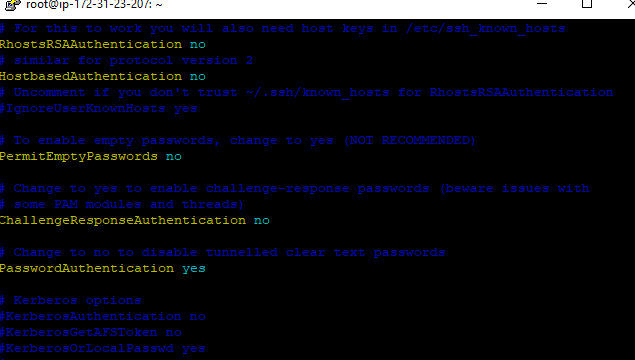
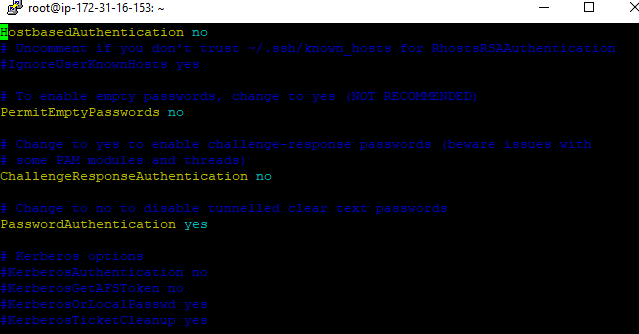
Install Ansible

1. Configuration

Ansible can be installed and run from on any machine. One machine is to be considered a Control Node/Management node and should be Linux. Second machine to be considered as a Target node. The Control Node/Management node will manage the Target nodes/Remote nodes. Ansible uses SSH to manage Target nodes/Remote nodes.

1. Login to the instances provided. Consider one instance as Control node and the other node as target node.
2. Execute the command sudo su – to get root access
3. Edit the /etc/ssh/sshd\_config file on both the machines and uncomment the lines for PasswordAuthentication variable.



1. Restart the sshd service using the below command:

service sshd restart

1. Execute the command ssh-keygen on both the machines(machine1 and machine2)
2. Copy the public key to the clipboard from the first machine (machine1) using the below command :

cat ~/.ssh/id\_rsa.pub

1. Append the copied contents to the authorized\_keys file in the second machine (machine2)

$ cat >> ~/.ssh/authorized\_keys

[Paste the copied content from the clipboard]

[Press ctrl+d to exit]

1. SSH from machine1 to machine2 and vice-versa using the below command:

$ssh -i ~/.ssh/id\_rsa <public\_ip>

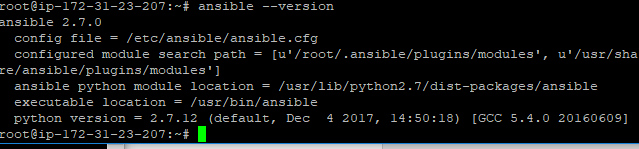
1. Installation of Ansible
2. Install Ansible on machine1 which will be the Control node to manage the target node, machine2

$ sudo yum install epel-release

$sudo yum install ansible

1. Verify that ansible is installed on the machine1

$ ansible –version



1. Verify python is installed on machine2

$python –version

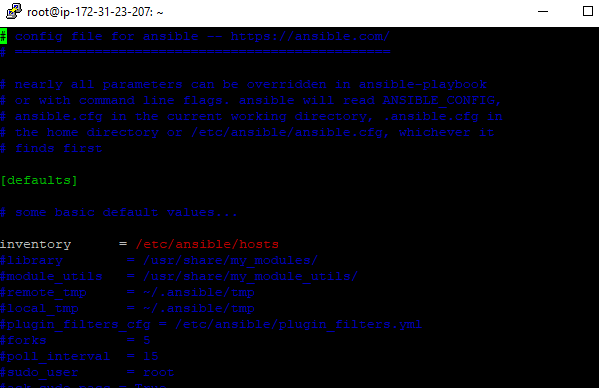


1. Configure Ansible hosts

Note: Ansible keeps track of all the remote machines that it knows about through a "hosts" file known as inventory file. We need to set up this file first before we can begin to communicate with our remote machines.

* 1. Edit the ansible.cfg file with root privileges and uncomment inventory variable as shown

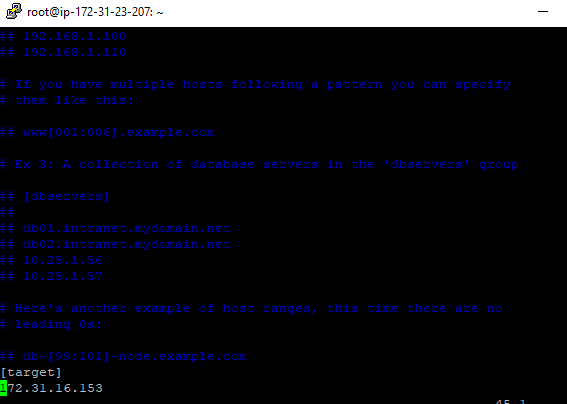
$ sudo vi /etc/ansible/ansible.cfg #



* 1. Edit the hosts file with root privileges and add the machine2 public ip or the hostname

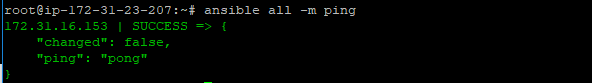
$ sudo vi /etc/ansible/hosts

Note: Ansible uses the concept of Inventory to manage and track the target machines. By default, this file is located in /etc/ansible/hosts and can be modified tool. A host file consists of groups of ip addresses or hostnames under the group.



* 1. Test the connectivity of the target nodes, machine2 using ansible ping module:

$ ansible all -m ping



$ ansible all –list-hosts



1. Ansible modules

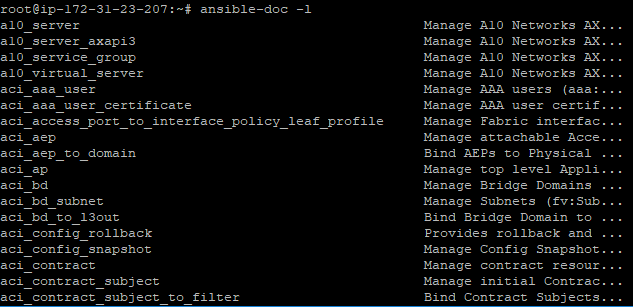
Modules are the main building blocks of Ansible and are basically reusable scripts that are used by Ansible playbooks. Ansible comes with a number of reusable modules. These include functionality for controlling services, software package installation, working with files and directories etc.

The syntax is as follows while running the ad-hoc commands which help in running single or simple tasks just once and which need not be run later. For E.g. just installing Tomcat on all servers.

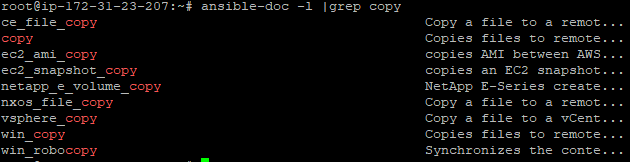
$ ansible hostORgroup -m module\_name -a "arguments" -u username --become

* 1. To list all the modules use the below command:

$ ansible-doc -l

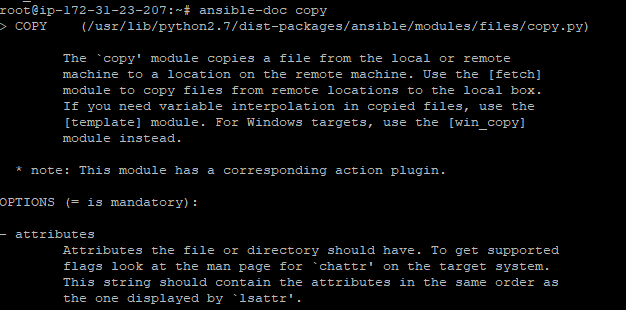


* 1. To find a specific module use the below command:

$ ansible-doc -l | grep copy

* 1. To find the documentation of specific module

$ ansible-doc copy

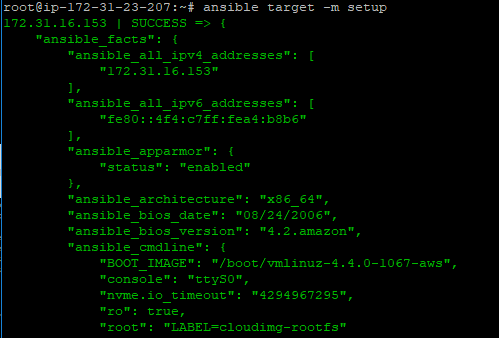


Ansible ad-hoc commands

Note: An ad-hoc command is something that you might type in to do something really quick, but don’t want to save for later.

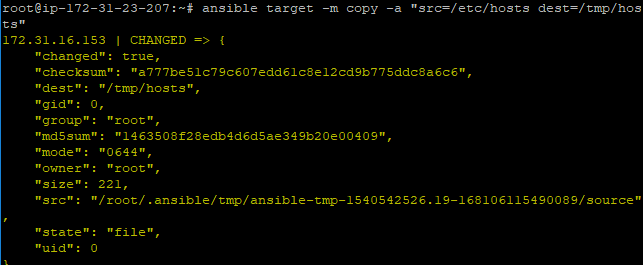
1. To get information about the network,machine or the metadata of the infrastructure ,use the setup module run the following command:

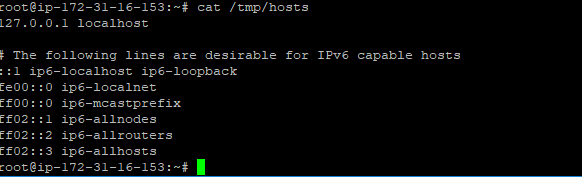
$ ansible target -m setup



1. To do File transfer from control machine to remote machine ,use the copy module

$ ansible target -m copy -a "src=/etc/hosts dest=/tmp/hosts"

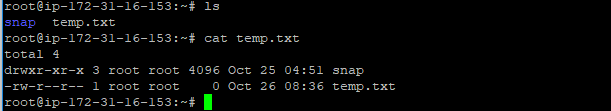




1. Using the shell module redirect the output of ls -l to temp.txt file on the remote machine.

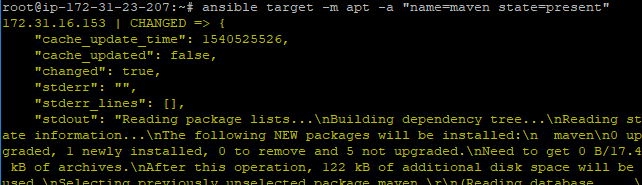
$ **ansible target -m shell -a 'ls -l > temp.txt'**





1. To install any software use apt/yum depending on the Linux distribution,use apt or yum module:

$ansible target -m apt -a “name=maven state=present”



1. To execute any comman,use command module:

$ansible -m command -a “arch” target



$ansible -m command -a "df -h" target > /tmp/df\_outpur.txt

Ansible Playbooks

Playbook have the following sections:

* Host Section
* Variable Section
* Tasks Section
* Handlers Section
* Conditionals Section
* Notify Section
* Until Section

To run any playbook use the following command:

$ Ansible-playbook <playbook.yaml>

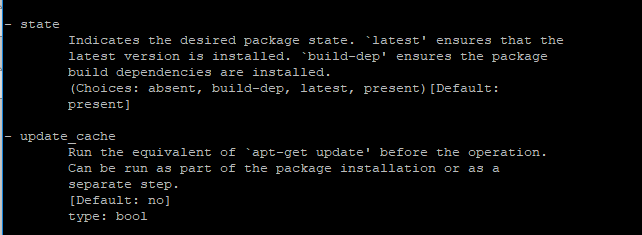
To check the playbook for syntax error

$ ansible-playbook <playbook.yml> --syntax-check

First 3 sections (Target, Variable and Tasks sections) are mandatory sections.

1. Sample-playbook.yaml : Install nginx and wget package in the remote machine and copy the file on the target machine.

* The playbook will have three tasks for installing nginx, wget and copying the files to the target machine.
* apt module installs wget and nginx on ubuntu.yum module installs on CentoS
* Copy module copies the file from control machine to target machine.
* Become elevates the user privileges to install the required package.
* Host section specifies the target machines on which the tasks will execute
* vars section is used to declare and define the variables like package1 and package2
* Check the parameters of any module e.g. apt module ,execute the following command:
  + ansible-doc apt



---

- hosts: target

vars:

package1 : "nginx"

package2 : "wget"

tasks:

- name: Installing package nginx

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

become: true

- name: Installing wget

apt: name={{package2}} state=present update\_cache=true

become: true

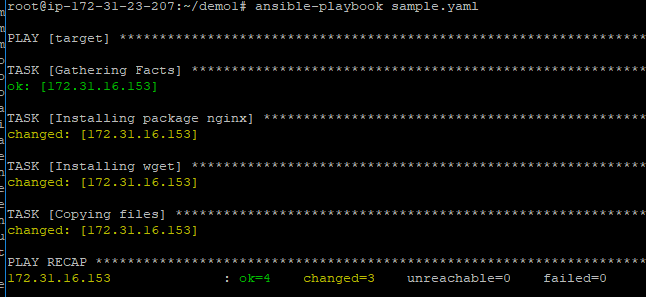
- name: Copying files

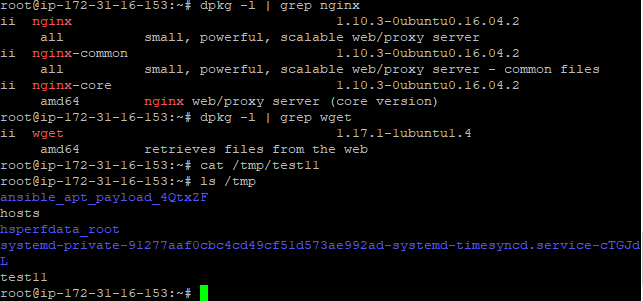
copy: src=/tmp/test11 dest=/tmp/test11

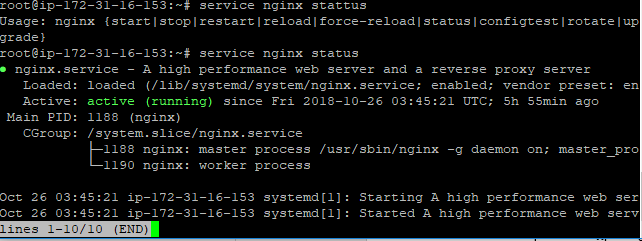
Paste the above code on the editor and save the file as sample.yaml.

Execute the following command to run the playbook:

$ansible-playbook sample.yaml







Output Analysis:

* It executed four tasks : Gathering facts, install nginx,wget and copy the files.3 changes are done on the remote machine installing wget,nginx and copying the files .
* Execute the below command to verify the installation and copy functionality:
  + dpkg -l | grep nginx
  + dpkg -l | grep wget
  + ls /tmp
  + service nginx status

1. Install Apache with Ansible Playbook (apache\_install.yaml)

* It executes 3 tasks of updating cache,installing apache2 and copying the index.html file to the /var/www/html
* Apt module installs the required packages and copy module will copy the file to the target node.
* Host section points to the target machine

---

- hosts: target

tasks:

- name: Installing apt-get update

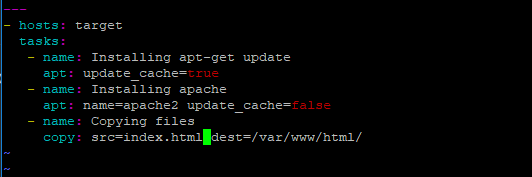
apt: update\_cache=true

- name: Installing apache

apt: name=apache2 update\_cache=false

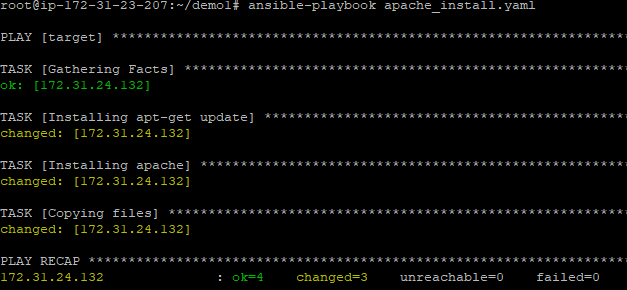
- name: Copying files

copy: src=index.html dest=/var/www/html/

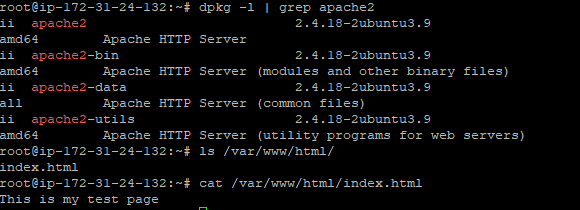


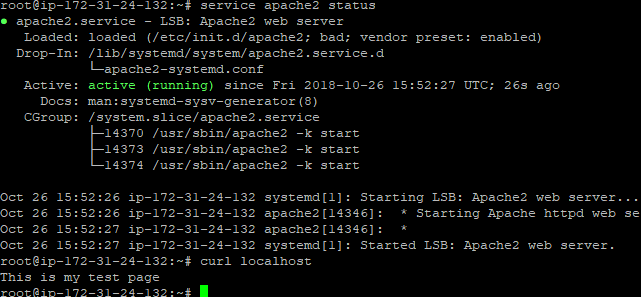
* Execute the playbook with the below command:

ansible-playbook apache\_install.yaml



* Analyze the output using the below commands:
  + dpkg -l | grep apache2
  + ls /var/www/html
  + service apache2 status
  + curl localhost





1. Run and Stop service

Add the following tasks to the above playbook:

* Service module start, stop, restart the service i.e. manage the services
* handlers and notify section manage the service only if the changes are done to configuration files:
  + For the first time ,Apache server will be installed and started.If you run the playbook even multiple times,nothing would happen if the service do not stop.
  + If the apache2 service gets stopped than notify section will notify Ansible to start the apache and handler will restart the apache service again once all other tasks complete.
  + Every handler task run at the end of the playbook if notified

- name: Stop service

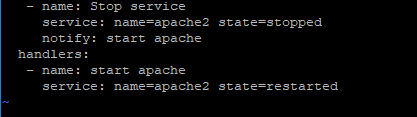
service: name=apache2 state=stopped

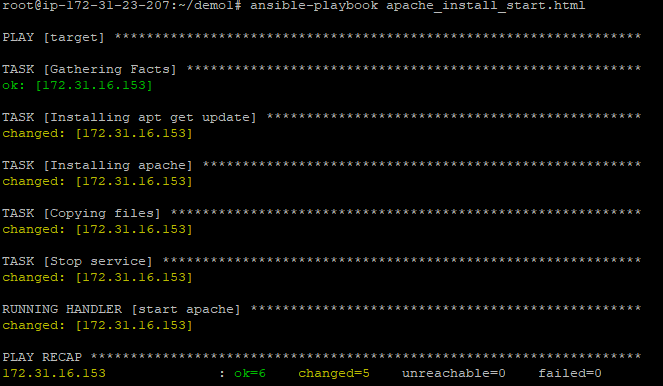
notify: start apache

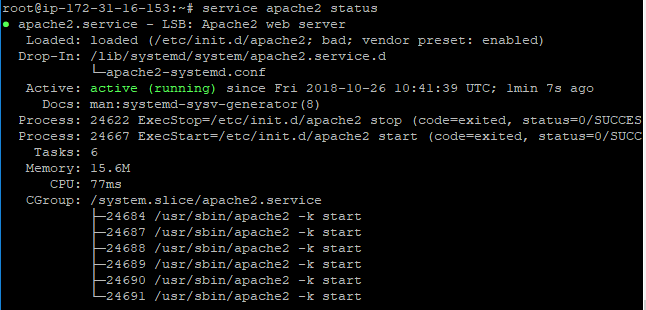
handlers:

- name: start apache

service: name=apache2 state=restarted







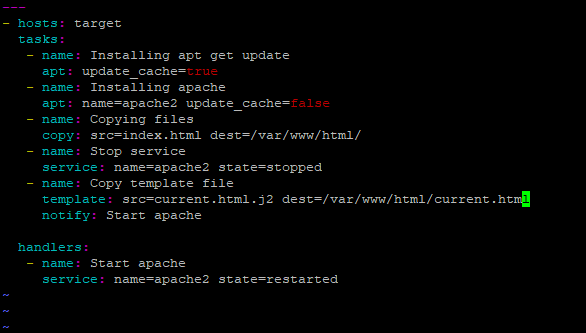
1. Ansible Template : Jinja Filters

Jinja2 filters are simple Python functions that take some arguments, process them and return the result.

(( myvar | filter }}

Ansible will pass myvar to the Jinja2 filter as an argument.The Jinja2 filter will process and retun the returning result.

* Refer to the playbook below. It is using a template module to work with Jinja2 file which will modify the content of current.html.j2 file. It will fetch the value of ansible\_hostname in the current.html. .j2 file and will save the current.html file with the actual value of ansible\_hostname in the target node.
* When the current.html file change on the target machine than notify directive will notify Ansible to restart the apache given in handler section.

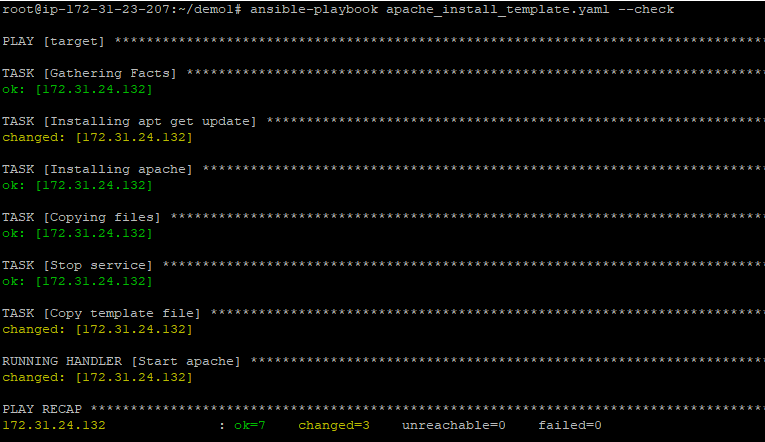


* Create the current.html.j2 file in the current directory.

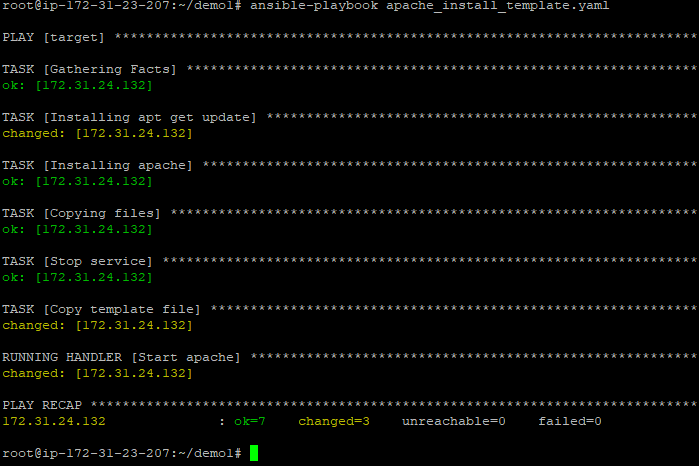
Add the following code to current.html.j2 (It is a Jinja file)



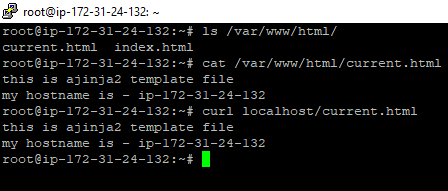
* Execute the playbook in DRY run mode
  + DRY run mode: To execute the playbook as a simulation rather than doing the changes actually on the target machine.It is a basic verification of ansible playbook using the –check option
  + ansible-playbook apache\_install\_template.yaml –check



* Execute the playbook to do the changes on the target node:
  + ansible-playbook apache\_install\_template.yaml



* Analyze the output



1. Debug statements and register variables in playbook

Register variable captures the output of any task to a register variable an debug module print the output.This module print statements during execution and printing the values without holding the execution.

Debug module will print the value of the uptime\_var varaible

---

- hosts: target

tasks:

- name: Installing apt get update

apt: update\_cache=true

- name: Installing apache

apt: name=apache2 update\_cache=false

- name: Copying files

copy: src=index.html dest=/var/www/html/

register: copy\_status

- name: Stop service

service: name=apache2 state=stopped

notify:

- start apache

- name: Check uptime

shell: uptime

register: uptime\_var

- name: Print uptime

debug: var=uptime\_var

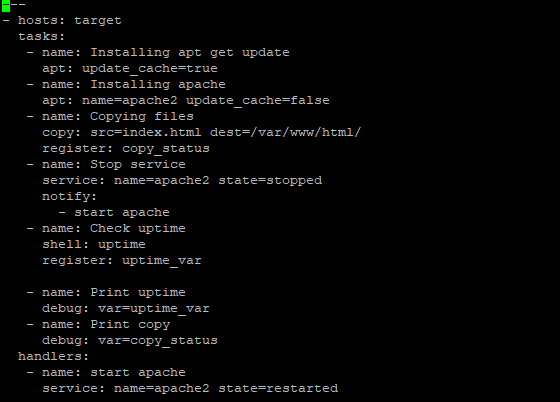
- name: Print copy

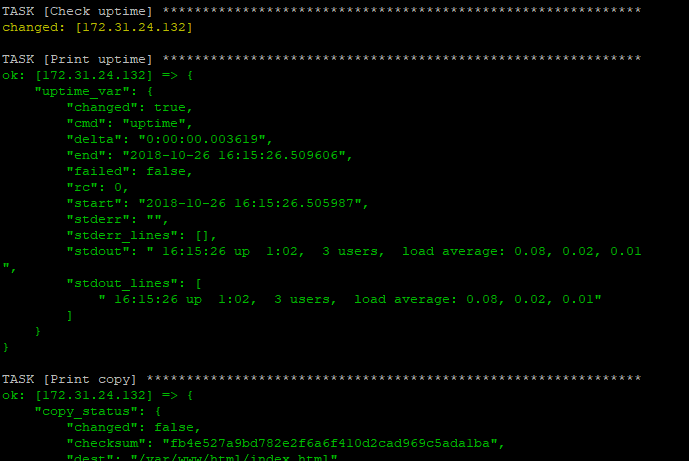
debug: var=copy\_status

handlers:

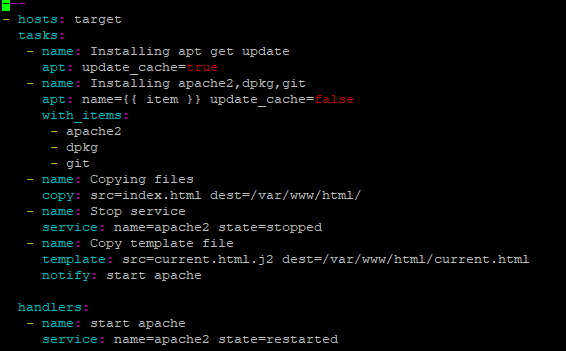
- name: start apache

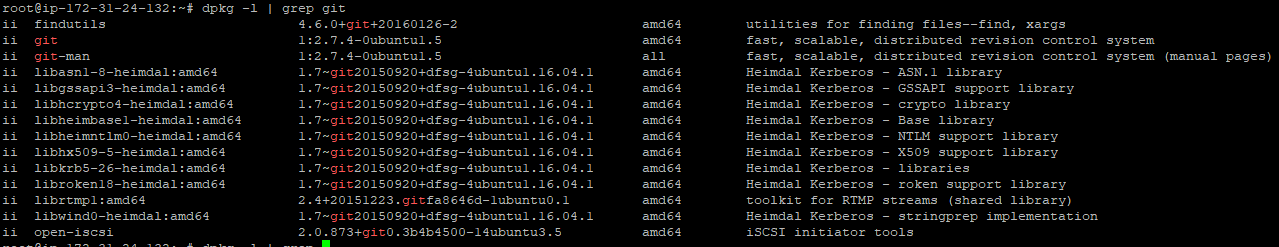
service: name=apache2 state=restarted

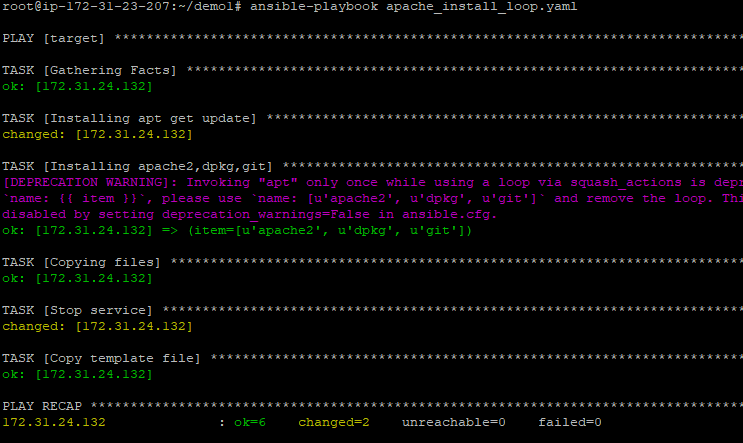




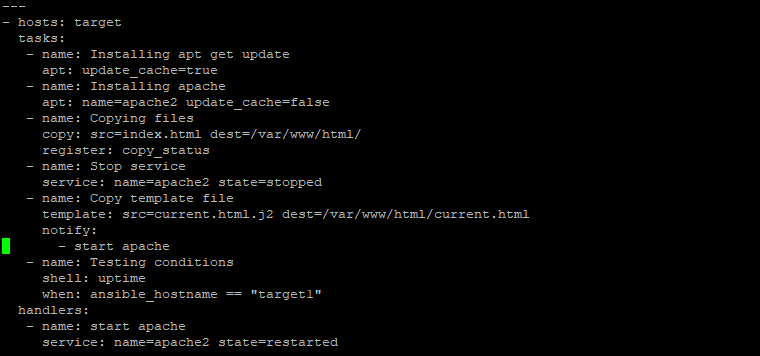
1. Loops and Conditions:Consider the code below below where we want to install multiple softwares and

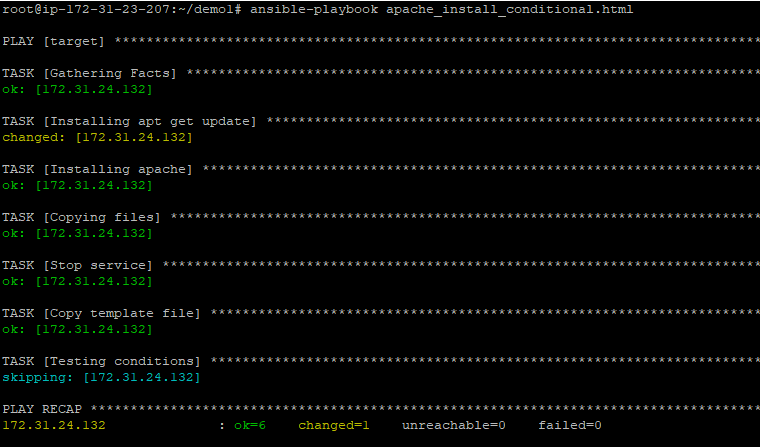






* Conditional:





If we replace target1 with target in when statement and run the playbook again

Ansible Vault

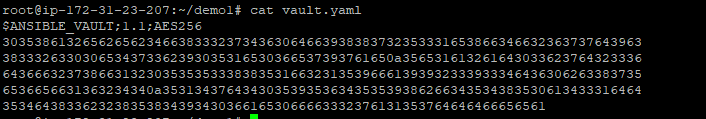
Most of the times when sensitive or confidential data need to be protected in the playbook, then it can be encrypted rather than just keeping it in a text file which is readable by all. Ansible Vault allows you to encrypt the playbook to protect the confidential data.

To create a new file encrypted with vault,use the ansible-vault create command

$ ansible-vault create vault.yaml

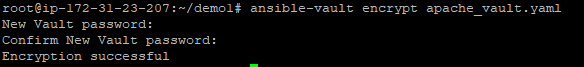


When you have confirmed your password, Ansible will immediately open an editing window where you can enter your desired contents.

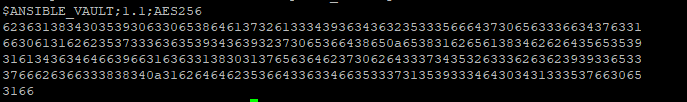


To encrypt an existing file,use the following command:

$ansible-vault encrypt filename

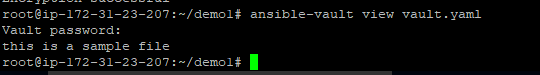


When you open the apache\_vault.yaml using vi/cat encrypted contents can be seen rather than the actual content



To reference the contents of a vault-encrypted file without needing to edit it or write it to the filesystem unencrypted.

$ ansible-vault view vault.yaml



To edit an encrypted files, use the ansible-vault edit command,It will prompt for the password and edit window opens to edit the existing content in the file:

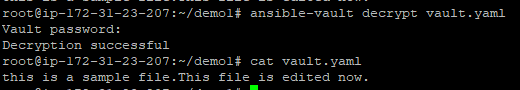
$ ansible-vault edit vault.yaml





To decrypt a vault encrypted file, use the ansible-vault decrypt command.

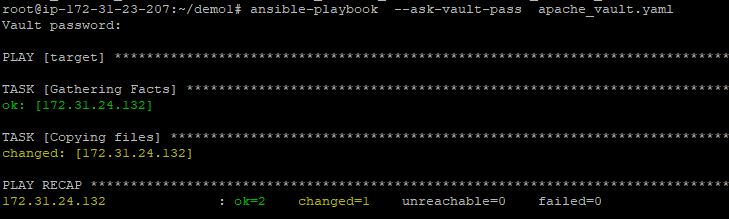
$ansible-vault decrypt vault.yml



to change the password of an encrypted file, use the ansible-vault rekey command:

$ansible-vault rekey encrypt\_me.txt

To use the encrypted file with playbook during execution use the flag –ask-vault-pass use the following command:

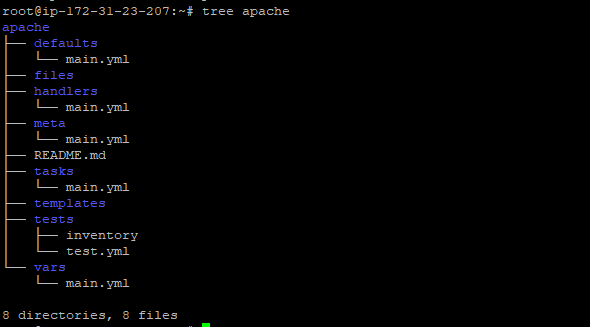


**Roles**

Roles are used for

The first step in creating a role is creating its directory structure.Use the below command to create the directory structure for roles:





A role’s directory structure consists of defaults, vars, files, handlers, meta, tasks and templates.

Create the roles for installing wordpress. It has four roles:

1. common for installing and configuring common dependencies like pythn
2. mysql for installing and configuring mysl-db
3. php for installing and configuring php
4. wp for installing and configuring wordpress

