**ANSIBLE**

**========================================================================**

1. **What is Ansible?**

Ans:- Ansible is an opensource automation platform. It is a simple automation language that can perfectly describe an IT application infrastructure in an ansible-playbook.

It is also an automation engine that runs Ansible playbook.

**2) what are the features and advantages of Ansible?**

Ans:-

1. Ansible is simple:-

--Human readable automation

--No special coding skills needed

--Tasks executed in order

--Get productive quickly

b) Ansible is Powerful:-

--App deployment

--configuration management

--workflow orchestration

--orchestrate the app lifecycle

c) Ansible is Agentless:-

--Agentless architecture

--Uses OpenSSH and WinRM

--No agents to explore or update

--More efficient and secure

d) Free:

-Ansible is an open-source tool.

e) Flexible:

-You can orchestrate the entire application environment no matter where it’s deployed. You can also customize it based on your needs.

**3)Use of ansible?**

Ans:- 1) Provisioning

2) Configuration Management

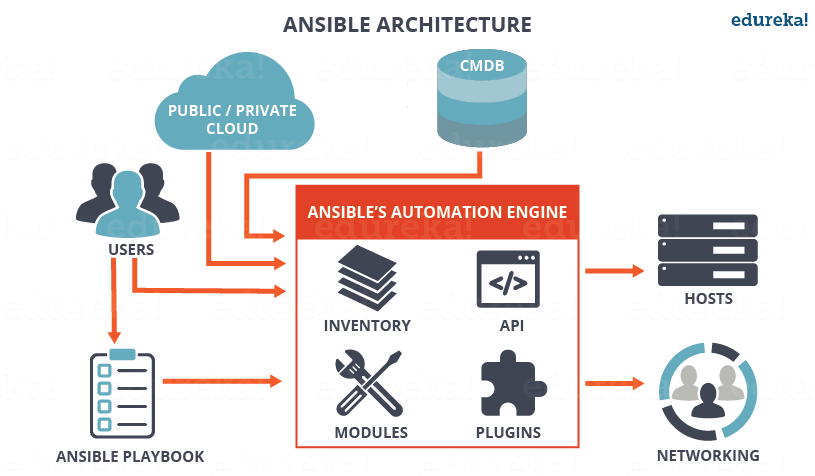
3) Application Deployment

4) Security and Compliance

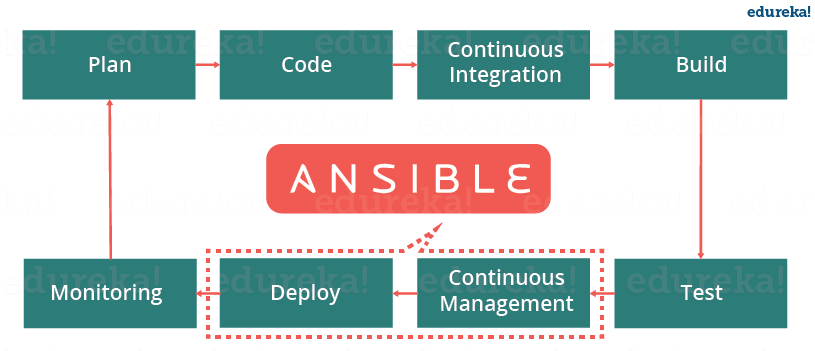
5) Orchestration

6) Continuous Delivery (CI/CD)

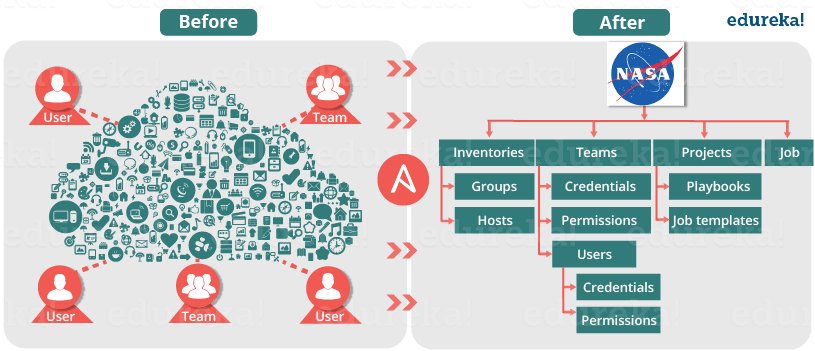
**4) The architecture of an Ansible?**



**5) Ansible in DevOps.**



**6) Ansible Case Study – A Real Life Usage by NASA**



Full Article:- <https://www.edureka.co/blog/what-is-ansible/>

=====================================================================

**Installation Of Ansible:-**

Ansible is simple to install. The Ansible software only needs to be installed on the control node form which Ansible will be run. Host by Ansible do not need to have Ansible installed. This installation involves relatively few steps and has minimal requirements.

The control node should be Linux or Unix systems. Microsoft Windows is not supported as the control node, although windows systems can be managed hosts.

Python 2 (version 2.7 or later) or Python 3 (version 3.5 or later) needs to be installed on the control node.

Below are the steps to install Ansible.

**Step 1>>** first, checked the python version

# yum list installed python (rhel/fedora/centos)

# apt list --installed | grep python (ubuntu/debian)

Or

# python -V

**Step 2>>** yum install ansible

Check if it is installed properly or not.

# ansible -V

**---------------------------------------------------------------------------------------------------------------------**

**How to install ansible in RHEL 8 AWS instances:-**

Launch 2 ec2 instances ---> **1 for Control node** and **1 for Host node**

**We have to configure our ansible control node as well as our hosts' nodes as below.**

1) If you’re using Python3, install python3-pip package.

#dnf -y install python3-pip

In rhel 8 we have a dnf package installation tool.

2) Add EPEL repository to your CentOS 8 / RHEL 8 system.

# sudo dnf -y install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm>

3) Then Enable EPEL playground repository and install Ansible on CentOS 8 / RHEL 8 from it.

#dnf install --enablerepo epel-playground ansible -y

4) Now, check whether it is installed or not.

#ansible --version

5) After completing the installation of ansible, then we have to create user ansible on both control node and slave nodes

#useradd ansible

#passwd ansible

6) Now, give sudo access to ansible user to perform necessary actions. (both on ansible and slave nodes)

#echo "ansible ALL=(ALL) ALL" >> /etc/sudoers

7) Enable password authentication no to yes. (both on ansible and slave nodes)

#sed -ie 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd\_config

8) after configuring sshd file make sure restart sshd service. (both on ansible and slave nodes)

#service sshd restart

9) Login as ansible user (on control node)

#su ansible

10) Then, generate ssh-keygen

#ssh-keygen

Or you can type below cmd

echo -e "\n"|ssh-keygen -t rsa -N ""

11)Copy keys onto all ansible client nodes ------> hosts nodes (public IP)

#ssh-copy-id <target public ip add>

11) check whether an ansible user is able to ssh to host nodes.

# ssh ansible@public\_ip\_hosts (ssh username@targetipadd)

12)configure ansible hosts file.

echo (public\_ip\_of \_hosts) > /etc/ansible/hosts

13) Finally ping your hosts nodes by using ansible adhoc cmd to check connectivity and configuration.

#ansible all -m ping

**Please find the below boot-strap script for the same.**

**Script for ansible installation**

--------

#!/bin/bash

dnf -y install python3-pip

sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

dnf install --enablerepo epel-playground ansible -y

useradd ansible

echo 'ansible:123' | chpasswd

echo "ansible ALL=(ALL) ALL" >> /etc/sudoers

sed -ie 's/PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd\_config

service sshd restart

----------

**#Then, switch user to ansible**

su ansible

**#Then, generate keygen and copy ssh id with the host public id.**

echo -e "\n"|ssh-keygen -t rsa -N ""

ssh-copy-id <target public ip add>

---------------------------------------------------------------------------------------------------------------------

**How to install ansible on ubuntu AWS instance.**

**Step1>>** sudo apt-add-repository ppa:ansible/ansible -y

**Step2>>** sudo apt-get update && sudo apt-get install ansible -y

Step3>> please repeat 5 to 13 steps as given above.

This, we can install and configure Ansible.

=====================================================================

**Building an Ansible Inventory**

**What is Inventory in Ansible?**

**---->** An Ansible defines a collection of hosts that Ansible will manage. These hosts can also be assigned to groups, which can be managed collectively. Groups can contain child groups, and hosts can be members of multiple groups. The inventory can also set variables that apply to the hosts and groups that it defines.

These host inventory can be defined in two different ways.

1. **static host:-** a static host inventory can be defined by a text file. You can write this file using various formats, including INI-format and a format expressed as a YAML document.

INI format inventory file:-

mail.example.com

[webservers] ---------------------------------------> hosts group name

foo.example.com

bar.example.com

[dbservers]-----------------------------------------> hosts group name

one.example.com

two.example.com

Three.example.com

192.0.2.43

**Defining the nested groups:-** Ansible hosts inventories can include groups of hosts. This is accomplished by creating a host group name with the **:children suffix.** Refer below example.

**[atlanta]**

**host1**

**host2**

**[raleigh]**

**host2**

**host3**

**[southeast:children]**

**atlanta**

**raleigh**

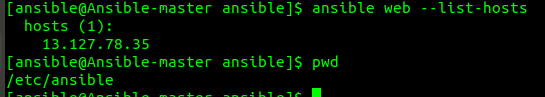
**Simplifying Host specification with Ranges.**

[START:END]

1. 192.168.[4.7].[0:255]
2. Server[01:20].example.com
3. [a:c].dns.example.com
4. 2001:db8::[a:f] ---> it matches all ipv6 IP address from 2001:db8::a to 2001:db8::f

**How to verify the inventory file:-**

# ansible web -i inv.ini --list-hosts

****

**Overriding the location of the inventory:-**

The **/etc/ansible/hosts** file is considered the default **s**tatic inventory file in ansible. However, the ansible and ansible-playbook commands that you use to run Ansible ad hoc commands and playbooks. The location of an inventory file on the command line with the **--inventory PATHNAME or -i PATHNAME,** where PATHNAME is the path to desired inventory file.

**2) Dynamic Inventory:-**

Ansible inventory information can also be dynamically generated, using information provided by external databases. The open source community has written a number of dynamic inventory scripts that are available from the upstream Ansible project.

Ansible easily supports all of these options via an external inventory system. The contrib/inventory directory contains some of these already – including options for EC2/Eucalyptus, Rackspace Cloud, and OpenStack, examples of some of which will be detailed below.

---------------------------------------------------------------------------------------------------------------------

**How to create Dynamic Inventory for the AWS EC2 instances.**

1) yum install python-pip

2) pip install boto

3) cd /etc/ansible

4) we have to download to file

a)ec2.ini b)ec2.py

5) wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.ini

6)wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/ec2.py

7) then, create a file and edit as below.

# touch /root/.aws/credentials

[personal]

aws\_access\_key\_id = AKIATC2DXDLOXV3JOFOL #(differnt in your case )

aws\_secret\_access\_key = t/MwEIyaQ5vQG8W4DVfceKL7GNK+QVMS9fIWpAb1 #(differnt in your case )

region = ap-south-1

#do the necessary changes

8) then edit ec2.ini and ansible.cfg file accordingly

a) ec2.ini --->

regions = ap-south-1

regions\_exclude = us-gov-west-1, cn-north-1

Note:-by default regions = all (change if you want)

b) ansible.cfg

[defaults]

# some basic default values...

inventory = /etc/ansible/ec2.py

Note:- mention proper path for inventory as your hosts file become an ec2.py

9) export

a) export ANSIBLE\_HOSTS=/etc/ansible/ec2.py

b)export EC2\_INI\_PATH=/etc/ansible/ec2.ini

c)export AWS\_ACCESS\_KEY\_ID='AK123' --> do changes

d)export AWS\_ACCESS\_KEY\_ID='AKIATC2DXDLOXV3JOFOL --> do changes

10) then, finally test your configuration as per below cmd.

# ./ec2

note:- (this would be different in other case)

aws secret and access key id

AKIATC2DXDLOXV3JOFOL

t/MwEIyaQ5vQG8W4DVfceKL7GNK+QVMS9fIWpAb1

Reference:-

https://jee-appy.blogspot.com/2016/10/dynamic-inventory-ansible.html

https://www.youtube.com/watch?v=Hr8I7pxtG9U

**====================================================================**

**Managing Ansible Configuration File:-**

**Configuration Ansible:-** The behavior of an ansible installation can be customized by modifying the setting in the Ansible config file. Ansible chooses its configuration file from one several possible locations on the control node.

**3) Using /etc/ansible/ansible.cfg**

**2) Using ~/.ansible.cfg**

**1) ./ansible.cfg**

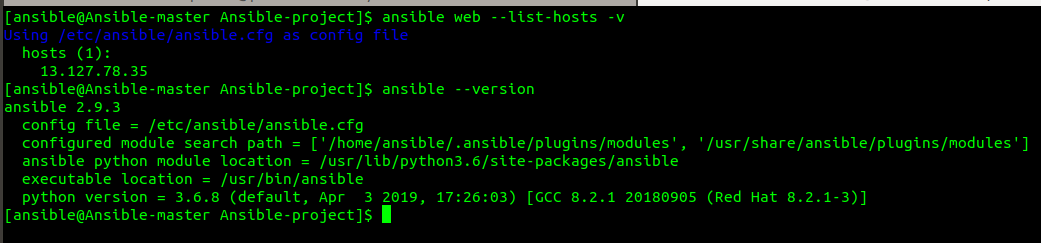
Configuration File Precedence:-

The search order for a configuration file is the **reverse of the preceding list.** That’s why we numbering in reverse order for the above 3 file location of ansible.cfg

How to check which ansible configuration file currently being used in our ansible env.

---> # ansible --version OR

# ansible web --list-hosts -v



**Managing setting in the configuration file:-**

In ansible there are two sections which we can configure ansible.cfg file.

1. [defaults] :- sets defaults for ansible operation.
2. [privilege\_escalation]:- configure how ansible performs privilege escalation on managed hosts

For example, the following is a typical **ansible.cfg file.**

[defaults]

# some basic default values...

#inventory = /etc/ansible/hosts

#forks = 5

#poll\_interval = 15

#sudo\_user = root

#ask\_sudo\_pass = True

#ask\_pass = True

#transport = smart

[privilege\_escalation]

#become=True

#become\_method=sudo

#become\_user=root

#become\_ask\_pass=False

**Ansible-Configuration options:-**

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Cmd options** |
| Inventory | The location of the ansible inventory file. | -i |
| remote\_user | The remote user account to establish connections to managed hosts | -u |
| ask\_pass | Prompt for password to use when connecting as the remote user. |  |
| become | Enable or disable privilege escalation for operations or managed hosts. | --become, -b |
| become\_method | The privilege escalation method to use on managed hosts. | --become-method |
| become\_user | The user account to escalate privilege to on managed hosts. | --become-user |
| become\_ask\_pass | Defines whether privilege escalations on managed hosts should prompt for a password. | --ask-become-pass,  -K |

=====================================================================

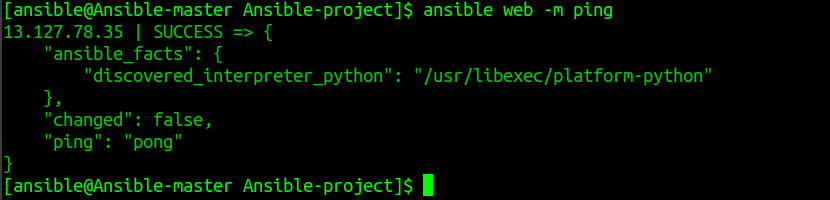
**Running AD HOC Commands**

An ad hoc command is a way of executing a single Ansible task quickly, one that you do not need to save to run again later. They are a simple, online operation that can be run without writing a playbook.

**Syntax of an ad hoc command:-**

*ansible host-pattern -m module [-a ‘module arguments’] [-i inventory]*

One of the simple ad hoc commands uses the **ping module.**



Performing tasks with modules using Ad Hoc commands

The **ansible-doc -l** command list all the modules that are installed on the systems.

E.g. **ansible-doc ping**

**The common module which we can use in the ansible ad hoc command.**

1. **Files modules:-** 
   1. copy b) file c) lineinfile d)synchronize
2. **Software package modules:-**
3. package b) yum c) apt d) dnf e) gem f) pip g) yum

**3) System modules:-**

a) firewalld b) reboot c) service d) user

**4) Net tools modules:-**

1. get\_url b) nmcli c) uri

**Ad Hoc Commands:-**

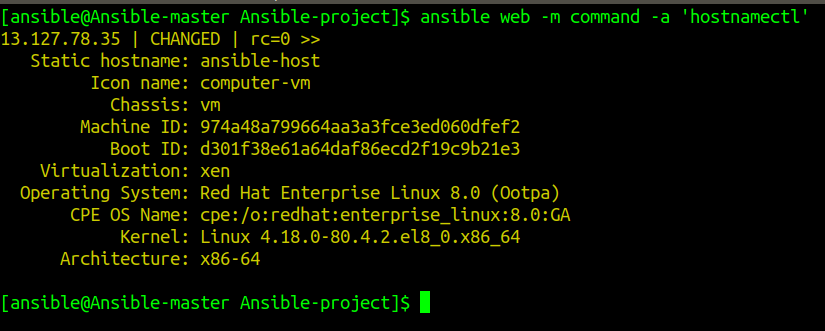
1) for creating user at managed hosts.

#ansible web -m user -a 'name=praffultest uid=4000 state=present' -b -K



2)to check the hostname of the host node

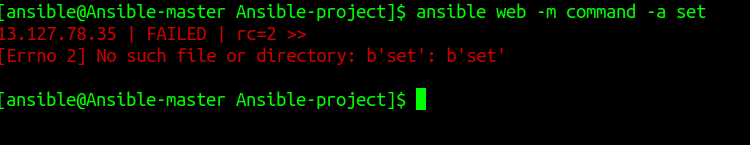
#ansible web -m command -a ‘hostnamectl’

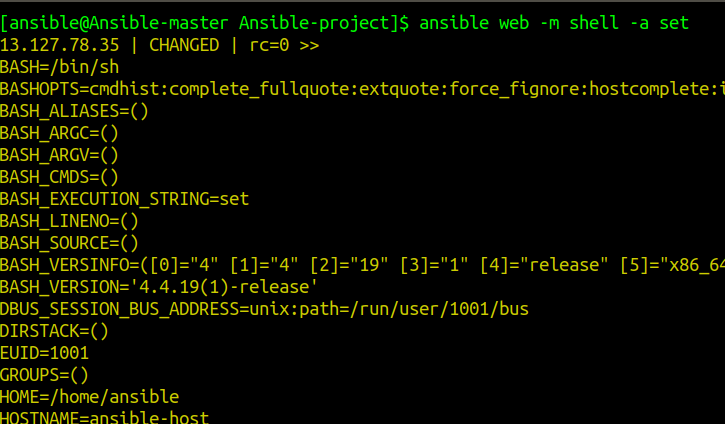


Note:- use the **-o option** to display the output of Ansible ad hoc commands in a single-line format.

4) The following example illustrates the **difference between the command and shell module**.

Note:- try to execute the built-in Bash command set with these two modules, it will only succeed with the **shell module.**





Note:- shell command is not **idempotent.**

**What is idempotent ansible?**

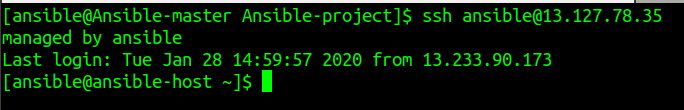
---> An idempotent operation is one that can be applied multiple times without changing the result beyond the initial application, such as multiplication by zero. Ansible modules are idempotent.

5) Copy from one dest to another.  
#ansible 172.31.28.157 -m copy -a 'src=/etc/passwd dest=/home'

6) how to edit /etc/motd file

#ansible all -m copy -a 'content="managed by ansible\n" dest=/etc/motd'

o/p:-



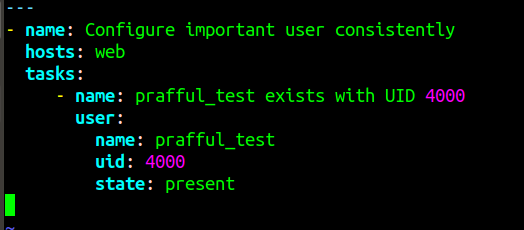
=====================================================================

**Implementing Playbooks.**

**“A play is an ordered set of tasks that are run against hosts selected from your inventory. A playbook is a text file (yml format) that contains a list of one or more plays to run in order.”**

**Format of ansible-playbook:-**

A simple playbook

****

**Same in ad hoc command:-**

#ansible -m user -a 'name=praffultest uid=4000 state=present'

**>>How to run the playbook:-**

Syntax:- ansible-playbook playbook\_name.yml

**>>How to check the syntax of the playbook.**

#ansible-playbook playbook\_name --syntax-check

**>>Increasing output verbosity.**

The **ansible-playbook -v** command provides additional information with up to four total levels.

**---> -v , -vv , -vvv, -vvvv**

Refer below screenshots for the same.

****

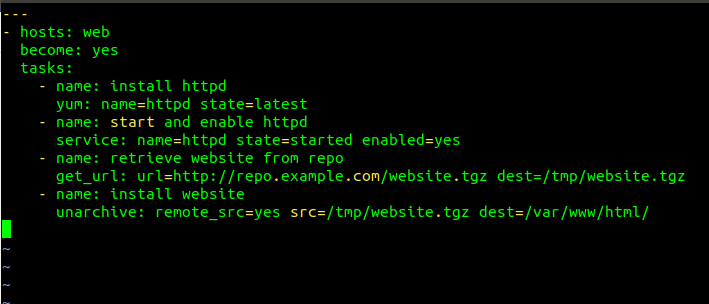
**>>Executing dry Run:-**

This causes an inability to report what changes would have occurred if the playbook were executed,but does not make any actual changes to managed hosts.

# ansible-playbook -C playbook\_name.yml

---------------------------------------------------------------------------------------------------------------------

**Apache web server configuration playbook**

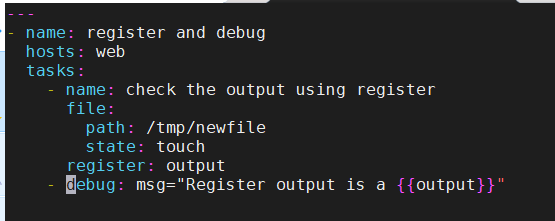


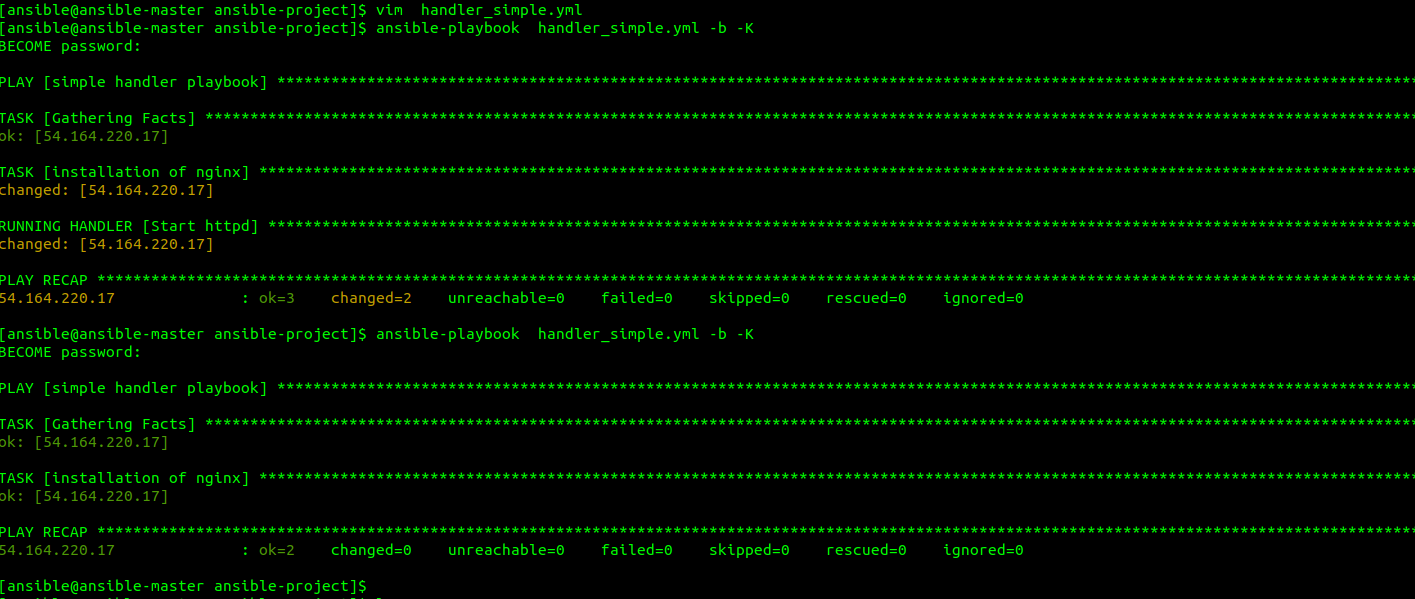
**Use a variable to retrieve the result of the running command.**

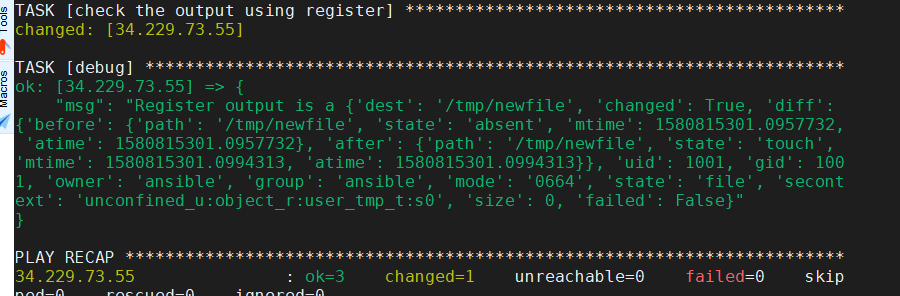
Ansible registers are used when you want to capture the output of a task to a variable. You can then use the value of these registers for different scenarios like a conditional statement, logging etc.

The variables will contain the value returned by the task. The common return values are documented in Ansible docs.

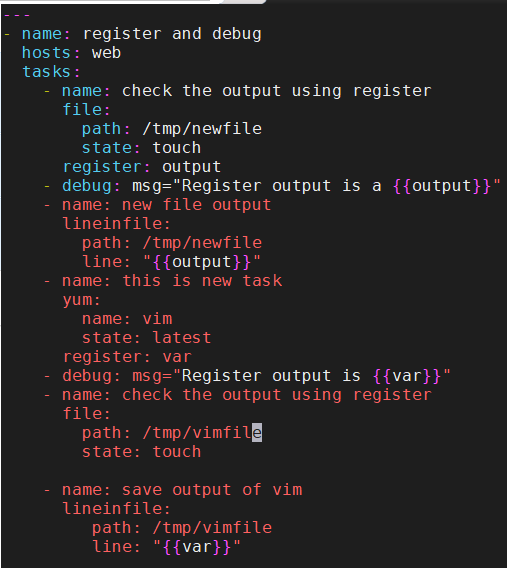
Playbook:- how to use register and debug module to capture the output of task.



output:-



Playbook2:-

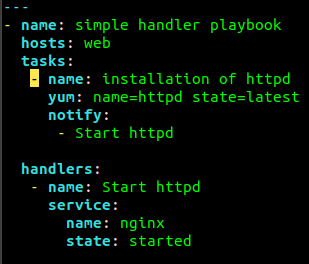


**Handler:-**

A Handler is exactly the same as a Task, but it will run when called by another Task. A Handler will take 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.

Simple handler playbook:-



Output:-

**Managing Variable in Ansible**

**Introduction to variables:-** Ansible supports variables that can be used to store values that can be reused throughout files in an Ansible Project. This can simply the creation and maintenance of a project and reduce the number of errors.

Example of valid variables which we can use in Ansible.

1. web\_server
2. first1
3. first\_variable\_define
4. remote\_file

Defining Variables:-

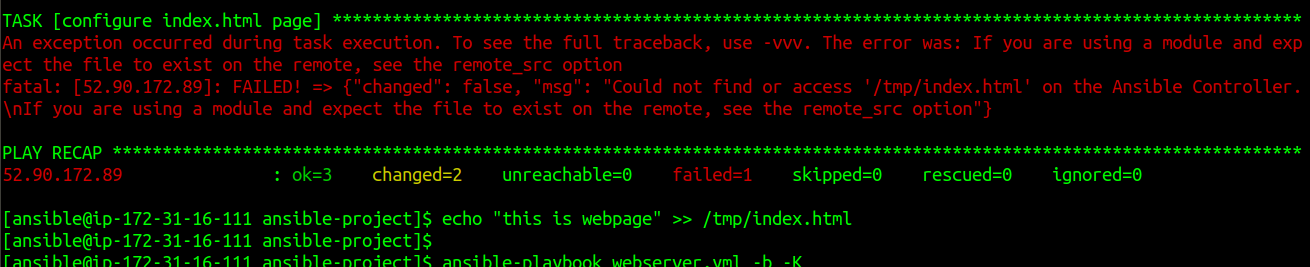
Variables can be defined in a variety of places in an Ansible Project. However, this can be done by three basic scope leveles.

1. Global scope:- variables which can be set from the command line or ansible configuration.
2. Play Scope:- Variables set in the play and related structures.
3. Host Scope:- variables set on host groups and individual hosts by the inventory, fact gathering or registered tasks.

Note:- If the same variable name is defined at more than one level, the level with the highest precedence wins.

How we can defining variables in ansible:-

Playbook variables can be defined in multiple ways. One common method is to place a variable in the vars block at beginning of the playbook.

****

**====================================================================**

**Ansible AWS**

**---**

**- name: ec2-launching**

**hosts: all**

**connection: local**

**tasks:**

**- name: ec2**

**ec2:**

**aws\_access\_key: AKIAST6R2Z2LKJOAALFU**

**aws\_secret\_key: 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**

**region: us-east-1**

**key\_name: test**

**instance\_type: t2.micro**

**image: ami-0fc61db8544a617ed**

**wait: yes**

**group: SG**

**count: 1**

**vpc\_subnet\_id: subnet-0041358580b7d0569**

**assign\_public\_ip: yes**