Date Submitted: October 23, 2022**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 2022-2023**

Activity 9: Install, Configure, and Manage Performance Monitoring tools

1. Objectives

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

2. Discussion

Github Link: https://github.com/jozshua/HOA9_Alonzo.git

Performance monitoring is a type of monitoring tool that identifies current resource consumption of the workload, in this page we will discuss multiple performance monitoring tool.

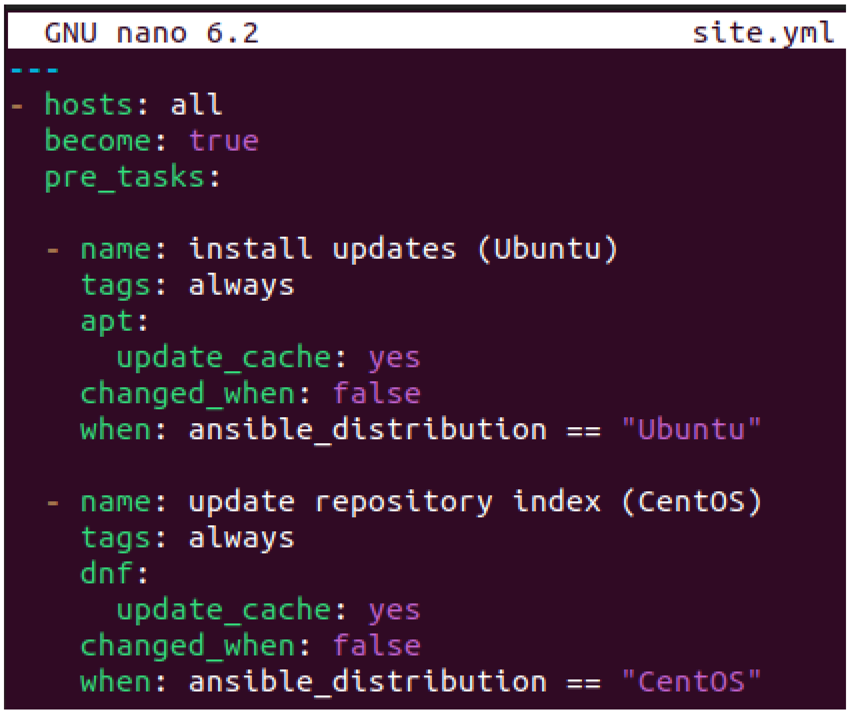
Prometheus
Prometheus fundamentally stores all data as timeseries: streams of timestamped values belonging to the same metric and the same set of labeled dimensions. Besides stored time series, Prometheus may generate temporary derived time series as the result of queries. Source: Prometheus - Monitoring system & time series database

Cacti
Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrapped in an intuitive, easy to use interface that makes sense for LAN-sized installations up to complex networks with thousands of devices. Source: Cacti® - The Complete RRDTool-based Graphing Solution

3. Tasks

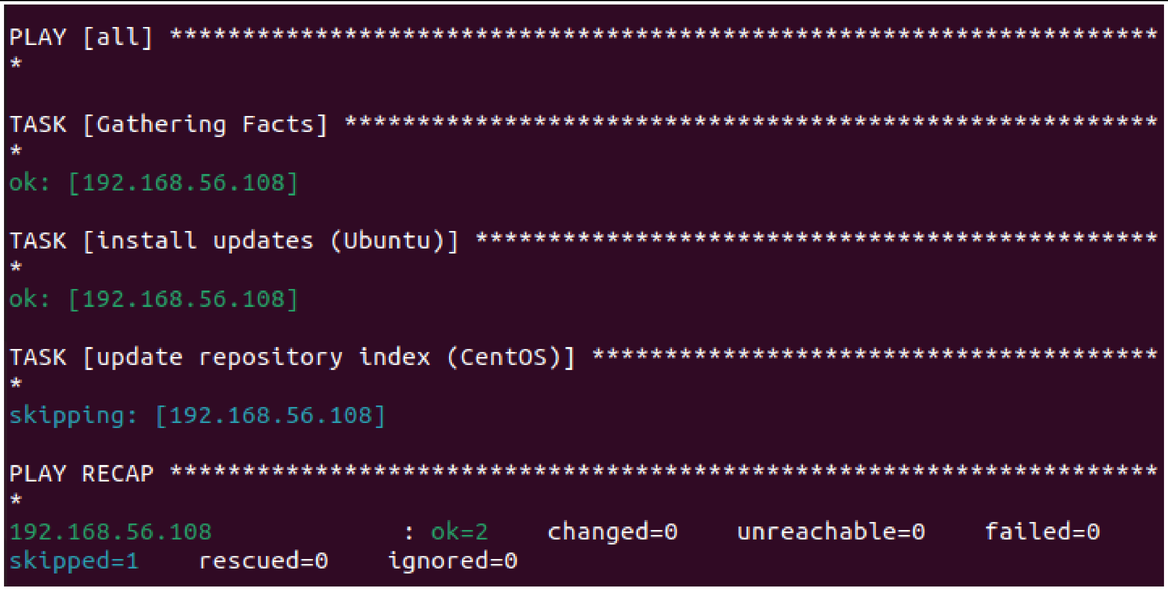
1. Create a playbook that installs Prometheus in both Ubuntu and CentOS. Apply the concept of creating roles.
2. Describe how you did step 1. (Provide screenshots and explanations in your report. Make your report detailed such that it will look like a manual.)
3. Show an output of the installed Prometheus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

4. **Output** (screenshots and explanations)

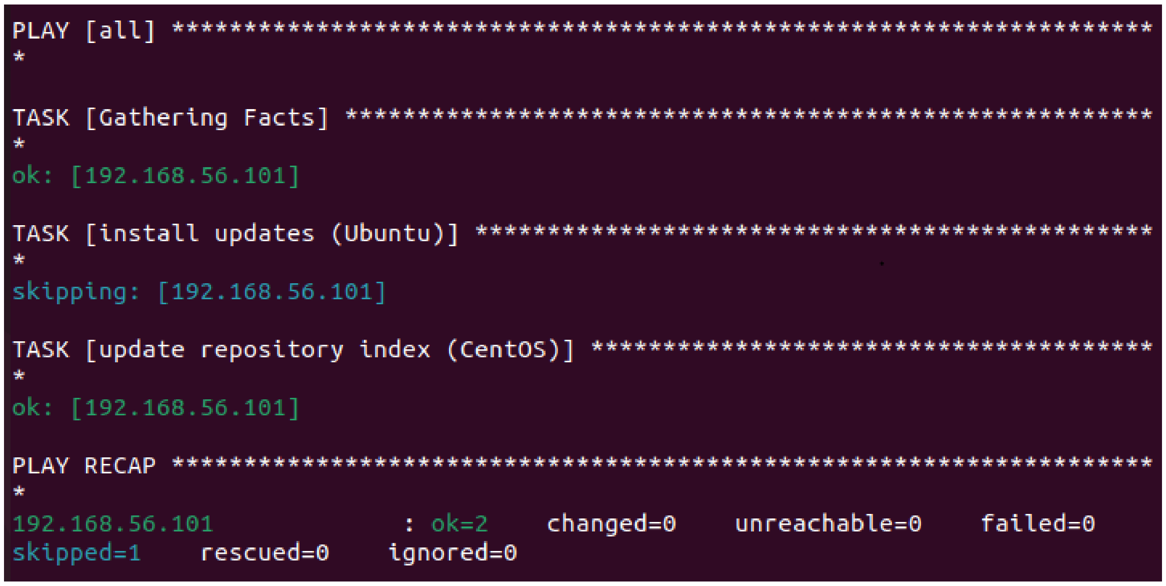


```
  GNU nano 6.2                                    site.yml
---
- hosts: all
  become: true
  pre_tasks:

  - name: install updates (Ubuntu)
    tags: always
    apt:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "Ubuntu"

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

Create the site.yml file and type these codes inside, these codes are for the two server distributions. Run the playbook.



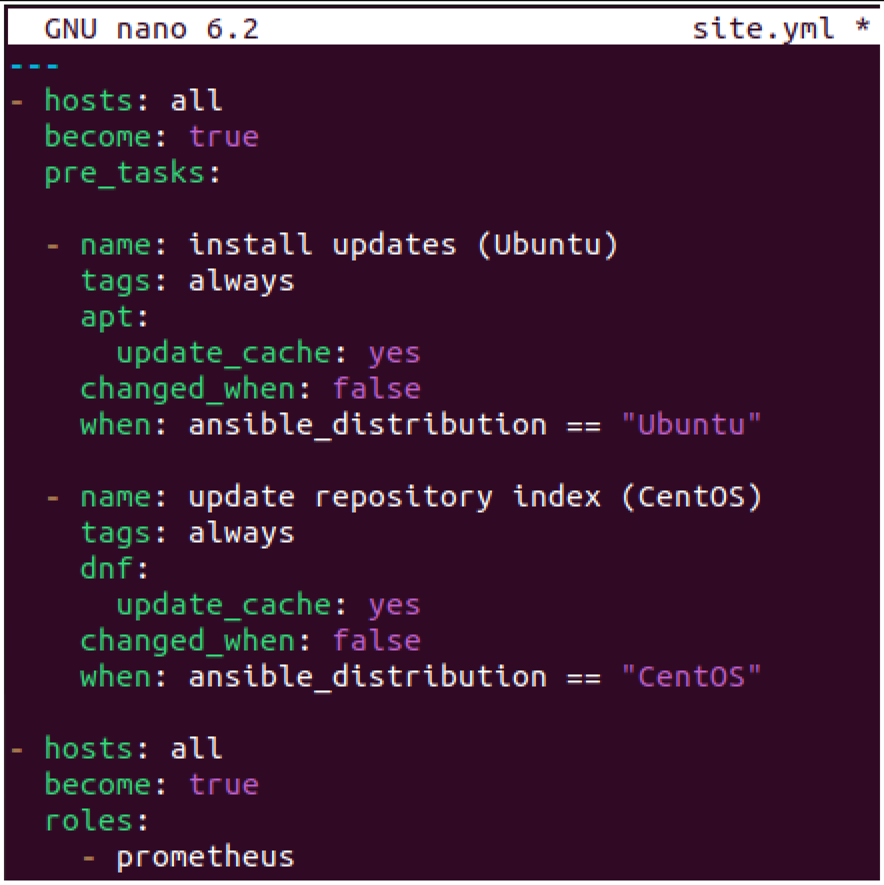
```
PLAY [all] *******************************************************************
*

TASK [Gathering Facts] *******************************************************
*
ok: [192.168.56.108]

TASK [install updates (Ubuntu)] **********************************************
*
ok: [192.168.56.108]

TASK [update repository index (CentOS)] **************************************
*
skipping: [192.168.56.108]

PLAY RECAP *******************************************************************
*
192.168.56.108             : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```



```
PLAY [all] *******************************************************************
*

TASK [Gathering Facts] *******************************************************
*
ok: [192.168.56.101]

TASK [install updates (Ubuntu)] **********************************************
*
skipping: [192.168.56.101]

TASK [update repository index (CentOS)] **************************************
*
ok: [192.168.56.101]

PLAY RECAP *******************************************************************
*
192.168.56.101             : ok=2    changed=0    unreachable=0    failed=0
skipped=1    rescued=0    ignored=0
```

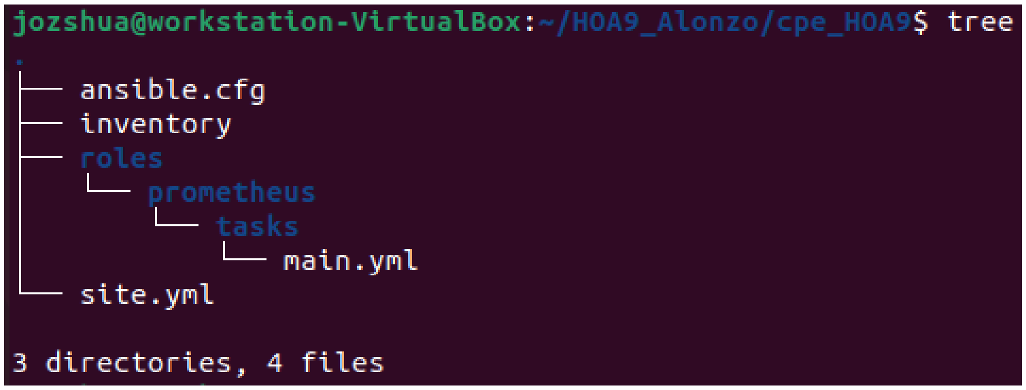
From running this playbook the tasks were successfully applied on each server distribution. There are 2 skipped states in each server because of having a different distribution from the ansible command.

Applying the concept of creating roles:



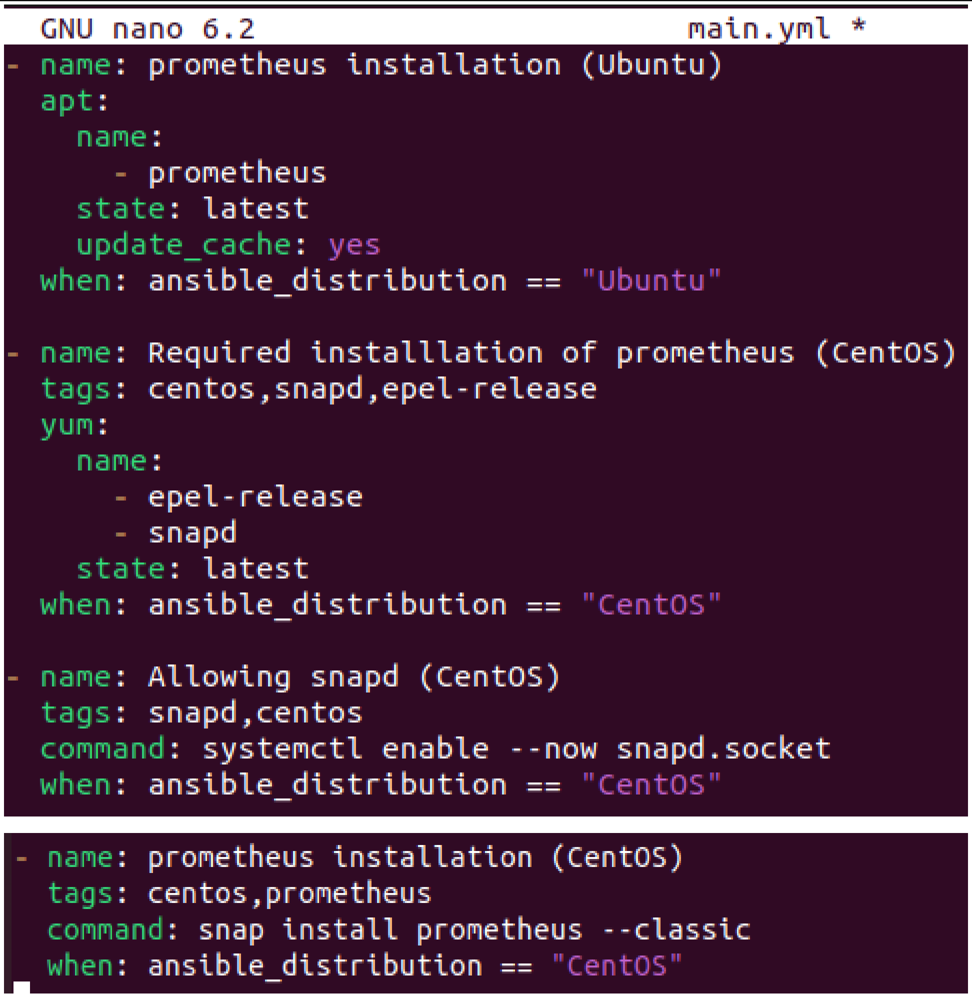
```
  GNU nano 6.2                        site.yml *
---
- hosts: all
  become: true
  pre_tasks:

  - name: install updates (Ubuntu)
    tags: always
    apt:
      update_cache: yes
    changed_when: false
    when: ansible_distribution == "Ubuntu"

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

- hosts: all
  become: true
  roles:
    - prometheus
```

Edit the site.yml file and insert these codes below from applying the main.yml file later.



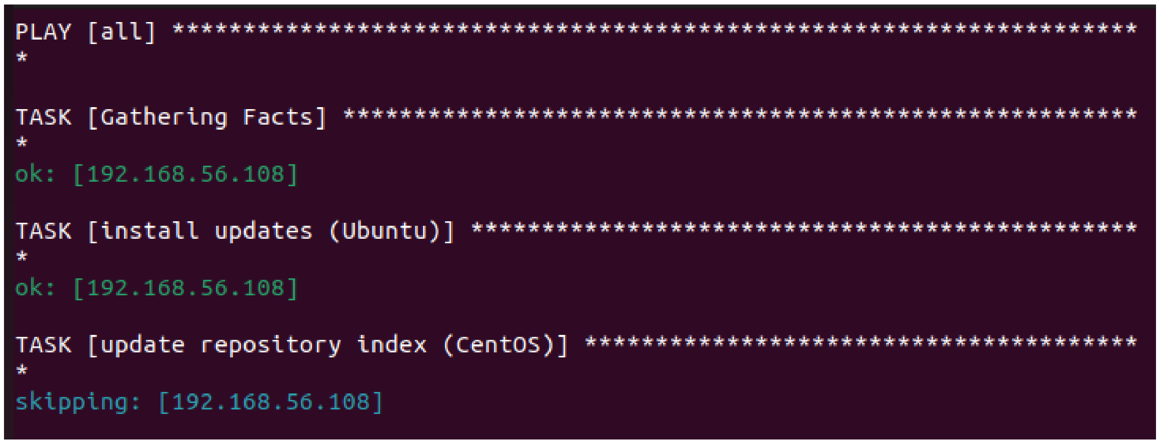
```
jozshua@workstation-VirtualBox:~/HOA9_Alonzo/cpe_HOA9$ tree
.
├── ansible.cfg
├── inventory
├── roles
│   └── prometheus
│       └── tasks
│           └── main.yml
└── site.yml

3 directories, 4 files
```

Create the roles, Prometheus, and tasks directories for creating the main.yml file. Issue the command tree to display the route.

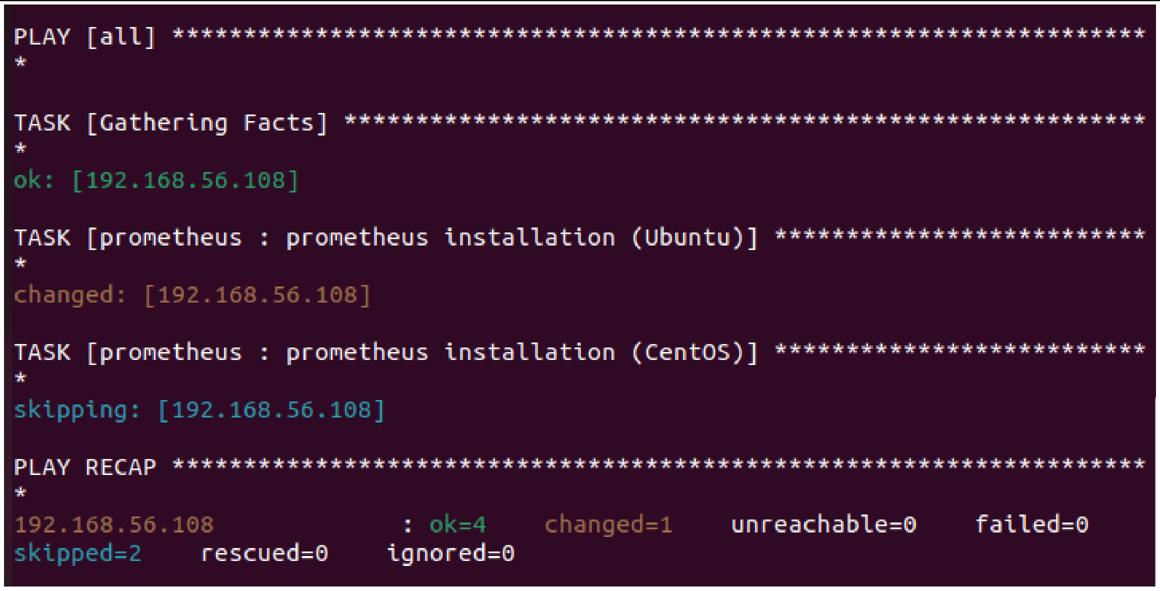


```
  GNU nano 6.2                                  main.yml *
- name: prometheus installation (Ubuntu)
  apt:
    name:
      - prometheus
    state: latest
    update_cache: yes
  when: ansible_distribution == "Ubuntu"

- name: Required installlation of prometheus (CentOS)
  tags: centos,snapd,epel-release
  yum:
    name:
      - epel-release
      - snapd
    state: latest
  when: ansible_distribution == "CentOS"

- name: Allowing snapd (CentOS)
  tags: snapd,centos
  command: systemctl enable --now snapd.socket
  when: ansible_distribution == "CentOS"

- name: prometheus installation (CentOS)
  tags: centos,prometheus
  command: snap install prometheus --classic
  when: ansible_distribution == "CentOS"
```

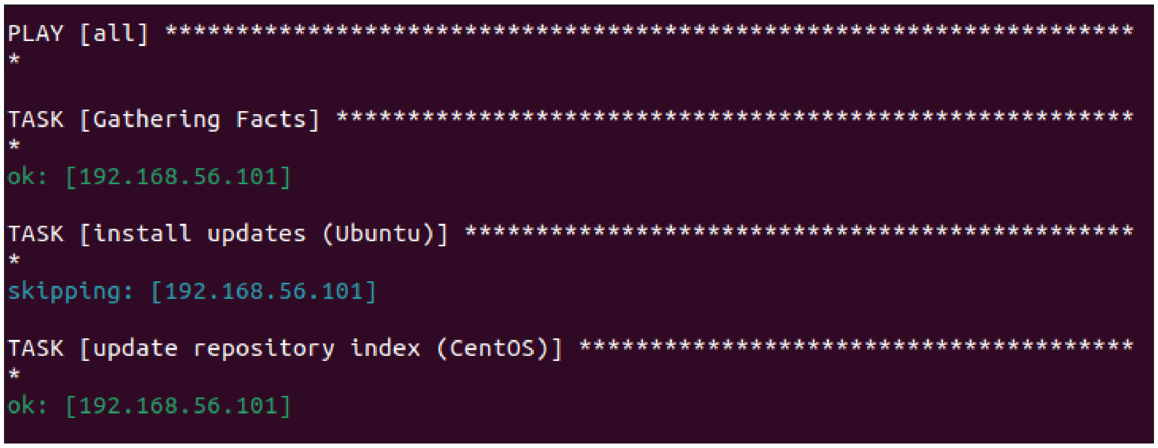
Apply the concept of creating roles and from the tasks, directory create the main.yml file. After that go back to the original directory by issuing the command “cd ..” Run the playbook once again.



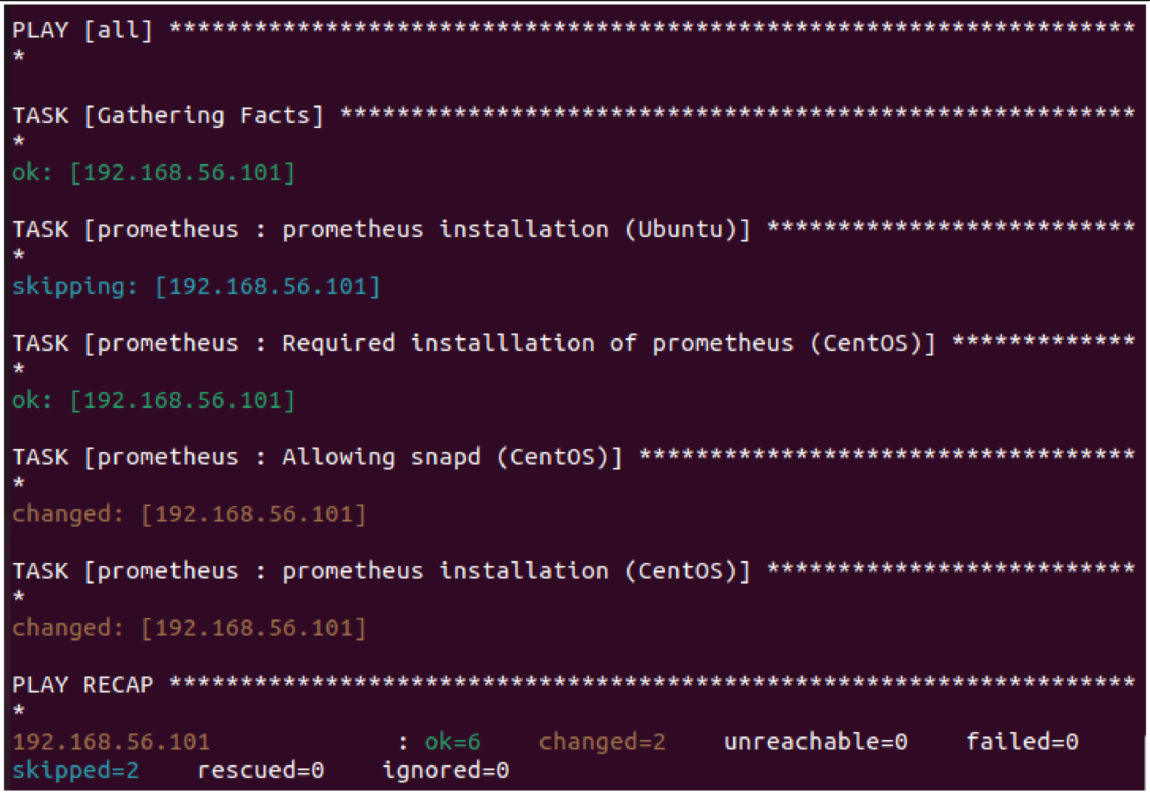
```
PLAY [all] **********************************************************************
*

TASK [Gathering Facts] **********************************************************
*
ok: [192.168.56.108]

TASK [install updates (Ubuntu)] *************************************************
*
ok: [192.168.56.108]

TASK [update repository index (CentOS)] *****************************************
*
skipping: [192.168.56.108]
```



```
PLAY [all] *********************************************************************
*

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.108]

TASK [prometheus : prometheus installation (Ubuntu)] ***************************
*
changed: [192.168.56.108]

TASK [prometheus : prometheus installation (CentOS)] ***************************
*
skipping: [192.168.56.108]

PLAY RECAP *********************************************************************
*
192.168.56.108             : ok=4    changed=1    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
```



```
PLAY [all] *********************************************************************
*

TASK [Gathering Facts] *********************************************************
*
ok: [192.168.56.101]

TASK [install updates (Ubuntu)] ************************************************
*
skipping: [192.168.56.101]

TASK [update repository index (CentOS)] ****************************************
*
ok: [192.168.56.101]
```

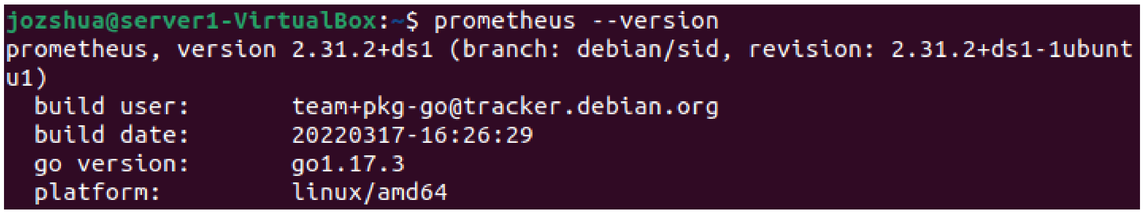


```
PLAY [all] *******************************************************************
*

TASK [Gathering Facts] *******************************************************
*
ok: [192.168.56.101]

TASK [prometheus : prometheus installation (Ubuntu)] *************************
*
skipping: [192.168.56.101]

TASK [prometheus : Required installlation of prometheus (CentOS)] ************
*
ok: [192.168.56.101]

TASK [prometheus : Allowing snapd (CentOS)] **********************************
*
changed: [192.168.56.101]

TASK [prometheus : prometheus installation (CentOS)] *************************
*
changed: [192.168.56.101]

PLAY RECAP *******************************************************************
*
192.168.56.101             : ok=6    changed=2    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
```

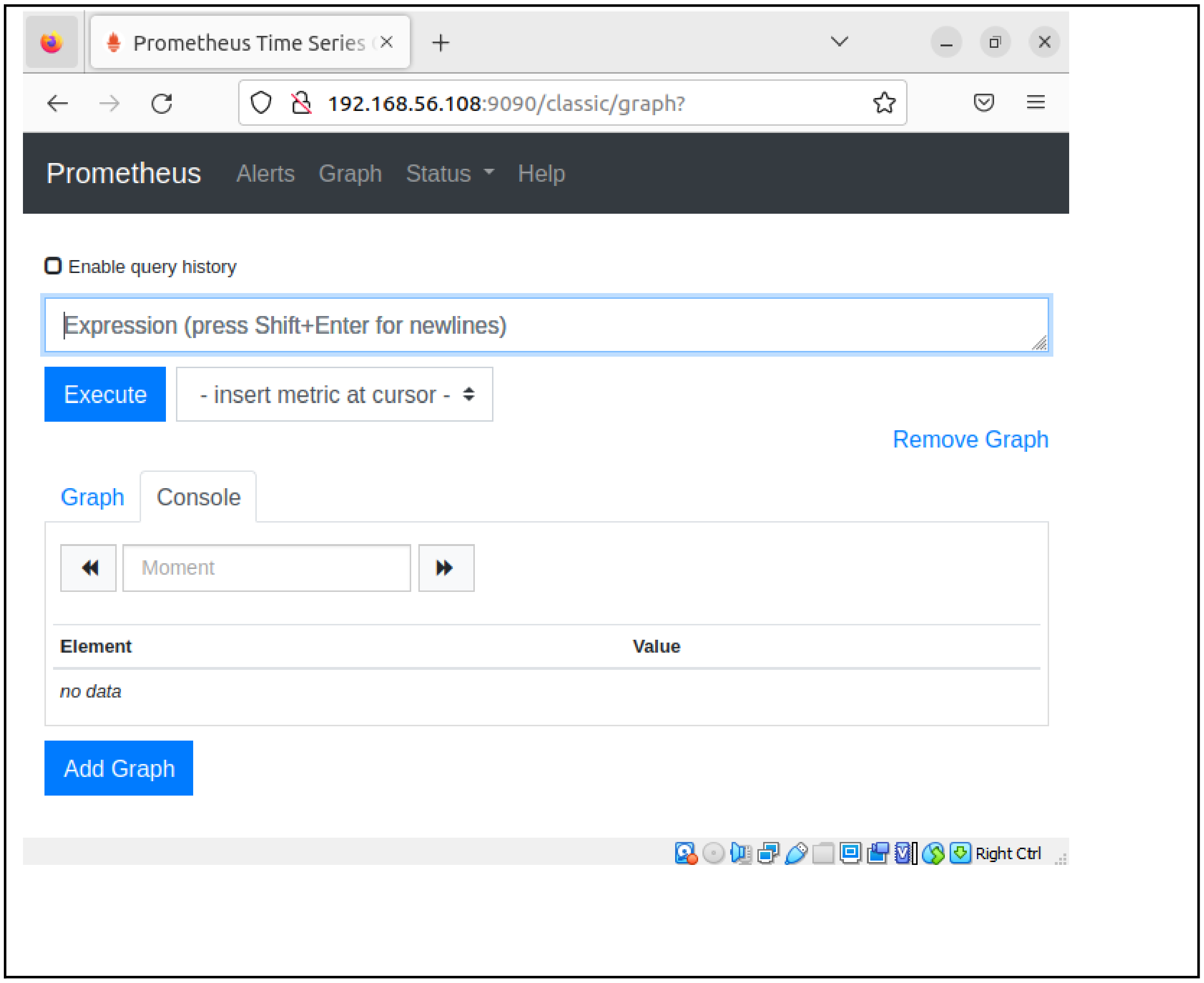
There are a total of 10 tasks that were successfully executed from running this playbook. Also, there are a total of 3 changed states wherein the tasks from the installation of the Prometheus in each remote server. Lastly, there are a total of 4 skipped states that is because of having different ansible distributions when running the tasks for each remote server in the files that we created.

Showing the installed Prometheus output:

From Ubuntu

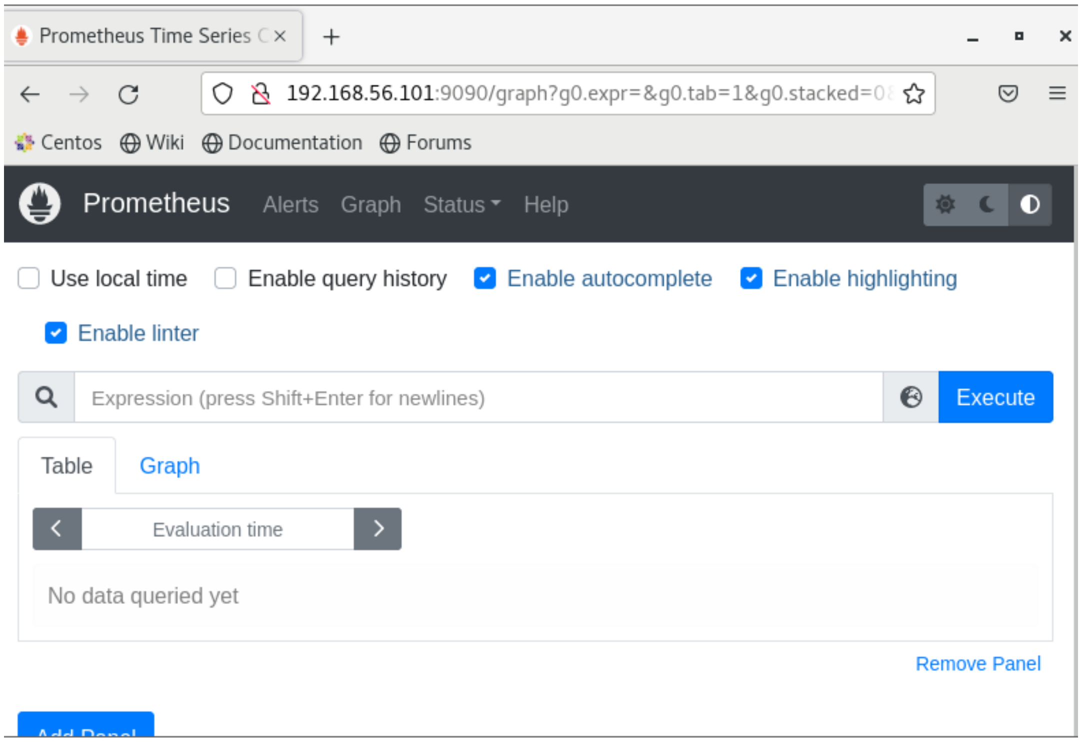


```
jozshua@server1-VirtualBox:~$ prometheus --version
prometheus, version 2.31.2+ds1 (branch: debian/sid, revision: 2.31.2+ds1-1ubunt
u1)
  build user:       team+pkg-go@tracker.debian.org
  build date:       20220317-16:26:29
  go version:       go1.17.3
  platform:         linux/amd64
```

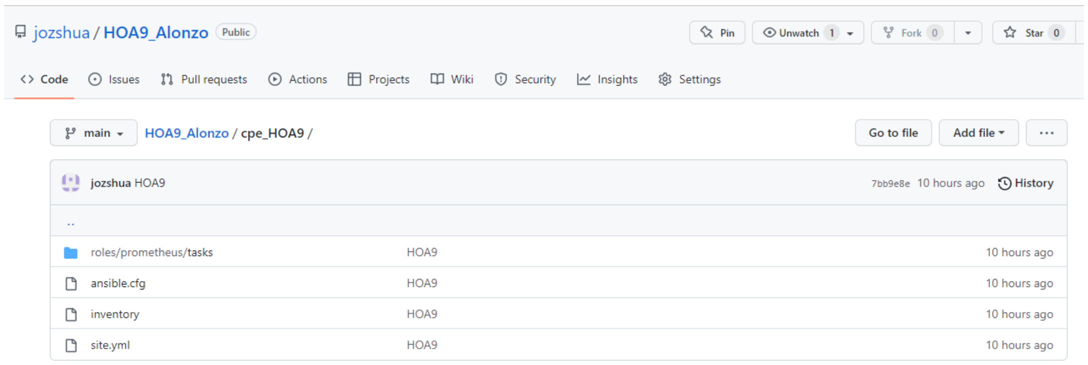


Prometheus Time Series
192.168.56.108:9090/classic/graph?
Prometheus
Alerts
Graph
Status
Help
Enable query history
Expression (press Shift+Enter for newlines)
Execute
- insert metric at cursor -
Remove Graph
Graph
Console
Moment
Element
Value
no data
Add Graph
Right Ctrl

From CentOS



As we can see from the browser of both remote servers after searching the IP address of each remote server, you can see the Prometheus site indicating that Prometheus was already installed from the remote servers.



..		
roles/prometheus/tasks	HOA9	10 hours ago
ansible.cfg	HOA9	10 hours ago
inventory	HOA9	10 hours ago
site.yml	HOA9	10 hours ago

Each file that we created was already committed to the GitHub repository.

Reflections:

Answer the following:

1. What are the benefits of having a performance monitoring tool?

   The benefit of having a performance monitoring tool is that we can monitor the performance of the device or application that we are using. By monitoring it you can also check if you are having any issues with the device that you are using and you could fix it right away. By doing that you can guarantee that your application on the device is in a good condition at all times and you can prevent system downtime.

Conclusions:

In this activity, Using ansible as an Infrastructure as Code tool I created and designed the workflow of installing the Prometheus for both server distributions. Installing the Prometheus for both server distributions needs a site.yml file and main.yml file. Inside these files, there are codes that are necessary for executing the tasks for the ansible playbooks for the output. After running the playbook successfully I am able to display the installation of Prometheus for both servers by issuing the Prometheus –version command and by putting the IP address from their browser. Lastly, I make sure that all files that I created for this activity were committed to my Github repository. Therefore there is no sign of failure from the outputs so I conclude that I had finished this activity well.