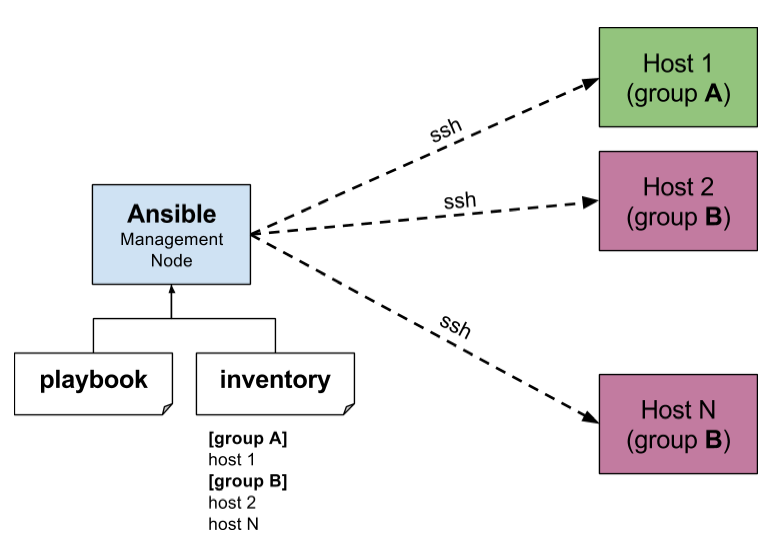
Ansible categorize into two types of server: controlling machines and nodes.

The controlling machine, where Ansible is installed and Nodes are managed by this controlling machine over SSH. The location of nodes is specified by controlling machine through its inventory.

The controlling machine (Ansible) deploys modules to nodes using SSH protocol and these modules are stored temporarily on remote nodes and communicate with the Ansible machine through a JSON connection over the standard output.



Ansible is agent-less, that means no need of any agent installation on remote nodes

Playbooks are bunch of commands which can perform multiple tasks and each playbooks are in YAML file format.

**Install Ansible in Linux OR CentOS7**

1. Unfortunately, there are no official Ansible repository for RedHat based clones, but we can install Ansible by enabling epel repository under RHEL/CentOS 6, 7 and currently supported fedora distributions.

Fedora users can directly install Ansible through default repository, but if you are using RHEL/CentOS 6, 7, you have to enable EPEL repo.

* **EPEL** (Extra Packages for Enterprise Linux) is open source and free community based repository project from Fedora team which provides 100% high quality add-on software packages for Linux distribution including RHEL (Red Hat Enterprise Linux), CentOS, and Scientific Linux. To Enable EPEL Repository in RHEL/CentOS 7 use below lines

# wget http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-10.noarch.rpm

# rpm -ivh epel-release-7-10.noarch.rpm

* verify that the EPEL repository is enabled

# yum repolist

1. After configuring epel repository, you can install Ansible using following command.

**$ sudo yum install ansible -y**

1. After installed successfully, you can verify the version by executing below comman**.**

**# ansible --version**

**Install Ansible Ubuntu**

Here we are going to use official Ansible **PPA** repository on the system, just run the below commands to add the repository.

# apt-add-repository -y ppa:ansible/ansible

# apt-get update

# apt-get install -y ansible

1. Preparing **SSH Keys to Remote Hosts**

# ssh-keygen -t rsa -b 4096 -C admin@eis.com

1. After creating SSH Key successfully, now **copy the created key to hosts**(to all hosts).
2. ssh-copy-id root@192.168.33.36
3. **check the connect** to 192.168.33.36

#ssh root@192.168.33.36

# logout (logout from 36, (now we are in 35))

1. To check **the partitions on all remote hosts**

# ansible -m command -a "df -h" web-servers

1. **Memory usage** on all remote hosts.

# ansible -m command -a "free -mt" web-servers

1. Checking **Uptime** for all 3 servers.

# ansible -m command -a "uptime" web-servers

1. Check for hostname and **Architecture**

# ansible -m command -a "arch" web-servers

# ansible -m shell -a "hostname" web-servers

1. If we need the output to any file we can **redirect** as below.

# ansible -m command -a "df -h" web-servers > /tmp/df\_outpur.txt

**Ansible Inventory Hosts File (backup actual file)**

# cd /etc/ansible

# mv hosts hosts.bak

# touch hosts

In host file having

[test]

192.168.2.10

[test] represents the group we want to group our hosts under

**# ansible -m ping test (OR) #ansible test -m ping**

**# ansible -m ping all**  (Check the ssh connection from Ansible to host)

This is going to run the ping module against all the hosts in our Hosts file

# ansible -m ping -vvvv test

# ansible -m ping -k test (we will be asked for the SSH password)

playbook will look like this:

---

- hosts: test

tasks:

- name: Install NTP

apt: pkg=ntp state=installed updated\_cache=true

**Modules**

Ansible uses "modules" to accomplish most of its Tasks. Modules can do things like install software, copy files, use templates and much more**.**

ansible all -s -m apt -a 'pkg=nginx state=installed update\_cache=true'

127.0.0.1 | success >> {

"changed": false

}

Ex: apt – install remove packages , copy – copy files from local to host , file – set the attribute file directory , service – start, stop, restart services.

**Basic Playbook**

Playbooks can run multiple Tasks and provide some more advanced functionality that we would miss out on using ad-hoc commands

Playbooks and Roles in Ansible all use Yaml.

Create file nginx.yml:

---

- hosts: local

tasks:

- name: Install Nginx

apt: pkg=nginx state=installed update\_cache=true

**Handlers**

A Handler is exactly the same as a Task (it can do anything a Task can), but it will run when called by another Task. You can think of it as part of an Event system; A Handler will take an action when called by an event it listens for

This is useful for "secondary" actions that might be required after running a Task, such as starting a new service after installation or reloading a service after a configuration change.

---

- hosts: local

tasks:

- name: Install Nginx

apt: pkg=nginx state=installed update\_cache=true

notify:

- Start Nginx

handlers:

- name: Start Nginx

service: name=nginx state=started

We can add a notify directive to the installation Task. This notifies any Handler named "Start Nginx" after the Task is run.

**More Tasks**

we can add a few more Tasks to this Playbook and explore some other functionality

---

- hosts: local

vars:

- docroot: /var/www/serversforhackers.com/public

tasks:

- name: Add Nginx Repository

apt\_repository: repo='ppa:nginx/stable' state=present

register: ppastable

- name: Install Nginx

apt: pkg=nginx state=installed update\_cache=true

when: ppastable|success

register: nginxinstalled

notify:

- Start Nginx

- name: Create Web Root

when: nginxinstalled|success

file: dest={{ '{{' }} docroot {{ '}}' }} mode=775 state=directory owner=www-data group=www-data

notify:

- Reload Nginx

handlers:

- name: Start Nginx

service: name=nginx state=started

- name: Reload Nginx

service: name=nginx state=reloaded

**Roles**

Roles are good for organizing multiple, related Tasks and encapsulating data needed to accomplish those Tasks. For example, installing Nginx may involve adding a package repository, installing the package and setting up configuration.

Roles have a directory structure like this:

rolename

- default

- files

- handlers

- meta

- templates

- tasks

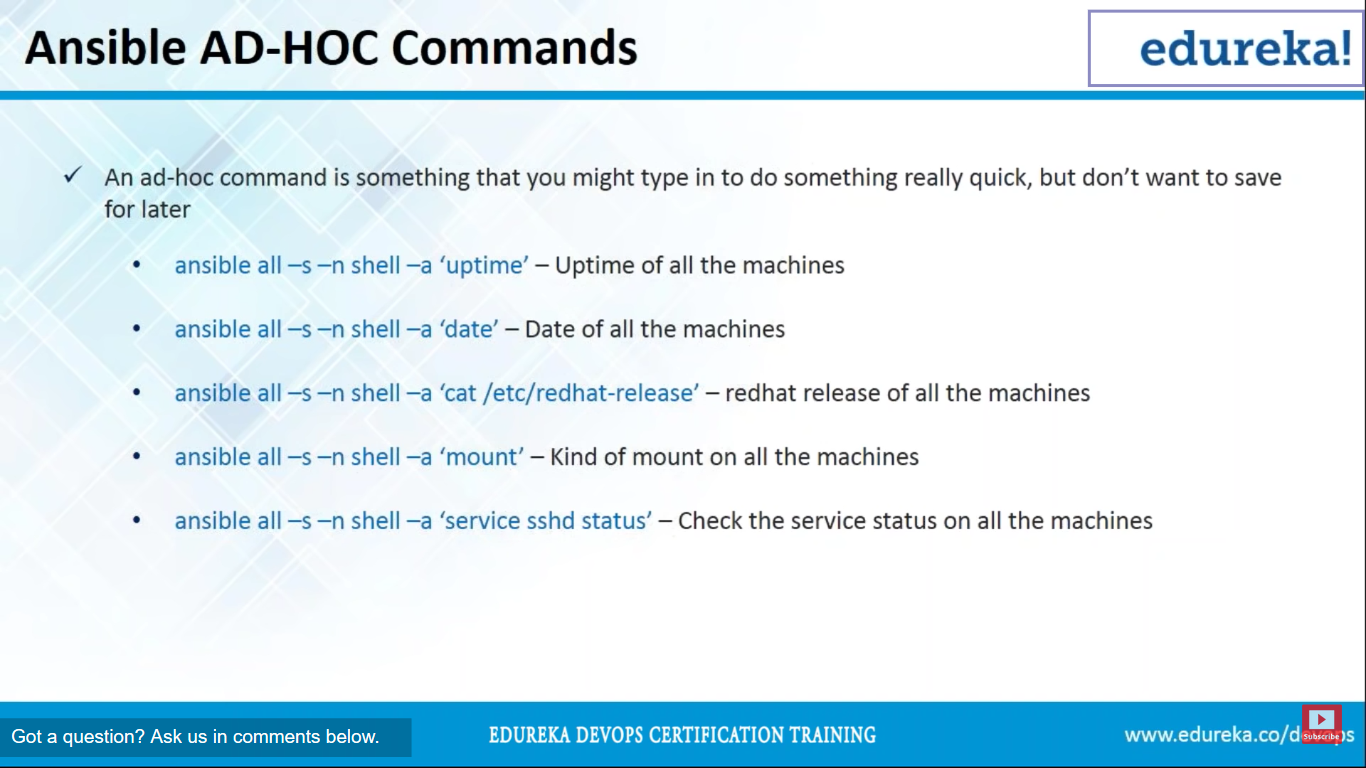
- vars

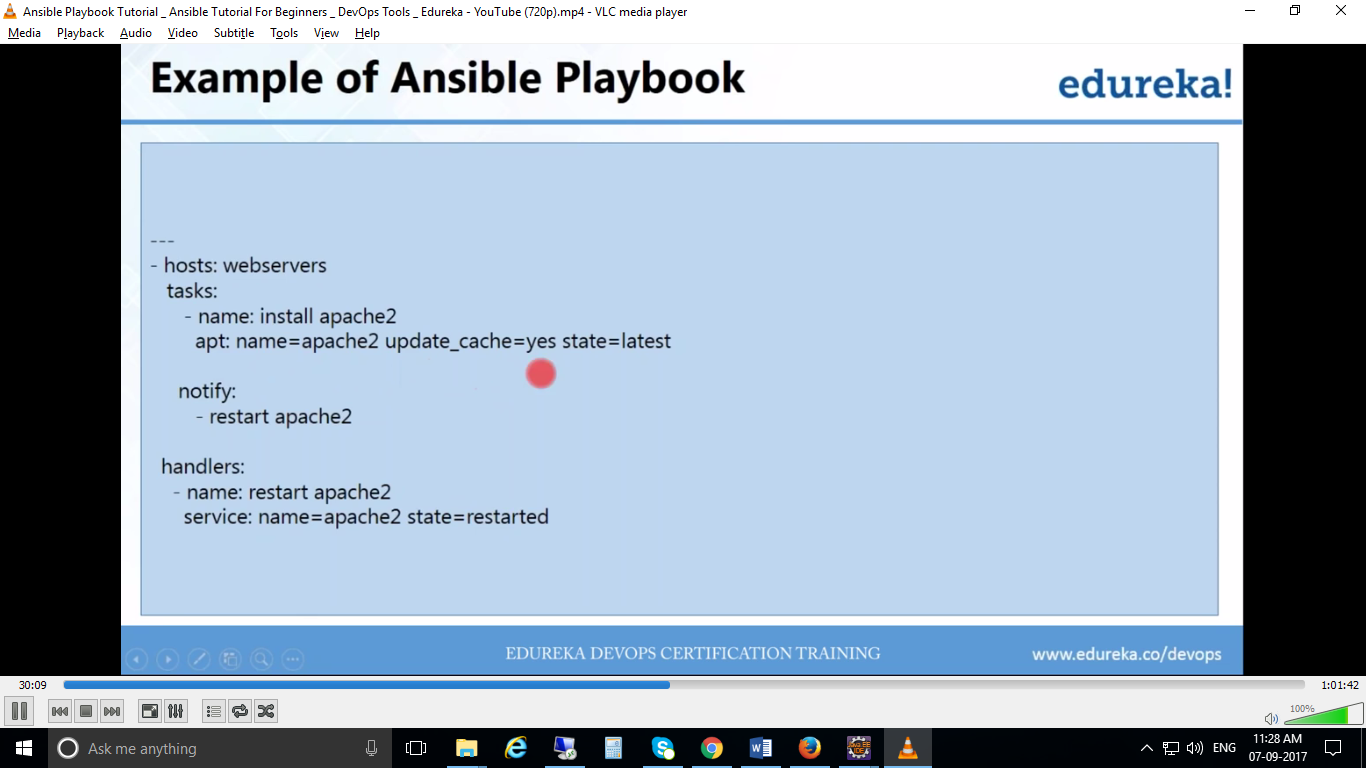
* **files**: put the static files here for copied to remote hosts. This may also include script files to run.
* **handlers**: All handlers that were in your playbook previously can now be added into this directory.
* **meta**: information about the role, like author supported platforms etc. dependencies if any.
* **templates**: You can place all files that use variables to substitute information during creation in this directory.
* **tasks**: This directory contains all of the tasks that would normally be in a playbook. These can reference files and templates contained in their respective directories without using a path.
* **vars**: Variables for the roles can be specified in this directory and used in your configuration files.

Within each directory, Ansible will search for and read any Yaml file called main.yml automatically.

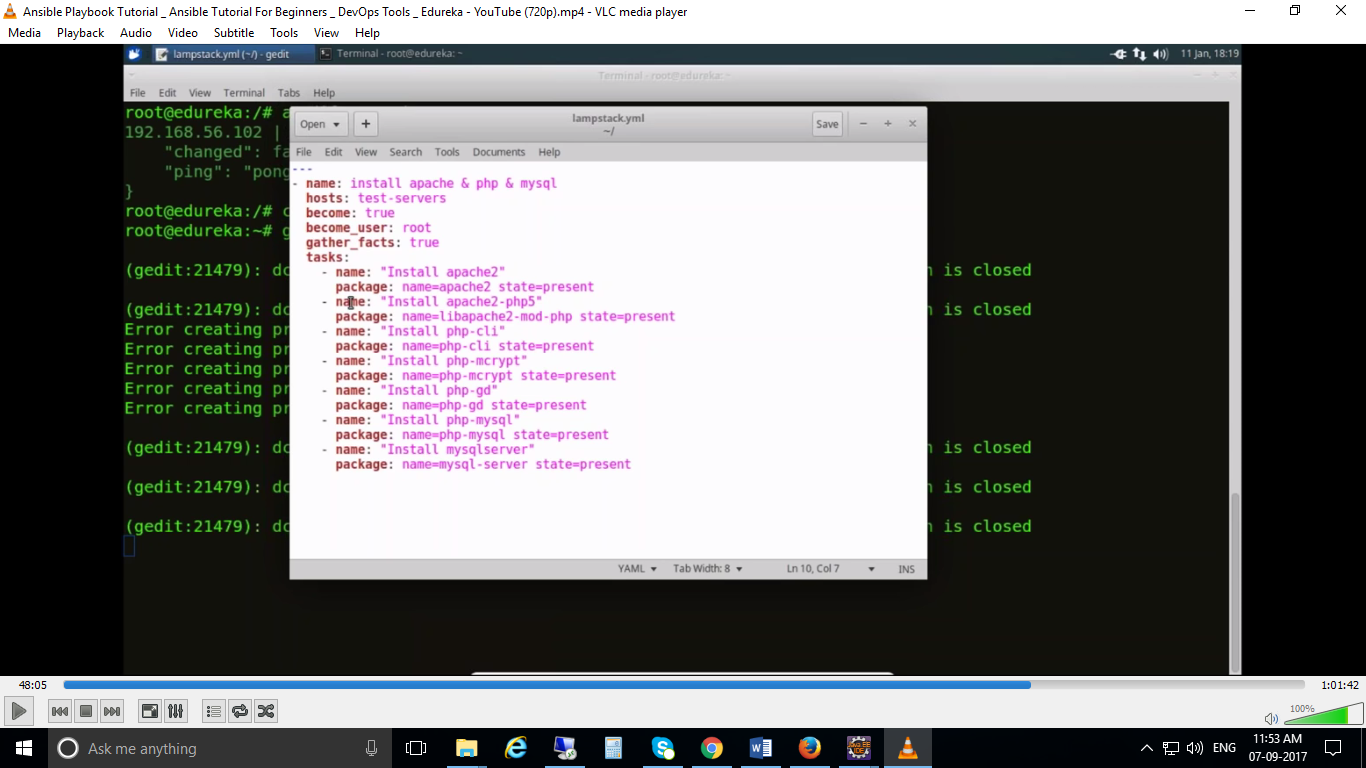
Create Role: go to /etc/ansible/roles ->

$ ansible-galaxy init apache (<role name>) (Now it will create apache folder)





Ex:2

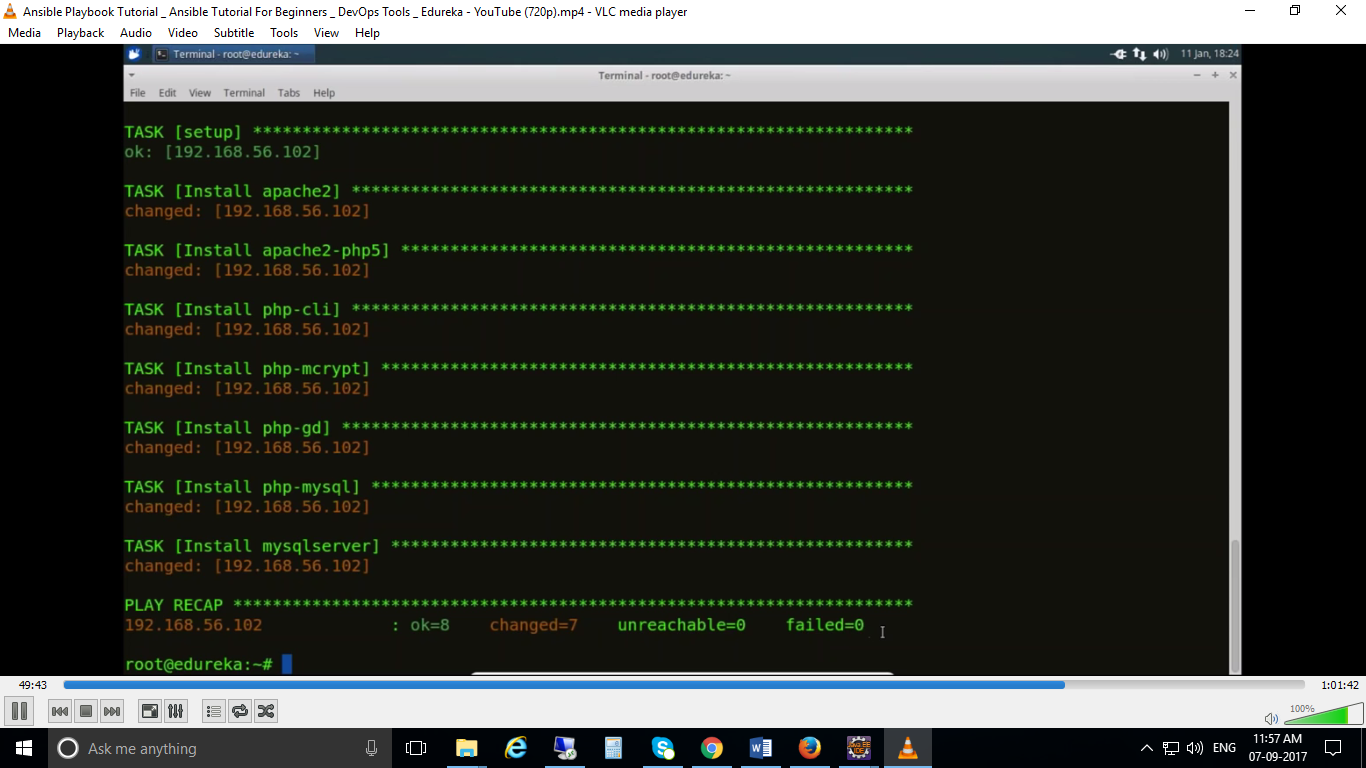


**Check the syntax of playbook:**

# ansible-playbook playbook\_name.yml –syntax-check

**Run playbook:**

# ansible-playbook playbook\_name.yml



* if host is windows machine. Linux is ansible machine, if it fails when we want to connect Linux to windows, then go to windows firewall -> Allow apps -> check the option “allow WinRM HTTPS”
* for debug this write playbook file

tasks:

-debug: msg= “{{ ansible\_os\_family}}”