**Ansible Hands on Guide**

Contents

[Pre-requisite 2](#_Toc22233344)

[Install Vagrant 2](#_Toc22233345)

[Install VirtualBox 2](#_Toc22233346)

[Disable the Hyper-V role 2](#_Toc22233347)

[Downloading the Lab files 3](#_Toc22233348)

[Vagrant setup explained 3](#_Toc22233349)

[Vagrantfile – Explained 3](#_Toc22233350)

[Managing the Lab VMs 4](#_Toc22233351)

[Creating the VMs 4](#_Toc22233352)

[Validating the VM status 5](#_Toc22233353)

[Powering on/off VMs 5](#_Toc22233354)

[SSH to VMs 6](#_Toc22233355)

[Deleting the VMs 6](#_Toc22233356)

[Accessing the Hands on code 7](#_Toc22233357)

[Lab 01 – Installing Ansible on Control Node 7](#_Toc22233358)

[Configure SSH key for remote access 9](#_Toc22233359)

[Lab 02 - Creating Inventory 10](#_Toc22233360)

[Lab-03 – Using common modules 12](#_Toc22233361)

[04-Running Playbooks 14](#_Toc22233362)

[Playbook syntax-check 14](#_Toc22233363)

[Running the playbook 14](#_Toc22233364)

[Install nginx on webservers and jdk on app server 15](#_Toc22233365)

[Lab-05 – Running Playbook with roles 16](#_Toc22233366)

[File structure of the role based playbook 17](#_Toc22233367)

[Tasks on Roles - explained : 17](#_Toc22233368)

[Running the playbook with roles: 18](#_Toc22233369)

# Pre-requisite

## Install Vagrant

Download the vagrant installer for the respective OS and validate the vagrant version using command

***Vagrant --version***

<https://www.vagrantup.com/downloads.html>



## Install VirtualBox

Install the virtual box from below location as we will be using VirtualBox based VM to provision VMs.

<https://www.virtualbox.org/wiki/Downloads>

Note : Vagrant comes with support out of the box for VirtualBox, a free, cross-platform consumer virtualization product.

## Disable the Hyper-V role

If you wish to use VirtualBox on Windows, you must ensure that Hyper-V is not enabled on Windows. You can turn off the feature by running this PowerShell command:

***Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-All***

**OR**

You can also disable it by going through the Windows system settings:

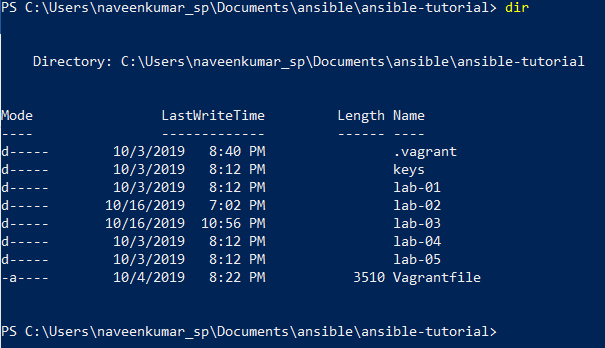
* Right click on the Windows button and select ‘Apps and Features’.
* Select Turn Windows Features on or off.
* Unselect Hyper-V and click OK.

Note: You might have to reboot your machine for the changes to take effect. More information about Hyper-V can be read here.

# Downloading the Lab files

Download the exercise code from GitHub repository

# Vagrant setup explained



The labs are aligned with the concepts which were covered in the training presentation.

## Vagrantfile – Explained

The Vagrantfile is a Ruby file used to configure Vagrant on a per-project basis. The main function of the Vagrantfile is to described the virtual machines required for a project as well as how to configure and provision these machines.

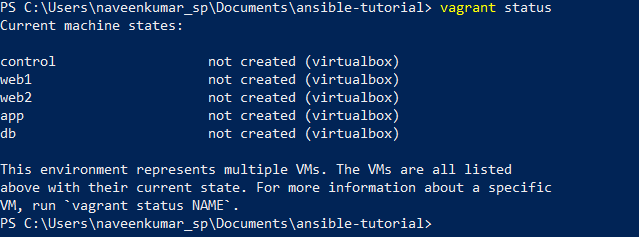
For more information , please refer the link <https://www.vagrantup.com/docs/vagrantfile/>

Vagrant Boxes are the package format for Vagrant environments. A box can be used by anyone on any platform that Vagrant supports to bring up an identical working environment.

In our Vagrantfile of the lab environment, we have defined to create following VMs from Vagrant box named “bento/Ubuntu-16.04”. The file will be created following VMs for our Ansible hands-on setup

* Control – Control VM for running Ansible commands with 1Gig memory and IP address of 192.168.35.1
* Web1 and Web2 – Webserver with 512Mb of memory and IP address of 192.168.35.101 and 192.168.35.102
* App - Middle tier server for managing application, configured with 512MB of memory and IP address is 192.168.35.103
* DB – Database servers for storing rational data, configured with 512MB memory and IP address is 192.168.35.104

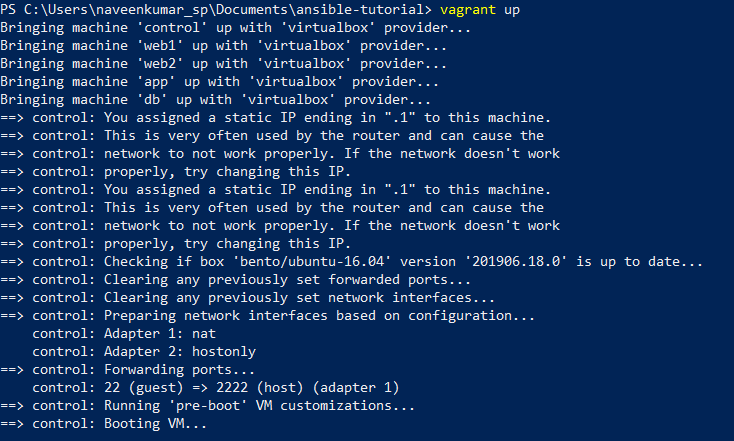
Validate the syntax of the Vagrantfile by running the vagrant status command on the directory where we have the Vagrantfile.



# Managing the Lab VMs

## Creating the VMs

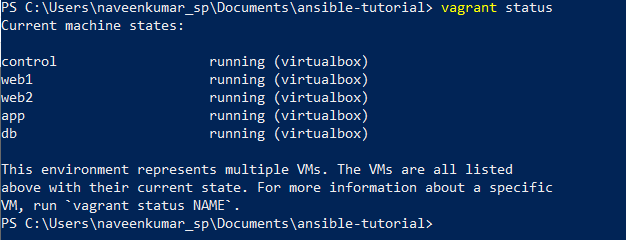
Using ***vagrant up*** command, we can power on all VMs which are part of the Lab exercise. You will be using the same command if incase the VMs were powered off.



**Note: It takes about 10-15 minutes to setup all four VMs.**

## Validating the VM status

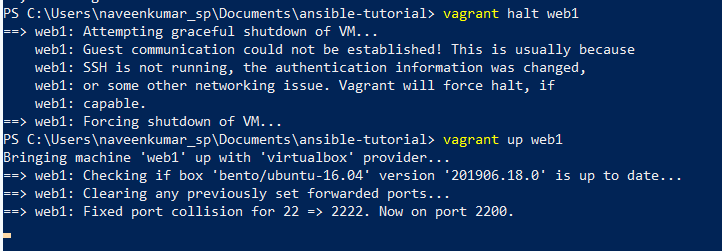
Issue the command ***vagrant status*** to validate the VMs which are part of the lab exercise.



## Powering on/off VMs

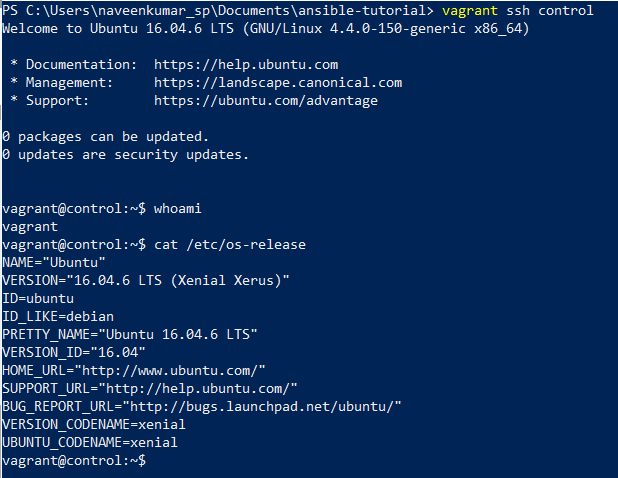
You may power on/off individual VMs by issuing the command **vagrant up/halt <vm\_name>**

If you have not mentioned any VM name, action will be applied on all VMs which are part of the Vagrant files. This helps to stop and resume the lab environment at your convenience.



## SSH to VMs

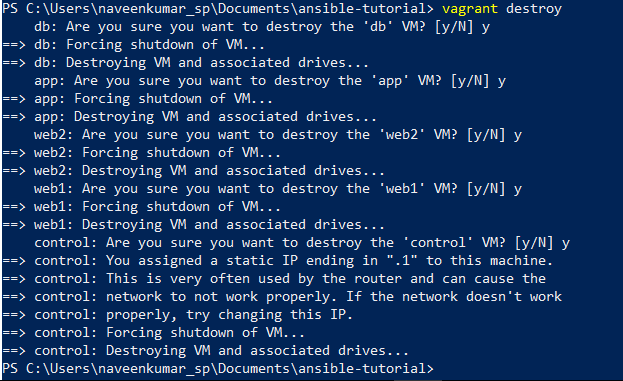
To SSH to VMs, issue the command ***vagrant ssh <vm\_name>***



## Deleting the VMs

Issue ***vagrant destroy*** command to delete the VMs. You may mention the specific VM to delete a specific VM. You may use **–f** parameter to delete all without any confirmation.

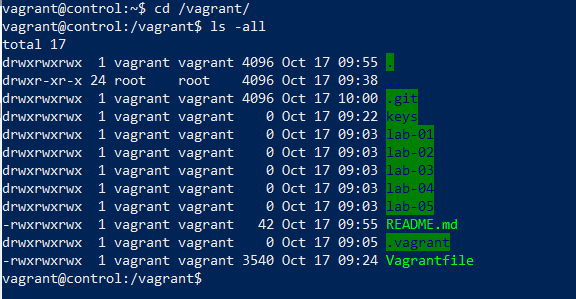
i.e ***vagrant destroy -f***



# Accessing the Hands on code

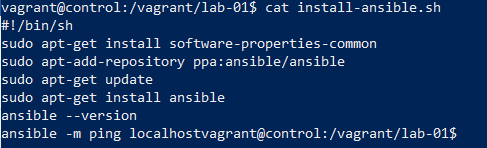
Vagrant loads the code directory i.e our repo data into /vagrant on each VM which was created. We will be using the lab scripts to execute the Ansible commands or playbooks.

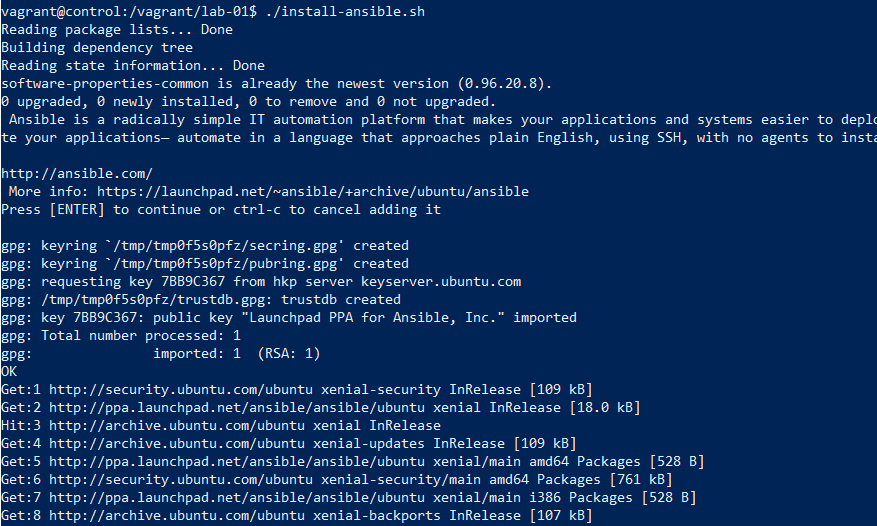
Also we have used the common key for all the VMs which were created. Hence we can use



# Lab 01 – Installing Ansible on Control Node

We will be installing Ansible on control node from which we will be executing all Ansible commands and playbooks. To install the Ansible and associated dependencies, change directory on control node VM to **/vagrant/lab-01** and execute the shell script named ‘**install-ansible.sh**’



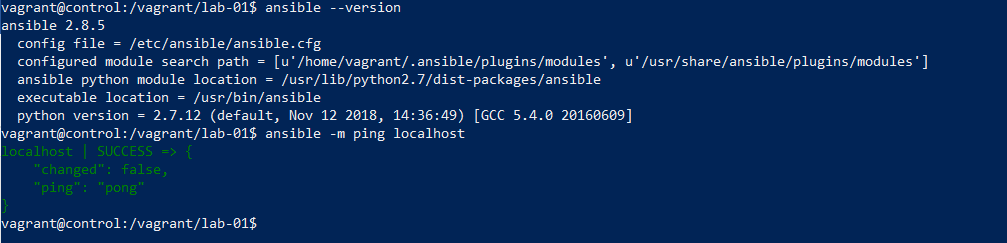


At the end of installation, you will be able to validate the version with the command

***ansible ---version***

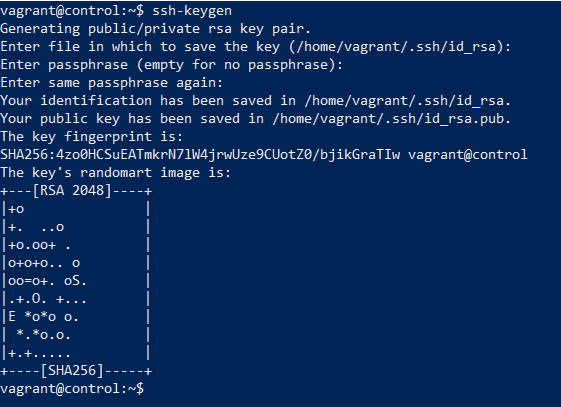
Also you may test the simple Ansible command to ping the local host by issuing below command

***ansible –m ping localhost***



# Configure SSH key for remote access

1. Create a SSH key using ***ssh-keygen*** on control node

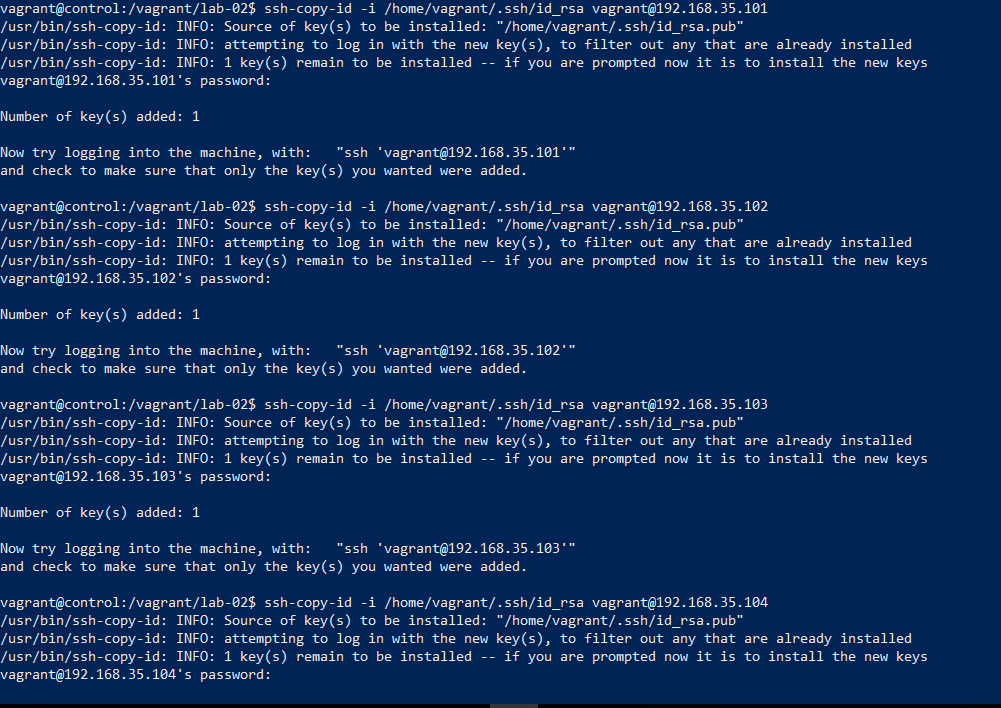


2. Copy the generated key into all other hosts for easier access by running the below command.

The password for ‘***vagrant’*** user on all machines is ***vagrant***

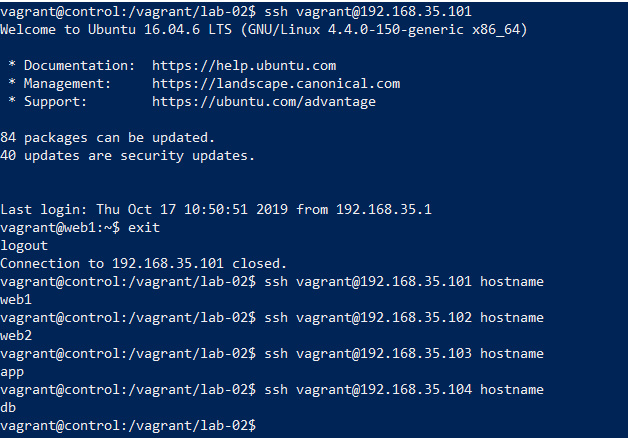
* ssh-copy-id -i /home/vagrant/.ssh/id\_rsa vagrant@192.168.35.101
* ssh-copy-id -i /home/vagrant/.ssh/id\_rsa vagrant@192.168.35.102
* ssh-copy-id -i /home/vagrant/.ssh/id\_rsa vagrant@192.168.35.103
* ssh-copy-id -i /home/vagrant/.ssh/id\_rsa vagrant@192.168.35.104

Note: In production, we will be using Ansible tower where encrypted password of user ID of the remote machine account is stored for remote. However, for our convenience we are using ssh key for remote access.



Validate the SSH access to all four machines by running below command

* ssh vagrant@192.168.35.101 hostname
* ssh vagrant@192.168.35.102 hostname
* ssh vagrant@192.168.35.103 hostname
* ssh vagrant@192.168.35.104 hostname

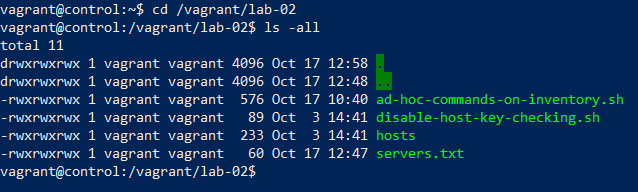


# Lab 02 - Creating Inventory

For running ad-hoc commands as well as to run playbooks, we will be creating an inventory file and update/replace the file /etc/Ansible/host in control node.

1. On control node, change directory to lab-02 directory running below command

***cd /vagrant/lab-02***



2. Replace the Ansible host file with /vagrant/lab-02 by running below command

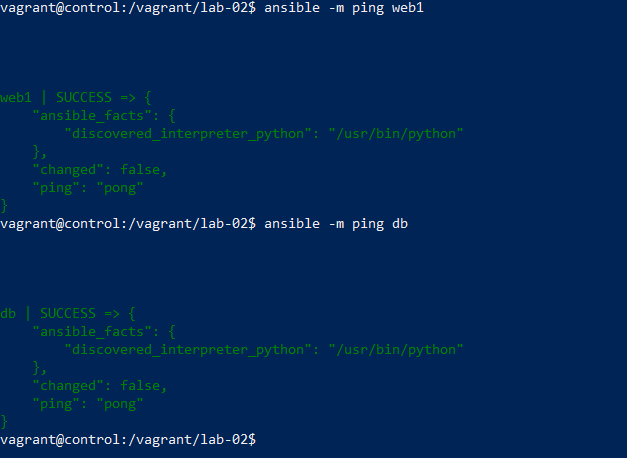
***sudo cp -v /vagrant/lab-02/host /etc/ansible/host***

*Note: Vagrant user on all VMs provisioned will have sudo access to perform administrative activity.*



3. Run below ad-hoc commands and validate the result.

* ansible -u vagrant -m ping web1
* ansible -u vagrant -m ping app
* ansible -u vagrant -m ping webservers
* ansible -u vagrant -m ping dc
* ansible -u vagrant -m shell -a 'ls -al' web1
* ansible -u vagrant -m shell -a 'whoami' app
* ansible -u vagrant -m shell -a 'ifconfig' webservers
* ansible -u vagrant -m shell -a 'hostname' dc
* ansible -u vagrant -m ping all
* ansible -u vagrant -m ping web\*
* ansible -u vagrant -m ping 'appservers:dbservers'
* ansible -u vagrant -m ping 'dc:!webservers'
* ansible -u vagrant -m ping 'dc:&webservers'

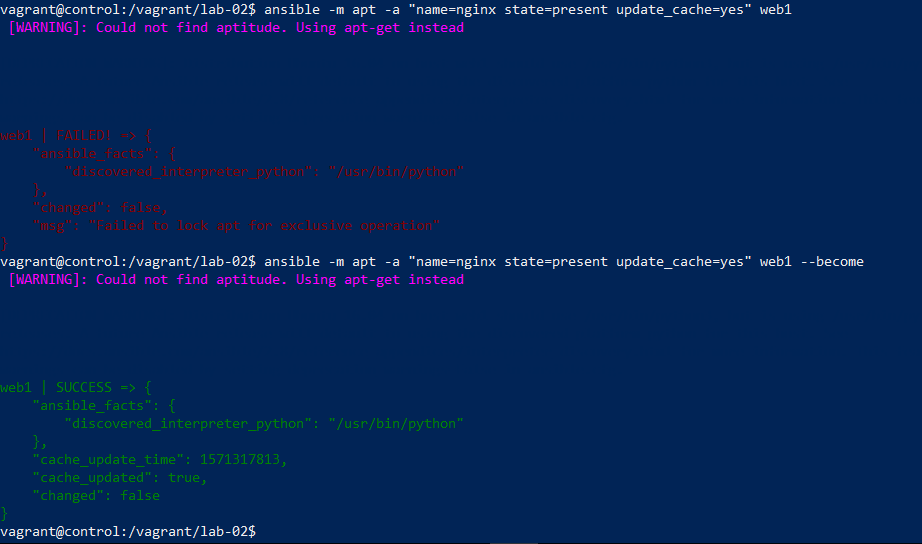


# Lab-03 – Using common modules

1. To install the nginx on webserver we can use the –m apt

***ansible -m apt -a "name=nginx state=present update\_cache=yes" web1***

as you may observe in below screenshot, nginx installation failed due to access privileges. Hence by using the parameter --***become*** helps to run the ansible module on remote server with sudo privileges.



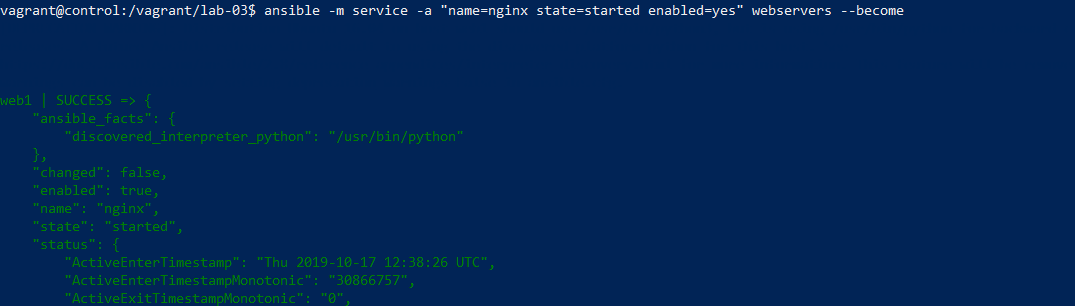
2. To un-install the application, we need to mark the application state as absent.

***ansible -m apt -a "name=nginx state=absent" web1 --become***

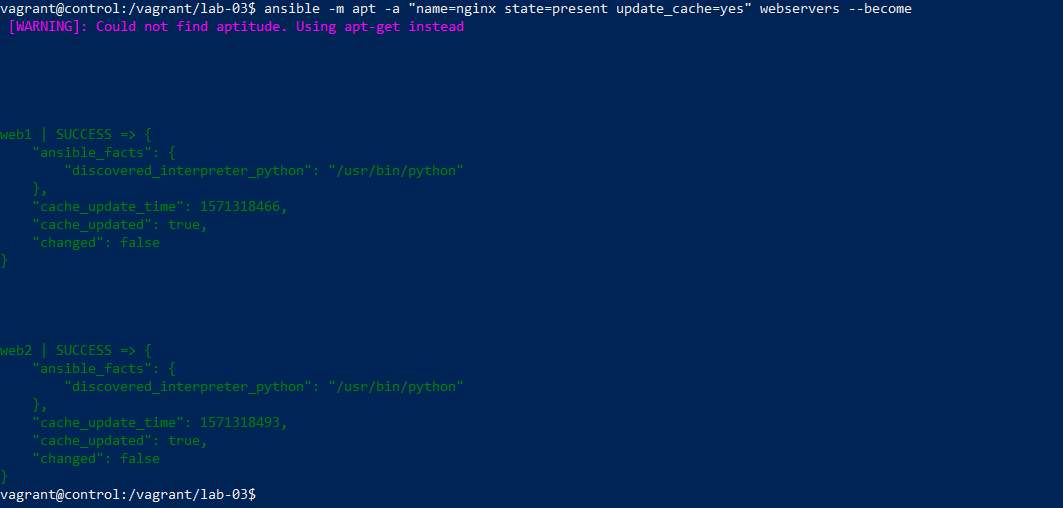


3. Issue the below command to execute Ansible module to start the service on remote node.

***ansible -m service -a "name=nginx state=started enabled=yes" webservers --become***

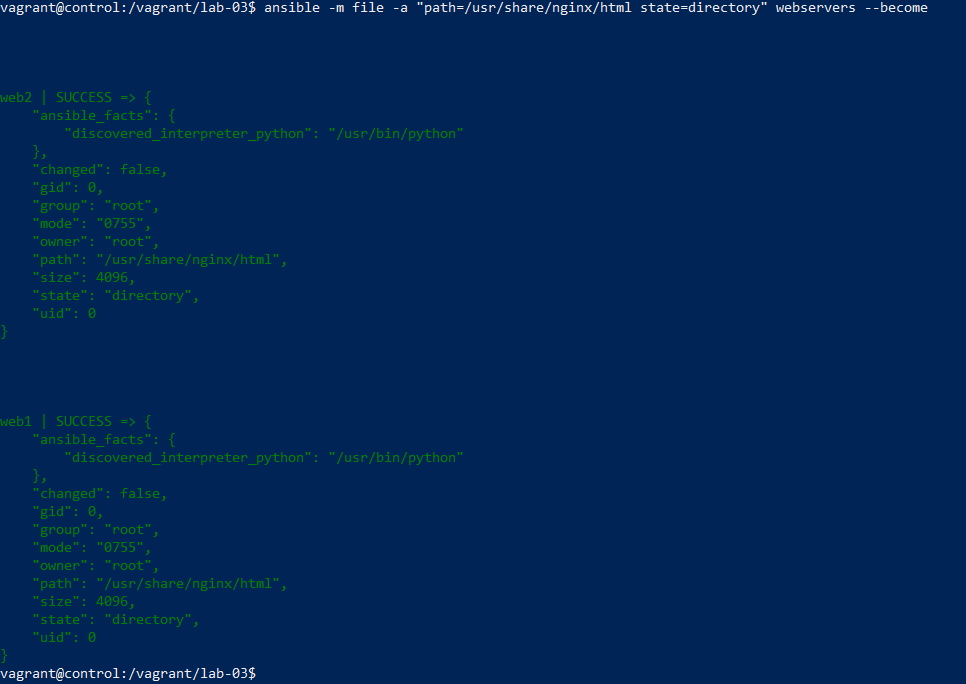


4. We can run the ansible module on groups as well. Here is the example of how ansible module be run to install nginx on webserver (web1 and web2)



5. Ansible provides various modules to perform configuration activity on remote server.

Here is the example where index.html file from control node is been copied to /usr/share/nginx/html directory.



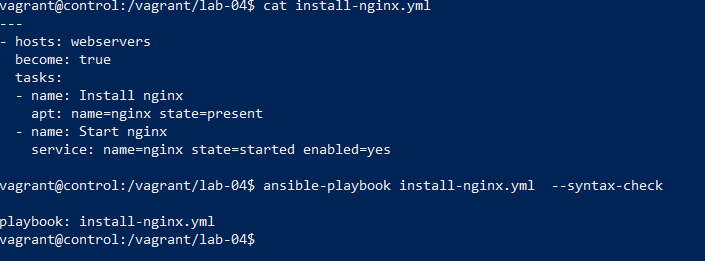
# 04-Running Playbooks

Ansible playbook is the easier way to run the Ansible modules on remote machines. These are files which can be developed and tested on local environments before running on production environments.

Here is the example of how we can install nginx on webservers.

## Playbook syntax-check

Ansible play throws errors if there any whitespace. In playbook space matters a lot in YAML, as it strictly uses space character as indent. To validate the written playbook, you may use the –syntax-check parameter. If there is no error, then the playbook name will be displayed as output.



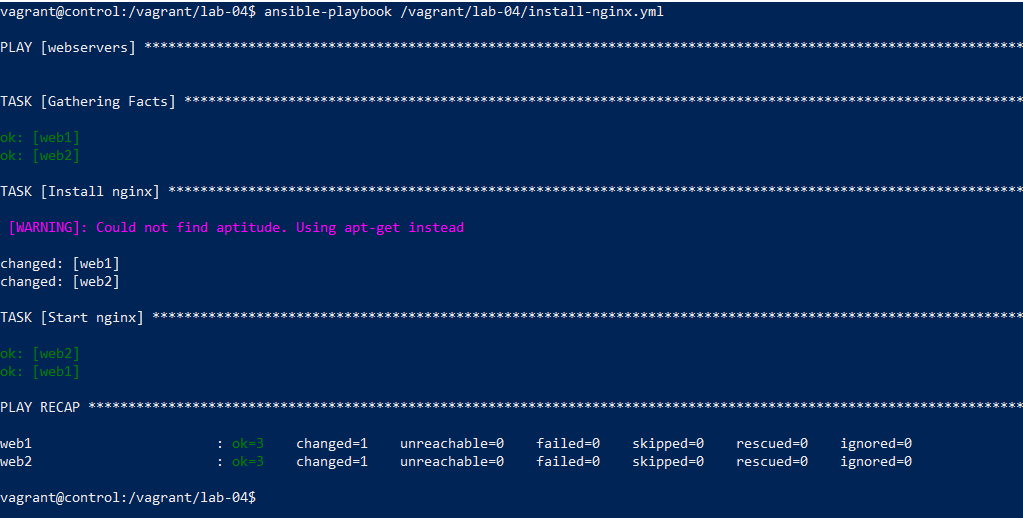
## Running the playbook

Before running the playbook, un-install the nginx on webservers using below command

***ansible -m apt -a "name=nginx state=absent update\_cache=yes" webservers –become***

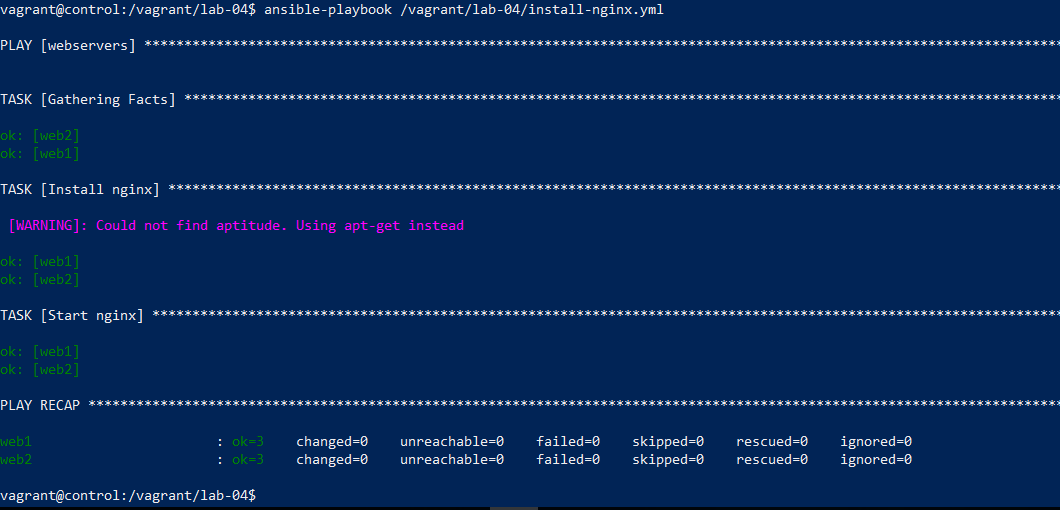
run the playbook using below command to install nginx on webservers

***ansible-playbook /vagrant/lab-04/install-nginx.yml***

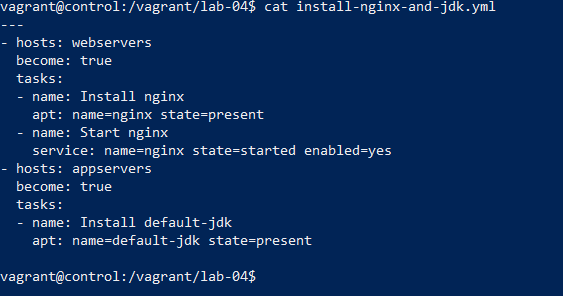


You may observe the task Install nginx indicate as changed. Which implies that ansible has installed the nginx on both servers.

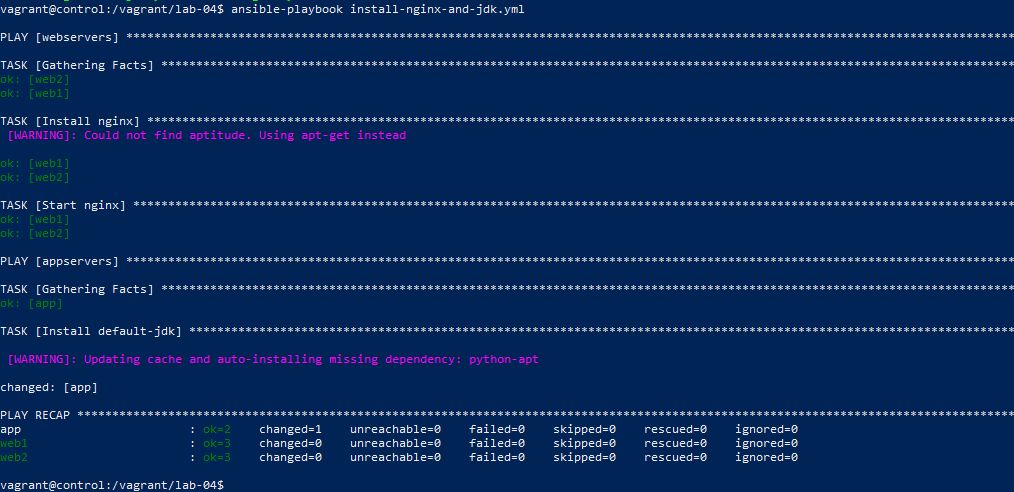
Since Ansible is idempotent, if we run the playbook once again, ansible will skip the installation and proceed with rest of the playbook step. This feature of any configuration management tool helps to ensure there will not be any issues especially when replacing the configuration or files of remote system.



## Install nginx on webservers and jdk on app server



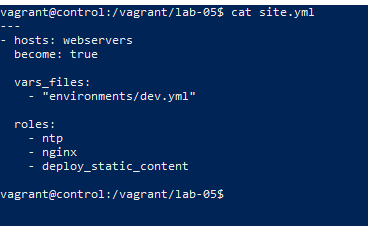
As you can see in below screenshot, nginx installation was not performed as it was already present on webserver. However, JDK was installed on app server. The play recap will summaries the status of changes which was performed by the playbook.



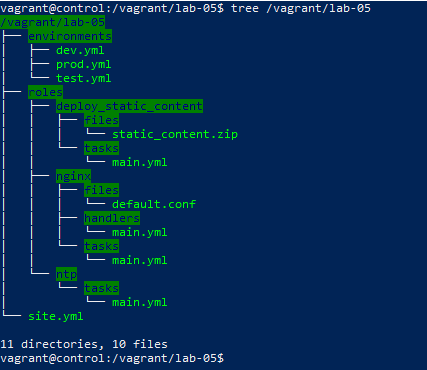
## Lab-05 – Running Playbook with roles

We a playbook with roles which performed the below activity.

* ntp: installs the ntp on webservers.
* nginx – install nginx on webservers and updates the default configuration
* deploy\_static\_content – copies the zip files containing static html and unzips on webservers under /opt/html directory

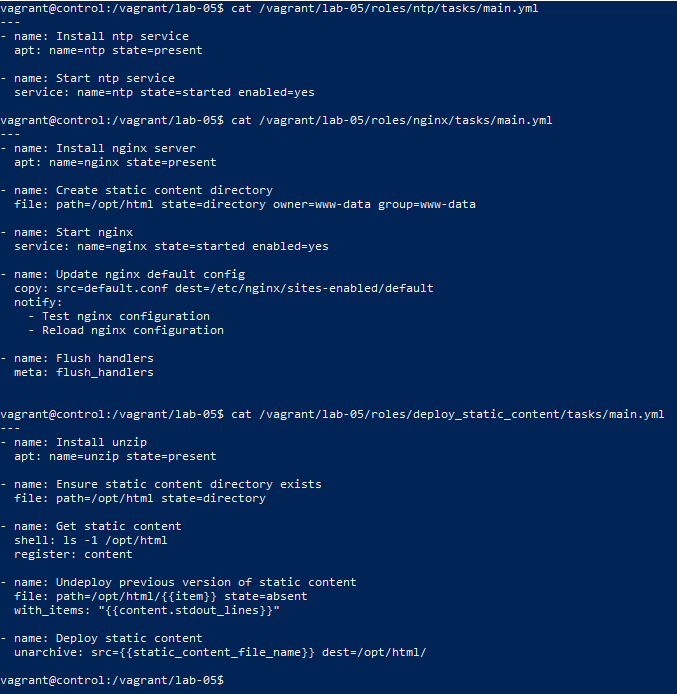
.

File structure of the role based playbook



## Tasks on Roles - explained :

Each role will have tasks which details what is shall it performed on mentioned host. Here are the three roles which were used in our playbook.



## Running the playbook with roles:

Running playbook with roles is as similar as any regular playbook. However, we need to ensure the file structure is been followed so that Ansible can look up the roles and performed tasks as defined in each role. Here is the screenshots of playbook run for your reference.

