Name: Davonn Escobilla	**Date Performed: 27/10/2022**
Course/Section: CPE31S24	**Date Submitted: 27/10/2022**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st Sem, 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

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

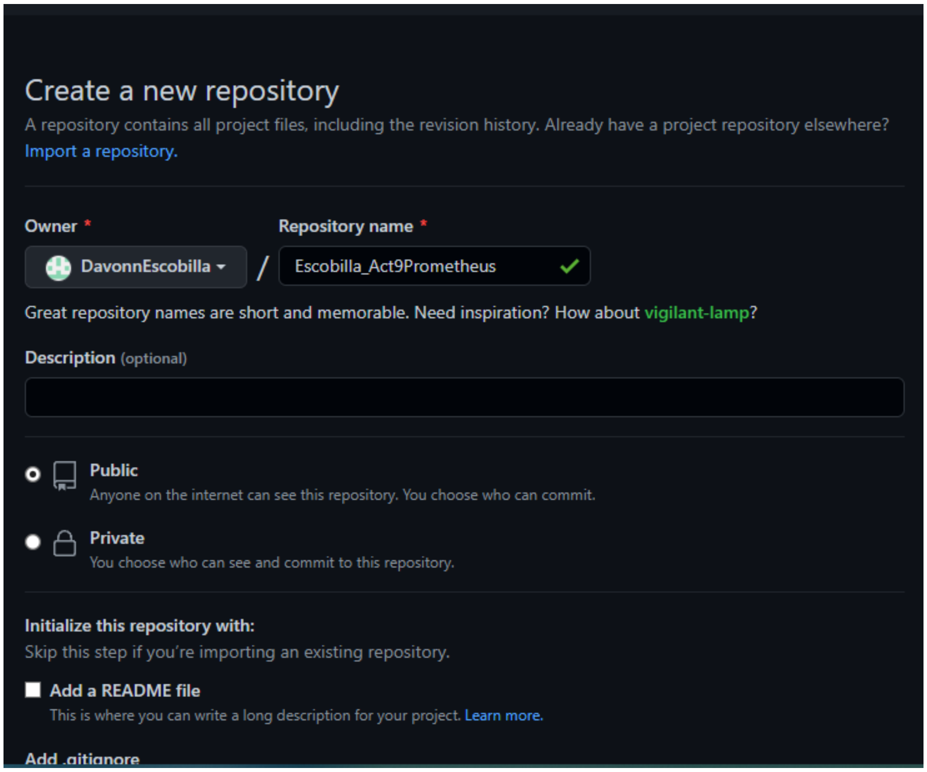
Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrapped in an intuitive, 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)

First step is to create a repository for Activity 9.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * / Repository name *

DavonnEscobilla / Escobilla_Act9Prometheus ✓

Great repository names are short and memorable. Need inspiration? How about vigilant-lamp?

Description (optional)

○ Public
Anyone on the internet can see this repository. You choose who can commit.

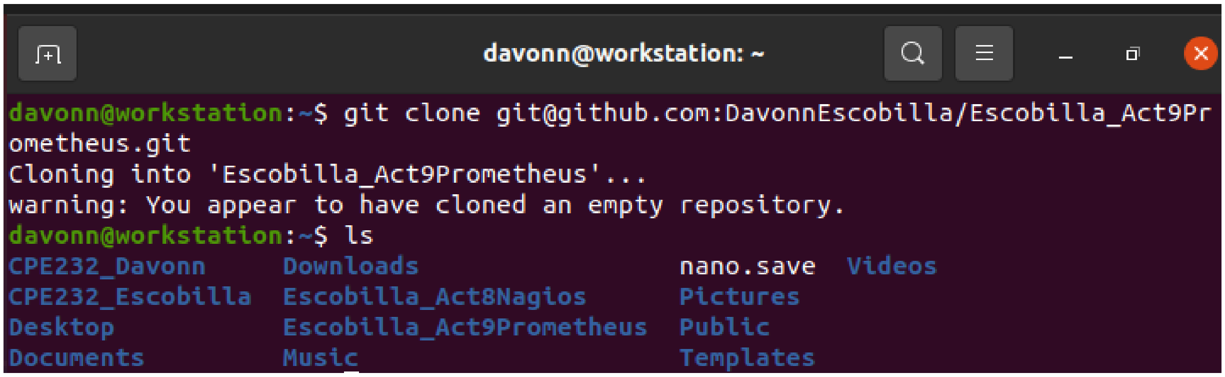
● Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

□ Add a README file
This is where you can write a long description for your project. Learn more.

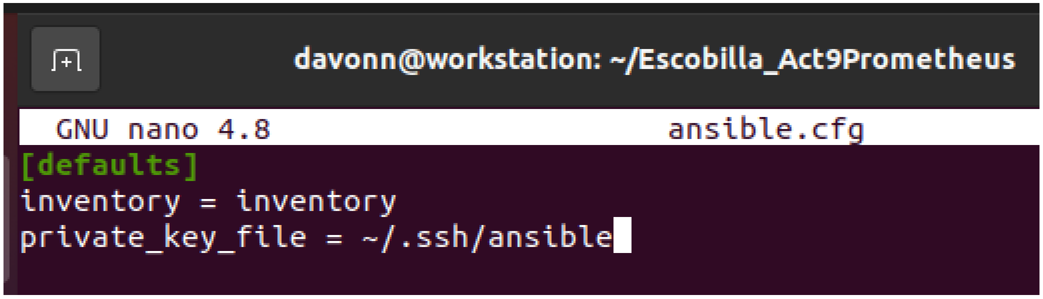
Add .gitignore

Second step is to clone the created repository on a managed node.

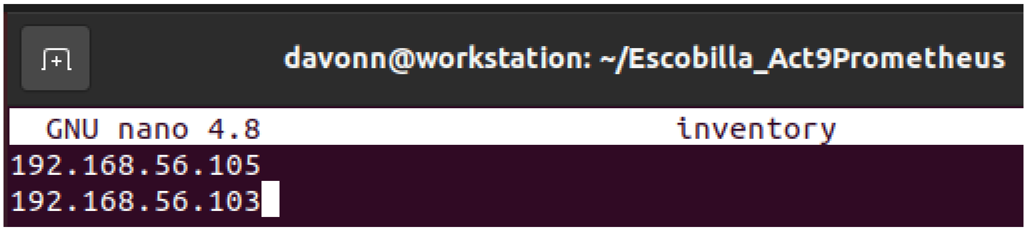


```
davonn@workstation:~$ git clone git@github.com:DavonnEscobilla/Escobilla_Act9Pr
ometheus.git
Cloning into 'Escobilla_Act9Prometheus'...
warning: You appear to have cloned an empty repository.
davonn@workstation:~$ ls
CPE232_Davonn      Downloads                 nano.save  Videos
CPE232_Escobilla   Escobilla_Act8Nagios      Pictures
Desktop            Escobilla_Act9Prometheus  Public
Documents          Music                     Templates
```

Third step, create the ansible configuration file and inventory for connection on control nodes.

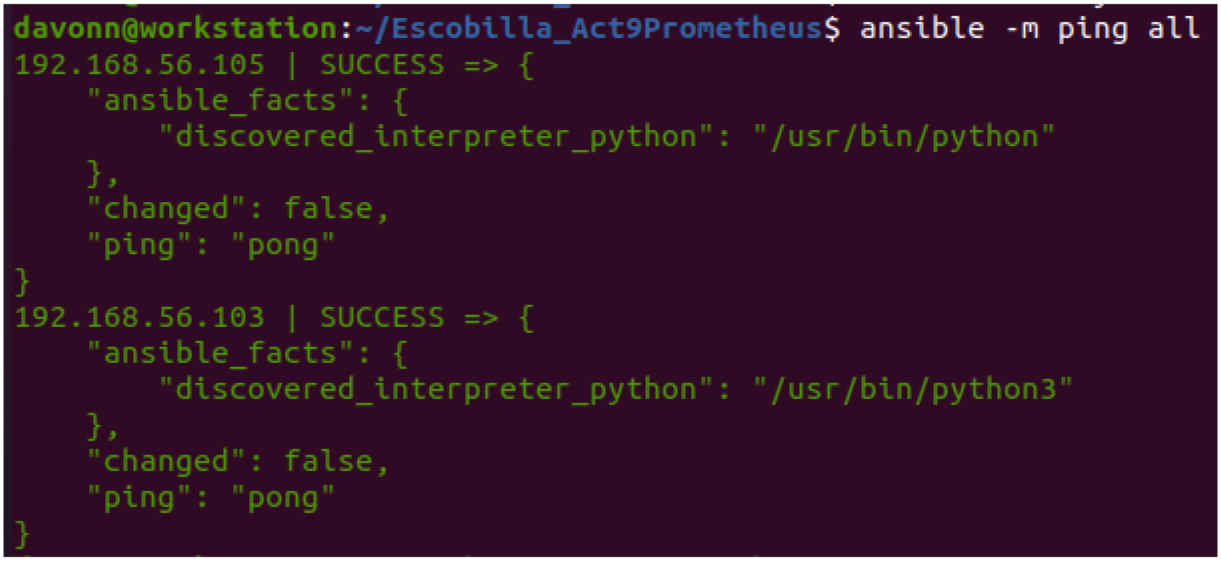


```
  GNU nano 4.8                          ansible.cfg
[defaults]
inventory = inventory
private_key_file = ~/.ssh/ansible
```



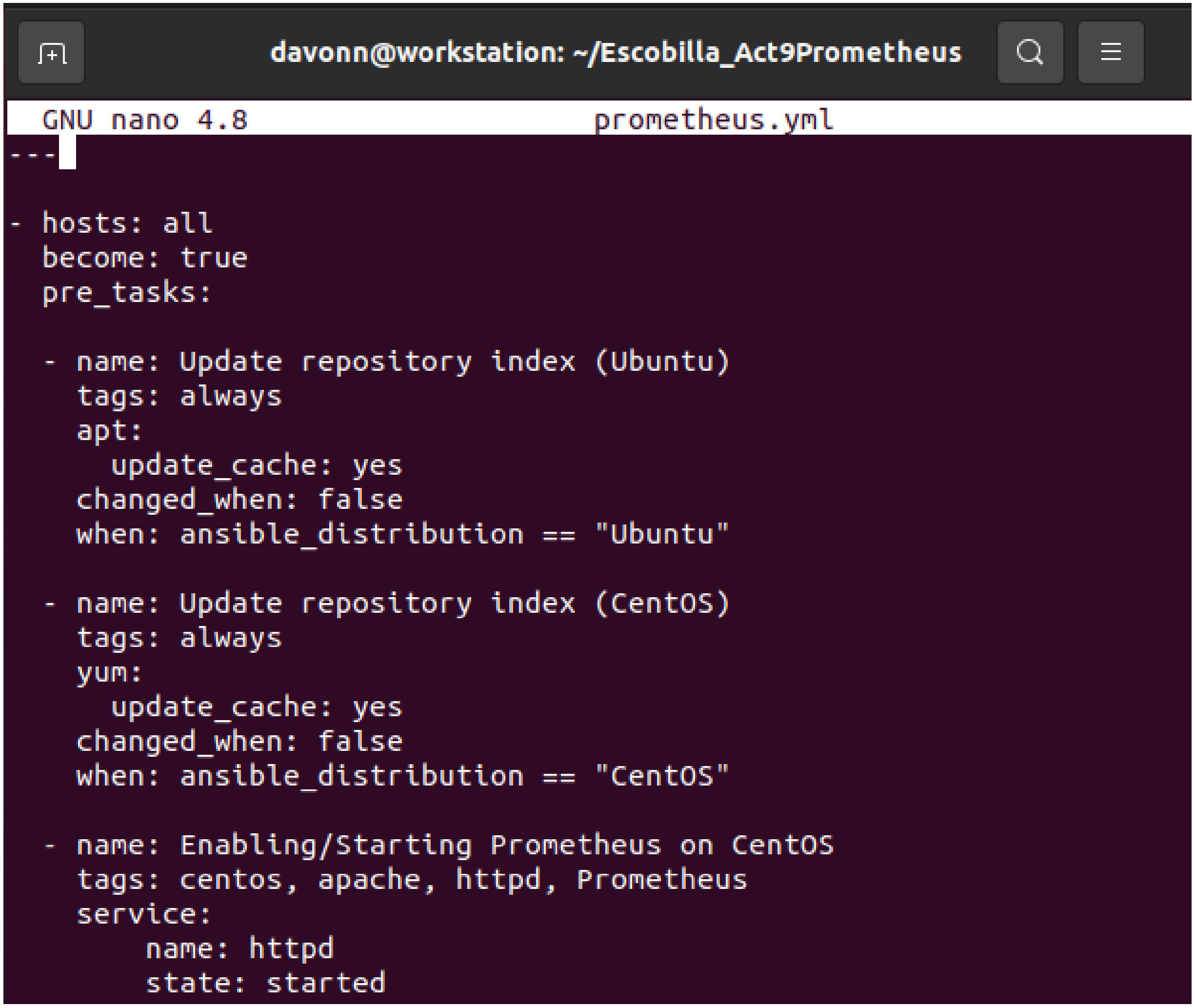
```
  GNU nano 4.8                            inventory
192.168.56.105
192.168.56.103
```

Before proceeding to the next step, the connection between managed node to control nodes must be verified.



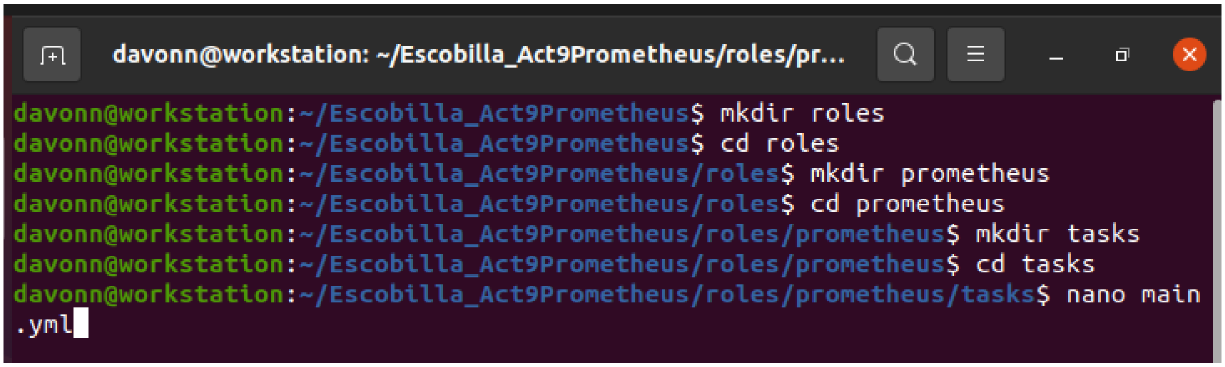
```
davonn@workstation:~/Escobilla_Act9Prometheus$ ansible -m ping all
192.168.56.105 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.56.103 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Fourth step, create prometheus.yml that contains the basic tasks such as updating.

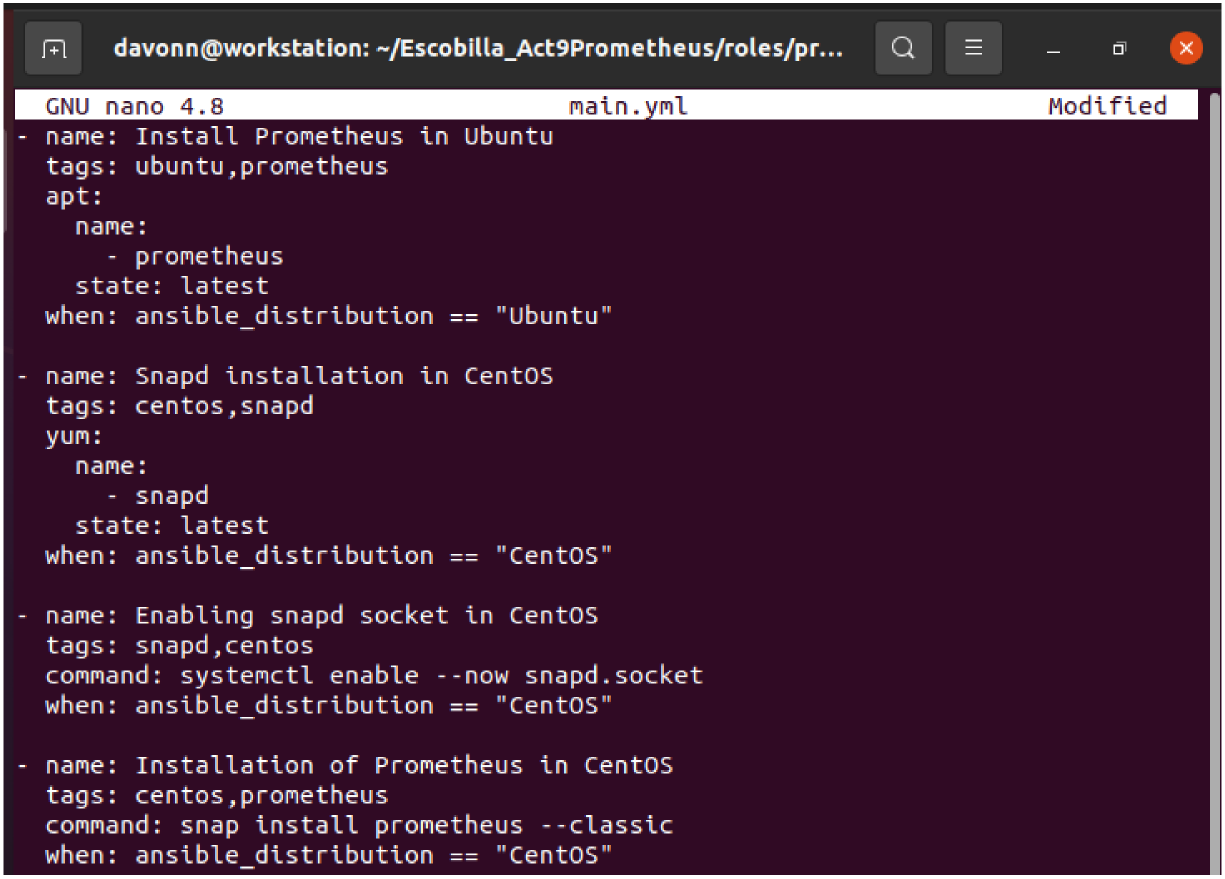


```
---

- hosts: all
  become: true
  pre_tasks:

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

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

  - name: Enabling/Starting Prometheus on CentOS
    tags: centos, apache, httpd, Prometheus
    service:
        name: httpd
        state: started
```

Fifth step, create directories for roles inside the repository. Under that directory create main.yml for the assigned task on installing prometheus to control nodes.

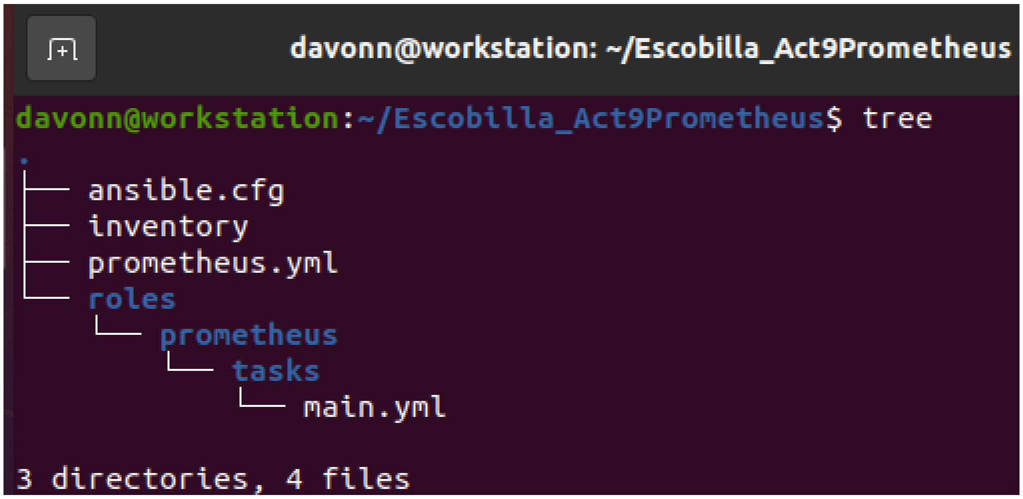


```
davonn@workstation:~/Escobilla_Act9Prometheus$ mkdir roles
davonn@workstation:~/Escobilla_Act9Prometheus$ cd roles
davonn@workstation:~/Escobilla_Act9Prometheus/roles$ mkdir prometheus
davonn@workstation:~/Escobilla_Act9Prometheus/roles$ cd prometheus
davonn@workstation:~/Escobilla_Act9Prometheus/roles/prometheus$ mkdir tasks
davonn@workstation:~/Escobilla_Act9Prometheus/roles/prometheus$ cd tasks
davonn@workstation:~/Escobilla_Act9Prometheus/roles/prometheus/tasks$ nano main
.yml
```



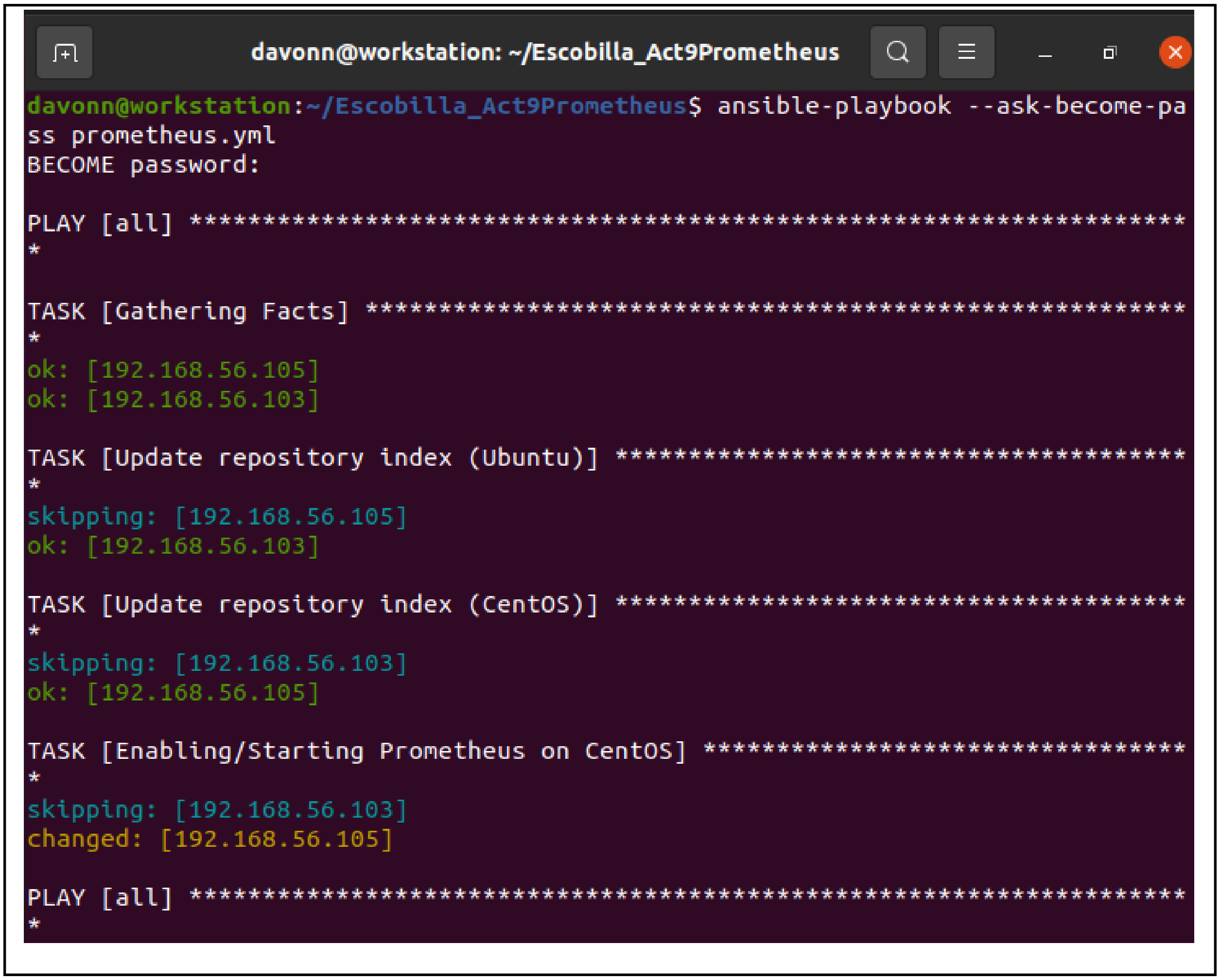
```
GNU nano 4.8                    main.yml                    Modified
- name: Install Prometheus in Ubuntu
  tags: ubuntu,prometheus
  apt:
    name:
      - prometheus
    state: latest
  when: ansible_distribution == "Ubuntu"

- name: Snapd installation in CentOS
  tags: centos,snapd
  yum:
    name:
      - snapd
    state: latest
  when: ansible_distribution == "CentOS"

- name: Enabling snapd socket in CentOS
  tags: snapd,centos
  command: systemctl enable --now snapd.socket
  when: ansible_distribution == "CentOS"

- name: Installation of Prometheus in CentOS
  tags: centos,prometheus
  command: snap install prometheus --classic
  when: ansible_distribution == "CentOS"
```

Before proceeding to the next step, check the directories using tree command.

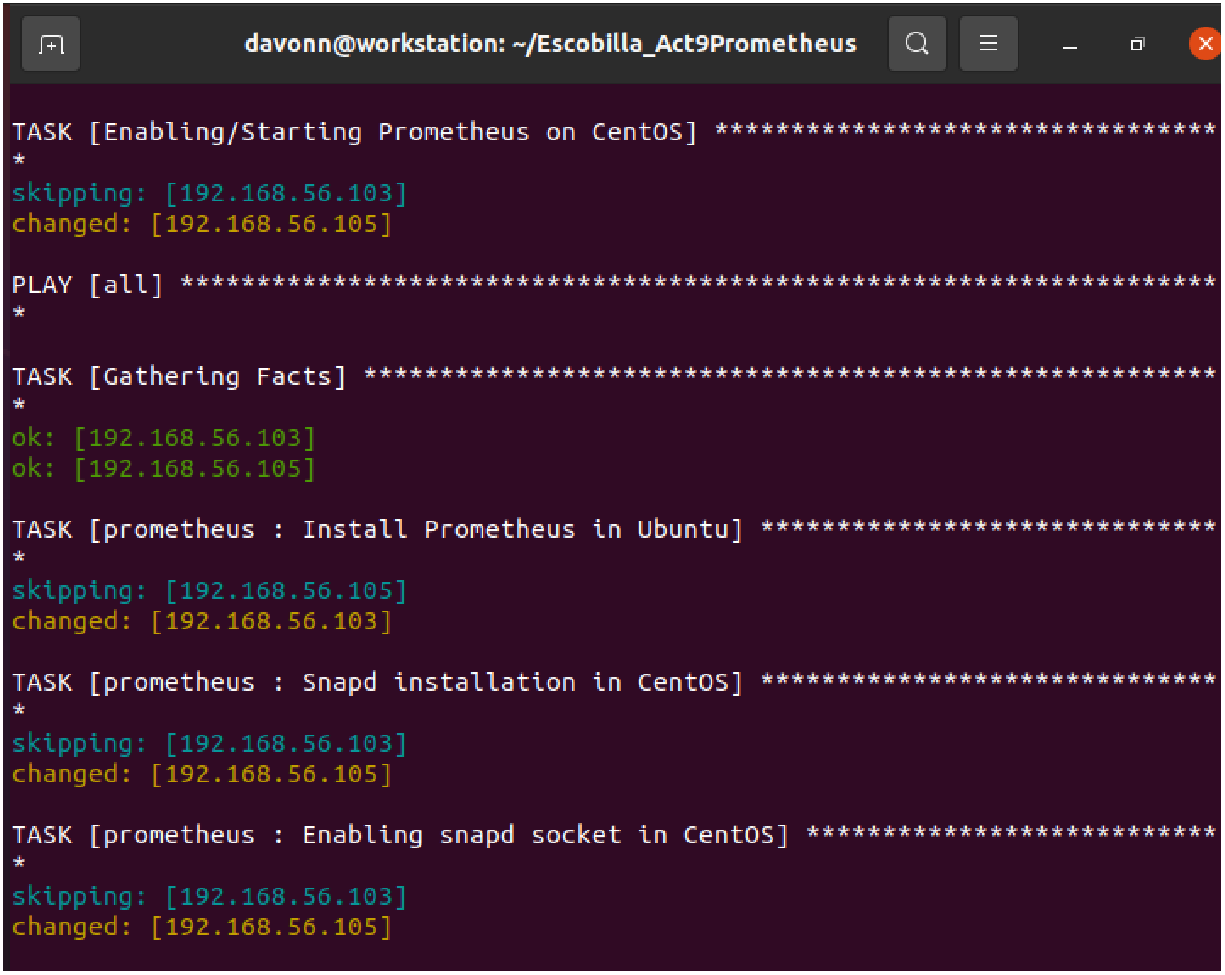


```
davonn@workstation:~/Escobilla_Act9Prometheus$ tree
.
├── ansible.cfg
├── inventory
├── prometheus.yml
└── roles
    └── prometheus
        └── tasks
            └── main.yml

3 directories, 4 files
```

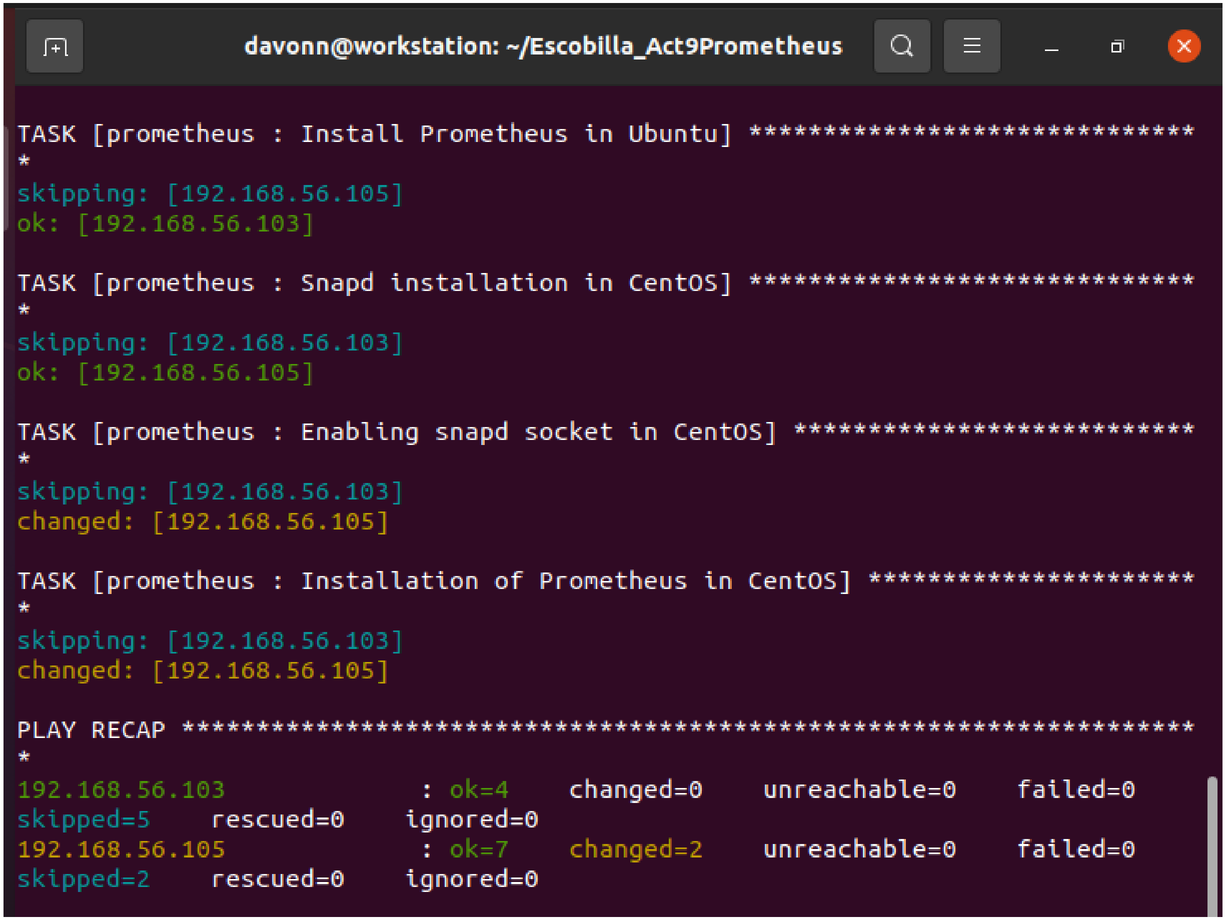
Now run the playbook.



```
davonn@workstation:~/Escobilla_Act9Prometheus$ ansible-playbook --ask-become-pa
ss prometheus.yml
BECOME password:

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

TASK [Gathering Facts] ********************************************************
*
ok: [192.168.56.105]
ok: [192.168.56.103]

TASK [Update repository index (Ubuntu)] ***************************************
*
skipping: [192.168.56.105]
ok: [192.168.56.103]

TASK [Update repository index (CentOS)] ***************************************
*
skipping: [192.168.56.103]
ok: [192.168.56.105]

TASK [Enabling/Starting Prometheus on CentOS] *********************************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]

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



```
TASK [Enabling/Starting Prometheus on CentOS] *********************************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]

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

TASK [Gathering Facts] ********************************************************
*
ok: [192.168.56.103]
ok: [192.168.56.105]

TASK [prometheus : Install Prometheus in Ubuntu] ******************************
*
skipping: [192.168.56.105]
changed: [192.168.56.103]

TASK [prometheus : Snapd installation in CentOS] ******************************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]

TASK [prometheus : Enabling snapd socket in CentOS] ***************************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]
```

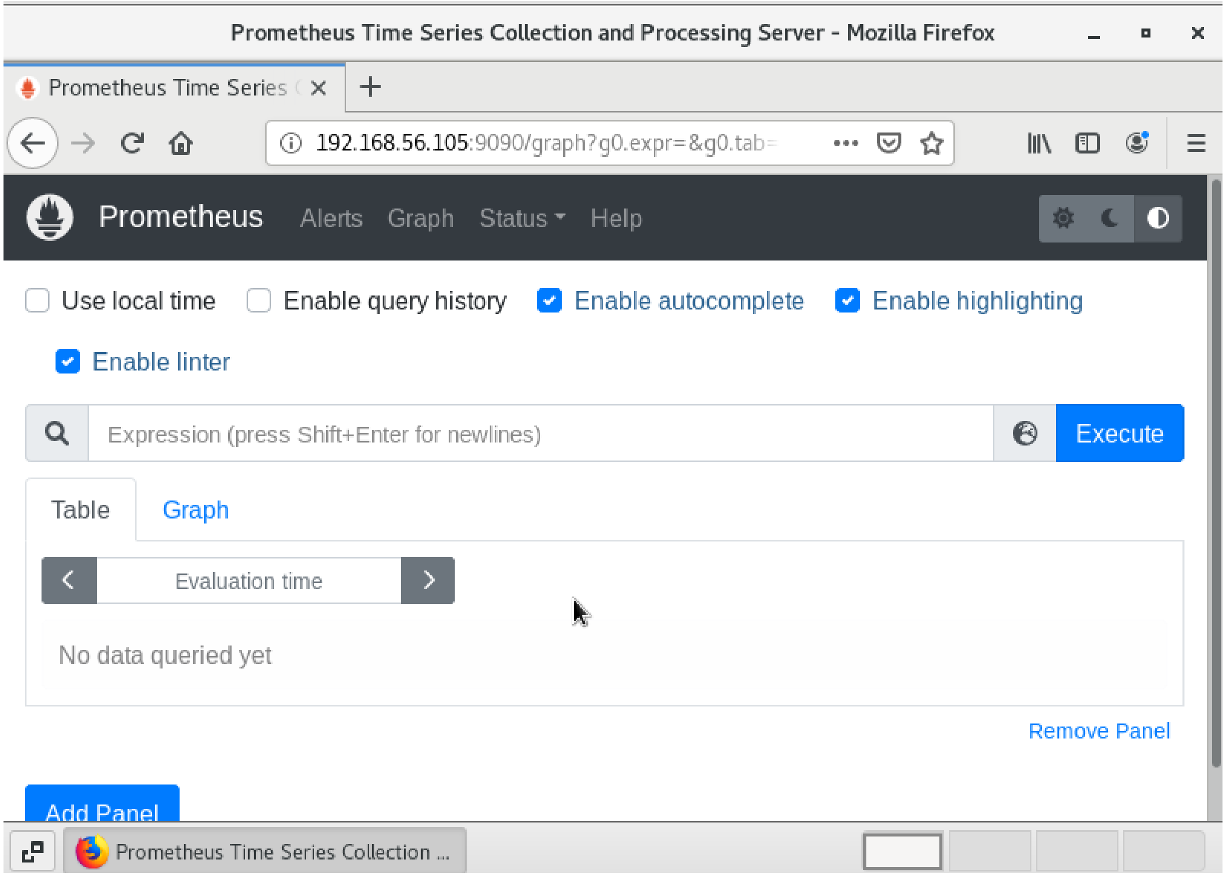


```
TASK [prometheus : Install Prometheus in Ubuntu] ******************************
*
skipping: [192.168.56.105]
ok: [192.168.56.103]

TASK [prometheus : Snapd installation in CentOS] ******************************
*
skipping: [192.168.56.103]
ok: [192.168.56.105]

TASK [prometheus : Enabling snapd socket in CentOS] ***************************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]

TASK [prometheus : Installation of Prometheus in CentOS] **********************
*
skipping: [192.168.56.103]
changed: [192.168.56.105]

PLAY RECAP ********************************************************************
*
192.168.56.103             : ok=4    changed=0    unreachable=0    failed=0
skipped=5    rescued=0    ignored=0
192.168.56.105             : ok=7    changed=2    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
```

All the tasks assigned on each control node are successfully executed as stated on the output of running the playbook.

Last step would be verifying the installation of the prometheus to the control nodes.

OUTPUT on CentOS:



Prometheus Time Series Collection and Processing Server - Mozilla Firefox
Prometheus Time Series (
Prometheus
Alerts
Graph
Status
Help
Execute
Table
Graph
Remove Panel
Add Panel
Prometheus Time Series Collection ...

192.168.56.105:9090/graph?g0.expr=&g0.tab=

□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting

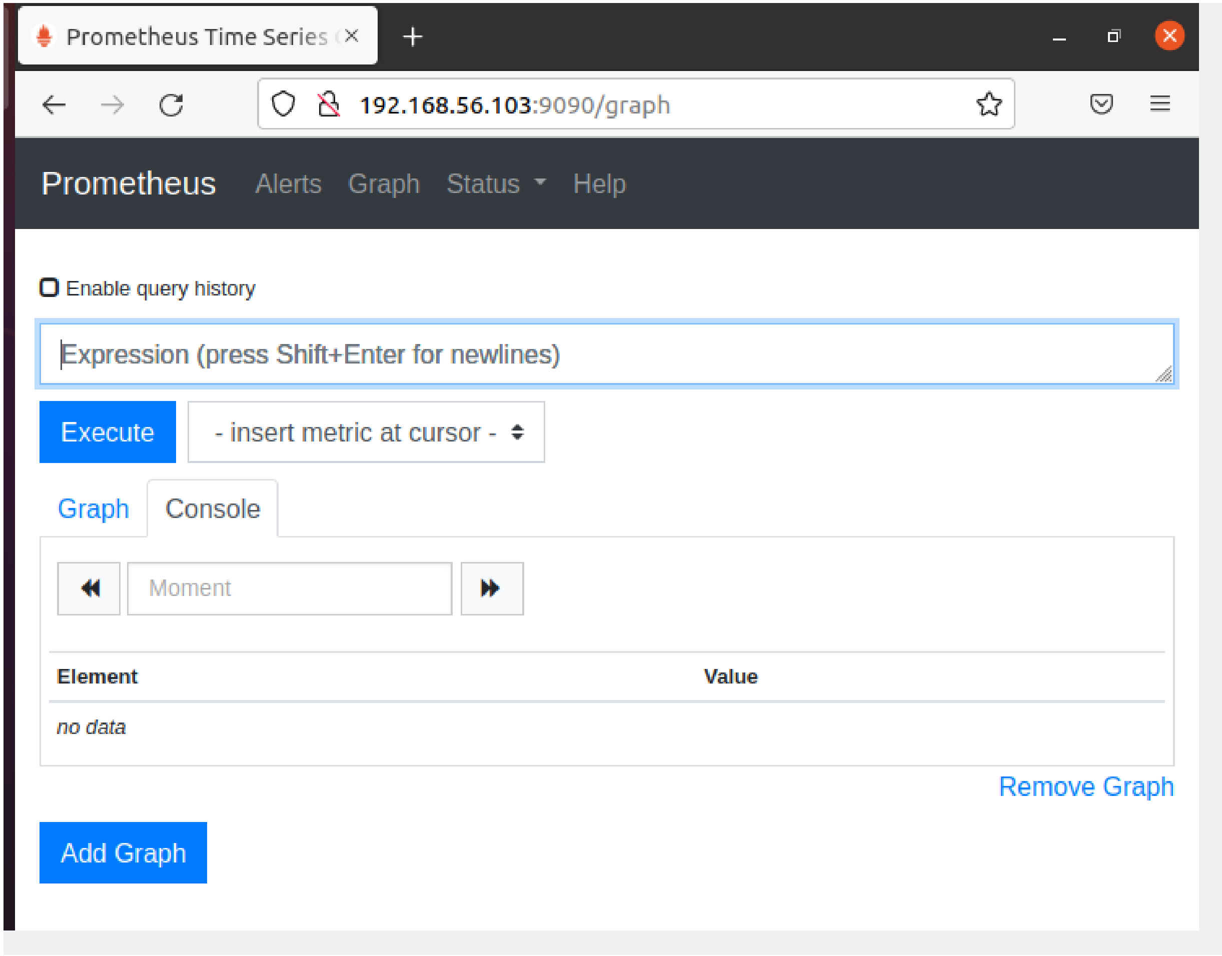
☑ Enable linter

Expression (press Shift+Enter for newlines)

Evaluation time

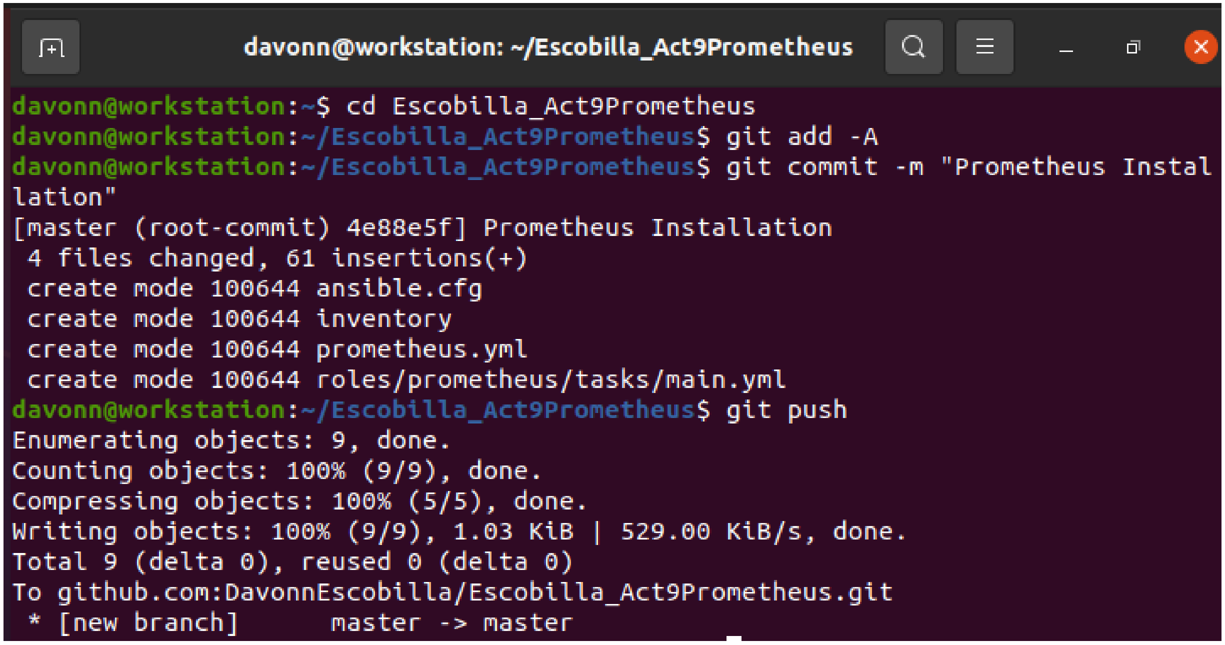
No data queried yet

OUTPUT on Ubuntu:



Element	Value
no data	

It can be now verified that the prometheus is installed on both control nodes, the last part is just saving the work on the repository.



```
davonn@workstation:~$ cd Escobilla_Act9Prometheus
davonn@workstation:~/Escobilla_Act9Prometheus$ git add -A
davonn@workstation:~/Escobilla_Act9Prometheus$ git commit -m "Prometheus Instal
lation"
[master (root-commit) 4e88e5f] Prometheus Installation
 4 files changed, 61 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 inventory
 create mode 100644 prometheus.yml
 create mode 100644 roles/prometheus/tasks/main.yml
davonn@workstation:~/Escobilla_Act9Prometheus$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 1.03 KiB | 529.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To github.com:DavonnEscobilla/Escobilla_Act9Prometheus.git
 * [new branch]      master -> master
```

Reflections:

Answer the following:

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

   The important benefits of a performance monitoring tool is that it can tell the amount of workload and current resources consumed. Example of this is that the prometheus stores all the data in a timeseries contains timestamped values on the same metric and labeled dimensions. In this way performance tool helps to measure out the proper distribution of resources and how it could affect the performance.

Conclusions:

This activity clarifies that a performance monitoring tool is a great program to use in order to identify the workload consumption of resources, as well as track all the problems that might occur in relevance with the data storage or graphing functionality.