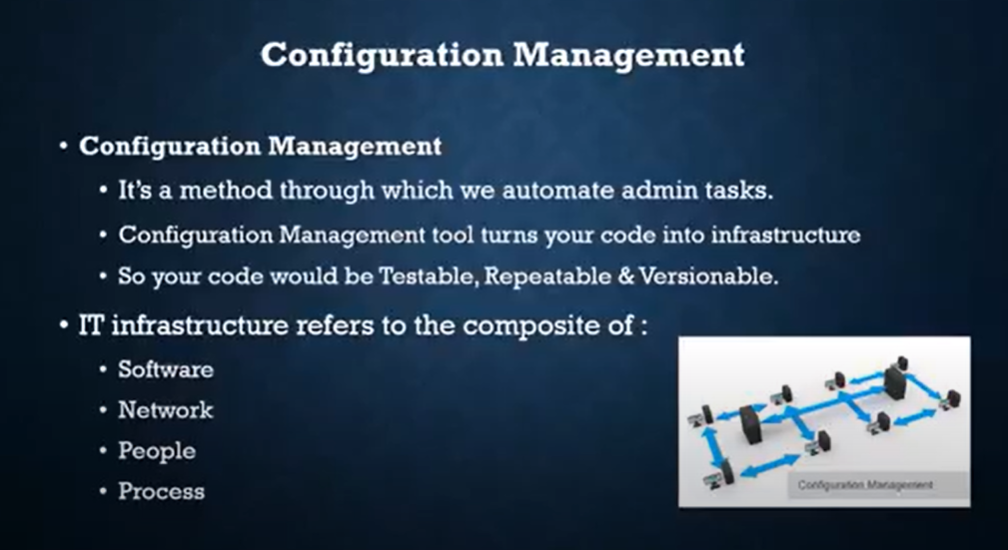
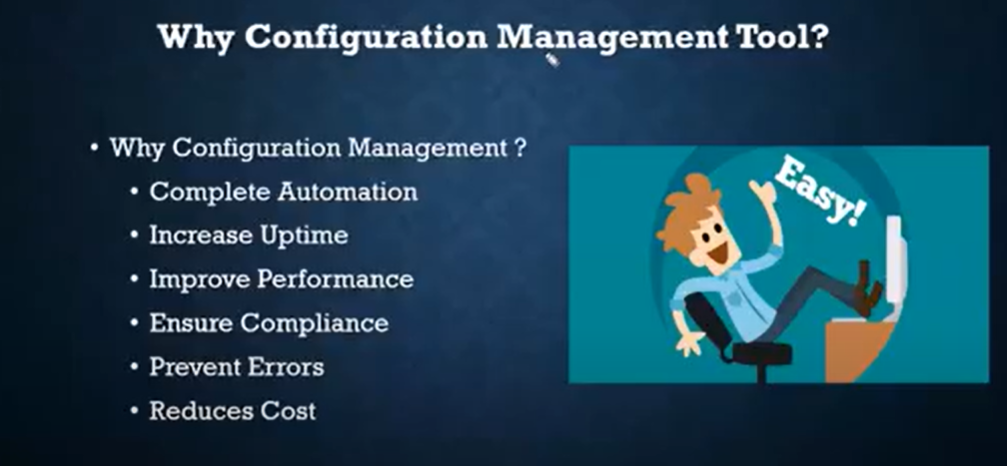
**Configuration Management:**



Configuration management in terms of Ansible means that it maintains configuration of the product performance by keeping a record and updating detailed information which describes an enterprise’s hardware and software.

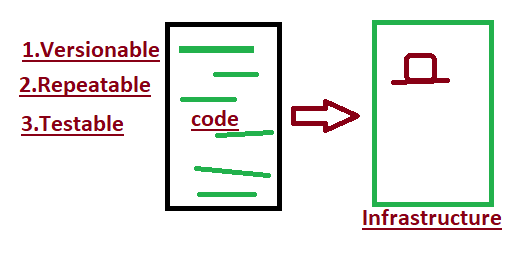
Such information typically includes the exact versions and updates that have been applied to installed software packages and the locations and network addresses of hardware devices. For e.g. If you want to install the new version of **WebLogic/WebSphere** server on all of the machines present in your enterprise, it is not feasible for you to manually go and update each and every machine.



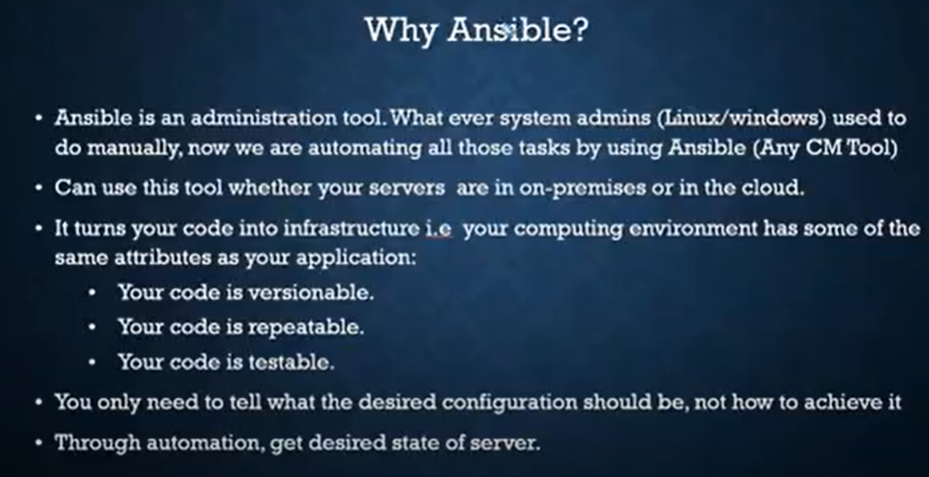
**Ansible** is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.

**Devops engineers will manage the configuration of a machine in automated way by using the tool called Ansible**. **That is why we called these tools as configuration management tools. Ex. Ansible, Chef, Puppet…**

**We are achieving this by IAC – Infrastructure as code, here we write some code to create packages , to create users in all the required machines**



Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

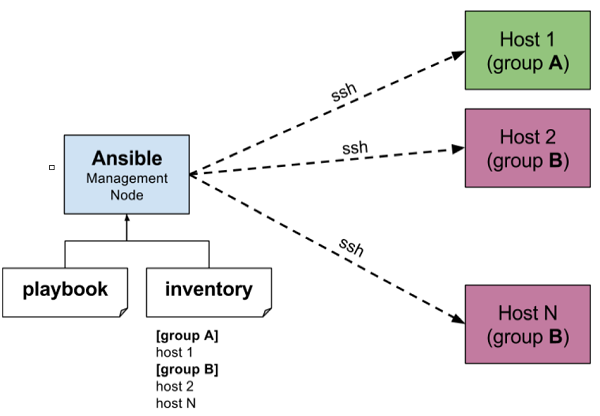


Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. **YAML** (It’s a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored)which is very easy for humans to understand, read and write. Hence the advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML – It is in human readable form).

Ansible is designed for multi-tier deployment. Ansible does not manage one system at time, it models IT infrastructure by describing all of your systems are interrelated. Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default). But if you want other method for connection like Kerberos, Ansible gives that option to you.

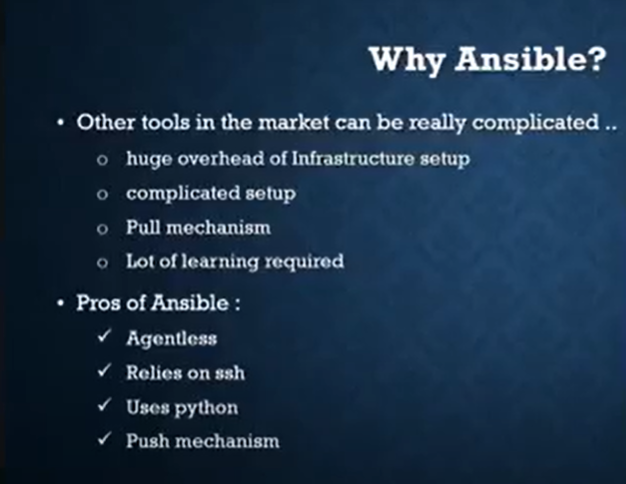
After connecting to your nodes, Ansible pushes small programs called as “Ansible Modules”. Ansible runs that modules on your nodes and removes them when finished. Ansible manages your inventory in simple text files (These are the hosts file). Ansible uses the hosts file where one can group the hosts and can control the actions on a specific group in the playbooks.

**Ansible works** by connecting to your nodes and pushing out small programs, called "**Ansible** modules" to them. **Ansible** then executes these modules (over SSH by default), and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.



The management node in the above picture is the controlling node (managing node) which controls the entire execution of the playbook. It’s the node from which you are running the installation. The inventory file provides the list of hosts where the Ansible modules needs to be run and the management node does a SSH connection and executes the small modules on the hosts machine and installs the product/software.

**Beauty** of Ansible is that it removes the modules once those are installed so effectively it connects to host machine , executes the instructions and if it’s successfully installed removes the code which was copied on the host machine which was executed.



## Installation Process

Mainly, there are two types of machines when we talk about deployment −

* **Control machine** − Machine from where we can manage other machines.
* **Remote machine** − Machines which are handled/controlled by control machine.

There can be multiple remote machines which are handled by one control machine. So, for managing remote machines we have to install Ansible on control machine.

### Control Machine Requirements

Ansible can be run from any machine with Python 2 (versions 2.6 or 2.7) or Python 3 (versions 3.5 and higher) installed.

**Note** − Windows does not support control machine.

By default, Ansible uses **ssh** to manage remote machine.

Ansible does not add any database. It does not require any daemons to start or keep it running. While managing remote machines, Ansible **does not** leave any software installed or running on them. Hence, there is no question of how to upgrade it when moving to a new version.

Ansible can be installed on control machine which have above mentioned requirements in different ways. You can install the latest release through Apt, yum, pkg, pip, OpenCSW, pacman, etc.

### Installation through Apt on Ubuntu Machine

For installing Ansible you have to configure PPA on your machine. For this, you have to run the following line of code −

$ sudo apt-get update

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible $ sudo apt-get update

$ sudo apt-get install ansible

After running the above line of code, you are ready to manage remote machines through Ansible. Just run Ansible–version to check the version and just to check whether Ansible was installed properly or not.

\*\*\*\* Ansible Installation from apel repo which contains all dependency packages like Ansible, Python, git…

**Here 1st we download apel and then install**

Launch Amazon Linux (need not to install ansible in nodes)

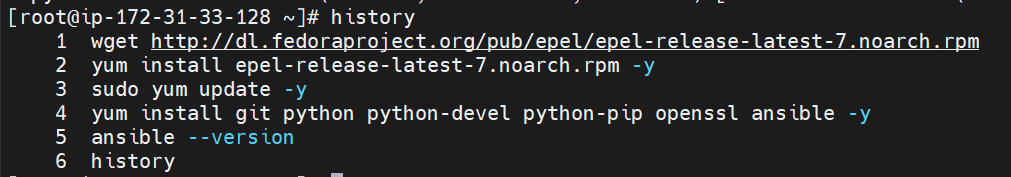
wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

yum install epel-release-latest-7.noarch.rpm -y

sudo yum update -y

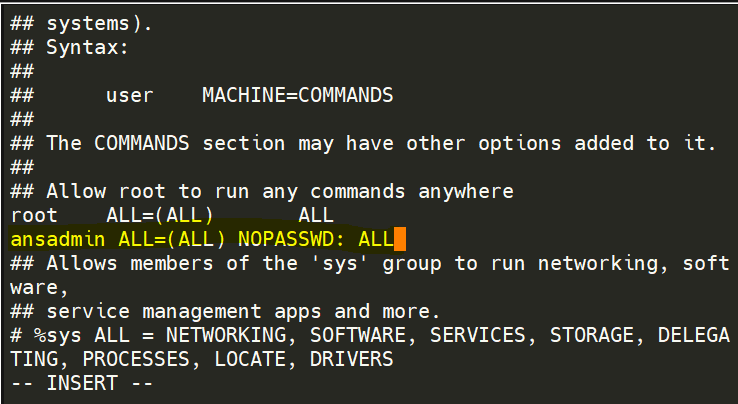
sudo yum install git python python-devel python-pip openssl ansible -y

ansible --version

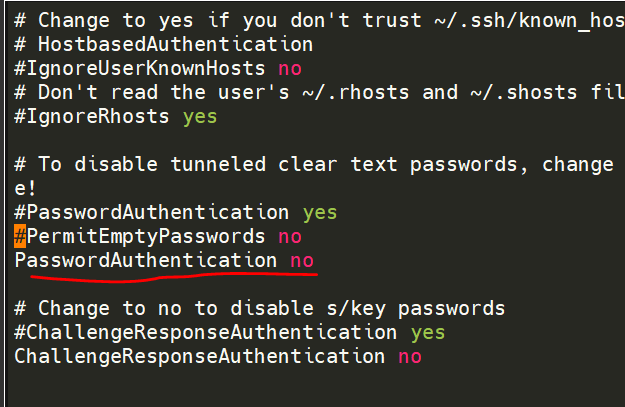
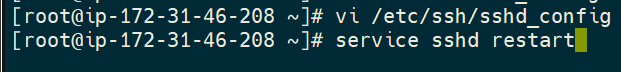


Create the required nodes using EC2 service

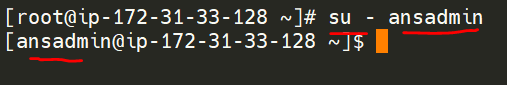
Using MultiExec in MobaXterm do the below actions

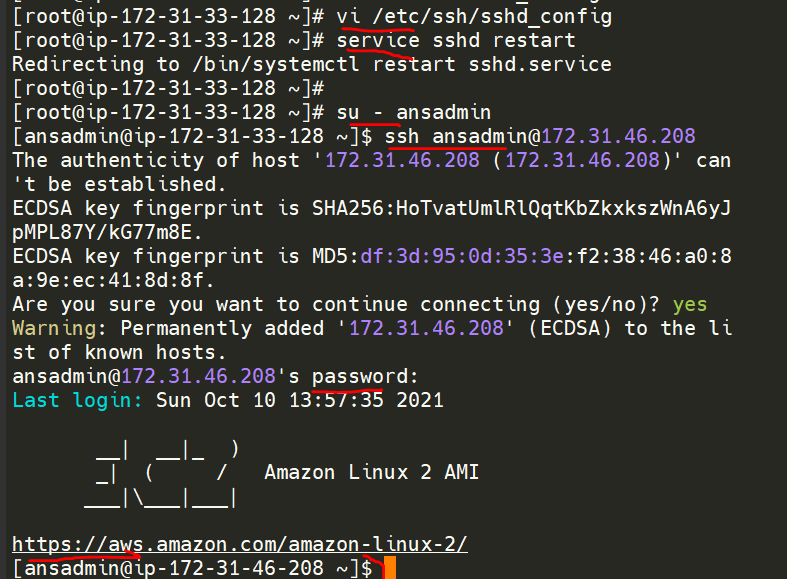
To add a new user in the machine: useradd <username>  
To add password :passwd <password>  
To give root access to the user: open sudo file  
1.visudo  
2.Add username under root user  
  
3.Save the file **ESCAPE>:wq!**

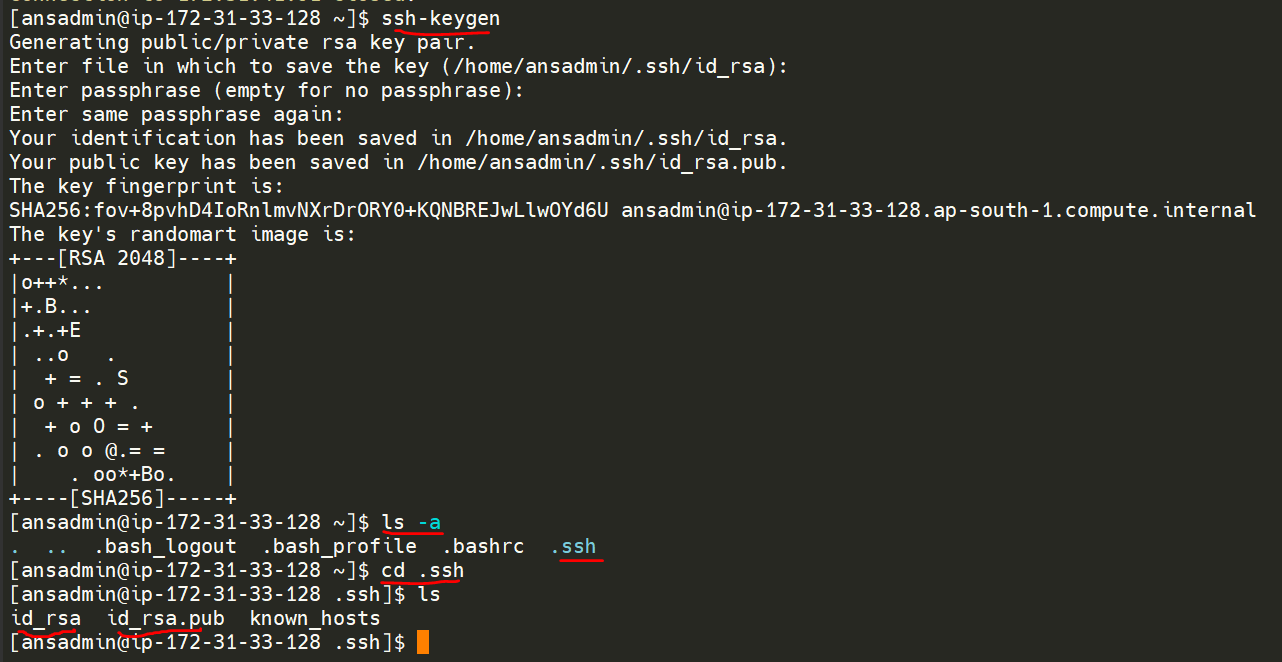
**Add ssh in the configuration file in all the machines, servers and nodes, then restart the service**

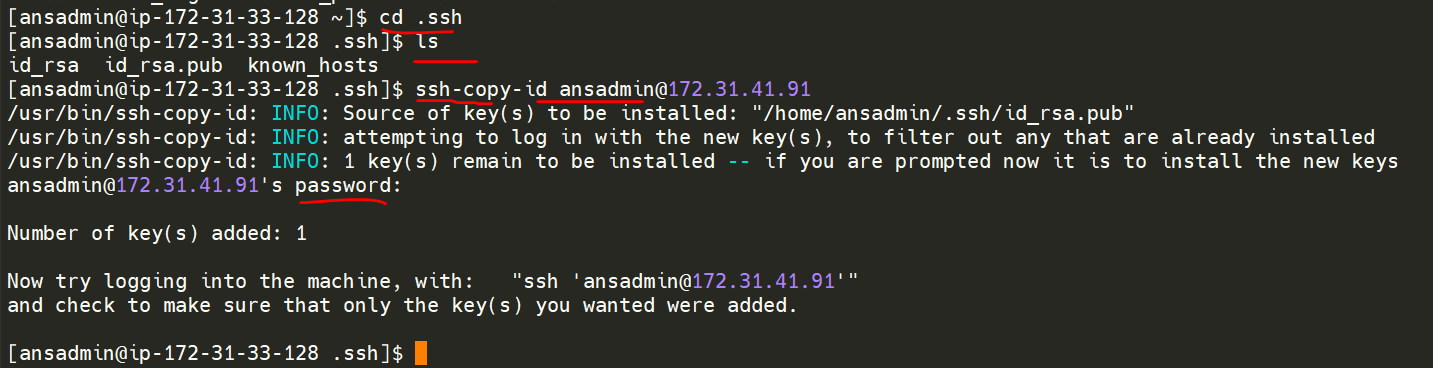
**vi /etc/ssh/sshd\_config  
passwordAuthentication YES** **then restart the sshd service service sshd restart**

**Create connection between server to nodes:**

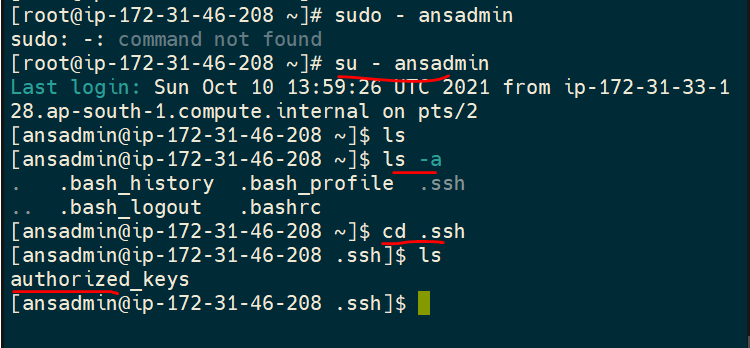
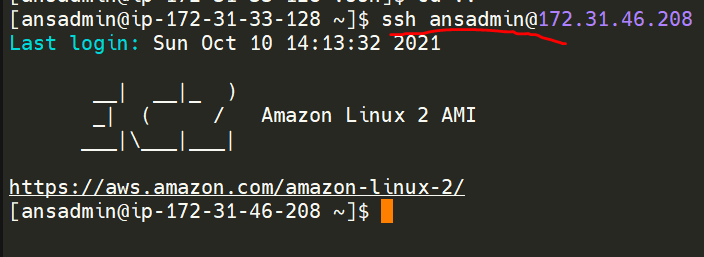
**To switch user**

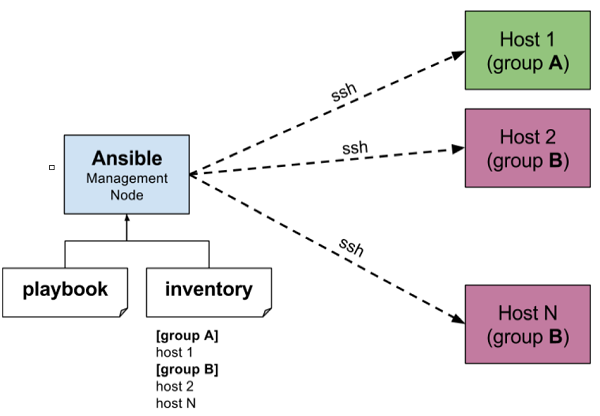
**ssh node\_username@ip\_of\_node   
Then enter password:**

**SSH without password: For this we will create keys and by using that we can connect all the node to server without password**

**To copy the public key to node  
ssh-copy-id <nodename@ip>**

Now node is connected to the server permanently

TO verify the public key copied or not  
  
Now onwards node won’t ask for password to connect.  
To connect to the node : ssh <nodename@ip>  


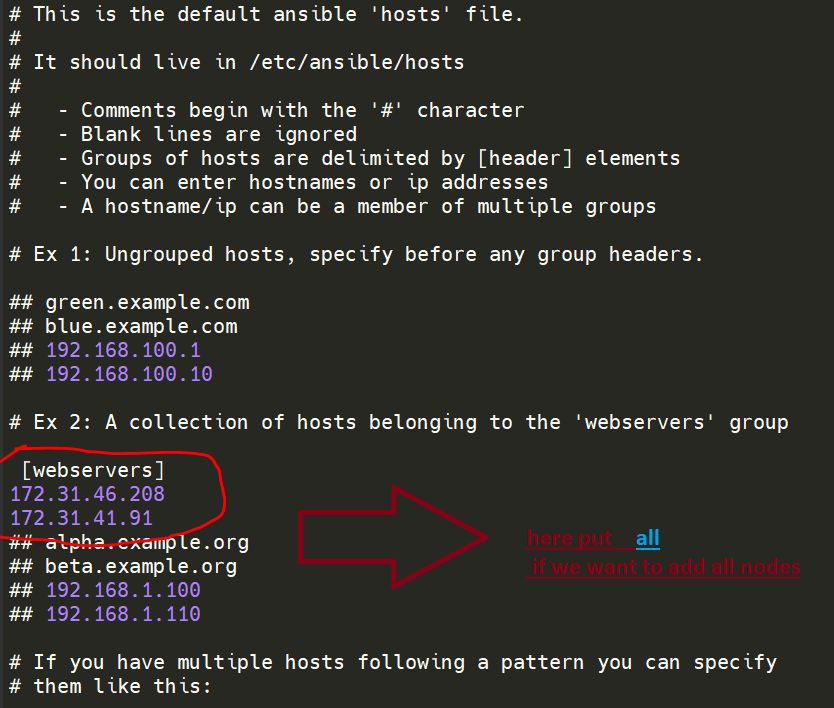


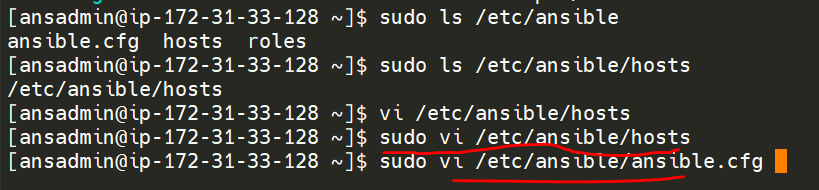
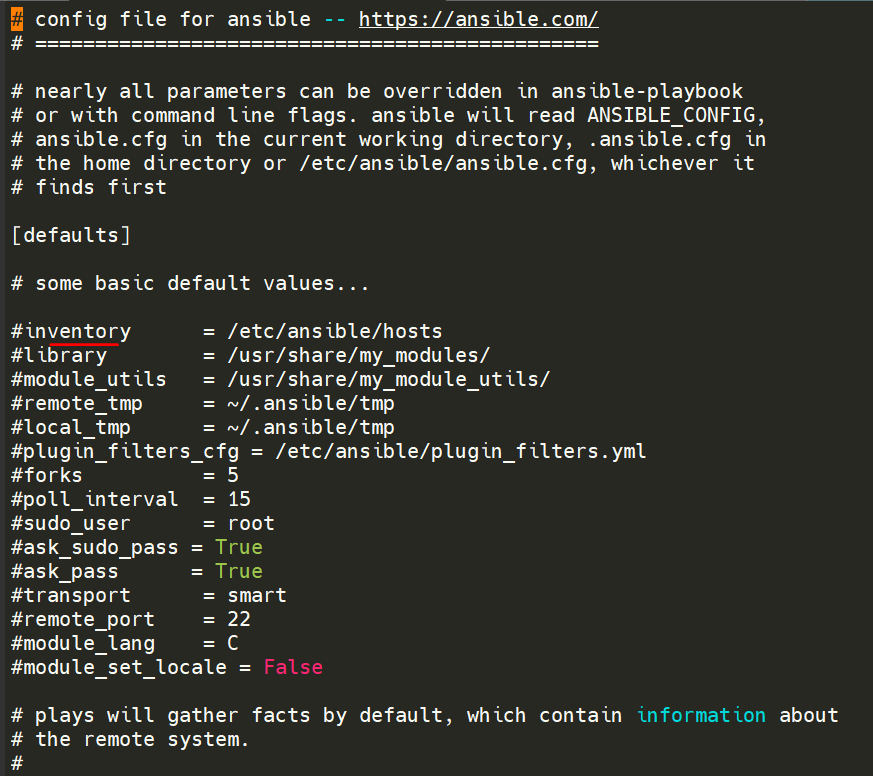
Add the nodes in server **inventory/hosts** file, then server will will check this for each push to know which nodes code needs to push.

**vi /etc/ansible/hosts**

**add hosts somewhere in this file**

**to add as a group like webservers, dbservers, testservers  
[webservers]  
ip1  
ip2  
ip3**

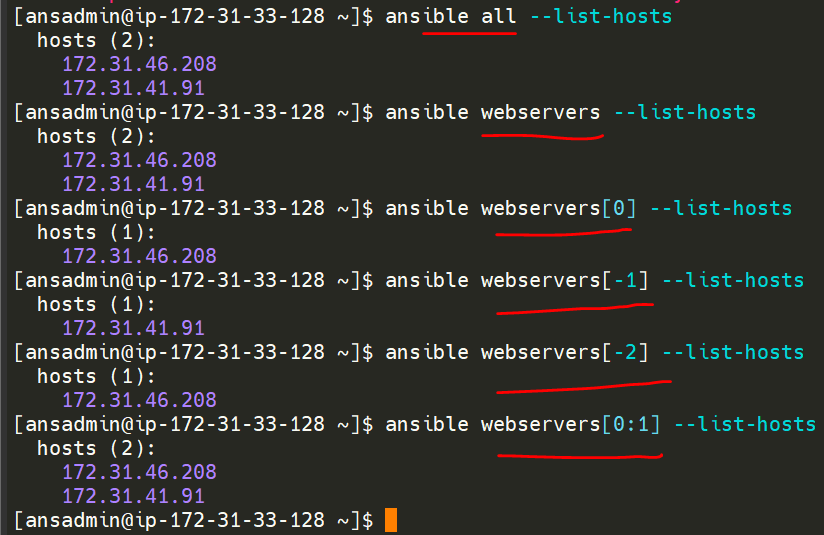
****

Then uncomment the **hosts** tag in ansible.config file  
**sudo vi /etc/ansible/ansible.config**   
  


To test whether we added the nodes correctly

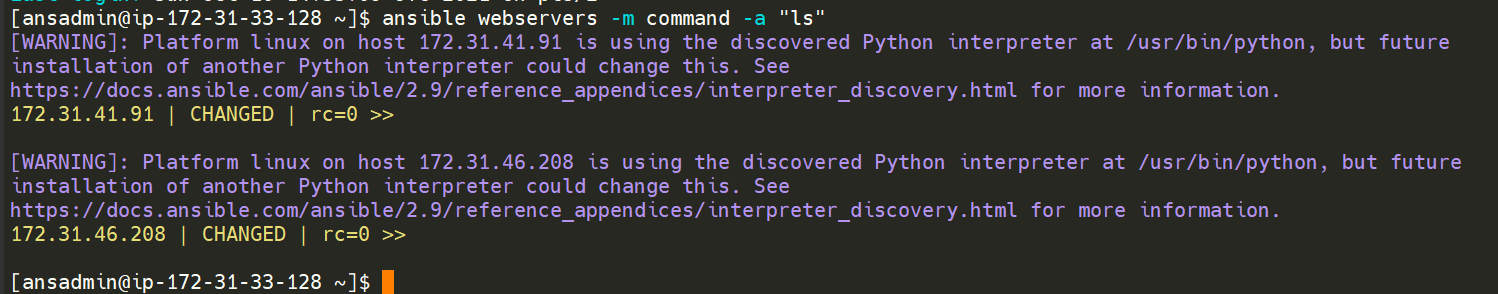
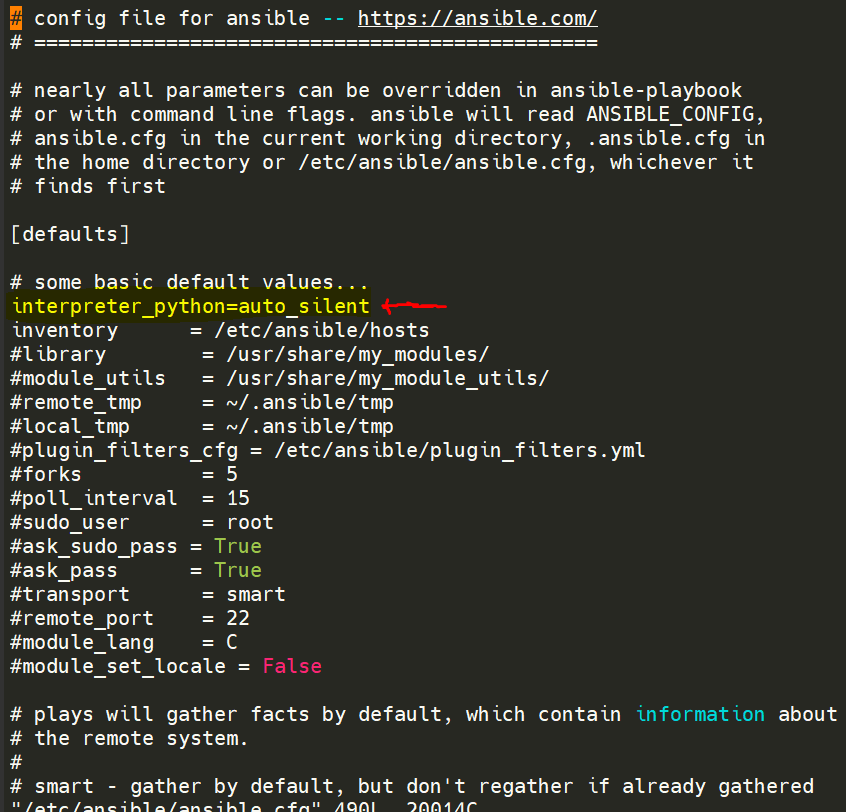
ansible **all –list-hosts**  
To know the nodes in a group: **ansible <groupname> - - list-hosts**   
To know the 1st node in list of nodes connected: **ansible <groupname>[index] - - list-hosts**

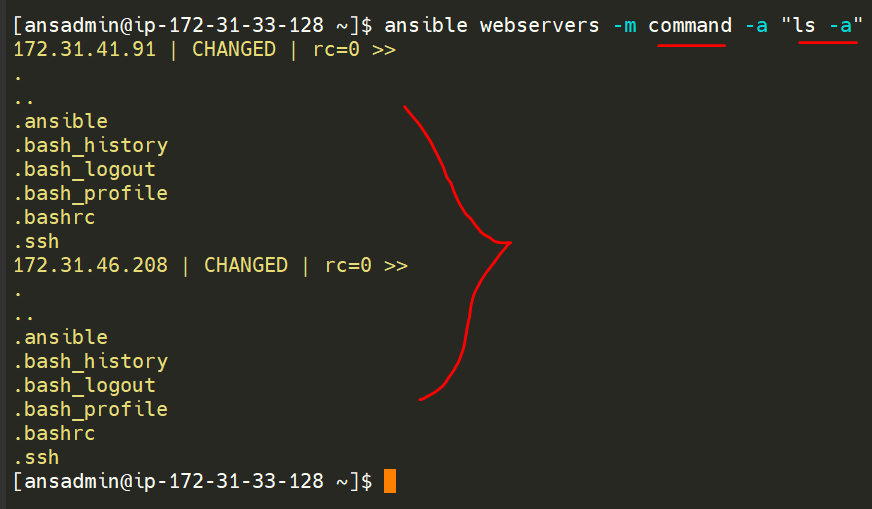
**To know the last node in list of connected : ansible <groupname>[-index] - - list-hosts  
Ex: ansible <groupname>[-1] - - list-hosts, last node  
Ex: ansible <groupname>[-2] - - list-hosts, 2nd from last  
Ex: ansible <groupname>[-3] - - list-hosts, 3rd from last**

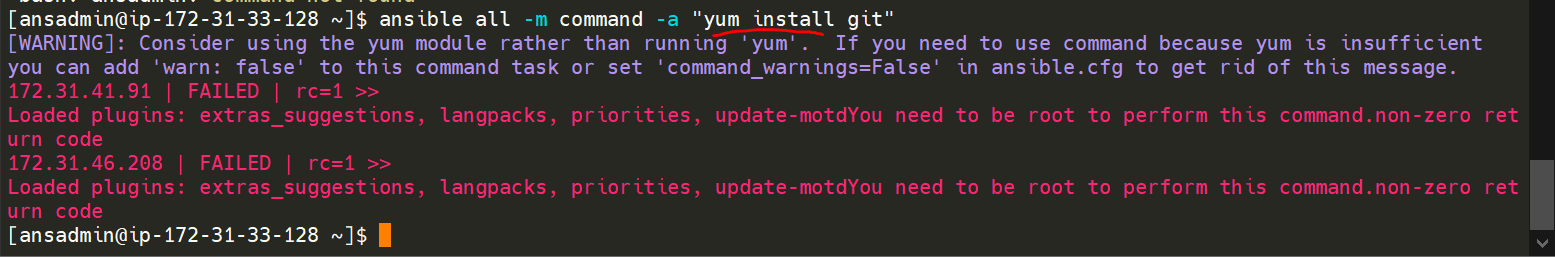
**To declare the nodes in range   
to push the code into first 5 nodes in a group  
ansible <groupname>[0:4] - - list-hosts  
To push code into multiple groups:  
ansible <groupname>[index]: <groupname>[index]: - - list-hosts  
Ex: ansible webservers[5]:dbservers[2] - - list-hosts**

**To push into the nodes from server we have 2 ways  
1.Adhoc (To perform any single task)  
2.Playbook(To perform more than 1 task)**

**1.Adhoc:**

**Command module:  
ansible <groupname>[index] -m <module> -a “action/task”  
Ex: ansible webservers[1] –m command –a “ls”  
Ex: ansible webservers[1] –m command –a “touch file1”** **To remove the warnings in above output  
add interpreter\_python=auto\_silent & command\_warnings=False in /etc/ansible/ansible.config  
for that   
sudo vi /etc/ansible/ansible.config**



**Here command means any linux command  
  
ansible webservers[1] –m command –a “yum install git”****To install a package as a user, we need sudo privilages  
so to add sudo we use  
ansible webservers[1] –m command –a “yum install git” –b  
here –b become   
\*\*If we don’t mention –m module in above, by default it will take command module**  
 **Copy module:   
ansible <servers\_groupname> -m copy -a “src=<path to source> dest=<path to destination>” –b**

**Ex: ansible webservers -m copy -a “src=/tmp/file1 dest=/tmp/dir/” –b**

**Yum module:**

**ansible <servers\_groupname> -m yum -a “pkg=<packagename> state=<state, installed/present” -b   
Ex: ansible webservers -m yum -a “pkg=httpd state=installed”–b  
Service module:   
ansible <servers\_groupname> -m service -a “name=<servicename state=<present/absent>” –b**

**Ex: ansible webservers -m service -a “name=httpd state=present” –b**

**user module:   
ansible <servers\_groupname> -m user -a “name=<username state=<present/absent>” –b**

**Ex: ansible webservers -m service -a “name=ganesh state=present” –b**

**setup module:   
ansible <servers\_groupname> -m setup**

**Ex: ansible webservers -m setup**

**Playbooks:**

Playbooks are the files where Ansible code is written. Playbooks are written in YAML format. YAML stands for Yet Another Markup Language. **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 run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

## Playbook Structure

Each playbook is an aggregation of one or more plays in it. Playbooks are structured using Plays. There can be more than one play inside a playbook.

The function of a play is to map a set of instructions defined against a particular host.

YAML is a strict typed language; so, extra care needs to be taken while writing the YAML files. There are different YAML editors but we will prefer to use a simple editor like notepad++. Just open notepad++ and copy and paste the below yaml and change the language to YAML (Language → YAML).

A YAML starts with --- (3 hyphens)

## Create a Playbook

Let us start by writing a sample YAML file. We will walk through each section written in a yaml file.

---

name: install and configure DB

hosts: testServer

become: yes

vars:

oracle\_db\_port\_value : 1521

tasks:

-name: Install the Oracle DB

yum: <code to install the DB>

-name: Ensure the installed service is enabled and running

service:

name: <your service name>

The above is a sample Playbook where we are trying to cover the basic syntax of a playbook. Save the above content in a file as **test.yml**. A YAML syntax needs to follow the correct indentation and one needs to be a little careful while writing the syntax.

## The Different YAML Tags

Let us now go through the different YAML tags. The different tags are described below −

### name

This tag specifies the name of the Ansible playbook. As in what this playbook will be doing. Any logical name can be given to the playbook.

### hosts

This tag specifies the lists of hosts or host group against which we want to run the task. The hosts field/tag is mandatory. It tells Ansible on which hosts to run the listed tasks. The tasks can be run on the same machine or on a remote machine. One can run the tasks on multiple machines and hence hosts tag can have a group of hosts’ entry as well.

### vars

Vars tag lets you define the variables which you can use in your playbook. Usage is similar to variables in any programming language.

### tasks

All playbooks should contain tasks or a list of tasks to be executed. Tasks are a list of actions one needs to perform. A tasks field contains the name of the task. This works as the help text for the user. It is not mandatory but proves useful in debugging the playbook. Each task internally links to a piece of code called a module. A module that should be executed, and arguments that are required for the module you want to execute.

Ansible uses YAML syntax for expressing Ansible playbooks. This chapter provides an overview of YAML. Ansible uses YAML because it is very easy for humans to understand, read and write when compared to other data formats like XML and JSON.

Every **YAML** file optionally starts with “---” and ends with “...”.

## Understanding YAML(Yaml ain’t markup language)

### key-value pair

YAML uses simple key-value pair to represent the data. The dictionary is represented in key: value pair.

**Note** − There should be space between : and value.

### Example: A student record

--- #Optional YAML start syntax

james:

name: james john

rollNo: 34

div: B

sex: male

… #Optional YAML end syntax

### Abbreviation

You can also use abbreviation to represent dictionaries.

### Example

James: {name: james john, rollNo: 34, div: B, sex: male}

## Representing List

We can also represent List in YAML. Every element(member) of list should be written in a new line with same indentation starting with “- “ (- and space).

### Example

---

countries:

- America

- China

- Canada

- Iceland

…

### Abbreviation

You can also use abbreviation to represent lists.

### Example

Countries: [‘America’, ‘China’, ‘Canada’, ‘Iceland’]

### List inside Dictionaries

We can use list inside dictionaries, i.e., value of key is list.

### Example

---

james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

…

### List of Dictionaries

We can also make list of dictionaries.

### Example

---

- james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

- robert:

name: robert richardson

rollNo: 53

div: B

sex: male

likes:

- biology

- chemistry

…

YAML uses “|” to include newlines while showing multiple lines and “>” to suppress newlines while showing multiple lines. Due to this we can read and edit large lines. In both the cases intendentation will be ignored.

We can also represent **Boolean** (True/false) values in YAML. where **boolean** values can be case insensitive.

### Example

---

- james:

name: james john

rollNo: 34

div: B

sex: male

likes:

- maths

- physics

- english

result:

maths: 87

chemistry: 45

biology: 56

physics: 70

english: 80

passed: TRUE

messageIncludeNewLines: |

Congratulation!!

You passed with 79%

messageExcludeNewLines: >

Congratulation!!

You passed with 79%

## Some common words related to Ansible.

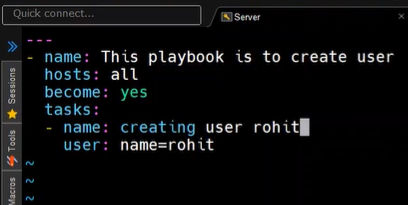
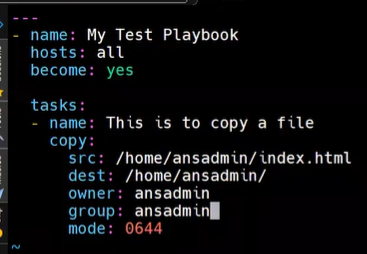
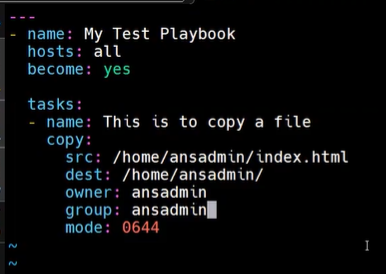
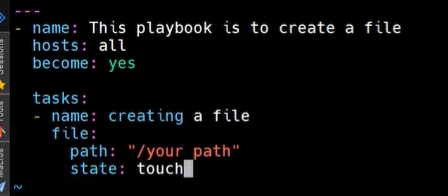
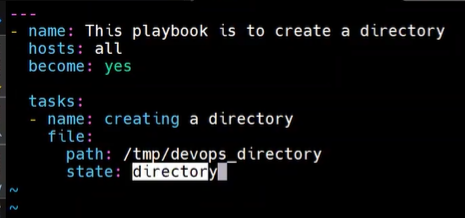
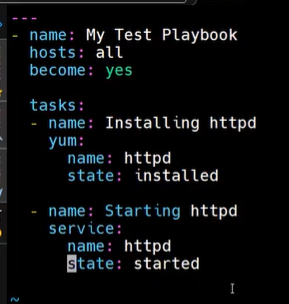
**Service/Server** − A process on the machine that provides the service.

**Machine** − A physical server, vm(virtual machine) or a container.

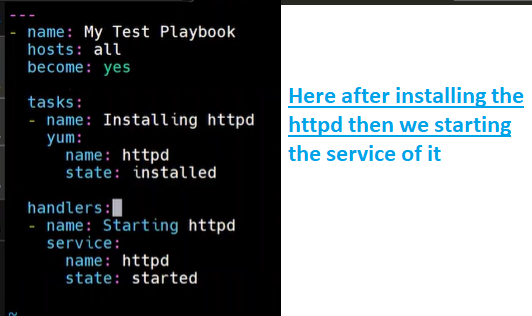
**Target machine** − A machine we are about to configure with Ansible.

**Task** − An action(run this, delete that) etc managed by Ansible.

**Playbook** − The yml file where Ansible commands are written and yml is executed on a machine.

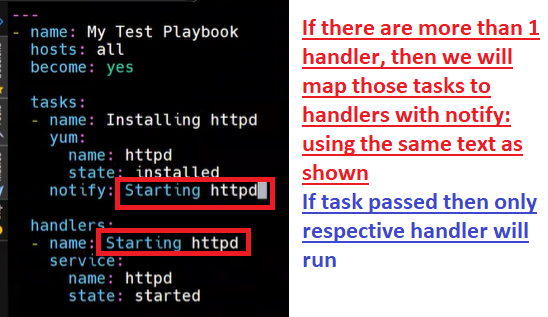
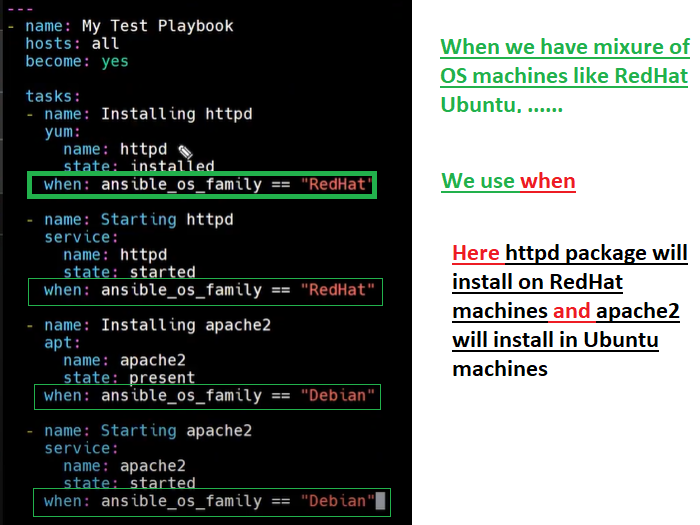
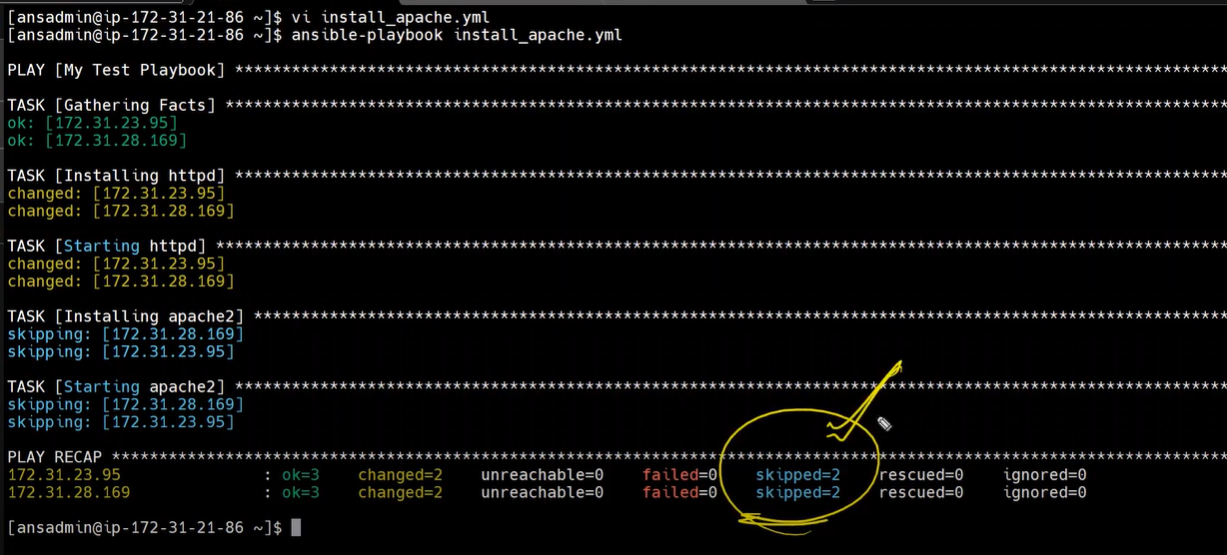
**To create a playbook** vi <filename>.yml  
Ex: vi create\_user.yml  
   
   
 

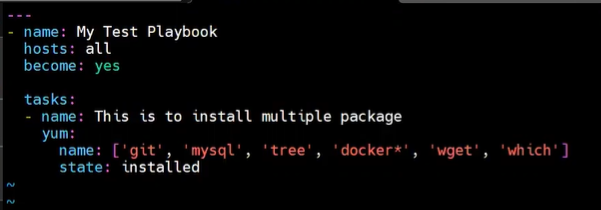
**To run the playbook:** ansible-playbook <playbookname.yml>

Notify & Handlers:  
Where two tasks are inter dependant on each other then we use handlers  
Ex:httpd and service start is inter dependant  


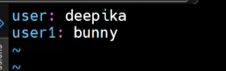
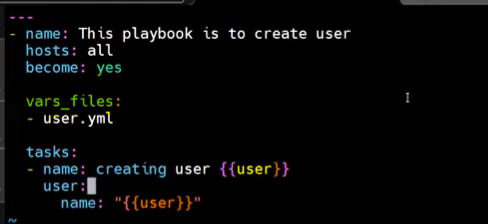
Here if httpd package installs then only httpd service will start

**Using handler and notify:**

  
  
If we run above playbook, having 2 Redhat nodes, it will skip the **Debian(Ubuntu)** tasks   


To make all nodes into webservers  
  
To install multiple packages using the Playbook  


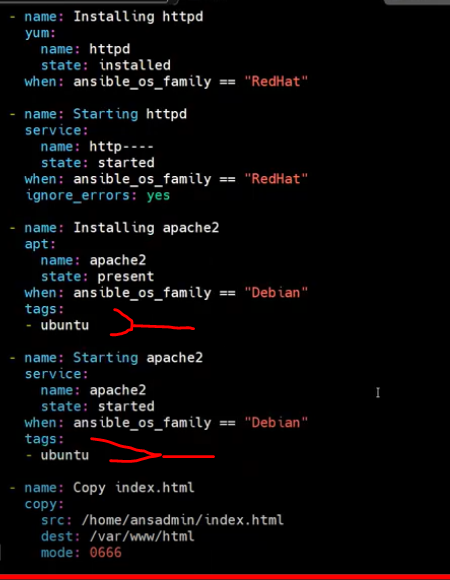
**Variables:  
To declare variables we use {{variable}}**

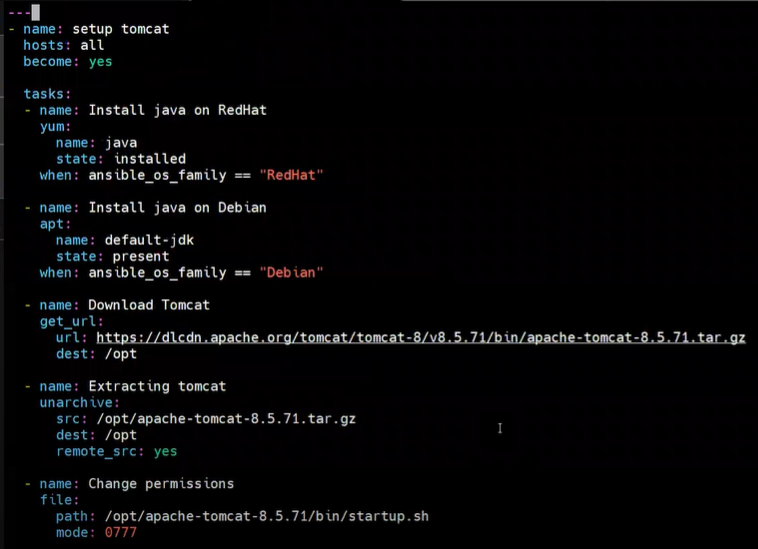
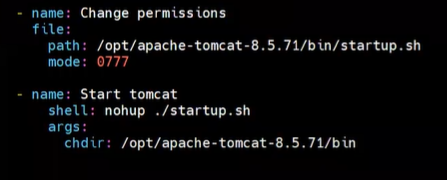
We can declare all the variables in one file and pass that file to another playbook  
 these varibles stored in **user.yml** file  
Then accessing those variable by declaring that file name in the Playbook  


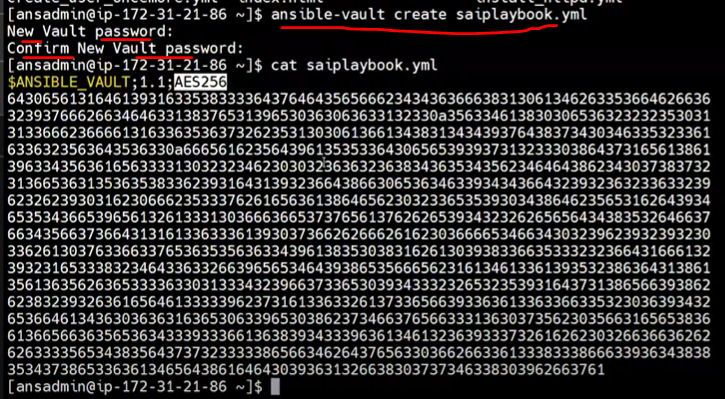
If we want to change the variable for temparary perpose we can use   
  
  
**How to handle errors:**If we feel there might be some error then we need to add : **ignore\_errors: yes**  

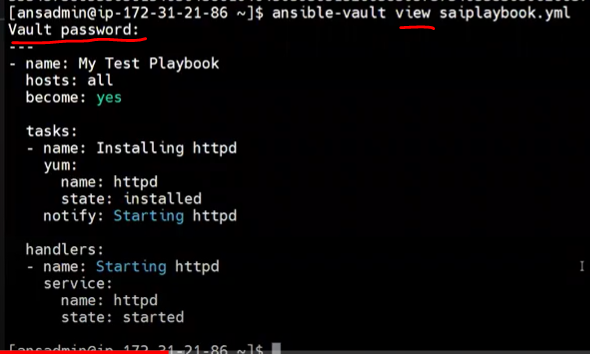

**Tags**:

We can give tags for Taks to which tasks from playbook has to run:

  
Then run the command as   

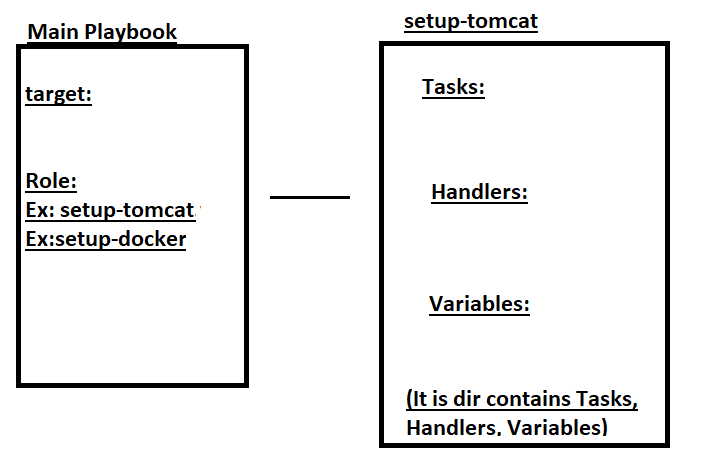
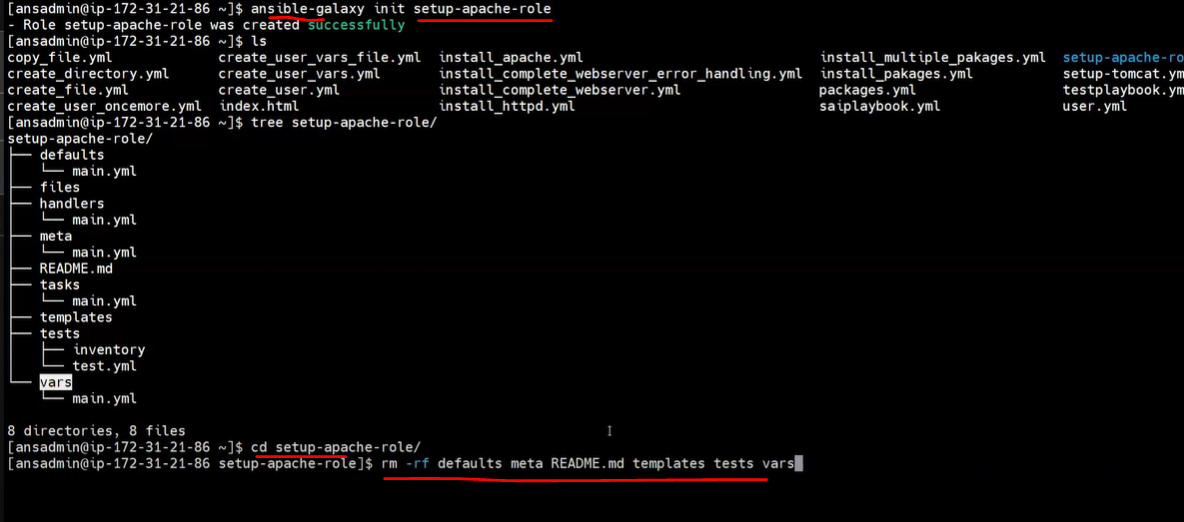

Installing Tomcat:  
  


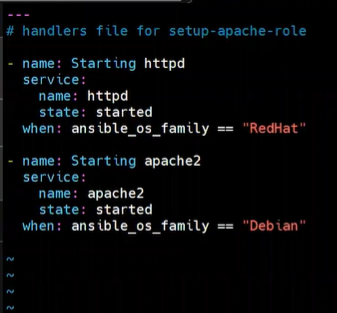
**To Encript Playbook while creating playbook**:  
ansible-vault create <Plybook.yml>  


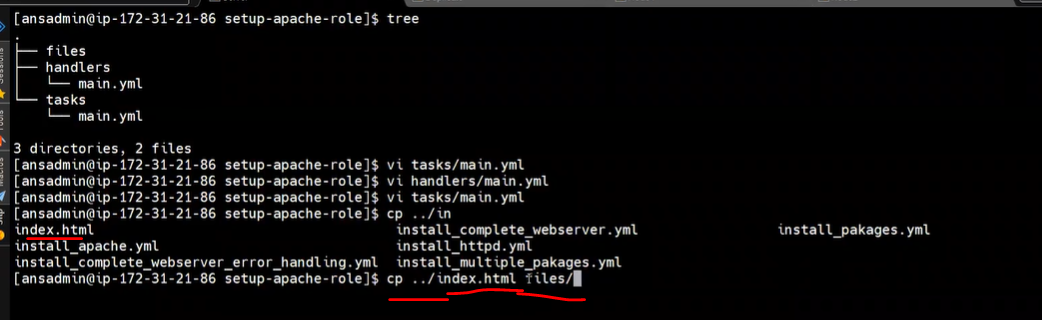
**To view the encripted playbook:**ansible-vault view <Plybook.yml>  
  
**To edit the encripted playbook:**ansible-vault edit <Plybook.yml>  
**To decrypt playbook:**ansible-vault decrypt <Plybook.yml>

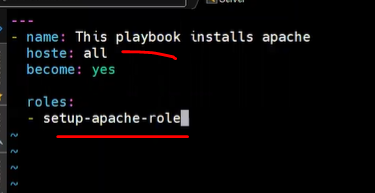
**To Encript the existing playbook :**ansible-vault encrypt <Existing\_Plybook.yml>  
**To decrypt the existing playbook :**ansible-vault **decrypt** <Existing\_Plybook.yml>

**Roles:**

Dividing playbooks into respective directory structure is called **Roles  
To create a role, ansible-galaxy init <role\_name>  
** **here add all the tasks in tasks/main.yml file  
add all the handlers in handlers/main.yml  
add all the variables in vars/main.yml  
add all the files in files folders  
Remove unwanted remaining folders like defaults, meta, templates**   
  
Added all tasks in tasks/main.yml  


Added all handlers in handlers/main.yml  


Add required files in file folders  


Add this role into main playbook  
  
  
Then run the main playbook  
