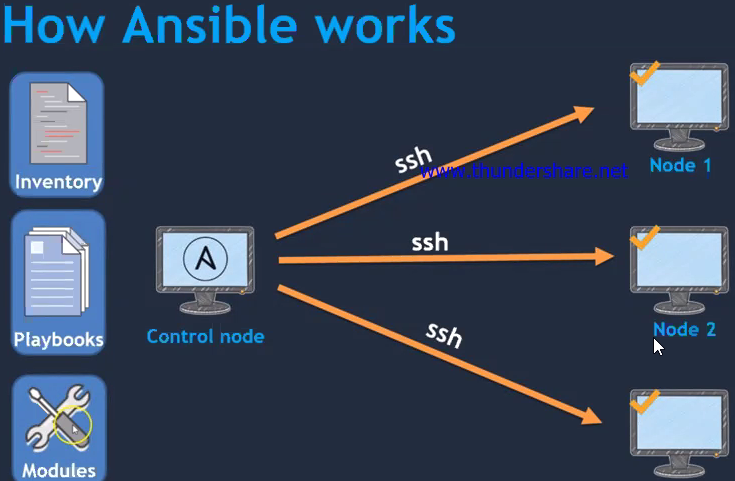
**What is Ansible?**

Ansible is a simple but powerful configuration management and orchestration intended for IT professionals, who use it for configuration management, cloud provisioning, application deployment, intra-service orchestration, and updates on workstations and servers, and nearly for anything a systems administrator does on a day-to-day

**Why Ansible and how it works?**

* Ansible does not just automate but also simplifies the repetitive, complex, and strenuous tasks that bring substantial time savings and increases overall productivity.
* It is straightforward to deploy; it leverages SSH to communicate between servers and thereby pushing out small programs, known as 'Ansible modules' to it.
* It uses the playbook to describe automation jobs, and playbook uses a very simple language YAML.
* Ansible is coded using the Python programming language. The servers that have to be configured need to have [Python libraries](https://www.upgrad.com/blog/python-libraries-for-data-science/).
* YAML is a digestible data serialization language (process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices) that is often utilized to create configuration files and works in accordance with any programming language. It is designed for human interaction



**Definitions**:

* **SSH** is a client-server based protocol. This means the protocol allows device requesting information or services (the client) to connect to another device (the server). When a client connects to a server over SSH, the machine can be controlled like a local computer.

**Note:** The main difference is that **sshd** is a server (like a web server serving https) and **SSH** is a client (think of a web browser). The client/user authenticates itself against the server using the user’s credentials

* **Inventory** files are nothing but host files where configuration details of nodes is stored. If you want to deploy, it will scan an inventory files whether configuration details of nodes is stored or not.
* **Playbooks** are the files where Ansible code is written. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks. Playbooks contain the steps which the user wants to execute on a particular machine.
* A **module** is a reusable, standalone script that Ansible runs on your behalf, either locally or remotely. Modules interact with your local machine, an API, or a remote system to perform specific tasks like changing a database password or spinning up a cloud instance.

**Advantages of Ansible**

* **Free**: Ansible is an open-source tool.
* **Very simple to set up and use**: No special coding skills are necessary to use Ansible.
* **Powerful**: Ansible lets you model even highly complex IT workflows.
* **Flexible**: You can orchestrate/manage/organize the entire application environment no matter where it’s deployed. You can also customize it based on your needs.
* **Agentless**: You don’t need to install any other software or firewall ports on the client systems you want to automate. You also don’t have to set up a separate management structure.
* **Efficient**: Because you don’t need to install any extra software, there’s more room for application resources on your server.

**What Can Ansible Do?**

The following tasks can be solved using Ansible:

* OS (de)installation
* OS configuration
* creation and removal of users
* management of users’ passwords and keys
* creation and removal of virtual environments and containers
* deployment of code
* Running scripts and tests, etc.

**Ansible’s unique feature set:**

* Based on an agent-less architecture (unlike Chef or Puppet).
* Accessed mostly through SSH
* No custom security infrastructure is required.
* Configurations (playbooks, modules etc.) written in the easy-to-use YML format.
* Shipped with more than 250 built-in modules.
* Full configuration management, orchestration, and deployment capability.
* Ansible interacts with its clients either through playbooks or a command-line tool.

**Ansible Terms:**

* **Controller Machine**: The machine where Ansible is installed, responsible for running the provisioning on the servers you are managing.
* **Inventory**: An initialization file that contains information about the servers you are managing.
* **Playbook**: The entry point for Ansible provisioning, where the automation is defined through tasks using YAML format.
* **Task**: A block that defines a single procedure to be executed, e.g. Install a package.
* **Module**: A module typically abstracts a system task, like dealing with packages or creating and changing files. Ansible has a multitude of built-in modules, but you can also create custom ones.
* **Role**: A pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of a provisioning.
* **Play**: A provisioning executed from start to finish is called a play. In simple words, execution of a playbook is called a play.
* **Facts**: Global variables containing information about the system, like network interfaces or operating system.
* **Handlers**: Used to trigger service status changes, like restarting or stopping a service.

**Setting up Ansible server:**

1. Set up ec2 instance
2. Set up host name
3. Create ansadmin user
4. Add user to sudoers file
5. Generate SSH keys
6. Enable password based login
7. Install ansible

**Set up ec2 instance**

* Open AWS management
* Launch instances
* Select free tier Amazon Linux 2 AMI and follow rest of the steps.
* Specify instance name as Ansible-server
* Open mobaXterm
* Create new SSH session, specify Remote host and username (ec2-user) for ansible-server
* Browse .pem key, click on OK

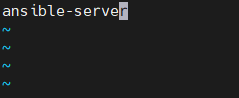
**Set up host name**

* Once command prompt is open, login as root, rename hostname from IP address to ansible-server by invoking below mentioned command

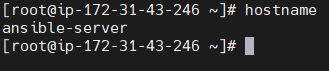


* Open vi /etc/hostname and delete whole content of file and specify “ansible-server”





* If just invoke a command, still it displays an IP address instead of hostname



* To rename the same, invoke a command : sudo su –



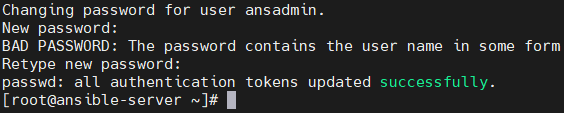
**Definition:**

**Sudo -** The sudo command allows you to **run** programs with the **security** privileges of another user (by default, as the super user). It prompts you for your personal password and confirms your request to execute a command by checking a **file**, called sudoers , which the system administrator configures

**Create ansadmin user**

* Add useradd and passwd as ansadmin





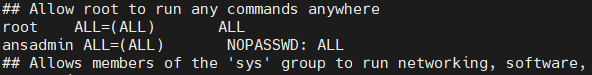
**Add user to sudoers file**

* Invoke command visudo
* The visudo command is a safe and secure way of editing the /etc/sudoers file on UNIX and Linux systems



* Grant permission to ansadmin to run any commands anywhere, save and exit.

Ansadmin ALL=(ALL) NOPASSWD: ALL



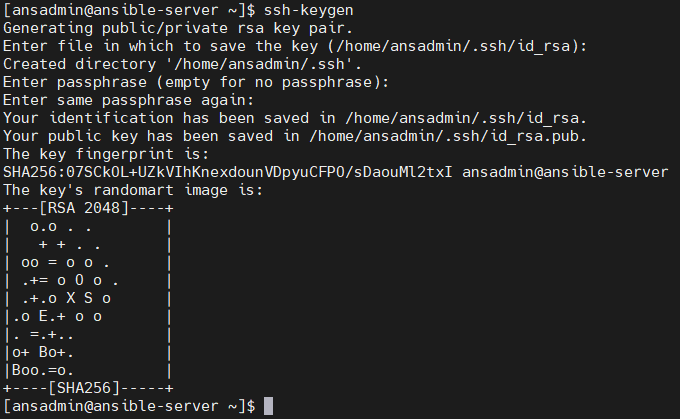
* Now we have added user to sudoers file
* You may check whether, you will be able to login as ansadmin as sudo by invoking below mentioned command



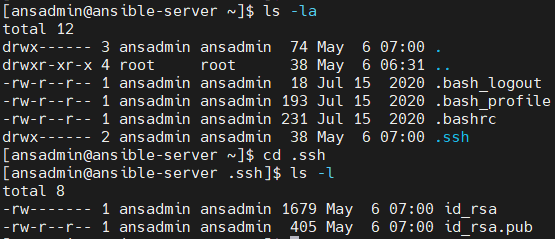
* If you observe in above screenshot, you have logged in as ansadmin

**Generate SSH Key**

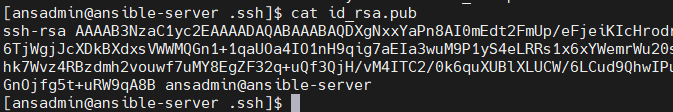
* Invoke a command ssh-keygen and press enter until key’s random art image is displayed.



* To check and confirm whether key is generated, list out files located in current directory
* Navigate to .pub file located in .ssh directory



* To check generated key, open a file id\_rsa.pub

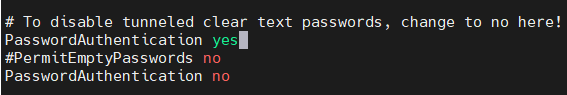


**Enable password based login**

* Exit from ansadmin and login as root
* Open file vi /etc/ssh/sshd\_config



* Change value of PasswordAuthentication from No to yes and delete “#” symbol present at the beginning of line. Save and exit

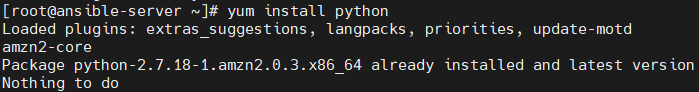


* Now to apply the same, I have to restart the service.



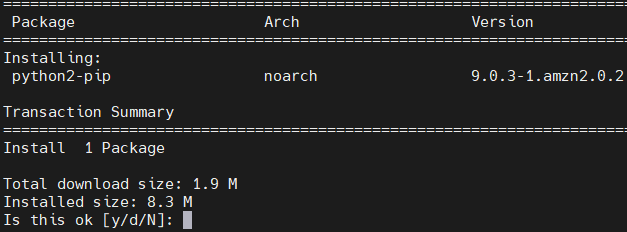
**Install Ansible**

* Since servers that have to be configured need to have [Python libraries](https://www.upgrad.com/blog/python-libraries-for-data-science/), we have to install python

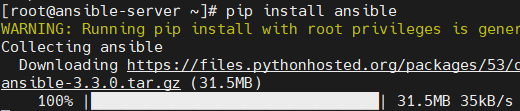


* Now install Python package-management system





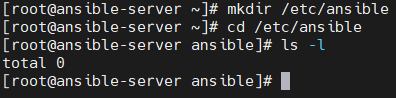
* Reply with comment yes.
* Install ansible : pip install ansible



* Now it is successfully installed ansible 3.3.0 version



* Create a directory by the name ansible



* Add ansible configuration file to etc/ansible directory
* Go to the browser and search for <https://github.com/ansible/ansible/blob/devel/examples/ansible.cfg>
* Copy whole content of file
* Go to MobaXterm command prompt
* Create and open file called vi ansible.cfg



* Check whether ansible.cfg file is created or not



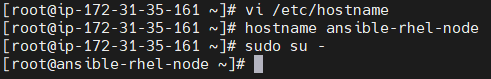
* Now I have create another file, Hosts also called as inventory file which stores configuration details of nodes



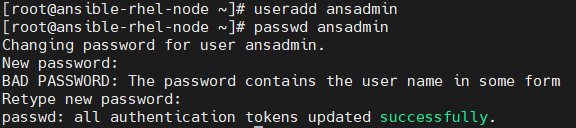
* Ansible is set now.
* Create another machine/node which will be added to the ansible-server
* Login AWS management
* Launch an instance with Red Hat Enterprise Linux 8 free tier
* Rename instance name as ansible-rhel-node
* Create new SSH session, specify Remote host and username (ec2-user) of ansible-rhel-node
* Browse .pem key, click on OK
* Once command prompt is open, login as root and rename the hostname



* Delete whole content of file and specify “ansible-rhel-node”



* Create ansadmin user



* Open sudoers file **Command** : visudo
* The visudo command is a safe and secure way of editing the /etc/sudoers file on UNIX and Linux systems



* Add ansadmin in sudoers file and grant permission to run any commands anywhere

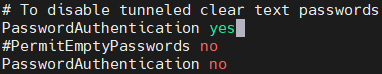




* Open file vi /etc/ssh/sshd\_config



* Delete “#” symbol present at the beginning of line “PasswordAuthentication yes”
* Save and exit



* Now to apply the same, I have to restart the service.



* Now we have added user to sudoers file
* You may check whether, you will be able to login as ansadmin as sudo by invoking below mentioned command



* Come back to the anisble-server command prompt
* Here we have to add copied IP address to “hosts” file
* Open hosts/inventory file and paste the Private IPv4 address of ansible-rherl-node, save and exit.

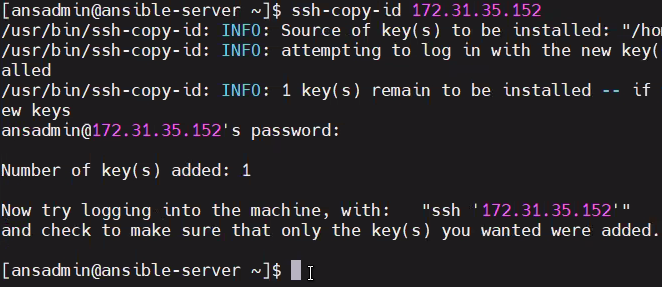


* Please note that we are in anisble-server command prompt
* I have to login as ansadmin



* Now copy the SSH key/Private IP address of node generated using ansadmin to the server
* Only then, it allows you to connect from server to the node
* **Command** : ssh-copy-id Private IP Address of node (172.31.35.161)
* Enter password: ansadmin, press Enter

**Note**: The Private IP address of node mentioned in the below screenshot is not the one which I have created. The mage is just for reference. Rests of screenshots are valid.



* If you observe, I am in ansible-server. Once you invoke command ssh Private IP Address of node, you will enter in ansible-rhel-node

**Note**: It won’t ask you for authentication details password anymore.

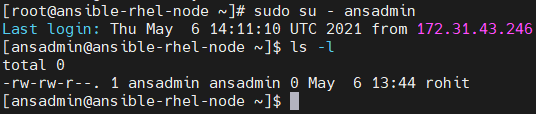


* Now create file by the name rohit under node



* File which you have created from ansible-server should reflect in ansible-rhel-node
* Go to ansible-rhel-node and list out files located in ansadmin

**Note**: You should be logged in as ansadmin



* Open rohit file and edit it



* Save and exit
* Now come to the ansible-server and check whether you will be to fetch the content updated in ansible-rhel-node/rohit file.



* If you want to close the connection, just pass command exit



**Introduction to ad-hoc commands**

An Ansible ad-hoc command uses the */usr/bin/ansible* command-line tool to automate a single task on one or more managed nodes. Ad-hoc commands are quick and easy, but they are not reusable.

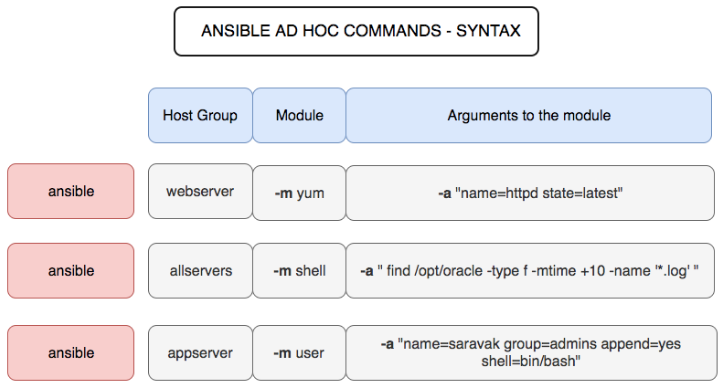
[**Why use ad-hoc commands?**](https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html#id4)

Ad-hoc commands are great for tasks you repeat rarely. For example, if you want to power off all the machines in your lab for Christmas vacation, you could execute a quick one-liner in Ansible without writing a playbook. An ad-hoc command looks like this:

**$ ansible [pattern] -m [module] -a "[module options]"**

* Ad-hoc tasks can be used to reboot servers, copy files, and manage packages and users, and gathering facts
* Ad hoc commands are commands which can be run individually to perform quick functions. These commands need not be performed later.
* For example, you have to reboot all your company servers. For this, you will run the Adhoc commands from ‘/usr/bin/ansible’.
* These ad-hoc commands are not used for configuration management and deployment, because these commands are of one time usage.

**Refer the following diagram to understand how the ansible ad hoc commands are framed.**



To run an ad hoc command, the command must be framed or have the following syntax.

1. **ansible <host-pattern> [options]**

for example. the command should be written as follows.

1. **ansible appserverhostgroup -m <modulename> -a <arguments to the module>**

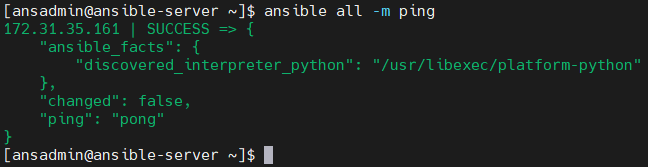
A single ansible ad hoc command can have multiple options. -m and -a are one amongst them and widely used.

Here are the 21 selected examples of ansible ad hoc commands. Please do visit below mentioned link for the explanation/description of commands.

Link: <https://www.middlewareinventory.com/blog/ansible-ad-hoc-commands/#ex2>

1. [ansible ad hoc ping command example](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex1)
2. [ansible ad hoc command to check uptime](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex2)
3. [ansible ad hoc command to check memory usage](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex3)
4. [ansible ad hoc command to get physical memory allocated to the host](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex4)
5. [Execute a command as root user (sudo) on host with ansible ad hoc command](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex5)
6. [ansible ad hoc command to Execute a command as a different user  (sudo su)](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex6)
7. [ad hoc command to Create a unix user group](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex7)
8. [ad hoc command to Create a unix user](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex8)
9. [ansible ad hoc command to Create a Directory with 755 permission](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex9)
10. [ansible ad hoc command to Create a file with 755 permission](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex10)
11. [ad hoc command to Change ownership of a file](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex11)
12. [ansible ad hoc command to  check free disk space of hosts](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex12)
13. [ansible ad hoc command to Install a package using yum command](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex13)
14. [ansible ad hoc command to Start or stop the service](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex14)
15. [Install and configure python Django application server with ansible ad hoc commands](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex14)
16. [Managing Cron Job and Scheduling with Ansible ad hoc](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex16)
17. [Running operations in the background asynchronous with Polling ansible](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex17)
18. [ansible ad hoc command to reboot remote system](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex18)
19. [ansible ad hoc command to check service status](https://www.middlewareinventory.com/blog/ansible-ad-hoc-command-examples-cheat-sheet/#ex19)
20. [ansible ad hoc command to copy file local to remote](https://www.middlewareinventory.com/blog/ansible-ad-hoc-commands/#Example_20_ansible_ad_hoc_command_to_copy_file_Local_to_remote)
21. [ansible ad hoc command to copy directory local to remote](https://www.middlewareinventory.com/blog/ansible-ad-hoc-commands/#Example_21_ansible_ad_hoc_command_to_copy_directory_Local_to_remote)
22. **How to check for the connectivity or whether server can ping to node/client using ad-hoc command**

* Please note that you have logged in as ansadmin and invoke below mentioned command.
* Ansible looks for the hosts file and Private IP address of nodes stored in it and confirms the status of connectivity.



* Ensure that private IP address of ansible-rhel-node/client is copied in the hosts file located at ansible-server
* To copy the same, login as root admin and open hosts file and edit.

**Command**: [root@ansible-server ~]# vi /etc/ansible/hosts

* In case you have already logged in as ansadmin, invoke below command

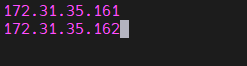


* Otherwise you will get below mentioned error



* What happens if you specify wrong Private IP address in hosts file
* Just to understand the concept, open hosts file and include an incorrect IP address.



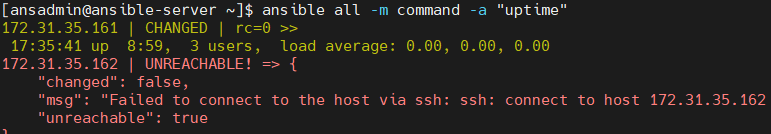


* Save and exit
* Now invoke command **ansible all –m ping** once again
* When you specify All, it considers all IP address of nodes present in hosts file
* If you observe, only the incorrect IP address is failed due to its non-existence/ not in operation.

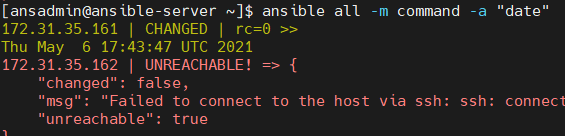


1. **How to check uptime (How long machine is in operation) command on nodes/clients configured in hosts file**

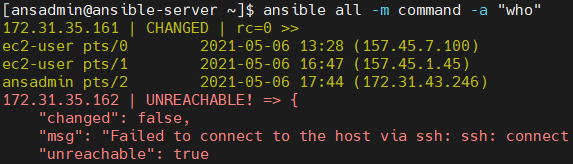
* Invoke a command, ansible all –m command –a “uptime”
* If you observe, 172.31.35.161 node is in operation since 8:59 hrs and the other node 172.31.35.162 is not existed/operational, ansible couldn’t reach



1. **How to check Date (Date of creation of node) command on nodes/clients configured in hosts file**

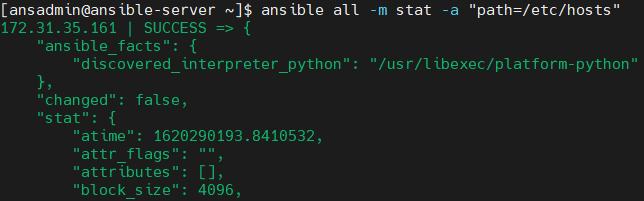
****

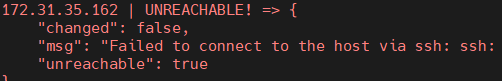
1. **How to check Who (Who all have logged into the node) command on nodes/clients configured in hosts file**



1. **How to check statistics of specific file/directory**

* It gives you detailed characteristic statistics of file you have specified in the command as well as data present in the file.





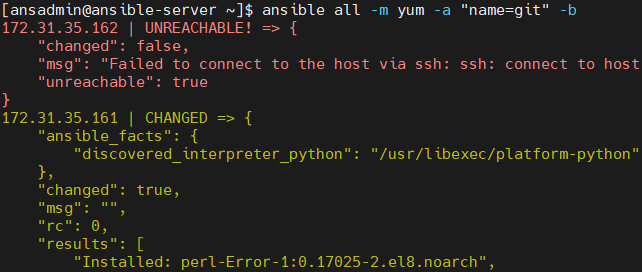
**Revisiting Definitions:**

**Sudo -** The sudo command allows you to run programs with the security privileges of another user (by default, as the superuser). It prompts you for your personal password and confirms your request to execute a command by checking a file, called sudoers , which the system administrator configures

**Visudo -** The visudo command is a safe and secure way of editing the /etc/sudoers file on UNIX and Linux systems. /etc/sudoers is instrumental for gaining privileged access via sudo command. Visudo is basically a wrapper for a text editor such as vi or nano.

1. **How to check GIT is installed on the System or not?**

* Once you invoke below mentioned command, go to ansible-rhel-node and just pass command “git”
* You should be logged in as ansadmin at the node

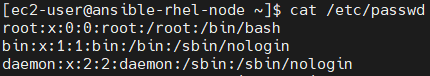


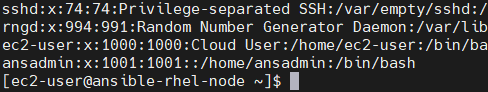
1. **How to create a new user in server and the same should reflect at node end**.

* Exit from ansadmin in node and login as ec2-user

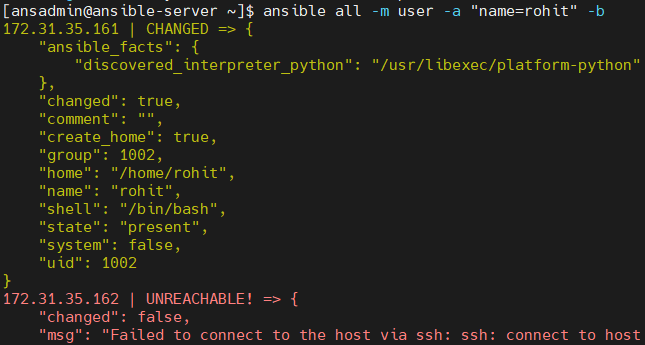


* Invoke a command, cat /etc/passwd to check who the last user is.





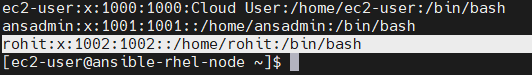
* If you observe in above screenshot, last user was ansadmin
* Now we will create new user in server and the same should be listed out at node end.
* Invoke below mentioned command at ansible-server end.



* If you are logged in as ansadmin in node, first Exit and login as ec2-user

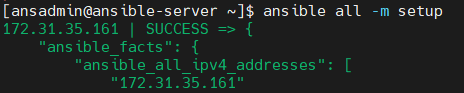


* Invoke a command, cat /etc/passwd
* If you observe, last user is “rohit”

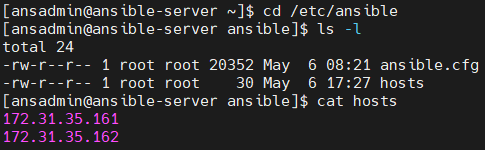


1. **How to fetch structural information of node**

* Invoke below mentioned command to fetch structural information of node

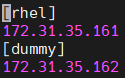
****

* We have hosts file in etc/ansible directory in the ansible-server

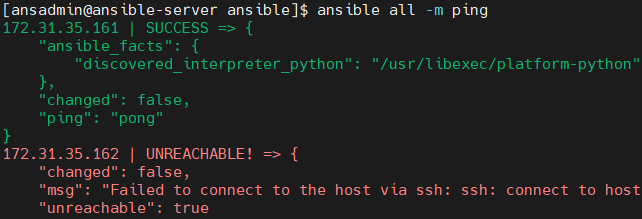


* Open hosts file and edit the script





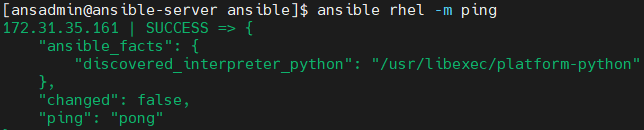
* Invoke a command **ansible all –m ping**
* As discussed earlier, Ansible looks for the hosts file and Private IP address of nodes stored in it and confirms the status of connectivity



**How to check connectivity of specific node/client instead of all at a time**

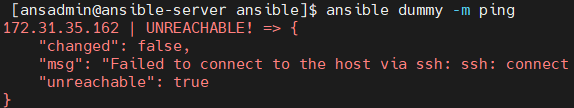
* When you specify All, it considers all IP address of nodes present in hosts file
* What if you want to check the connectivity of specific node

**Connectivity of rhel IP address**



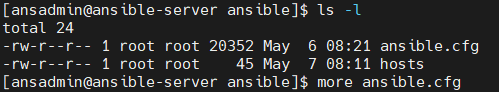
**Connectivity of dummy IP address**

* Since dummy IP address is not in operation, it is unreachable

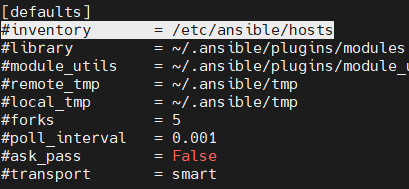


**Important:**

* As discussed earlier, whenever you invoke commands, ansible gathering up IP address of nodes from hosts file.
* It is because of the path specified in configuration file called ansible.cfg
* List out files located in server
* Invoke command, **more ansible.cfg**



* If you observe in the screenshot, /etc/ansible/hosts is the default path (Refer next page) specified as inventory in the file.
* Here, **Forks** represents max number of servers can perform deployment task parallel /at a time.



* Incase requirement is of more than 5 servers to be performed parallel, first it will process a group of first 5 servers and rest of them will be processed in group of five sequentially.
* If you observe in above screenshot, value of **ask\_pass** is false. This means, it will not ask for authentication details like password to enter every time you invoke a command.
* If value is true, password should be entered every time you invoke a command.

**Note:** In case you alter ansible.cfg file, behavior of anisble changes completely

**How to install GIT using ad-hoc command**

* Go to the ansible-rhel-node command prompt
* Login as root admin and remove GIT : **yum remove git**

****

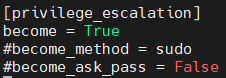
* Go back to ansible-server command prompt
* Please note that, git has to be installed under root. To do the same, specify –b at the end of command line.
* If you don’t, it will throw an error; saying “This command has to be run under the root user”



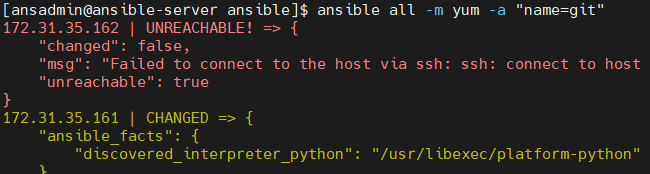
* So to install the git, you have to login as root user or else you may configure the same in ansible.cfg file.
* But you have to be more careful while editing/altering it. Otherwise behavior of ansible will change.
* Open ansible.cfg file and scroll down, search for privilege\_escalation



* Under privilege\_escalation, change the value of “become” from False to True and delete the # symbol mentioned at the beginning of line
* Save and exit.



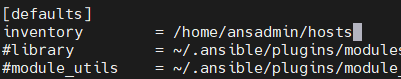
* Now run the same installation command under ansadmin without –b
* GIT will be installed successfully



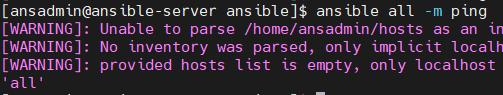
* You may check the same by passing a command git at node/client end.



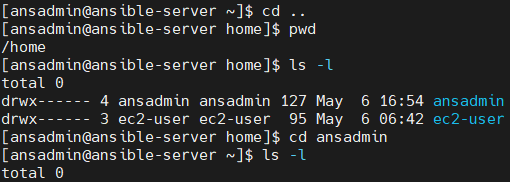
* What if I want to change the path of inventory file
* Open ansible.cfg file and change the path to /home/ansadmin/hosts and remove # symbol
* Save and exit



* Invoke ping command



* Since there is no hosts file present in /home/ansadmin, it couldn’t find/recognize inventory & thrown an error. You may list out the files located in home/ansadmin directory to confirm it



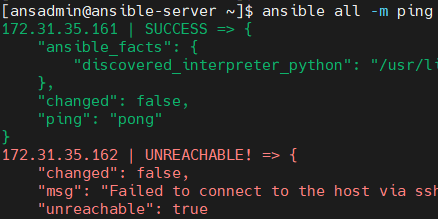
* Create hosts file in ansadmin



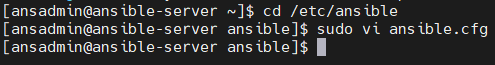
* Save the Private IP address of node as well as dummy IP address and exit



* Now invoke ping command once again.
* Since we have already created hosts file in /home/ansadmin, it will run successfully.



* Now go back to configuration file ansible.cfg and change the values (Privilege escalation and path of inventory file) which you have altered/edited earlier.



**Module**

* A module is a reusable, standalone script that Ansible runs on your behalf, either locally or remotely.
* Modules interact with your local machine, an API, or a remote system to perform specific tasks like changing a database password or spinning up a cloud instance.
* Ansible ships with a number of modules (called the 'module library') that can be executed directly on remote hosts or through Playbooks.
* These modules can control system resources, like services, packages, or files (anything really), or handle executing system commands.

Ansible is all about using modules in its Playbook. Before we go through list of modules, you need to know below terminologies:

1. **test-servers:** the group of my hosts which I have written in my [inventory file](https://gist.github.com/slathia15/6d830207e5dd6bcbfc48f5c9b7951d1a).
2. ***- m XYZ***: XYZ is the module name.
3. **-u ec2-user**: defines username that is ec2-user.
4. **SUCCESS:**means that my module has done its task on destination nodes.
5. **Changed:**If it is true, something has changed on the destination nodes. If it is false, something has not changed on the destination nodes.

**Let's start with the modules:**

* Ping Module

[Ping](https://gist.github.com/slathia15/450ecc43059ed73c75a74688cbdba35e) is used when we want to check whether the connection with our hosts defined in the inventory file is established or not.

**ansible test-servers -m ping -u ec2-user**

**ping**changes to **pong** if an SSH connection is established.

* Setup Module

The setup module is used when we want to see the information of all the hosts, their configuration, and detailed information.

**ansible test-servers -m setup -u ec2-user**

This is a [snapshot](https://gist.github.com/slathia15/7f6c81e8c60b02d9ac23f4ab9379f483) of the configuration of my machine running on AWS.

* Copy Module

The [copy module](https://gist.github.com/slathia15/65c164727aa572d1f913c9fcc8d63db3) is often used in writing playbooks when we want to copy a file from a remote server to destination nodes.

For example, suppose we want to copy a file from a remote server to all destination machines.

**ansible test-servers -m copy -a 'src=/home/knoldus/Personal/blogs/blog3.txt dest=/tmp' -u ec2-user**

* Yum Module

We use the [Yum](https://gist.github.com/slathia15/d228b0c2d638a45ce98555190680afb2) module to install a service.

**ansible test-servers -m yum -a 'name=httpd state=present' -become -u ec2-user**

Apache2 will be installed on our machines.

The key point to note here is that we have to use -become, which is new in version 2.6; before, we had to use -s.

* Shell Module\*

When we want to run UNIX commands then we use shell module

**ansible test-servers -m shell -a 'ls -la' -u ec2-user**

https://gist.github.com/slathia15/be3f84fa101ab39fb0d1969b8a99fe5d

This will display all the files present in our machine with their permissions.

* Service Module

When we want to ensure the state of a service that is service is running we use the service module.

**ansible test-servers -m service -a 'name=httpd state=started' -become -u ec2-user**

https://gist.github.com/slathia15/339cc8f6784bdec5037481f7dc225bbb

Apache2 is up on my machine.

* Debug Module

To print a msg on hosts we use Debug module.

**ansible test-servers -m debug -a 'msg=Hello' -u ec2-user**

https://gist.github.com/slathia15/d408ac54c5cc1cddbf07d6b14abcaa3b

Hello, a message is printed on my machine.

* Template Module

The Template module is used to copy a configuration file from the local system to the host server. It is the same as the copy module, but it dynamically binds group variables defined by us.

[Here](https://gist.github.com/slathia15/5890cfd642d49f6a6711acaae10ee905), I have vars in my source machine.

* Include Module

When we want to include another playbook in our playbook, then we use the [Include module](https://gist.github.com/slathia15/685fa46cbc6c85af5029617dfd7fbdb9).

* User Module

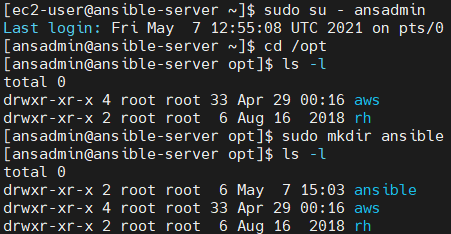
To add a particular user to our module we can use User module. [Here](https://gist.github.com/slathia15/a91dc8f67c31afc0978703674181ec40), we have added a user named Sachin to our module.

**Playbooks**

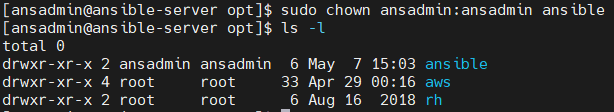
* Playbooks are the files where Ansible code is written.
* Playbooks are one of the core features of Ansible and tell Ansible what to execute.
* They are like a to-do list for Ansible that contains a list of tasks.
* Playbooks contain the steps which the user wants to execute on a particular machine.
* Playbooks are written in the YAML format and have a .yml file extension. Use this command to run a playbook: **$ ansible-playbook <playbook**.
* Use the ansible-playbook command to run the sample-playbook. yml file. Use the optional argument -i to point to the inventory file. If the -i option is not used, and there is no ansible
* The playbook contains 2 plays: The first checks whether or not web server software is up to date and runs the update if necessary. The second checks whether or not database server software is up to date and runs the update if necessary.

**How to create user at node/client using playbook**

* Login as ansadmin in server
* Navigate to directory “opt” which is basically an operational file
* Create directory called ansible in opt



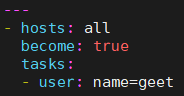
* If you observe in above screenshot, the permission granted under “root” for ansible directory
* Change it to ansadmin by invoking below mentioned command



* Navigate to ansible directory and create a .yml file called create-user



* Here task specified in the script is to create user at node end.



* Save and exit

**Let’s understand structure of YAML format**

**Rules for Creating YAML file:**

When you are creating a file in YAML, you should remember the following basic rules −

* YAML is case sensitive
* The files should have **.yaml** as the extension
* YAML does not allow the use of tabs while creating YAML files; spaces are allowed instead

**Basic Components of YAML File:**

The basic components of YAML are described below −

* **Conventional Block Format**

This block format uses **hyphen+space** to begin a new item in a specified list. Observe the example shown below

--- # Favorite movies

- Casablanca

- North by Northwest

- The Man Who Wasn't There

* **Inline Format**

Inline format is delimited with **comma and space** and the items are enclosed in JSON. Observe the example shown below −

--- # Shopping list

[milk, groceries, eggs, juice, fruits]

* **Folded Text**

Folded text converts newlines to spaces and removes the leading whitespace. Observe the example shown below −

- {name: John Smith, age: 33}

- name: Mary Smith

age: 27

The structure which follows all the basic conventions of YAML is shown below −

men: [John Smith, Bill Jones]

women:

- Mary Smith

- Susan Williams

**Synopsis of YAML Basic Elements:**

1. The synopsis of YAML basic elements is given here: Comments in YAML begins with the (**#**) character.
2. Comments must be separated from other tokens by whitespaces.
3. Indentation of whitespace is used to denote structure.
4. Tabs are not included as indentation for YAML files.
5. List members are denoted by a leading hyphen (**-**).
6. List members are enclosed in square brackets and separated by commas.
7. Associative arrays are represented using colon **(:)** in the format of key value pair. They are enclosed in curly braces **{}**.
8. Multiple documents with single streams are separated with 3 hyphens (---).
9. Repeated nodes in each file are initially denoted by an ampersand (**&**) and by an asterisk (**\***) mark later.
10. YAML always requires colons and commas used as list separators followed by space with scalar values.
11. Nodes should be labeled with an exclamation mark (**!**) or double exclamation mark (**!!**), followed by string which can be expanded into an URI or URL.

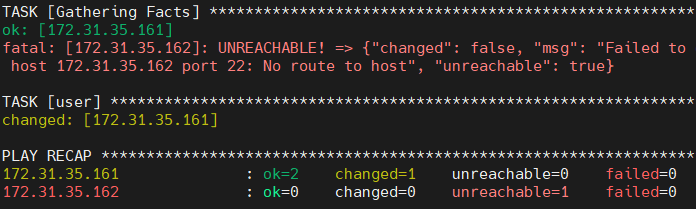
* Now open ansible-rhel-node command prompt and let’s check who was the last user





* Comeback to ansible-server and execute create-user.yml playbook





* Now open ansible-rhel-node command prompt and once again check who is the last user





* If you observe in above screenshot, geet is the last user which we have created.

**How to create user at multiple node/client using playbook**

* Open AWS management; create another node with Ubuntu Server 20.04 LTS image and launch.
* Rename instance as ansible-ubuntu-node
* Open MobaXterm and start new SSH session with Public IP address (13.232.182.184) of ansible-ubuntu-node
* Login as root and open a file hostname



* Change the host name, save and exit.

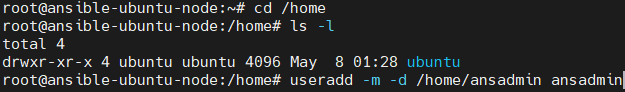


* Now check whether hostname has changed or not

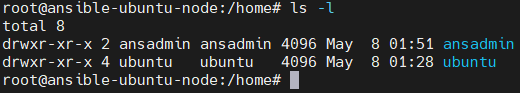


* In ubuntu, there won’t be any home directory created for this particular user/ansadmin.
* So first, navigate to ubuntu home directory and create home directory for user by invoking below mentioned command

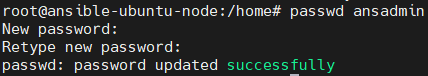
**Command:** useradd –m –d /home/ansadmin ansadmin



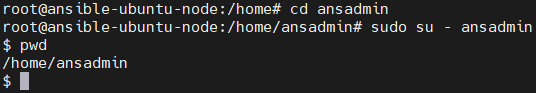
* Now list out files/directories located in ubuntu home directory.
* If you observe, new directory by the name ansadmin is created under ubuntu home directory



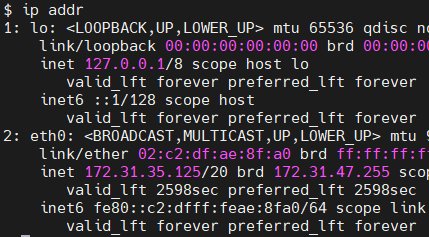
* Set the password: ansadmin



* Navigate to ansadmin



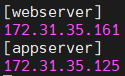
* Check Private IP address of ansible-ubuntu-node (172.31.45.137)



* Now I have to add this Private IP address (172.31.45.137) to ansible-server/hosts file
* Now come back to ansible-server command prompt
* Open hosts file



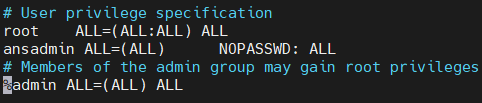
* Specify only red hat and ubuntu node’s Private IP addresses. Save and exit.



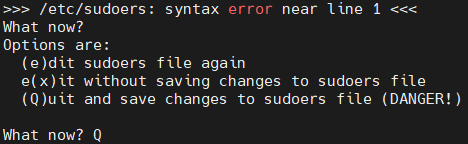
* Go to ansible-ubuntu-node and open duplicate tab/window, login as root and invoke a command visudo
* The visudo command is a safe and secure way of editing the /etc/sudoers file on UNIX and Linux systems



* Under user privilege specification, grant permission for ansadmin to execute commands without asking for password each time



* Type control +x and press Enter to exit.

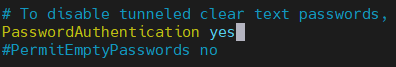


**Note**: In case you get an error like mentioned in above screenshot, select an option “Q” to quit and save changes to etc/sudoers file.

* Open a file /etc/ssh/sshd\_config



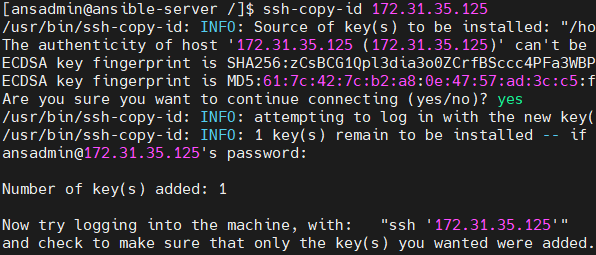
* Change value of PasswordAuthentication from no to yes, Save and exit



* Invoke service sshd reload command.



* Copy your SSH key/Private IP address of node to server to enable a keyed or password less login



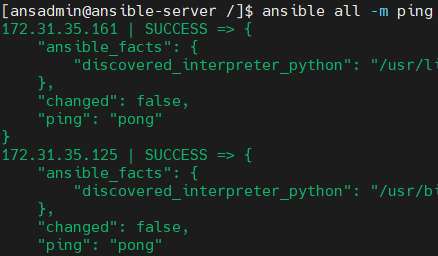
* 1. **service sshd reload**
* Only works when sshd is already running
* Stops sshd if the sshd\_config file has problems
* Returns error code 0 even if the sshd\_config file has problems
  1. **service sshd restart**
* Works regardless of whether sshd is already running
* Stops sshd if the sshd\_config file has invalid syntax or other problems
* Returns non-zero error code if the sshd\_config file has problems
* To have access over ansible-ubuntu-node though ansible-server, invoke command ssh Private IP address



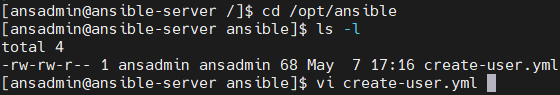
* To confirm the accessibility, just pass command hostname



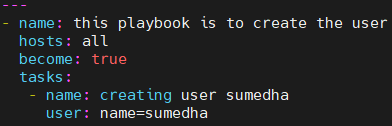
* To check the connectivity between ansible-server and red hat/ubuntu nodes invoke below mentioned.
* When you specify All, it considers all IP address of nodes present in hosts file



* Navigate to the/opt/ansible/create-user.yml file



* Open file create-user.yml file and specify some other username

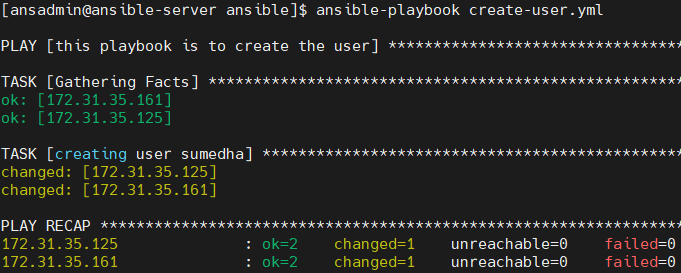


* Before you run the program, go to ansible-ubuntu-node and check who last user is.





* Comeback to ansible-server and run create-user.yml playbook



* Now go to the ansible-ubuntu-node as well as in ansible-rhel-node and check that who is the last user.







* If you observe the output screenshot of create-user.yml playbook, its human readable.

**PLAY** – This playbook is to create the user

**Task** – Creating user Sumedha

* Gathering facts is default tasks being performed by ansible playbook from configuration file.
* Facts are nothing but the system configuration which includes the hostname, IP address, file systems, OS releases, Users, Network parameters, CPU, memory and many more.
* Got to ansible-rhel-node and check whether GIT or Tree is installed.
* If it is, uninstall both by invoking a command: **yum remove git/tree**
* Got to ansible-ubuntu -node and check whether GIT or Tree is installed.
* If it is, uninstall both by invoking a command: **apt remove git/tree**
* Let’s create another node/client
* Go to AWS manager and launch a new instance with free tier Amazon Linux 2 AMI
* Once its launched, rename instance as ansible-Linux -node
* Go to MobaXterm and create new SSH session and specify public IP address (13.233.59.7) of ansible-Linux -node as remote hosts and user name as ec2-user.
* Now login as root (Sudo -i)
* As usual rename the hostname
* Open /etc/hostname file and specify node name in it.



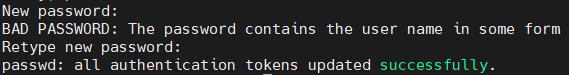


* Save and exit
* Invoke below mentioned commands in the screenshot to rename hostname



* Now create ansadmin user
* Add useradd and passwd as ansadmin



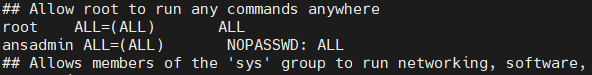


* Invoke command visudo
* The visudo command is a safe and secure way of editing the /etc/sudoers file on UNIX and Linux systems



* Grant permission to ansadmin to run any commands anywhere, save and exit.

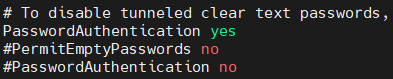
Ansadmin ALL=(ALL) NOPASSWD: ALL



* Open file vi /etc/ssh/sshd\_config



* Specify value of PasswordAuthentication as yes and delete “#” symbol present at the beginning of line. Save and exit



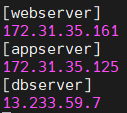
* Now to apply the same, I have to restart the service and.



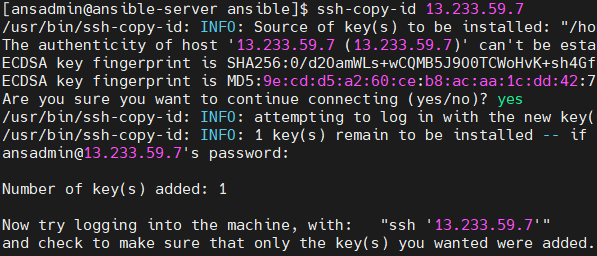
* Copy the Private IP address (13.233.59.7) of ansible-Linux -node.
* Go to Ansible-server command prompt
* Open /etc/ansible/hosts



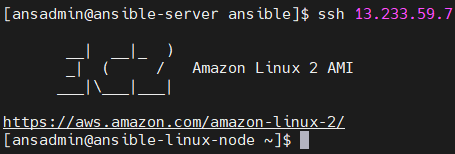
* Add Private IP address (13.233.59.7) of ansible-Linux -node as dbserver.



* Save and exit.
* Invoke a command : ssh-copy-id Private IP Address of Linux node (13.233.59.7)
* Enter password: ansadmin, press Enter



* If you observe, I am in ansible-server. Once you invoke command ssh Private IP Address of node, you will enter in ansible-rhel-node



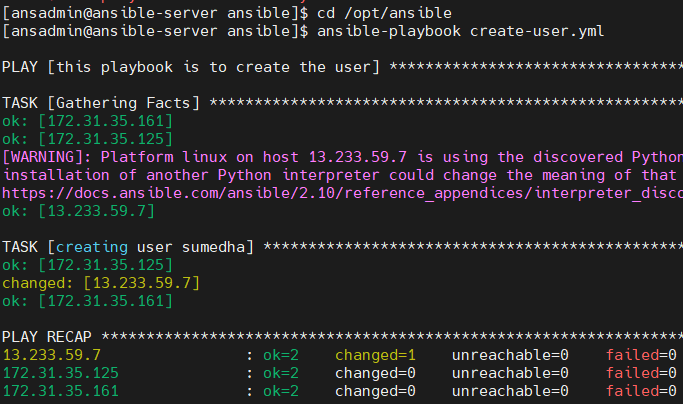
**Note**: It won’t ask you for authentication details password anymore.

* Now exit
* Before you add a user, go to ansible-Linux -node command prompt and check who the last user is.





* Now navigate to /opt/ansible and execute create-user.yml



* Now go to Ansible-Linux -node command prompt and check who the last user is.



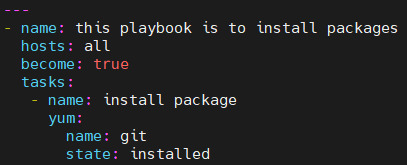


**How to install a package:**

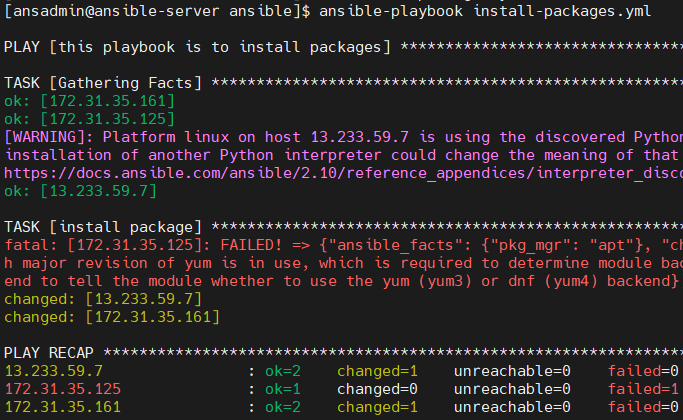
* First ensure git is not installed on all nodes specified in hosts file. (**yum remove git/apt remove** **git** (ubuntu))
* Go to ansible-server, navigate to /opt/ansible
* Create playbook called install-packages.yml



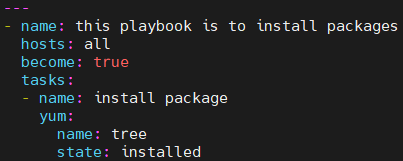
* Write playbook, save and exit. Then execute the playbook : ansible-playbook install-packages.yml



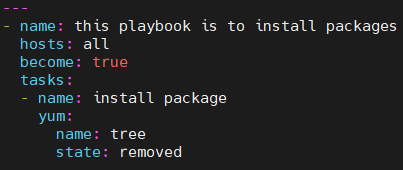
* Since yum install package works only with Red hat Linux , it failed for ubuntu node.
* For ubuntu, apt install package is preferred.



* To check the status of installation of git package in nodes, invoke command **git** in respective command prompts.
* In case you want to install some other package, specify the package name in install-packages.yml playbook, save and exit.



* Execute the install-packages.yml playbook
* Tree package will be successfully installed on Linux nodes. (ansible-rhel-node and ansible-Linux -node)
* To check the status of installation of tree package in nodes, invoke command **tree** in respective command prompts.
* As shown in below screenshot, you may write playbook to remove packages as well.



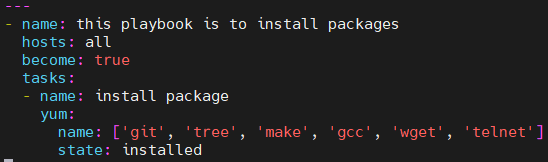
* According to you requirement, you may choose package and task to be executed.

**Note**:

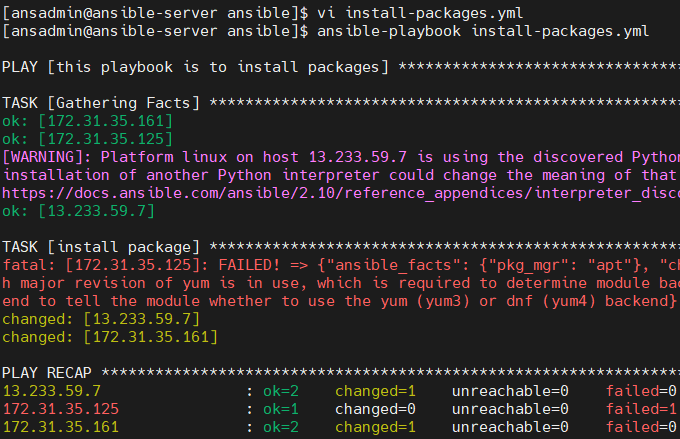
* **yum** is the primary tool for getting, installing, deleting, querying, and managing Red Hat Enterprise Linux  RPM software packages from official Red Hat software repositories, as well as other third-party repositories. yum is used in Red Hat Enterprise Linux  versions 5 and later
* **apt** is preferred for getting, installing, deleting, querying, and managing Debian and Debian based Linux distributions like Ubuntu

**How to install multiple numbers of packages:**

* Go to ansible-server
* Navigate to /opt/ansible
* Write script to install multiple numbers of packages



* Execute the playbook : ansible-playbook install-packages.yml

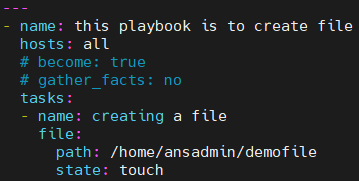


**How to create a file:**

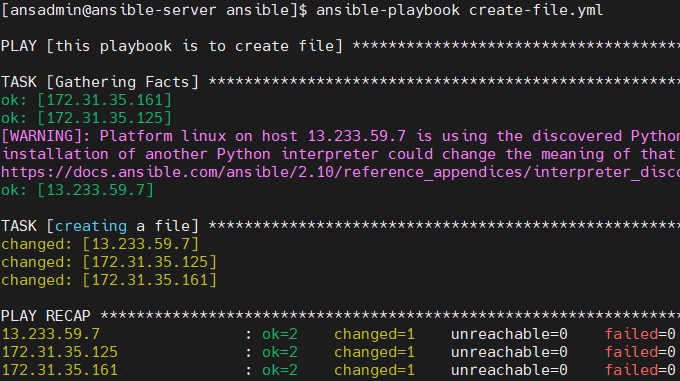
* Go to ansible-server
* Navigate to /opt/ansible
* Create a playbook.



* Task is to create a file

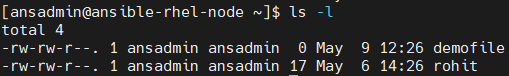


* Save, exit and execute the playbook



* Now go to all nodes and check whether file is created in all of nodes or not.

1. **ansible-rhel-node**

****

1. **ansible-ubuntu-node**

****

1. **ansible-Linux -node**

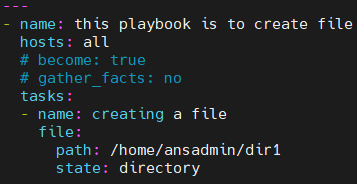


**How to create a directory:**

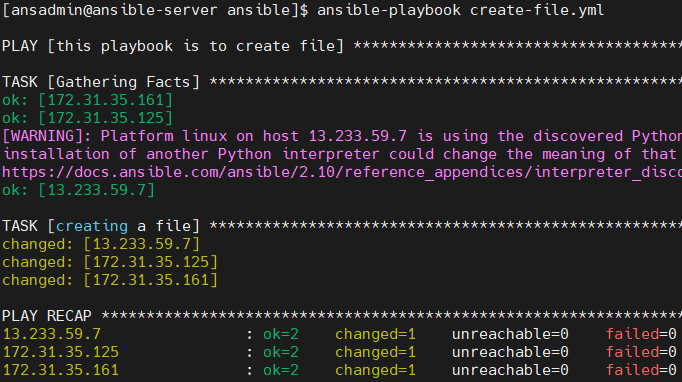
* Open same playbook, create-user.yml



* Edit the playbook to create directory

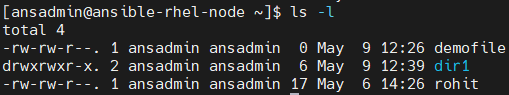


* Save, exit and run the playbook.

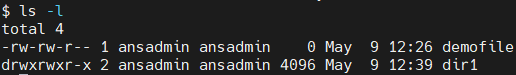


* Now visit all nodes and check whether file is created in all of nodes or not.

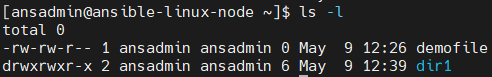
1. **ansible-rhel-node**

****

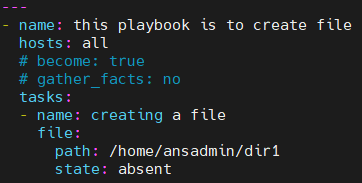
1. **ansible-ubuntu-node**

****

1. **ansible-Linux -node**

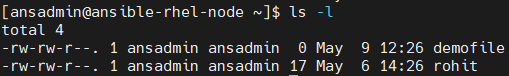


* In case, you want to remove the directory specify “State =absent”



* If you visit all nodes and dir1 will be removed from nodes successfully.

1. **ansible-rhel-node**

****

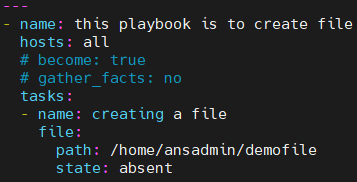
1. **ansible-ubuntu-node**

****

1. **ansible-Linux -node**

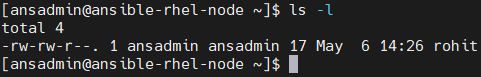


* In same way, file can also be removed.



* If you visit all nodes and demofile will be removed from nodes successfully.

1. **ansible-rhel-node**

****

1. **ansible-ubuntu-node**

****

1. **ansible-Linux -node**

****

**How to copy html file from server to all nodes:**

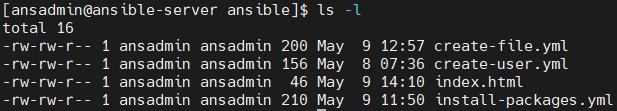
* Go to ansible-server
* Navigate to /opt/ansible and create html file



* Write script to display below mentioned content in the screen shot.



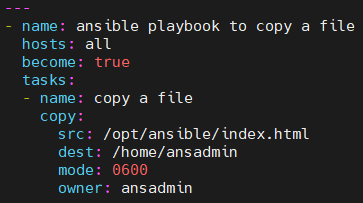
* Check whether file is created successfully or not.



* Now create a playbook to copy html file



* Here task is to copy a file

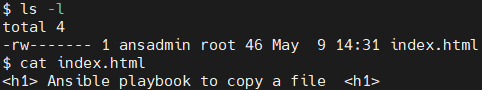


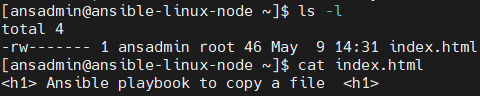
* Before you execute playbook copy-file.yml, list out files located in directory home/ansadmin of all nodes
* Ensure that, no file is copied yet
* Execute playbook copy-file.yml



* If you visit all nodes and index.html will be copied to nodes successfully and you may check the content of index file as well



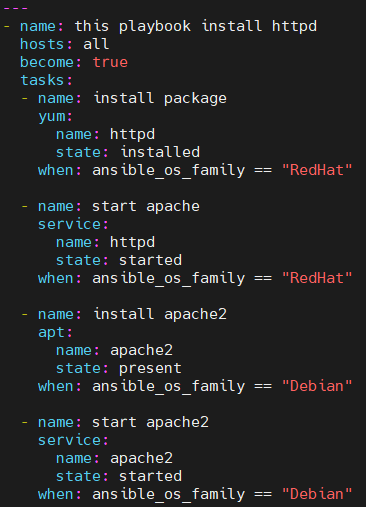




**How to install packages on different operating system using “When ” condition:**

* Let’s enable ports of all nodes created first and install apache package and check whether I will able to open that in port or not.
* Go back to AWS management, select instances one by one.
* Click on security => security group link
* Click on “Edit inbound rules”
* Click on Add
* Click on drop down icon next to Custom TCP and select “All Traffic”
* Click on drop down icon next to Custom and select “Anywhere”
* Scroll down to the page and apply Save rules
* Write a playbook to install packages on different operating system using “When ” condition



****

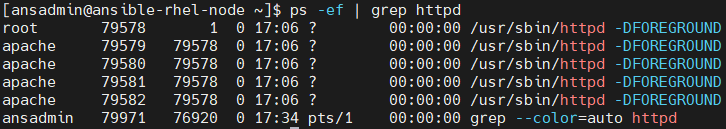
* Save, exit and run the playbook

****

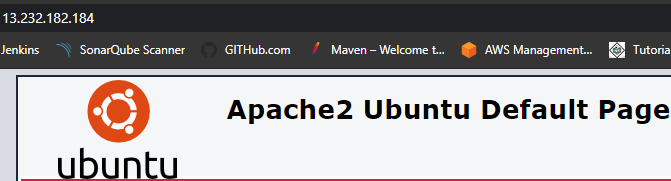
* Once playbook is run successfully, open browser, copy and paste public IP address of each nodes in text box of search Google. Press Enter.
* You may check current processes running in system, by invoking ps –ef | grep httpd (Linux )/apche2(ubuntu)

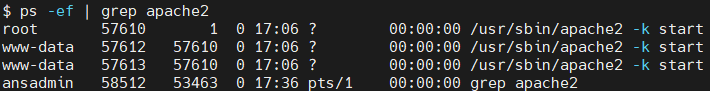
1. **ansible-rhel-node**

****

****

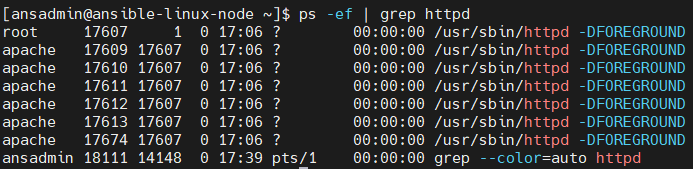
1. **ansible-ubuntu-node**

****

****

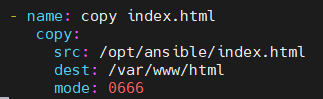
1. **ansible-Linux -node**



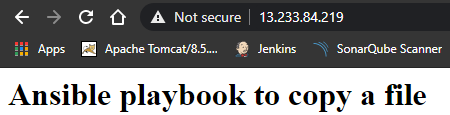


**How to push html file to all nodes and display string/content specified in a file:**

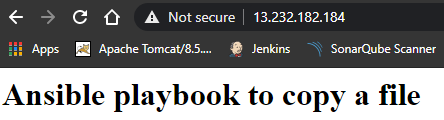
* Open same playbook : **vi install-apache-httpd.yml**
* Scroll down to end of the program and include below mentioned 5 lines of commands.



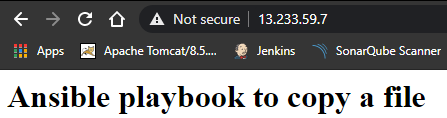
* Save, exit and run the playbook
* Once playbook is run successfully, open browser, copy and paste public IP address of each nodes in text box of search Google. Press Enter.
* String/content mentioned in index.html file will be displayed
  1. **ansible-rhel-node**

****

* 1. **ansible-ubuntu-node**

****

* 1. **ansible-Linux -node**

****

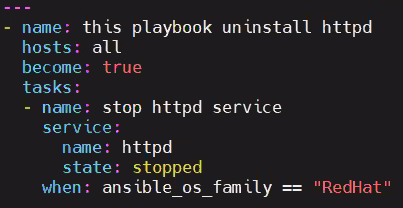
**Note:** Since my AWS account is deleted, below mentioned all screenshots are of sessions conducted by Keshav sir not mine. IP addresses mentioned in below screenshots are different from above ones.

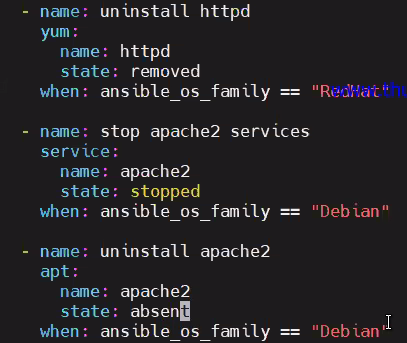
**How to uninstall packages on different operating system:**

* Go to ansible-server and login as ansadmin
* Create a playbook uninstall-apache.yml



* Write playbook to uninstall packages. (Below mentioned two screenshots are of single/same program)



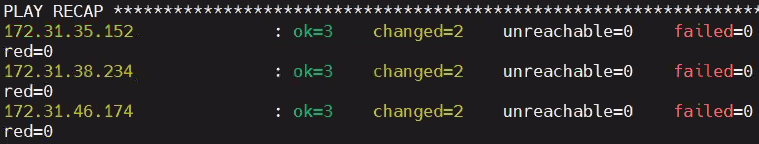


* Save and exit.
* Once ensure packages are installed.

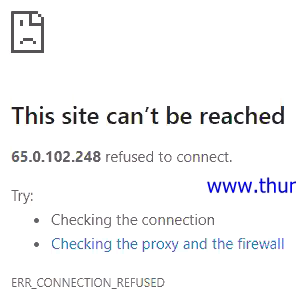


* Now execute uninstall-apache.yml





* Open browser, copy and paste public IP address of each nodes (rhel/ubuntu/Linux ) in text box of search Google. Press Enter.
* Since packages are uninstalled now, you will be denied access to above mentioned IPv4 address and refused to connect.



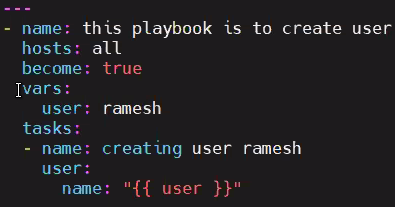
**Ansible Variables:**

* Ansible uses variables to manage differences between systems.
* You can define these variables in your playbooks, in your inventory, in re-usable files or roles, or at the command line.
* You can also create variables during a playbook run by registering the return value or values of a task as a new variable.

**How to create user using variable:**

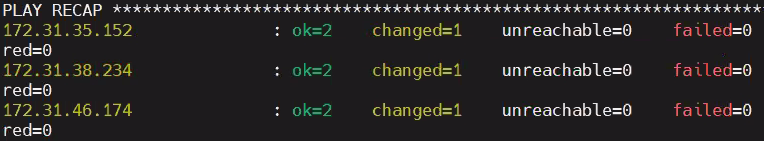
* Create a playbook called create\_file.yml





* Save and exit.
* Before you execute playbook, go to each nodes (rhel/ubuntu/Linux ) and to check who the last user is, invoke **command**: cat /etc/passwd





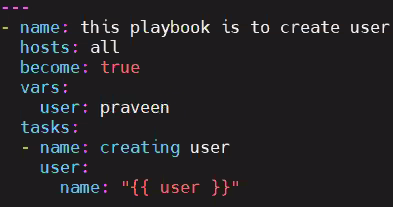
* Once playbook is run successfully, go back to nodes and invoke /etc/passwd command once again.
  + **ansible-rhel-node**

****

* + **ansible-Linux -node**



* Now we will change value of user name in the playbook and check once.



* Save, exit and run the playbook once again.



* Go back to nodes and invoke /etc/passwd command once again. If you observe, last user will be changed to Praveen accordingly.

**How to pass value of variable from one file to other (Task of primary file is to create user):**

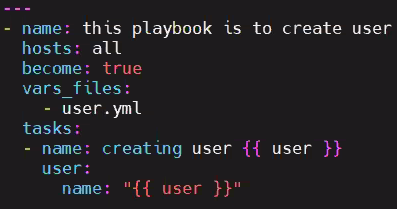
* Create a playbook user.yml and specify the value of variable to be passed.





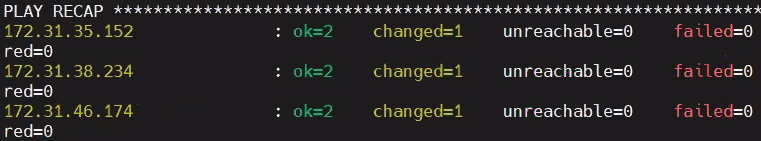
* Open and edit same previous playbook create\_user.yml as mentioned below.





* Save, exit and run the playbook





* Go back to nodes (rhel/ubuntu/Linux ) and invoke /etc/passwd command once again. If you observe, last user will be changed to Rohit accordingly.

.







**How to pass value of variable to the script in the command line (Task of primary file is to create user):**

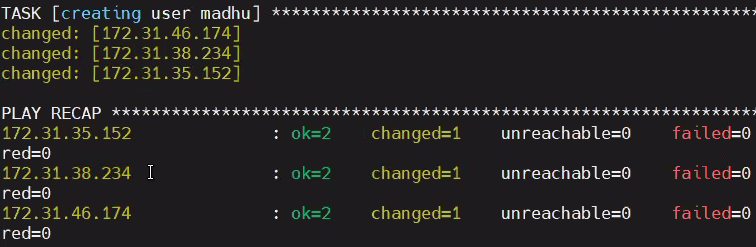
Using ansible ‘**– – exra-vars’** tag or ‘**-e**‘ tag we can pass the ansible variable to ansible playbook in the command line. In this post i will explain how to pass variable to ansible playbook with examples.

**Syntax:**

1. ansible-playbook --exra-vars "var1=value  var2=value .........."
2. ansible-playbook -e "var1=value  var2=value .........."

* Invoke a below mentioned command

****

****

* Go back to nodes (rhel/ubuntu/Linux ) and invoke /etc/passwd command once again. If you observe, last user will be changed to Madhu accordingly.
* The other way to pass value of variable to the script in the command line is

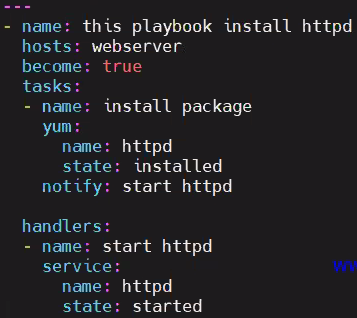


**Handlers**:

* Sometimes you want a task to run only when a change is made on a machine.
* For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged.
* Ansible uses handlers to address this use case. Handlers are tasks that only run when notified.
* Each handler should have a globally unique name.
* A *Handler* is exactly the same as a Task, but it will run when called by another Task. A *Handler* will take an action when called by an event it listens for.

**How to notify handler to start httpd once package is installed**

* Create playbook handlers.yml
* Write a script to notify handler to start httpd once package is installed

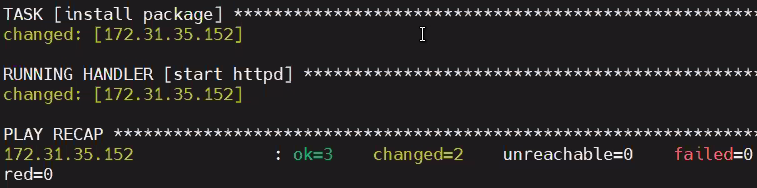


**Note**: Ensure httpd service is uninstalled on selected instance (hosts: webserver) in the program.

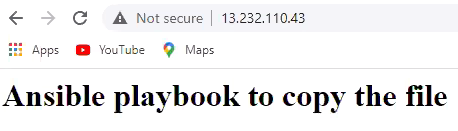
* Save, exit and run the playbook



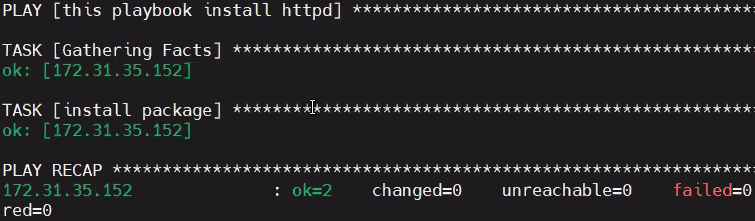
* If you observe in a screenshot (Refer next Page), once package is installed, handler is being notified to start httpd.



* Now go to respective host: webserver node in browser and refresh the page.
* httpd connection will be established



* In case you execute the same playbook once again, it will not start httpd service.
* Since httpd service is already running on node/client, it will skip command to “start hhtpd “
* Please look into difference between the output of previous execution and the current one (below screenshot).
* Post installation of package “Start httpd” is skipped now.



**Gathering Facts:**

* Ansible facts are data gathered about target nodes (host nodes to be configured) and returned back to controller nodes.
* Ansible facts are stored in JSON format and are used to make important decisions about tasks based on their statistics.
* Facts are in an ansible\_facts variable, which is managed by Ansible Engine

**How do you turn off Ansible gathering facts?**

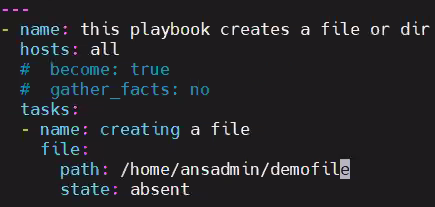
* To stop gathering facts, we need to disable it with “gather\_facts” attribute in ansible playbook.
* By default gather\_facts attributes value is yes, to disable it we need to set it as no.
* After updating gather\_facts: False in playbook if we execute, it will skip collecting facts and directly execute tasks listed

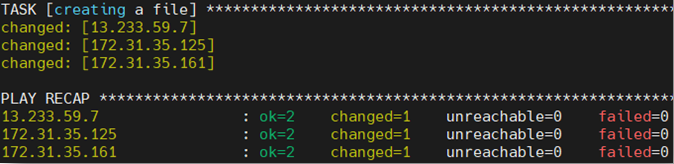
**Example:**

* Open create-file.yml and just to confirm, whether it gathers facts or not, execute the same as it is.

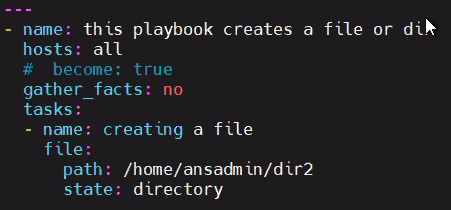
**Note**:

* Primary task of the playbook is to create file/directory
* “Gather\_facts: no” is commented.

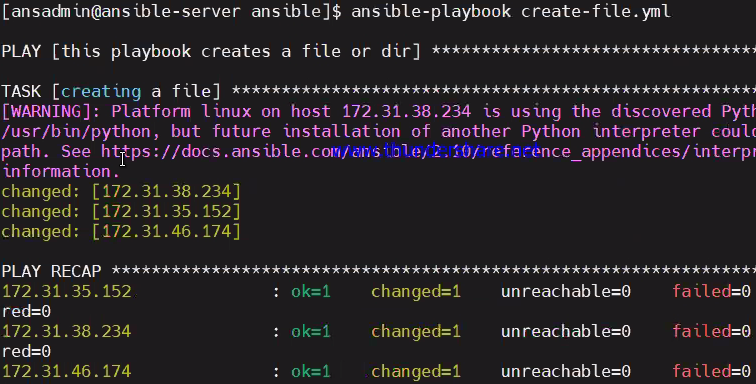
****

****

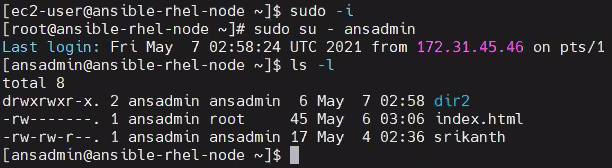
* Open the same file
* Primary task of the playbook is to create directory
* To execute commented line, gather\_facts: no; delete “#” symbol
* And change the path of directory.

****

* Execute the playbook.



* If you observe in the above screenshot, it has stopped gathering facts.
* Now go to all nodes and check whether directory (dir2) is created in all of nodes or not. (Refer next page)



**Ansible Tags:**

* A tag is an attribute that you can set to an Ansible structure (plays, roles, tasks), and then when you run a playbook you can use –tags or –skip-tags to execute a subset of tasks.
* Ansible runs or skips all tasks with tags that match the tags you pass at the command line.
* If you have a large playbook it may become useful to be able to run a specific part of the configuration without running the whole playbook.
* Both plays and tasks support a “tags:” attribute for this reason.
* If you wanted to just run the “configuration” and “packages” part of a very long playbook, you could do this:

**ansible-playbook example.yml --tags "configuration,packages"**

* On the other hand, if you want to run a playbook *without* certain tasks, you could do this:

**ansible-playbook example.yml --skip-tags "notification"**

* You may also apply tags to roles:

**roles: - { role: webserver, port: 5000, tags: [ 'web', 'foo' ] }**

* And you may also tag basic include statements:

**- include: foo.yml tags=web,foo**

* Both of these apply the specified tags to every task inside the included file or role, so that these tasks can be selectively run when the playbook is invoked with the corresponding tags.

**How to execute specific task without running the whole playbook:**

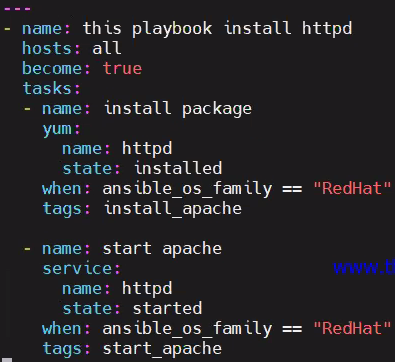
* First uninstall all packages.

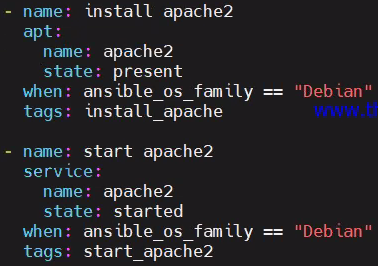


* Just to ensure whether packages are uninstalled or not, go to respective public IP address of nodes in browser and refresh the page.
* Open playbook, install-apache-httpd.yml and specify tags.



**Note**: Below mentioned two screenshots are of single/same program

****

****

* Save, exit and run the playbook.

**Command**: ansible-playbook playbook\_name.yml --tags “tag name (install\_apche)”



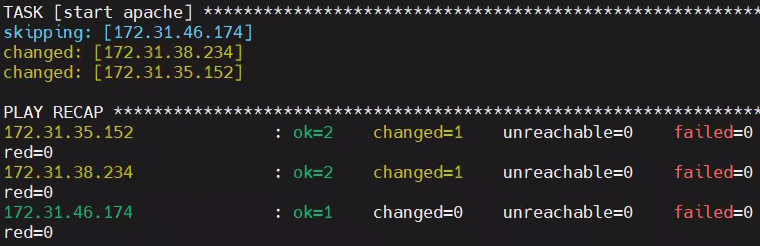
* If you observe in below screen shot, only specified tag “install\_apache” is executed.



* In case, you want to execute only “start\_apache” ,invoke below mentioned command.

**Command**: ansible-playbook playbook\_name.yml --tags “tag name (start\_apche)”





* Now go to respective public IP address nodes in browser and refresh the page.
* httpd service will be established successfully.



**How to skip certain tasks to ignore an error:**

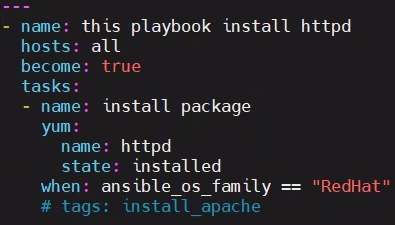
* First uninstall all installed packages.

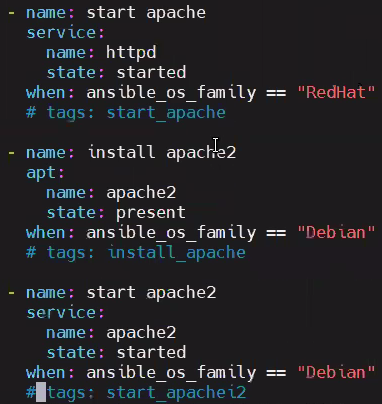


* Just to ensure whether packages are uninstalled or not, go to respective public IP address nodes in browser and refresh the page.
* Open playbook, install-apache-httpd.yml and comment (#) all tags.

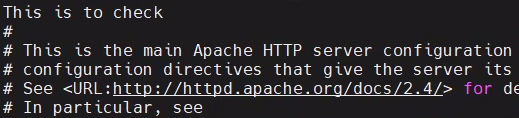


**Note**: Below mentioned two screenshots are of single/same program.

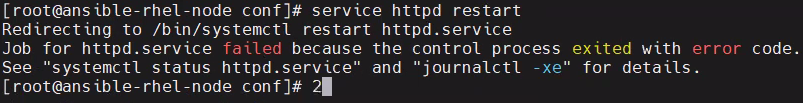
****

****

* Go to the ansible-rhel-node and login as root.
* Edit a file called /etc/httpd/conf/httpd.conf
* Invoke a command vi httpd.conf
* Scroll up to the first line of file and type something above the first line.



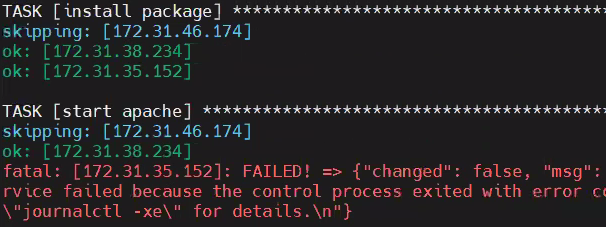
* Save and exit.
* Restart httpd service to apply the changes.



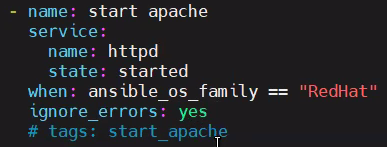
* Now execute playbook install-apache-httpd.yml to install packages back.



* If you observe, we got an error “Unable to start service httpd” at task called “start\_apache”



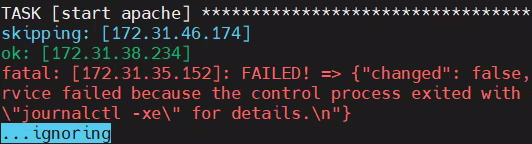
* Due to such errors, sometimes server will interrupt/skip other steps and stops a playbook from functioning given task.
* In such scenario, if you want ignore certain error and execute rest of tasks; go to specific section of script (start\_apache) where you have got an error and include command line ignore\_errors: yes



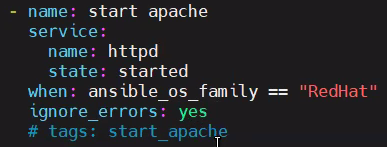
* Save and exit.
* Now execute playbook install-apache-httpd.yml to install packages.



* If you observe in below screen shot, start\_apache task is being ignored.



* Just to ensure whether playbook functioning properly or not, go to respective public IP address all nodes in browser and refresh the page.
* Since, we have skipped only ansible-rhel-node (RedHat) where we got an error, httpd service of RedHat node will be refused to connect,
* For other two nodes, httpd service will be established successfully.





* Go to the ansible-rhel-node and login as root.
* Modify a file /etc/httpd/conf/httpd.conf as it was earlier.
* Now if you try to install packages, it will be installed successfully without any error.



**Ansible Vault:**

* Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles.
* These vault files can then be distributed or placed in source control.
* It can encrypt entire files, entire YAML playbooks or even a few variables.
* It provides a facility where you can not only encrypt sensitive data but also integrate them into your playbooks.
* This provides the ability to secure any sensitive data that is necessary to successfully run Ansible plays but should not be publicly visible, like passwords or private keys.
* Open ansible-server
* Navigate to opt/ansible



* Create new vault in /opt/ansible.

**Command**: ansible-vault create vault-pass.yml

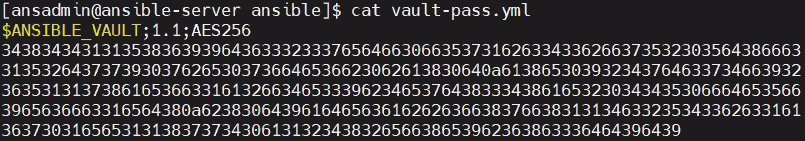
* Create new vault password (Any Password is OK), re-enter the same to confirm.



* Save data you want to encrypt.



* Save, exit.
* Once you save and exit, data will be encrypted.
* If you want to check the encrypted data of a file, invoke a command: cat vault-pass.yml



* To view content the file, invoke below mentioned command and enter vault password

**Command**: ansible-vault view vault-pass.yml

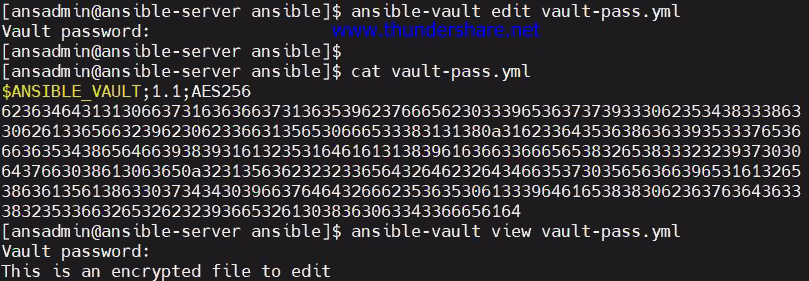
* Once vault password is entered, you will be able to view the content.



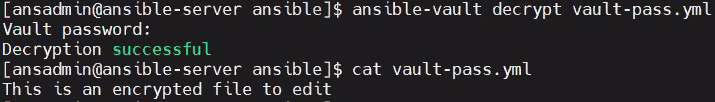
* In case you want to edit the file (vault-pass.yml), specify “edit” instead of “view” in the command line

**Command**: ansible-vault edit vault-pass.yml

* Enter vault password.
* Edit the file, save and exit.
* You may again view content of the file by invoking “view” command.



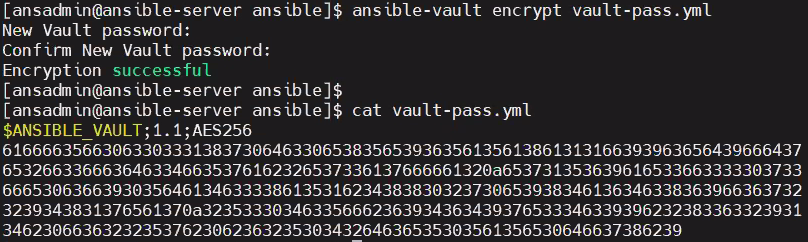
* To decrypt the file, invoke below mentioned command and enter vault password.
* File (vault-pass.yml) will be decrypted.
* Invoke a command cat vault-pass.yml to check the same.



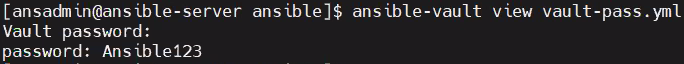
* Now open the same file vault-pass.yml and edit.
* Remove previous saved data and specify some password



* Save and exit.
* Encrypt a file, enter vault password.



* To view the password saved in vault-pass.yml, invoke view command.

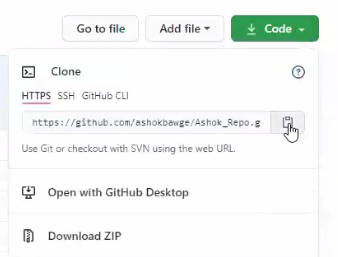


**Example**: Let’s say I have GIT Hub account and the requirement is whenever I clone a repository. I should pass encrypted username and password.

* First, ensure that Git is installed on ansible-server as well as on nodes
* Login to GIT Hub account
* Create new repository “Ashok\_Repo” and choose “Private” and “README file” options as show in below screenshot



* Click on OK
* Copy Web URL of repository



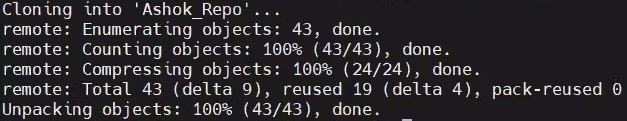
* Come back to the ansible-server’s command prompt
* Login as ansadmin



* Invoke below mentioned command. Here, I have to clone the repository.

**Syntax**: git clone https://username:password@ Web URL of repository

**Command:** git clone https://ashokbawge:Ashokkury96@github.com/ashokbawge/Ashok\_Repo.git



* Once cloning is done, directory by the name Ashok\_Repo will be created under ansadmin



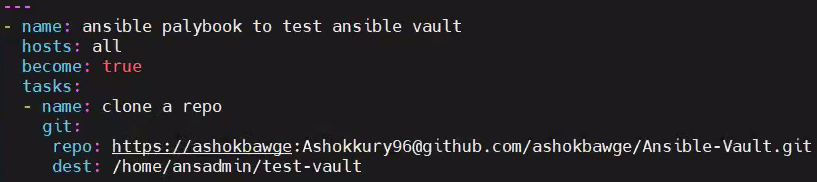
* Now create vault file “vault-pass.yml”



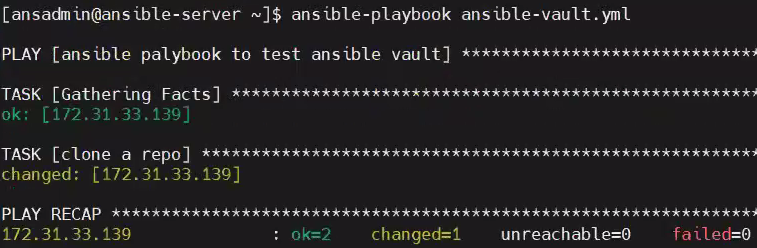
* Specify GIT account’s password, save and exit.



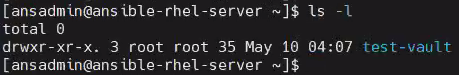
* Now go to GIT Hub, Create new repository “Ansible-Vault” and choose “Private” and “README file” options as show in below screenshot.
* Copy Web URL of repository from GIT Hub
* Come back to ansible-server; create playbook ansible-vault.yml and paste the copied URL as path of repo.
* And also specify the destination path for vault.



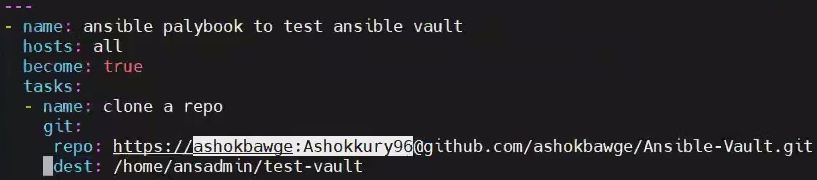
* Save and exit
* Execute the playbook, ansible-vault.yml



* Go to client node, check whether repository will be cloned or not
* Login as ansadmin and list out files contained in it
* test-vault directory is cloned successfully.



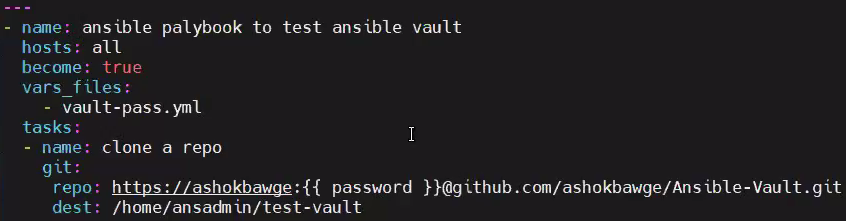
* Go back to ansible-server
* Open ansible-vault.yml
* If you observe, anyone can read/access authentication details, username and password.



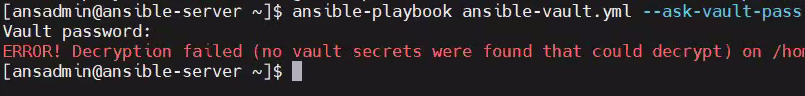
* In order to conceal such sensitive data, we will pass value of variable password from a playbook vault-pass.yml

**Note**: Password of repository saved in playbook vault-pass.yml

* Open ansible-vault.yml.
* Specify file and variable name which is to be passed.



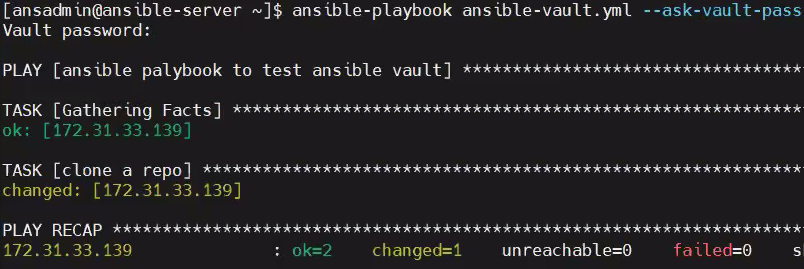
* Before we run the script, go to client node(ansible-rhel-node) and delete directory test-vault (rm -rf test-vault)
* Come back to ansible-server
* Since, we have not specified a password in the script, it will ask for password before we execute the playbook
* Now run the script



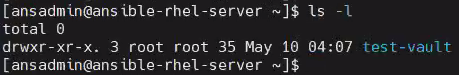
* If you observe, you will get an error saying “Decryption failed”
* Since “vault-pass.yml” is encrypted (Refer page 440), it couldn’t access the password of repository.
* To allow you to decrypt and use the file, you have to save password of vault in a specific file.
* Create file called vi pass.yml



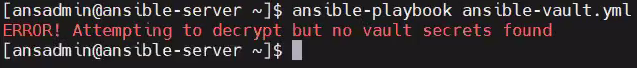
* Save and exit
* Now run the script once again



* Go to client node, check whether repository will be cloned or not
* Login as ansadmin and list out files contained in it
* test-vault directory is cloned successfully



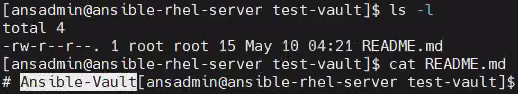
* Now delete directory test-vault (rm -rf test-vault) from the node ansible-rhel-node
* Let us see what happens, in case you do not specify “--ask-valut-pass” in the command line



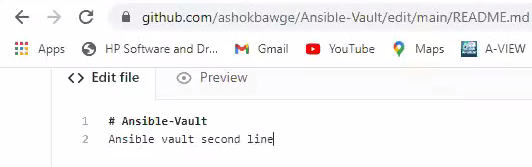
* This time, it’s attempting to decrypt but couldn’t find playbook pass.yml
* The other way to execute the script is:



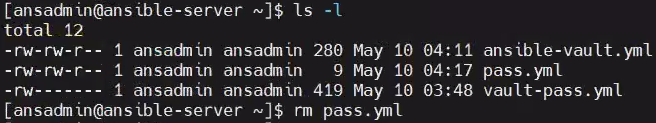
* Since we have passed file name (pass.yml) in the command line, it won’t ask you for the vault password.
* If you go to ansible-rhel-node, test-vault directory is cloned again successfully
* Navigate to test-vault file and list out files contained in it.
* Open README file



* Go to GIT Hub, edit README file and commit changes



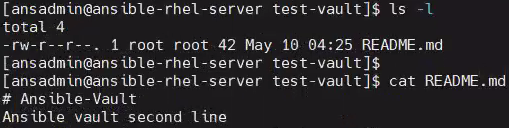
* Come back to ansible-rhel-node.
* Navigate back (cd ..) to test-vault and delete it.
* Go to ansible-server
* Before you execute the script, delete file pass.yml



* Now execute the playbook ansible-vault.yml
* This time, it will ask for you the password.
* Enter vault password and it will run successfully



* Now go to ansible-rhel-node, navigate to test-vault file and list out files contained in it.
* If you observe, changes made in README file in GIT Hub is cloned in Client/node



* Once again delete directory test-vault (rm -rf test-vault) from the node ansible-rhel-node
* Go back to ansible-server
* If you notice, in previous command we haven’t specified file name (pass.yml), it will you for the vault password



* In case you don’t want vault password to be entered each time you execute the command, create file

vi pass.yml



* Save and exit.



* Since we have passed file name (pass.yml) in the command line, it won’t ask you for the vault password.
* If you go to ansible-rhel-node, test-vault directory is cloned again successfully

**Ansible Role:**

* In Ansible, the role is the primary mechanism for breaking a playbook into multiple files.
* This simplifies writing complex playbooks, and it makes them easier to reuse.
* The breaking of playbook allows you to logically break the playbook into reusable components.
* Each role is limited to a particular functionality or desired output, with all the necessary steps to provide that result either within the same role itself or in other roles listed as dependencies.
* Roles are not playbooks. Roles are small functionality that can be used within the playbooks independently. Roles have no specific setting for which hosts the role will apply.
* Roles are ways of automatically certain var files, tasks, and handlers based on the known file structure.

**How to break a playbook into multiple files:**

* Go to ansible-server
* Navigate to /opt/ansible



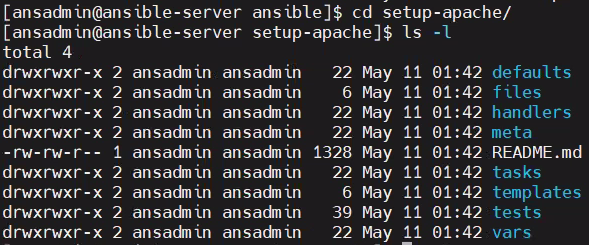
* Open playbook install-apache-httpd.yml



* Command to create a role is **ansible-galaxy init setup-apache**



* To check whether roles is created or not, list out (ls -l) files contained in /opt/ansible
* Navigate to directory setup-apache and list out files located in it.



* If you observe in the screenshot (Refer previous page), by default multiple directory structure is created
* If you invoke tree command, it will expand default directory structure of apache.



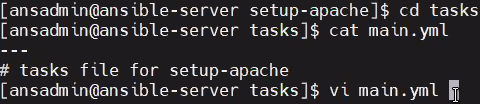
A role directory structure contains directories: defaults, vars, tasks, files, templates, meta, handlers. Each directory must contain a main.yml file which contains relevant content. Let’s look little closer to each directory.

1. **defaults**: contains default variables for the role. Variables in default have the lowest priority so they are easy to override.
2. **vars**: contains variables for the role. Variables in vars have higher priority than variables in defaults directory.
3. **tasks**: contains the main list of steps to be executed by the role.
4. **files**: contains files which we want to be copied to the remote host. We don’t need to specify a path of resources stored in this directory.
5. **Templates**: contains file template which supports modifications from the role. We use the [Jinja2 templating](https://docs.ansible.com/ansible/2.6/user_guide/playbooks_templating.html) language for creating templates.
6. **meta**: contains metadata of role like an author, support platforms, dependencies.
7. **handlers**: contains handlers which can be invoked by “notify” directives and are associated with service.

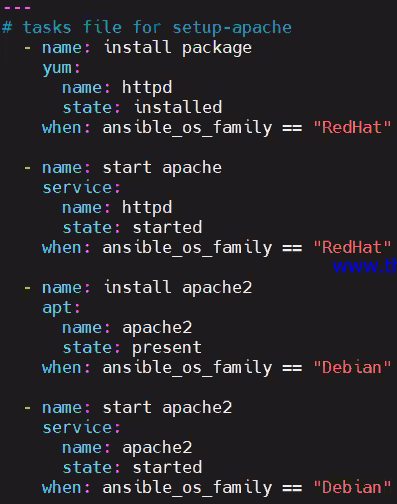
* Now first uninstall apache installed on node/client (rhel/ubuntu/Linux )
* Navigate or come back to directory opt/ansible and execute playbook uninstall-apache.yml

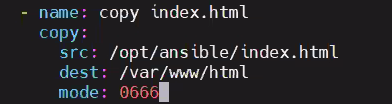


* Just to ensure whether packages are uninstalled or not, go to respective public IP address of nodes in browser and refresh the page.
* We will segregate playbook by adding all tasks in /tasks/main.yml file
* Navigate to directory tasks and open main.yml file

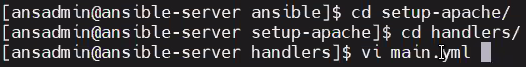


* Specify only tasks to be executed in the script. (Next continuous two screenshots are of same program )

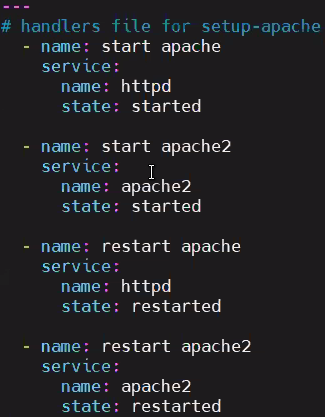




* Save and exit.
* Navigate to directory handler and open main.yml file



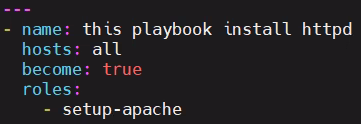
* Specify handlers which can be invoked by “notify” directives



* Save and exit. Please note that, we have already uninstalled packages in nodes.
* Now we will write a script/playbook where role is specified through which certain tasks will be executed.
* Create a playbook, setup-apache.yml and write a script to invoke a role.



* Specify role name which is to be invoked.



* Save and exit.
* Execute the playbook setup-apache.yml



* Now go to respective public IP address nodes in browser and refresh the page. httpd connection will be established successfully.

