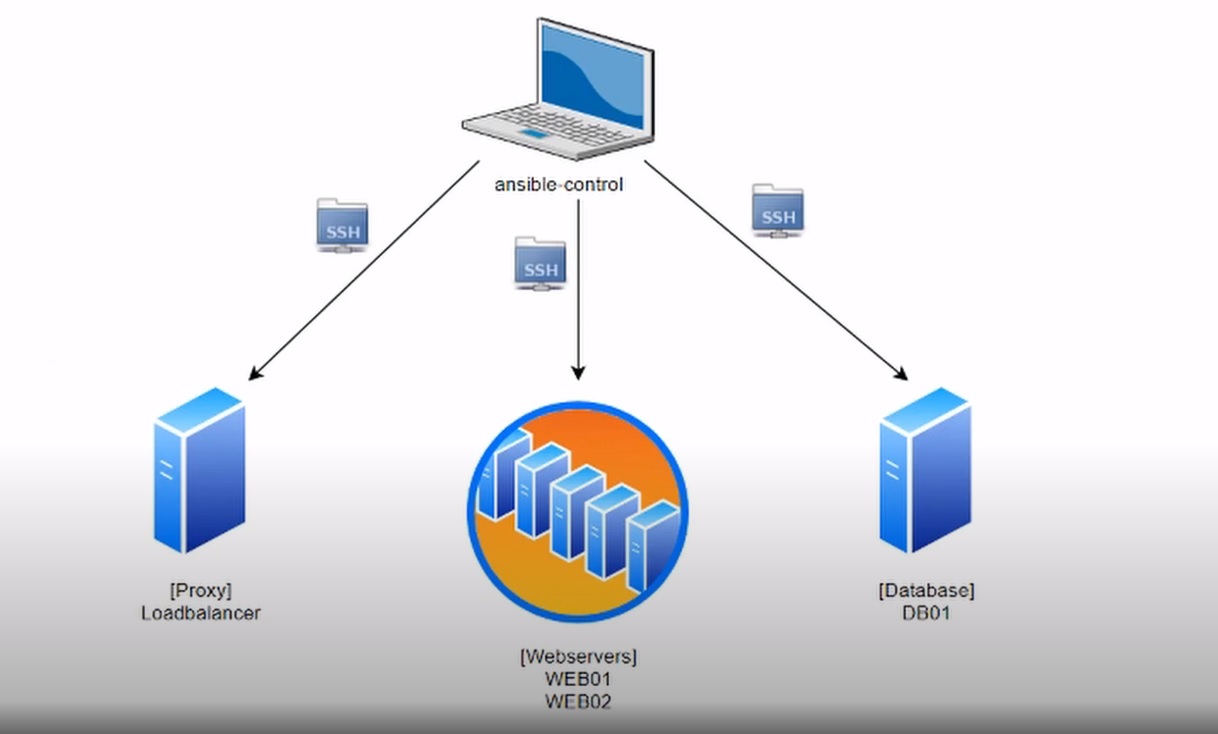
Before proceeding with the labs below, we have to install Oracle Virtual Box and Vagrant, following the steps in the pdf.

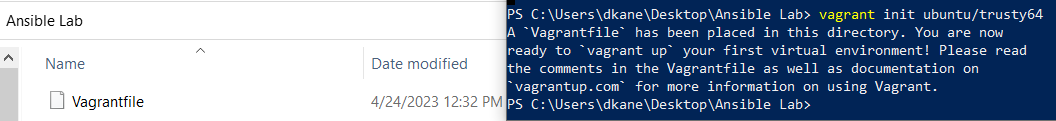
**Ansible Lab 1 - Installation and Inventory file basics.**

In the first lab, we will set up the Vagrant infrastructure that we need in order to learn Ansible.



**1. Create the Vagrantfile.**

- I created it using the following command below:

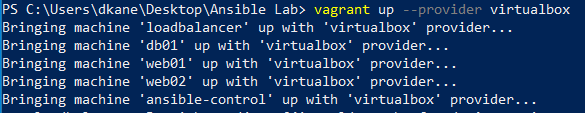


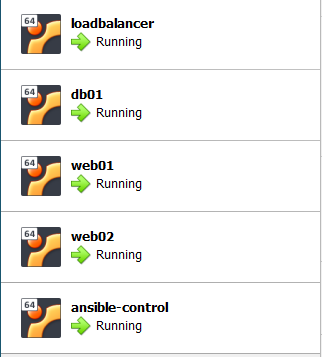
- I edited it with notepad++ and proceeded to the next step.

**2. Create VMs using vagrant and ssh to our control server.**

- We start off by setting up the Vagrant with the following command:

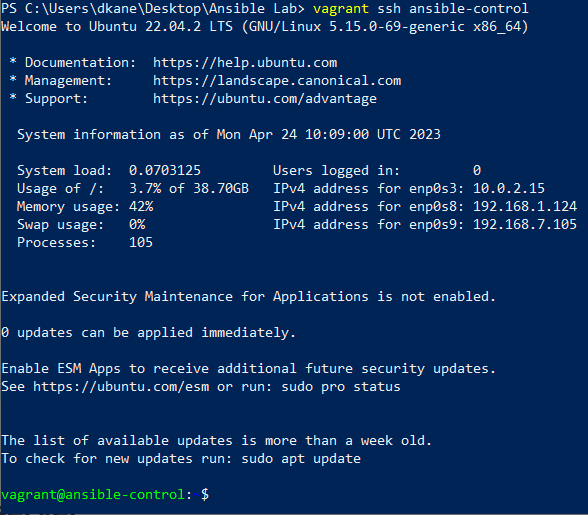
- vagrant up --provider virtualbox





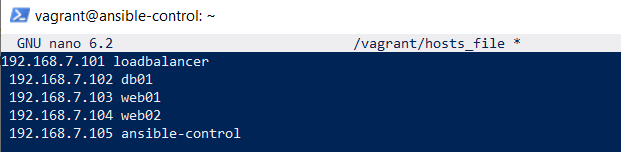
And then connect to the ansible-control server with:

- vagrant ssh ansible-control



**3. Create the /vagrant/hosts\_file.**





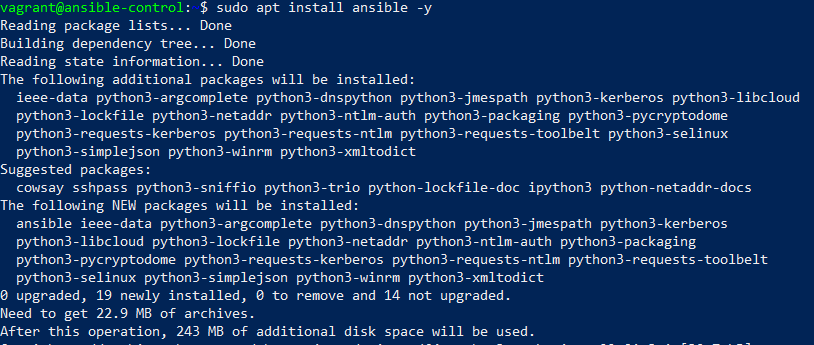
**4. Copy /vagrant/hosts\_file to /etc/hosts.**



**5. Install Ansible.**

- Since we are now connected to the ansible-control machine we can install Ansible with the following 2 commands:

- sudo apt update (to update and upgrade the machine) and - sudo apt install ansible -y (to install Ansible)



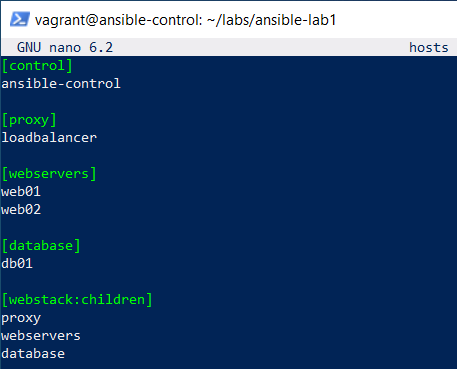
**6. Create an inventory file named hosts.**

- First, we create a directory structure for the labs:



- Then the hosts file:





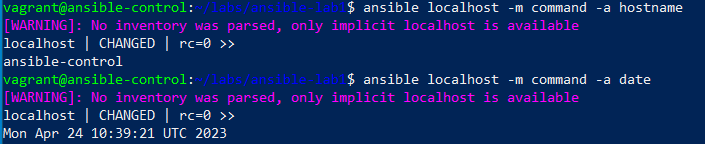
- Lastly, we copy the pre-defined hosts file:



**7. Test out a command.**

- Now we test if the setup works with the following commands:

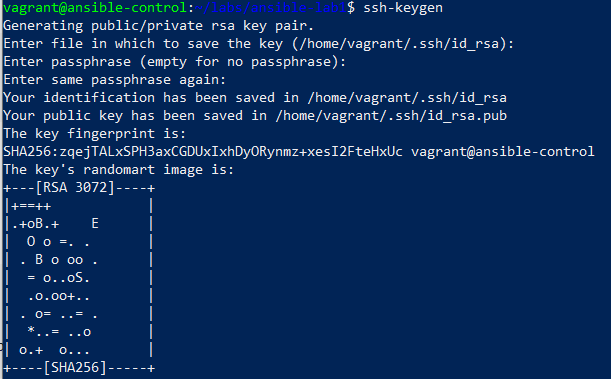
- ansible localhost -m command -a hostname and ansible localhost -m command -a date



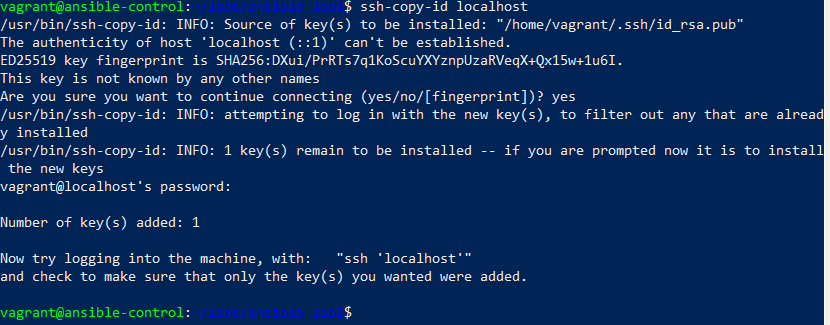
**8. Generate SSH Keys and copy to hosts.**

- We generate the keys with the following commands:

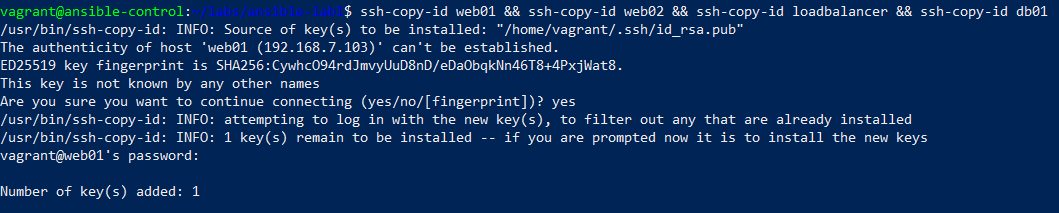
- ssh-keygen:



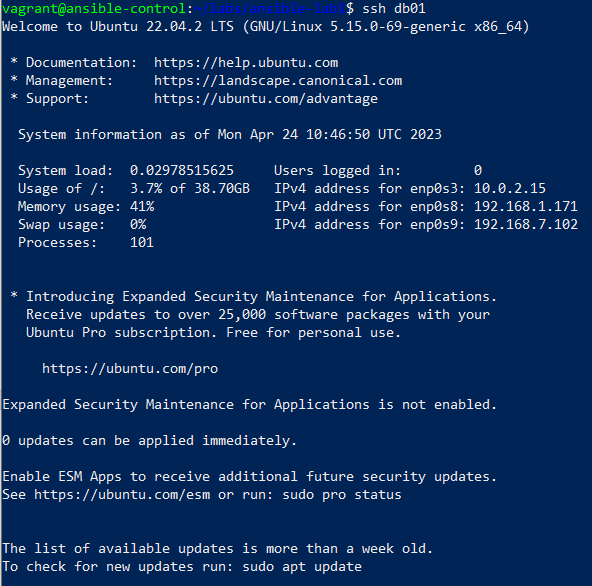
ssh-copy-id localhost:



ssh-copy-id web01 && ssh-copy-id web02 && ssh-copy-id loadbalancer && ssh-copy-id db01:

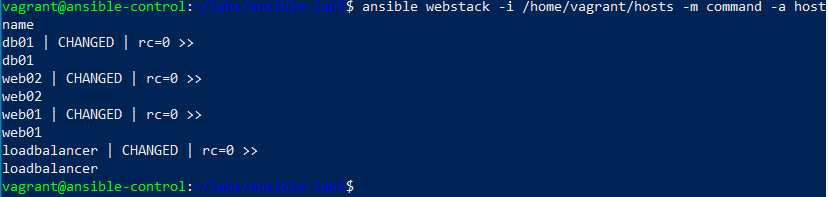


- And we can see that we are able to connect to any machine, for example db01:



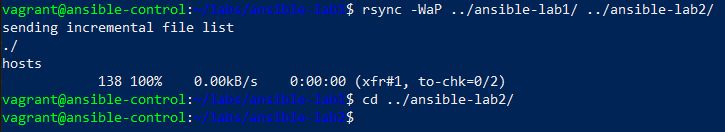
**9. Test running ad-hoc commands to all hosts.**

- Webstack are all the machines, apart from the ansible control.



**Ansible Lab 2 – Ad HOC tasks and Modules.**

- We start off by copying the content from lab1 to lab2 and changing the directory to lab2.



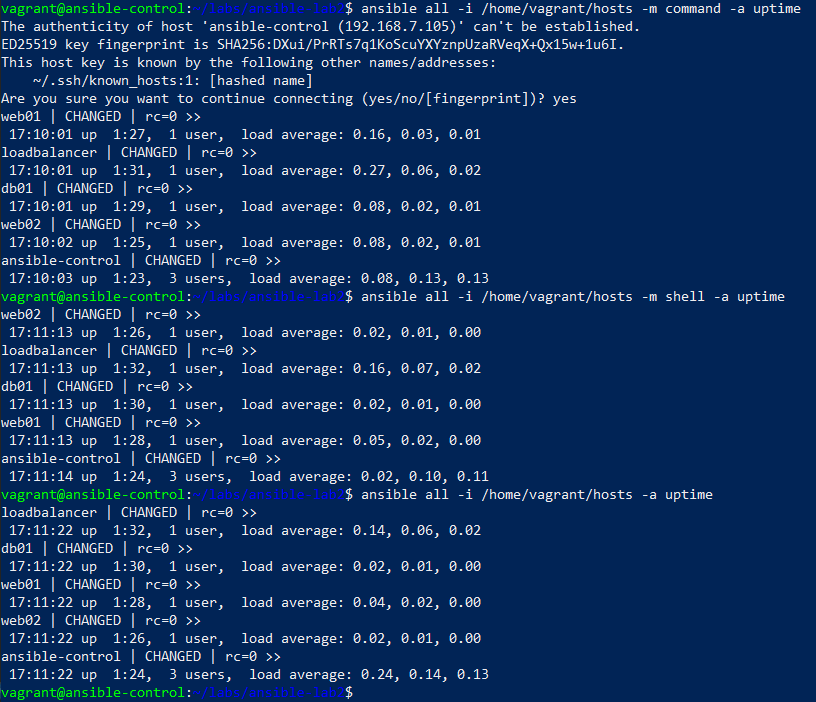
**1. Ansible ad hoc command for checking the uptime of the hosts.**

- The 3 different commands below would yield the same result:

ansible all -i /home/vagrant/hosts -m command -a uptime

ansible all -i /home/vagrant/hosts -m shell -a uptime

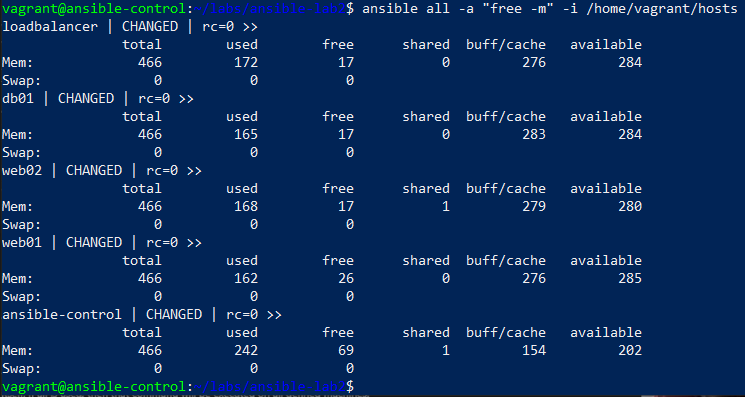
ansible all -i /home/vagrant/hosts -a uptime



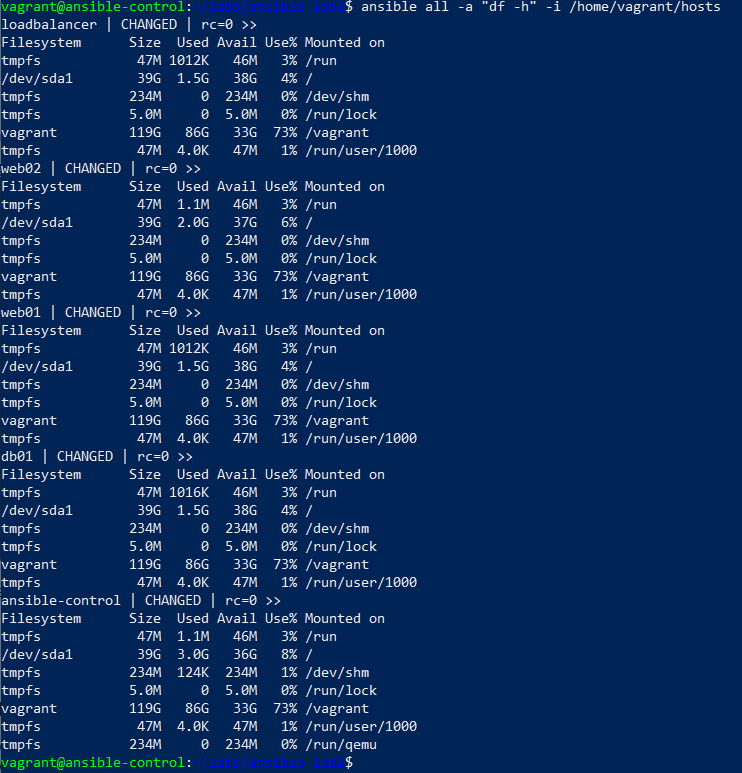
**2. Check the free memory or memory usage of hosts using ansible ad hoc command.**

- We check this with the following command:

ansible all -a "free -m" -i /home/vagrant/hosts



- And the different disk spaces of the machines with small modification of this command:

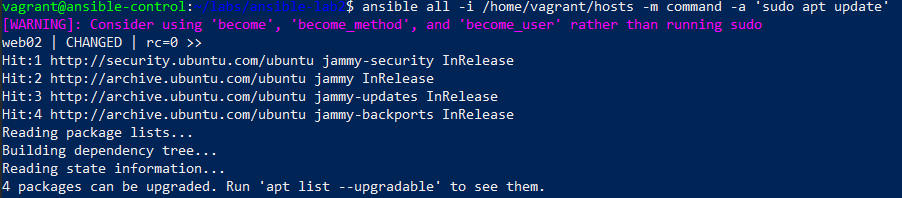


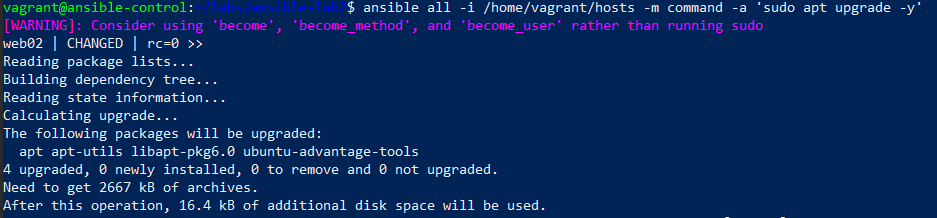
**3. Update and upgrade all machines.**

- We can do this with the following commands:

ansible all -i /home/vagrant/hosts -m command -a 'sudo apt update'

ansible all -i /home/vagrant/hosts -m command -a 'sudo apt upgrade -y'

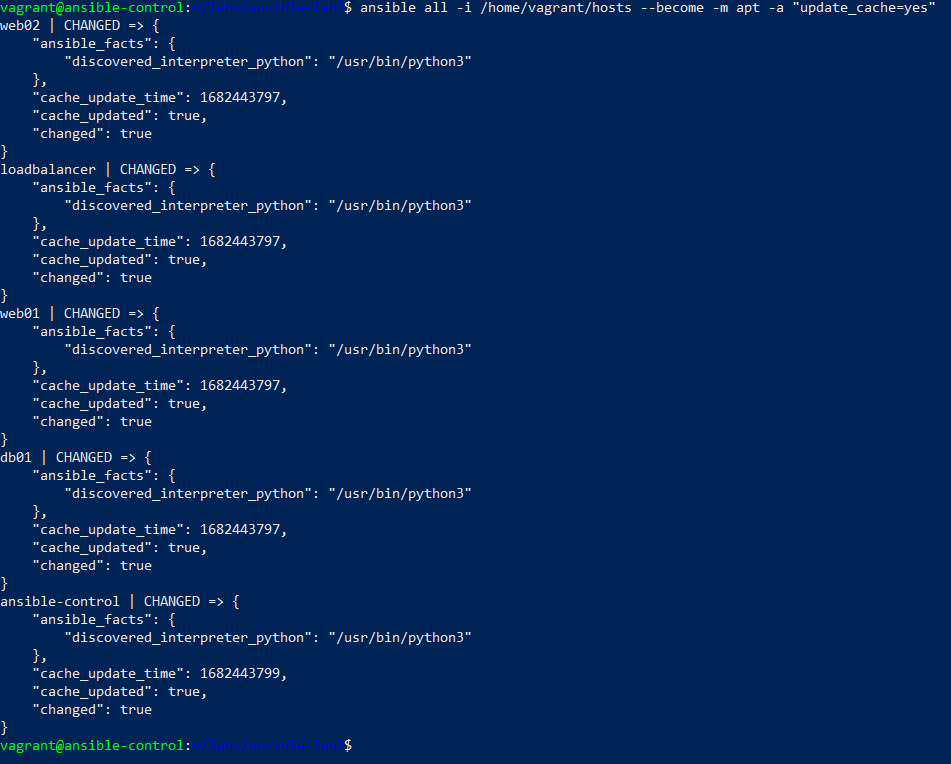


****

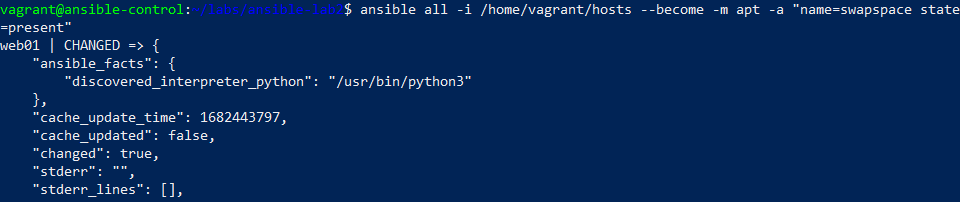
**4. Use APT module to install services.**

- We will use this module to install Apache web server in the webservers group and a MySQL server in the database group. For this, we will use the following commands below:

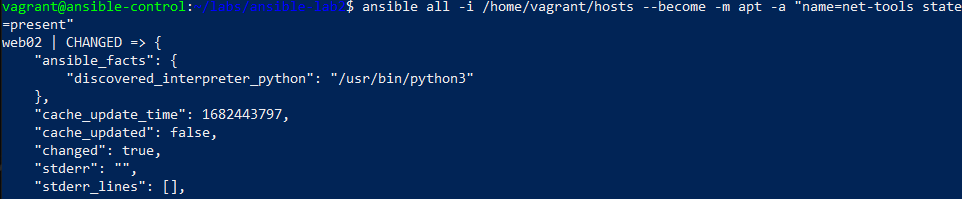
ansible all -i /home/vagrant/hosts --become -m apt -a "update\_cache=yes"



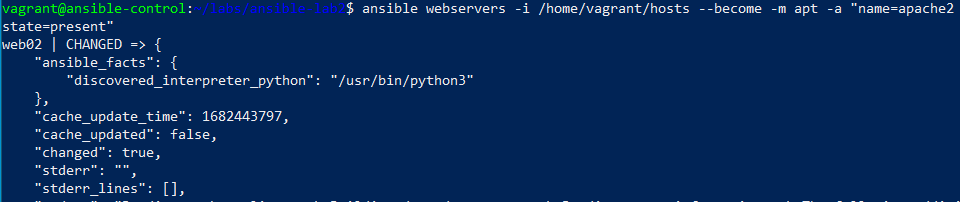
ansible all -i /home/vagrant/hosts --become -m apt -a "name=swapspace state=present"



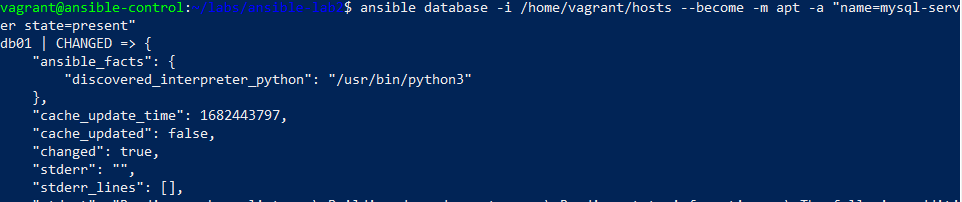
ansible all -i /home/vagrant/hosts --become -m apt -a "name=net-tools state=present"



ansible webservers -i /home/vagrant/hosts --become -m apt -a "name=apache2 state=present"



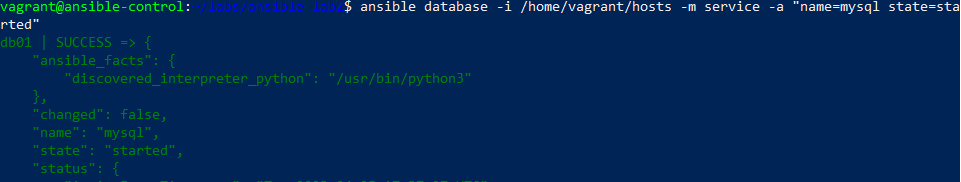
ansible database -i /home/vagrant/hosts --become -m apt -a "name=mysql-server state=present"



**5. Use service module to manage services.**

- The service module is useful for checking the status, or restarting all services. In the next two examples, we will learn how to check if your MySQL server is active and running and then restart it with the commands below:

ansible database -i /home/vagrant/hosts -m service -a "name=mysql state=started"



- and to restart it (not going to right now):

ansible database --become -i /home/vagrant/hosts -m service -a "name=mysql state=restarted"

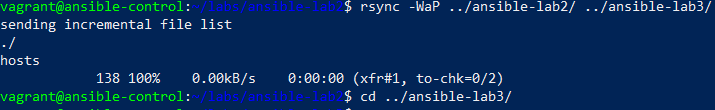
**6. Use Ansible to reboot webstack.**

- We can use the following command to restart the virtual machines from the webstack group, but we are not going to right now so we can finish the rest of the tasks.

ansible webstack -i /home/vagrant/hosts --become -a "reboot --reboot"

**Ansible Lab 3 – Playbooks, Templates and Handlers.**

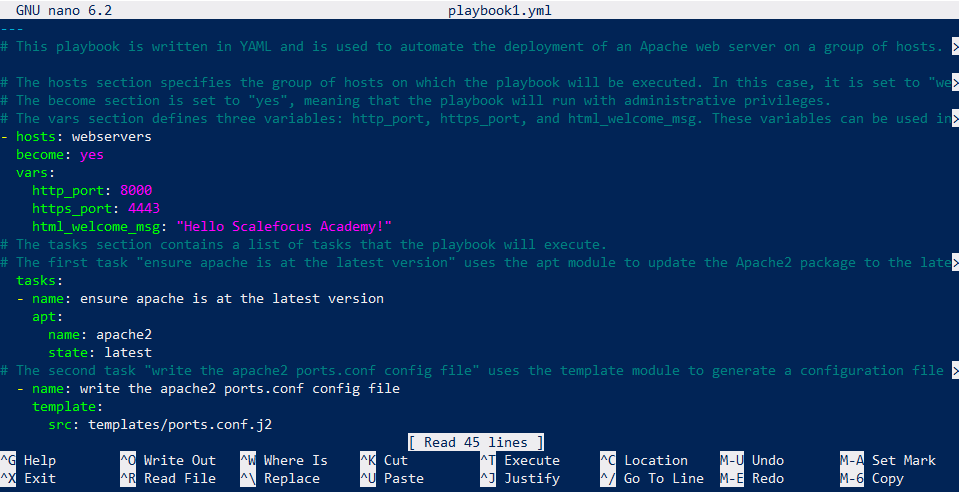
- First, we sync the current ansible-lab state to the newly created lab structure, and move to the third lab directory with the following command:



**1. Exam the playbook and look over the details of the YAML file.**

- Creating our playbook with the nano command and adding the required code:





**2. Check the Templates.**

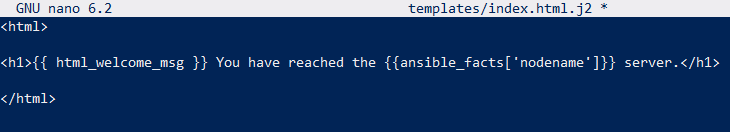
- Two Jinja templates have been added to the playbook1.yml file, which we should create and look at before running this playbook. For this, we use the following commands:

mkdir /vagrant/ansible-lab3/templates

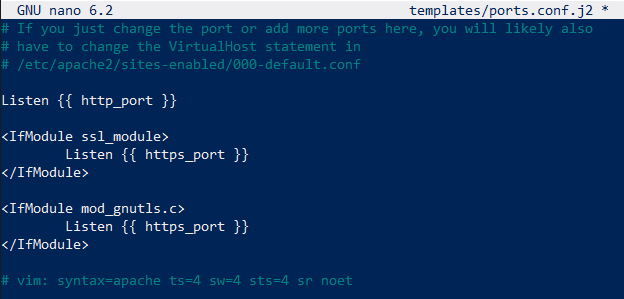
touch /vagrant/ansible-lab3/templates/index.html.j2

touch /vagrant/ansible-lab3/templates/ports.conf.j2

- In the index.html.j2 file, we add the following HTML content:



- In the ports.conf.j2 file, we add the following HTML content:

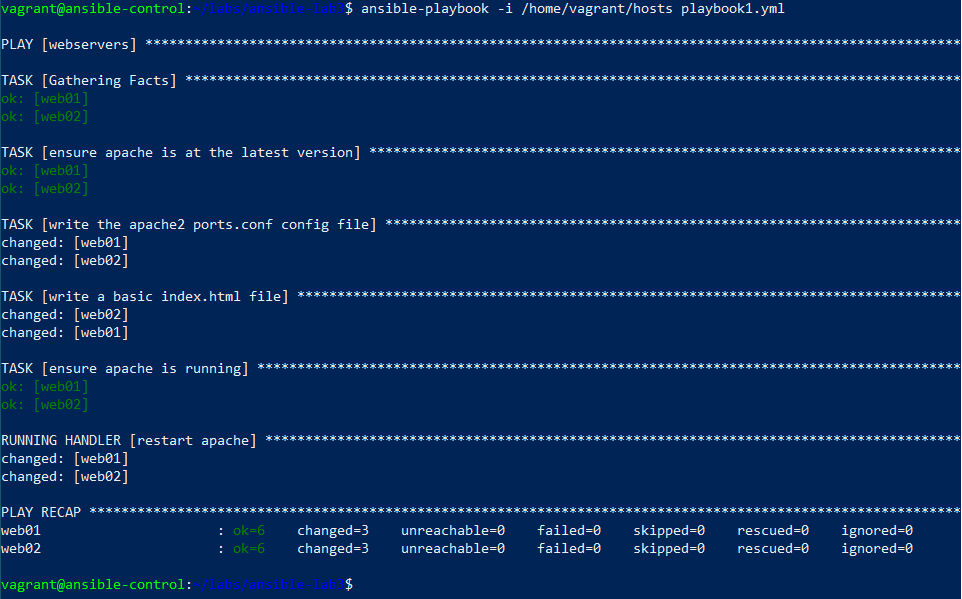


- The index.html.j2 is a simple HTML file with a message defined in the playbook.   
- The ports.conf.j2 template is intended as a configuration file for the the Apache servers. It defines the HTTP and HTTPS ports of this service.

**3. Run the Playbook.**

- We can now run it with the following command:

ansible-playbook -i /home/vagrant/hosts playbook1.yml

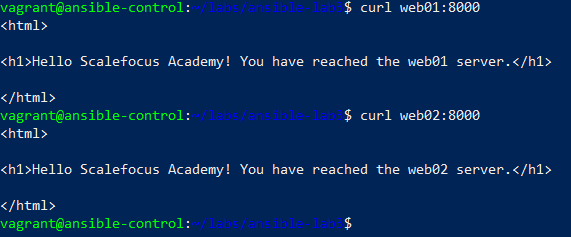


**4. Test connectivity to servers.**

- Here we can use the curl command to check if the newly installed Apache servers are available and responsive with the templates above. We can use the following commands:

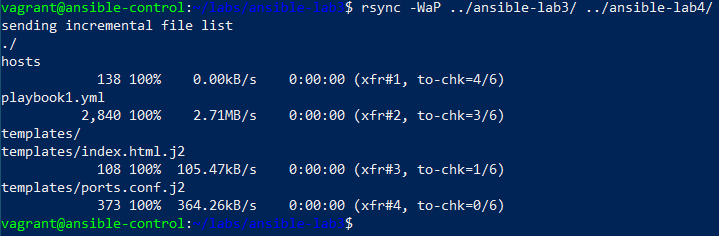
curl web01:8000

curl web02:8000



**Ansible Lab 4 – Re-usable playbooks, import\_tasks, roles and Ansible Galaxy.**

- First, we sync the current ansible-lab state to the newly created lab structure, and move to the fourth lab directory with the following command:



And we are going to use the playbook in order to enhance it and make it look leaner.

**1. Re-usable playbooks, import\_tasks.**

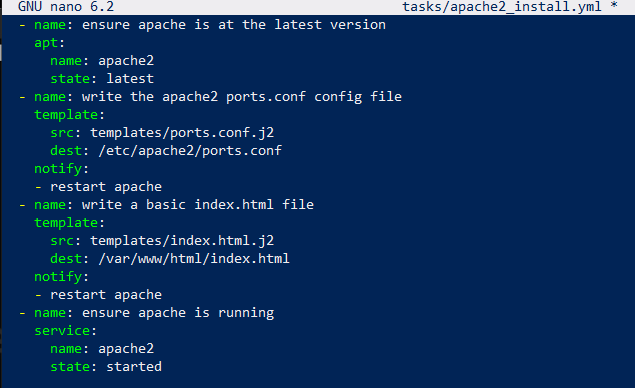
- We start off by creating a new tasks directory and a new yml file. Then edit the playbook1.yml and move the tasks code to the newly created file.

mkdir tasks ; touch tasks/apache2\_install.yml

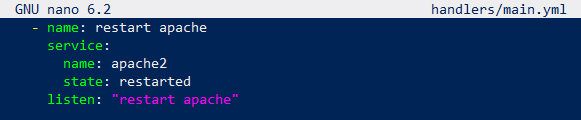
and handlers  
mkdir handlers ; touch handlers/main.yml



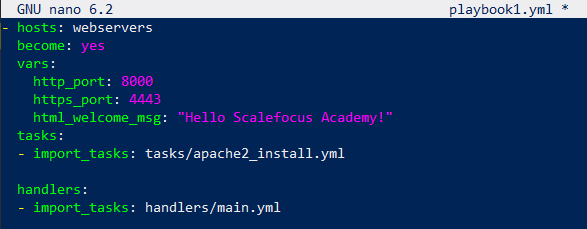
- Edit the apache2\_install.yml file:



- Edit the handlers/main.yml file:

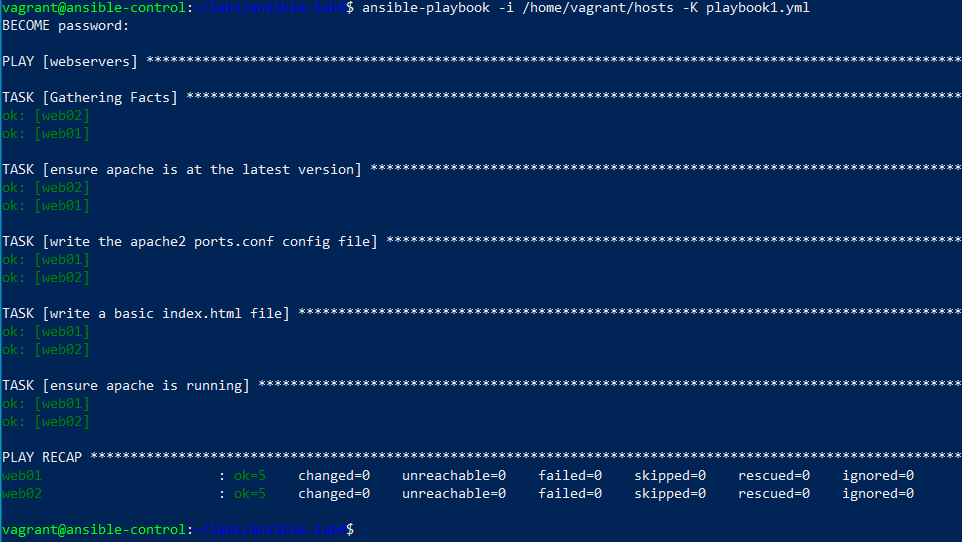


- Edit the playbook1.yml file:



Finally, we run the playbook in order to test the changes with the following command:

ansible-playbook -i /home/vagrant/hosts -K playbook1.yml



**2. Ansible Roles and Ansible Galaxy.**

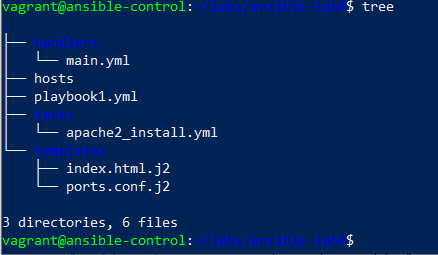
- Ansible Roles are a way to organize your playbooks and tasks into reusable units. A role is essentially a collection of related tasks, files, templates, and variables that can be easily shared and used across multiple playbooks.

Roles are useful for several reasons:

1. Reusability: You can use roles in multiple playbooks and projects, which saves time and effort.
2. Modularity: Roles make it easy to break down complex playbooks into smaller, more manageable units.
3. Organization: Roles provide a structured way to organize your playbooks and tasks, which makes it easier to understand and maintain them.
4. Collaboration: Roles can be shared with other teams or the community, which fosters collaboration and sharing of best practices.

[Ansible-galaxy](https://galaxy.ansible.com/) is a repository of reusable Ansible roles and collections that are contributed by the community. It provides a central location where Ansible users can search for, download, and use existing roles, rather than starting from scratch.

This is our current tree (has to be installed first with sudo apt install tree)



1. We start with creating the apache2 webserver role scaffolding with ansible-galaxy:

ansible-galaxy init roles/apache2



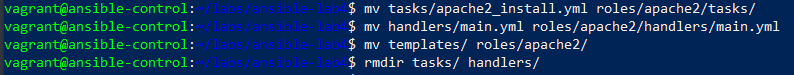
2. We move the tasks to the roles/webserver folder with the following commands:

mv tasks/apache2\_install.yml roles/apache2/tasks/

mv handlers/main.yml roles/apache2/handlers/main.yml

mv templates/ roles/apache2/

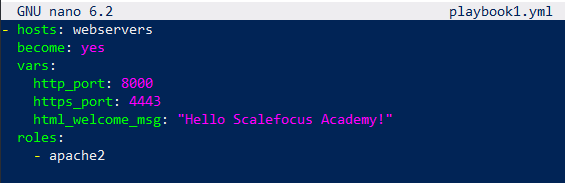
rmdir tasks/ handlers/



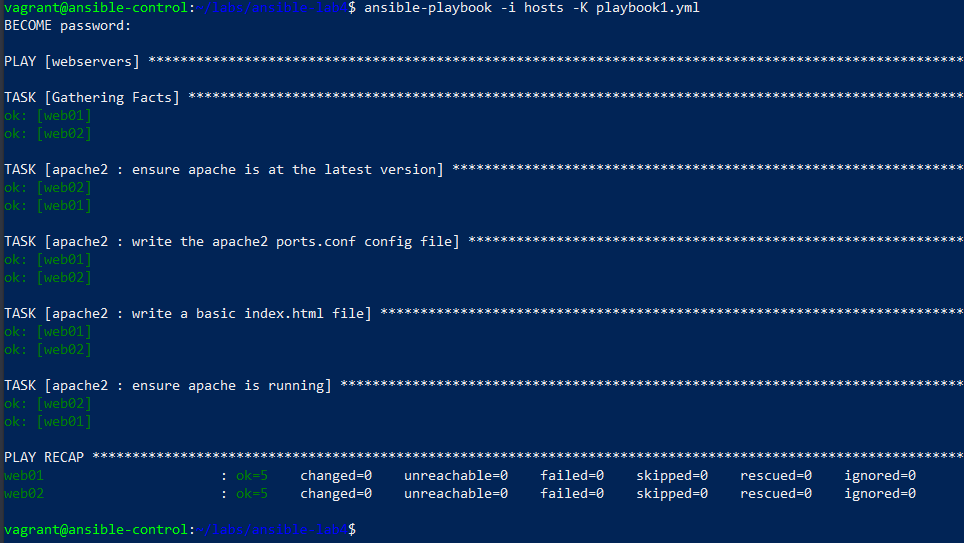
- Edit the roles/apache2/tasks/main.yml and we add the following:



- Edit the playbook1.yml and we add the following:



3. We can run the playbook again:



4. We proceed with creating 2 new roles with the following commands:

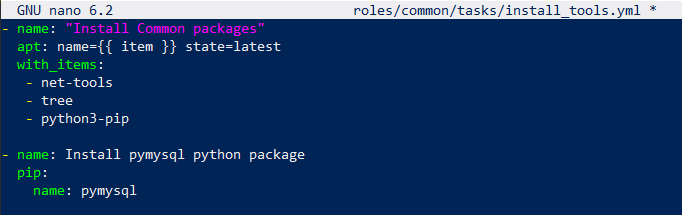
ansible-galaxy init roles/common

ansible-galaxy init roles/nginx



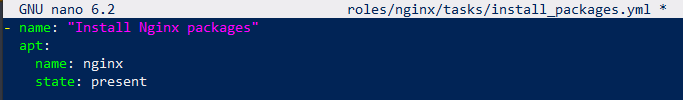
5. Setup 'common' role tasks/main.yml and tasks/install\_tools.yml by entering the following code:



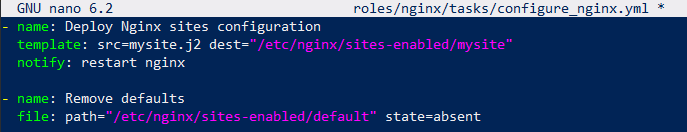


6. Setup the nginx role through code again.

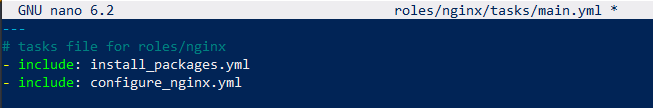
- A tasks file:



- Another task to configure the Nginx:



- The new tasks that we just created into the main.yml:



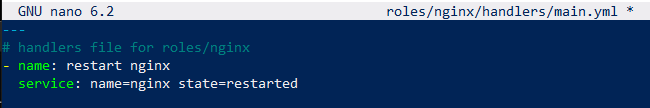
We proceed with configuring the template:

mkdir roles/nginx/templates

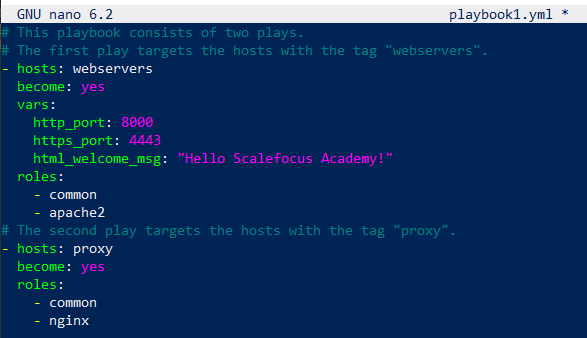




And the handler:



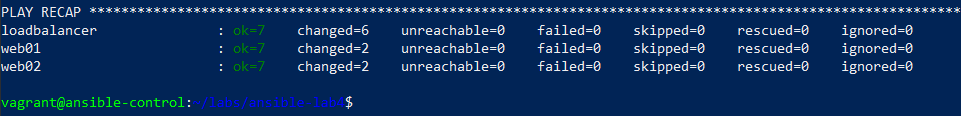
Finally, we update the playbook1.yml file:



7. Run the playbook.



And we can see the final result at the end:



**Ansible Lab 5 – Variables, Ansible Vault.**

- First, we sync the current ansible-lab state to the newly created lab structure, and move to the fifth lab directory with the following command:



In this lab, we can add the MySQL role and get a database up and running, with the appropriate configuration.

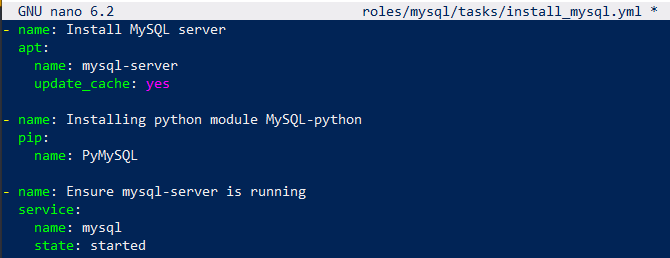
1. We start by creating a mysql role using ansible-galaxy:

ansible-galaxy init roles/mysql

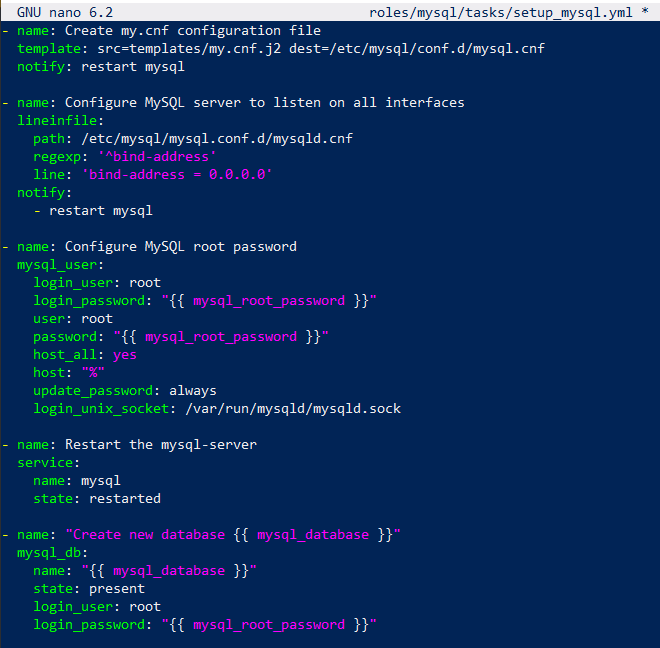


2. After that, we create tasks, handlers and templates for the new mysql role.

- The installation task:



- The configuration task:



- Import them in the main.yml file:



Then we proceed with the templates config, for which we create a new directory using:

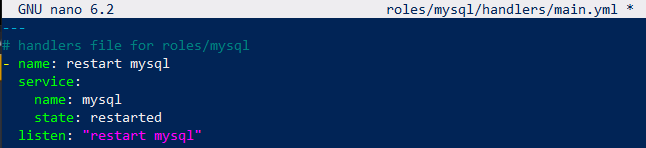
mkdir roles/mysql/templates



And edit the file:



Finally, we create the restart handler:

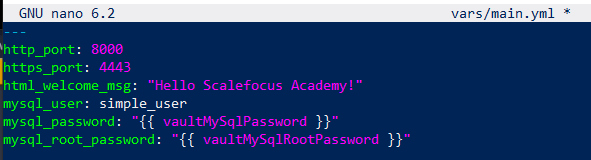


3. Set our variables and encrypt the important data.

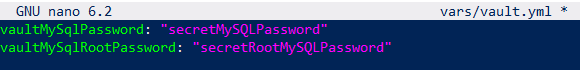
- In this step, we will move the variables that were declared in the playlist itself to a separate directory and encrypt the MySQL password with the Ansible vault.

Creating the vars directory:  


And place our variables in a main.yml file:

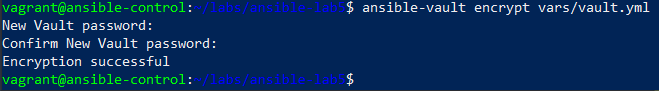


The mysql\_password is declared as a variable. That's because we have the plan to protect and encrypt it with Ansible Vault. For that, we will create another file:

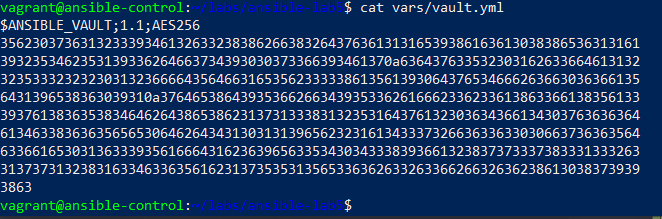


And encrypt it by using ansible-vault command:

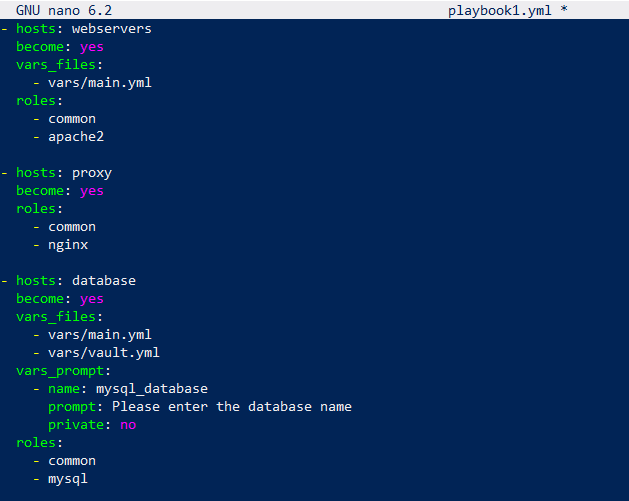
ansible-vault encrypt vars/vault.yml



When trying to access the file, we can see that it’s encrypted:

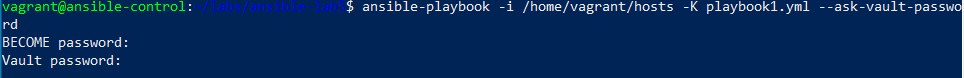


4. Modify the playbook1.yml and add our new play.



5. Run the playbook.

ansible-playbook -i /home/vagrant/hosts -K playbook1.yml --ask-vault-password



And the final result:

