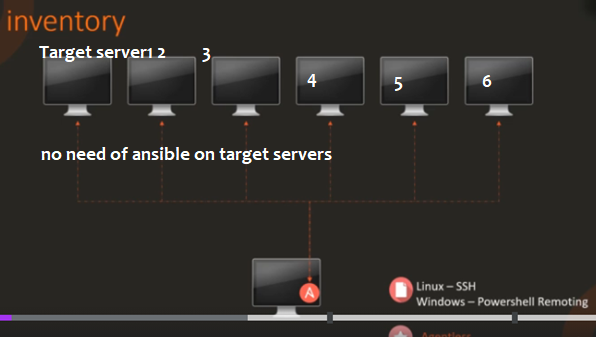
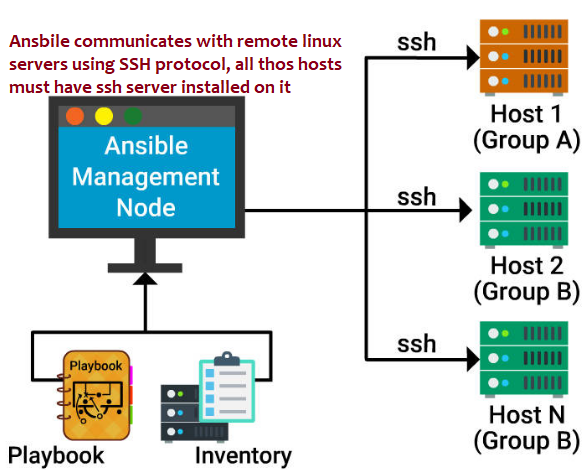
Ansible architecture

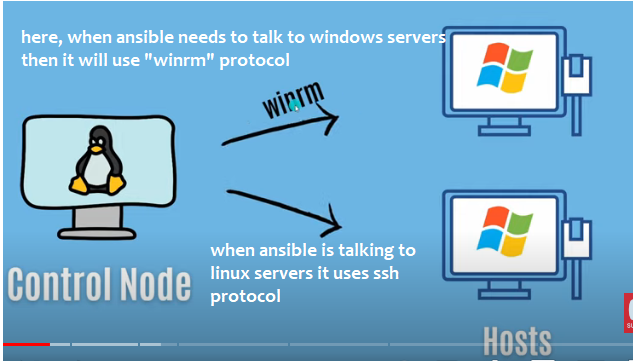


Ansible Is used to communicate with remote target servers like windows/linux/centos..

Ansible is used to communicate with remote linux servers using SSH

Ansible can communicate to windows server using winrm protocol





Ansible is agent less, when the source machine wants to connect to 10 machines, then on those 10 target machines, Ansible doesn’t need to be installed, hence Ansible is agent less, means ansible should only be on single source machine, no need Ansible installation on target machines,

### Why Ansible

1. Example-
2. U may want to start 100 servers at once
3. U may want to stop all servers at once
4. In remote server/machine , When u want to move files from 1 folder to another folder, because manually we cant go to each each server and move them na?
5. If u want to start docker engine, In all 10 remote servers, then we can’t go manually and start the server by executing some script right?
6. If u want to start ssh server on all 10 remote servers, then we can simply execute a playbook
7. Lets say u want to power on 10 machines /if u have to deploy code in 10 machines then u have to write a script and execute that script in 10 machines, here instead of writing that complex script

Ansible playbook (Ansible scripts) are easy to write and learn bec they are in YAML file

1. Ansible is agent less, Ansible agent need not to be installed on target machines, this source machine will connect via ssh, other orchestration tool needs ansible agents on target machines too

### What is Ansible

Ansible is an automation tool,

Ansible is a configuration management tool, based on push-based architecture to automate configuration of your hosts to achieve a ***desired state***.

Following are the components of **Ansible architecture**

* **Inventory** - Defines the list of target hosts, means it have all the remote server urls on whom we are going to execute tasks
* **Playbook (YAML file)** - Defines list of tasks
* **Module** - A python code invoked from tasks and executed on hosts
* **Control Machine** - Takes playbook and executes each task on particular group of host

Uses of ansible

Example u wrote some script, and u have 100 target servers, if u want to execute that same script on all servers,

then choose ansible , just give all 100 target ipaddresses, ansible will execute that script in all 100 machines

### Ansible installation

Ansible will communicate to all target servers using ssh , so if u want to work with ansible u have to install ssh server

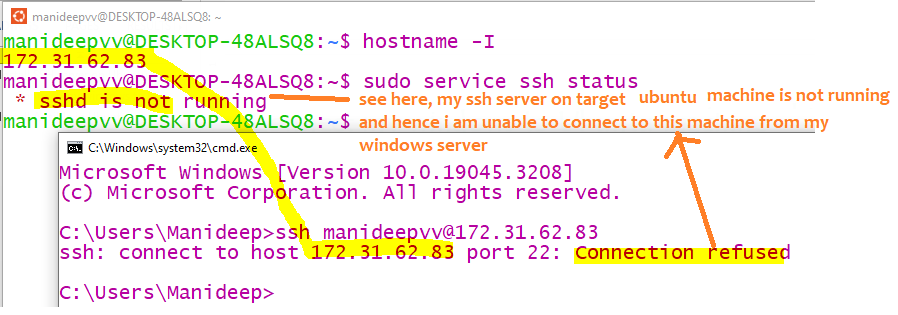
So on current machine ansible must be there and on target machine ssh server must be installed on target machines

### Install-start SSH server svc on remote machines

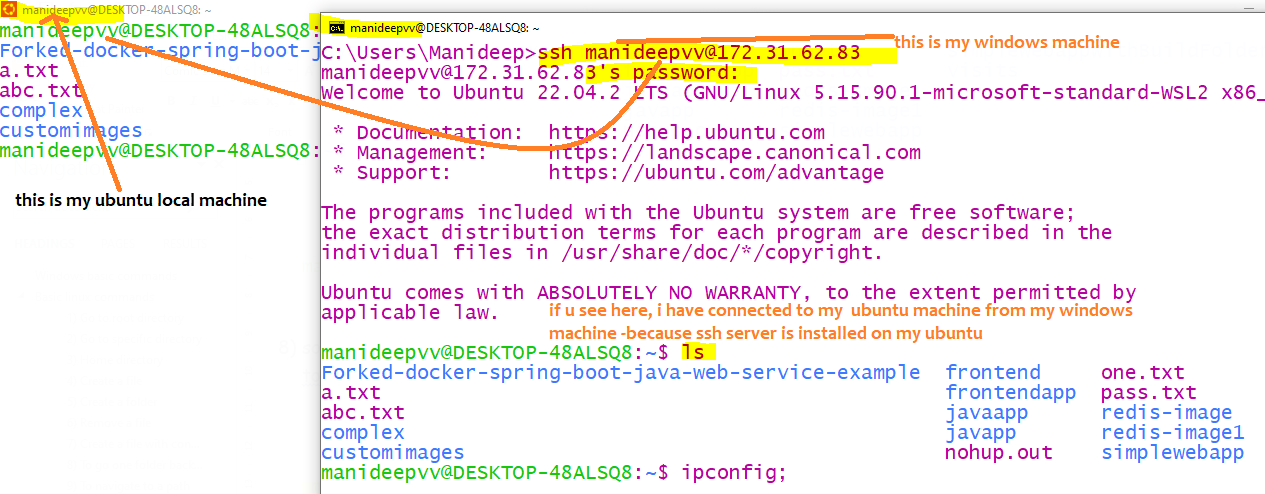
|  |  |
| --- | --- |
| Check ssh server is running on target machine- execute this command on target machine | “sudo service ssh status “ |
| To start ssh server | “sudo service ssh start”  To stop ssh server  **sudo service ssh stop** |
| Get the ip of target machine , so go & execute this cmnd on target machine | hostname –I |
| Connect from windows | ssh <target ssh machine username>@<ip of remote machine>  **ssh manideepvv@172.31.62.83** |

## **Connect my windows to Ubuntu machine**

1. if ssh server on target machine is stopped, we cant connect, so always ensure ssh server on target machine is running
2. when it asked pass I used my docker pass



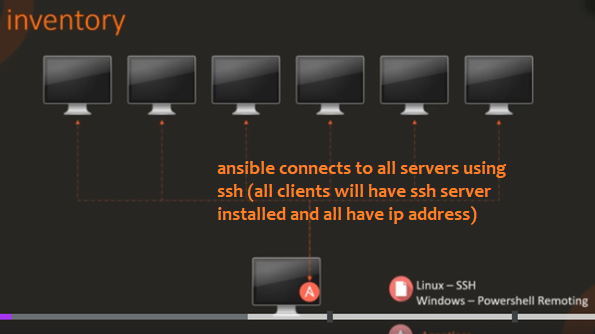
1. now from my windows machine I will connect to my Ubuntu machine



1. if u want to exit the connection from that machine, then simply type exit

# How ansible works

Using ip addresses of target machines+ ssh server on target machines, if target machine is not having the ipaddress then ansible cant connect to target machines,



## Inventory files

 Ansible captured the desired state through Playbook, but how would ansible know which remote machines it should connect to???

ans:inventory file

Inventory file contains the list of all target servers information to connect.

Here web,db,mail are the aliases to the server, now ansible will connect to all server urls mentioned in the inventory files

**The Inventory file in Ansible contains the list of all hosts (target systems/servers) that need to be configured. You can also group hosts under different names as shown:**

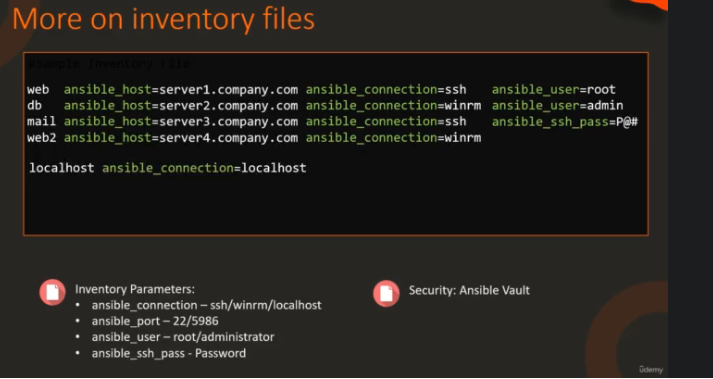
[group A]

Host 1

[group B]

Host 2

Host N



Here if u see parameter “ansible\_connection=ssh” means the

1. Ansible host connects with remote Ubuntu server then it will use ssh protocol,
2. If Ansible is connecting to target windows machine then it uses winrm protocol

### **Sample inventory files**

Create an inventory file with 1 target machine



#<alias of target machine> ansible\_host=<ipaddress of host> ansible\_ssh\_pass=<passoword>

Target1 ansible\_host=192.168.1.114 ansible\_ssh\_pass=osboxes.org

Target2 ansible\_host=192.168.1.115 ansible\_ssh\_pass=osboxes.org

Ansible cmnds with inv files

* **ansible**: *This is a tool that allows you to****run a single task****at a time*.

$ ansible <host-name> [-m module\_name] [-a args] [options]

ansible <host-group-name> -i <inventory-file-name> -m ping

m means module –a means argument

**Make a directory in remote host**:

 ansible <host alias name> -i <inventory file> -m command -a "mkdir folder1"

 ansible host01 -i myhosts -m command -a "mkdir folder1"

**Ansible Command to create a folder in localhost**

ansible localhost -m command -a "mkdir test"

* **Check files or folders in remote host**: ansible host01 -i myhosts -m command -a "ls"
* Create a file 🡪 ansible <alias-naem> -i <inv file name> -m command -a "touch aio.txt"

ansible target1 -i 3.inv-with-variables-v1.txt -m command -a "touch aio.txt"

echo something

=======

ansible <host alias name> -i <inventory-file-name> -m shell -a "echo abcd"

ansible target1 -i 3.inv-with-variables-v1.txt -m shell -a "echo abcd"

here –m says module, -a means argument

if u don’t have inventory file u directly use localhost as mentioned above

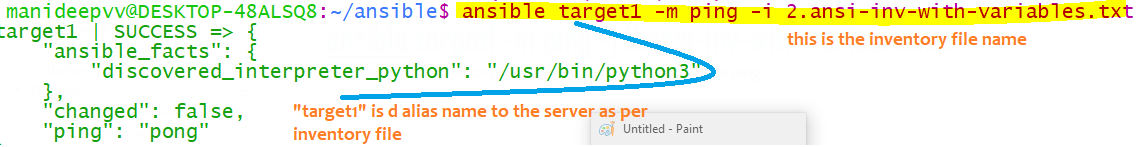
ansible target1 -i 2.ansi-inv-with-variables.txt -m ping

here target1 is an alias name declared in the inventory file , hence mentioning inv file immediately is better.

here I wrote an inventory file contains 1 server I just want to ping

//here –m means module, ping module or command module

//-i means inventory file

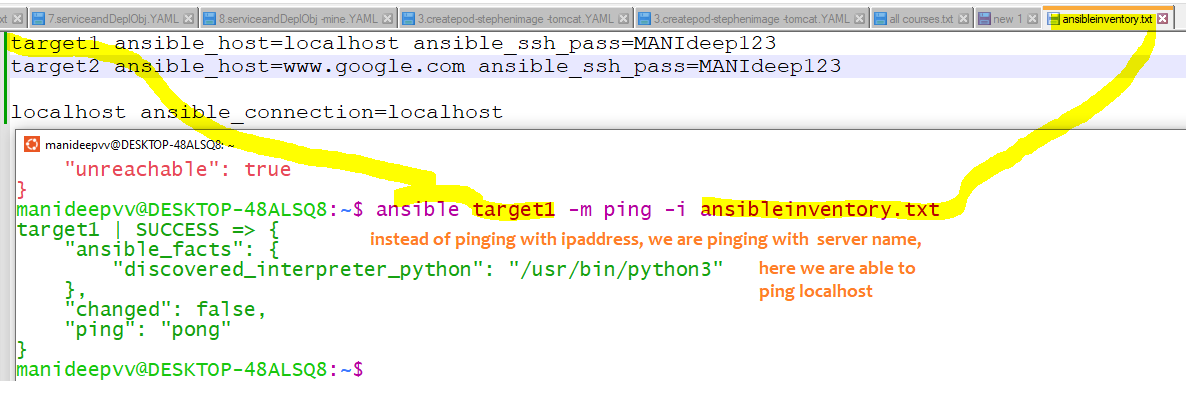


To ping a particular server in the inventory file

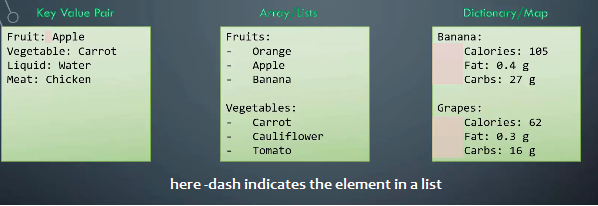
Generally there are many server names-urls present in the inventory file

ansible target1 –m ping –i inventory.txt

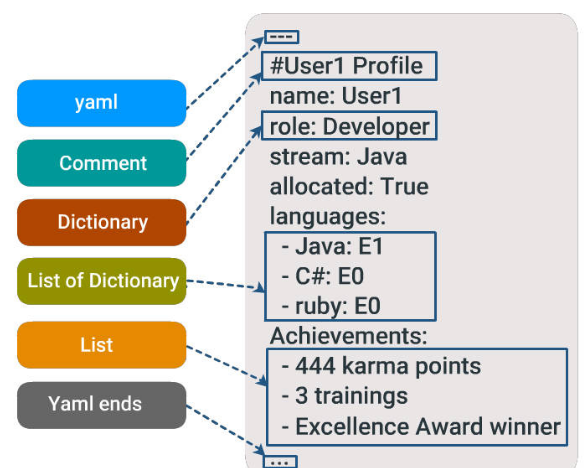
here “target1” is the alias name of the server(which we would have mentioned in the inventory.txt file) is the server alias, inventory.txt is the file where all these entries are present



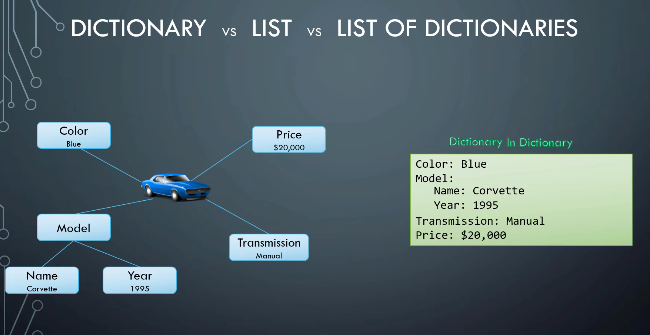
# Yaml standards



* **Members of a list should be at the same indentation level** starting with a dash(-) and space.
* Each item in the list is a **key: value pair (colon must be followed by a space)**, called as **dictionary**.
* At each level, exactly **two spaces are used for indentation**. Using tabs is not recommended here.



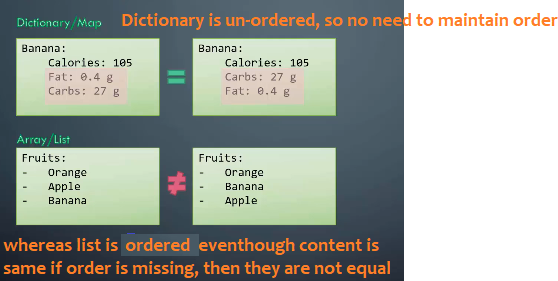
## Dictionary inside a dictionary



Actually, key-value pairs is nothing but a dictionary,

Here inside model object we have another small dictionary

## Dictionary vs list



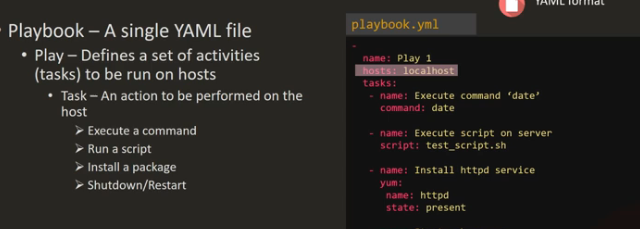
Example:- if we mention tasks as a list, in same order that tasks will be executed, because list is ordered, whereas dictionary is un-ordered

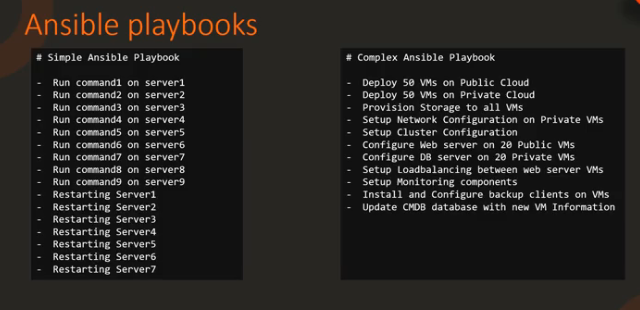
# Playbooks

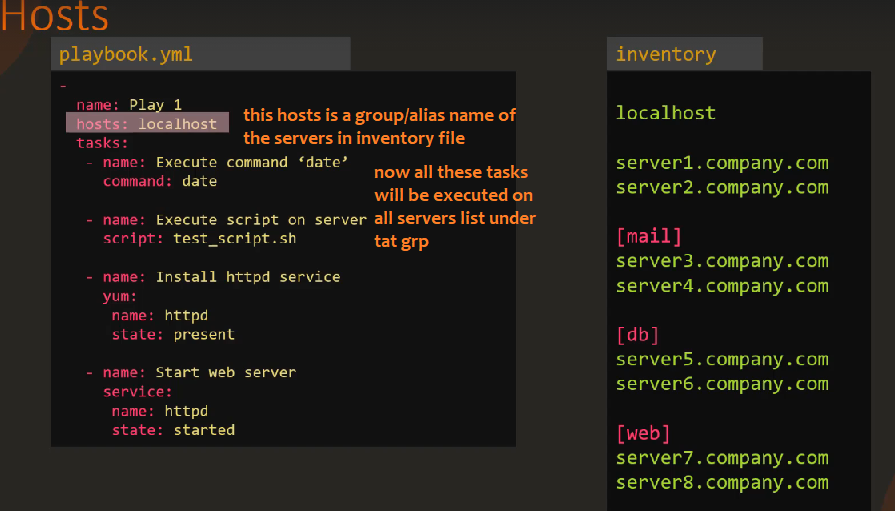
**A Playbook is a file that defines the desired state of your system.**

It contains plays, which has a list of tasks to run in sequence against a list of hosts.

* A play is set of tasks, grouped together to achieve an objective
* A task is an **instruction** you give to Ansible.
* They are written in **YAML** format, a **data serialization language**, that we discussed in previous cards.







Sample playbook files



attached sample playbook and sample inventory file



---

- name: this play displays "hello world"

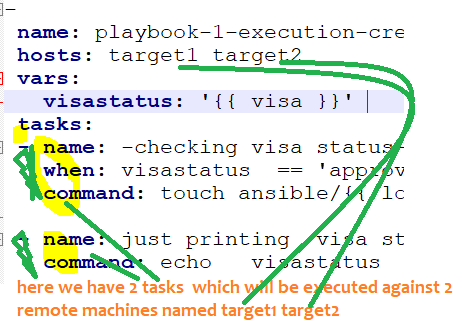
hosts: all

A Playbook always starts with three dashes ---.

* name tells the name of the play.
* hosts tell the list of hosts on which this play will be played.,

u can check the syntax of the playbook using following condition

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



mean all the tasks we mention will be executed on all the servers

##### **Playbook Step 2 - Tasks**

tasks:

- name: displaying "hello world"

shell: echo "hello world"

- name: displaying wishes for the day

shell: echo "have a good day"

//here name is the name of the task, can be any name

And shell is the module name, example module names are command,shell,copy,file …100’s of such are there

* name is optional but is always recommended as it improves readability
* shell: echo "hello world" is a single task. This executes **shell module** and calls its **echo argument** to display the message written

Issue while executing with ssh pass

Generally, the inventory files contains ssh passwords, whereas in prod or in kodekloud when u execute a playbook with an inventory

File contains a password u will get below error then execute this command (export ANSIBLE\_HOST\_KEY\_CHECKING=False)

After executing this command u will be able to successfully run the playbook file

Issue while executing the playbook

 "FAILED! Using a SSH password instead of a key in not possible because Host Jey checking is enabled..."

### **Executing a playbook**

If u don’t know type “ansible-playbook --help”

* **ansible-playbook**: *This is the tool used to****run ansible playbook***

$ ansible-playbook <filename.yml> ... [options]

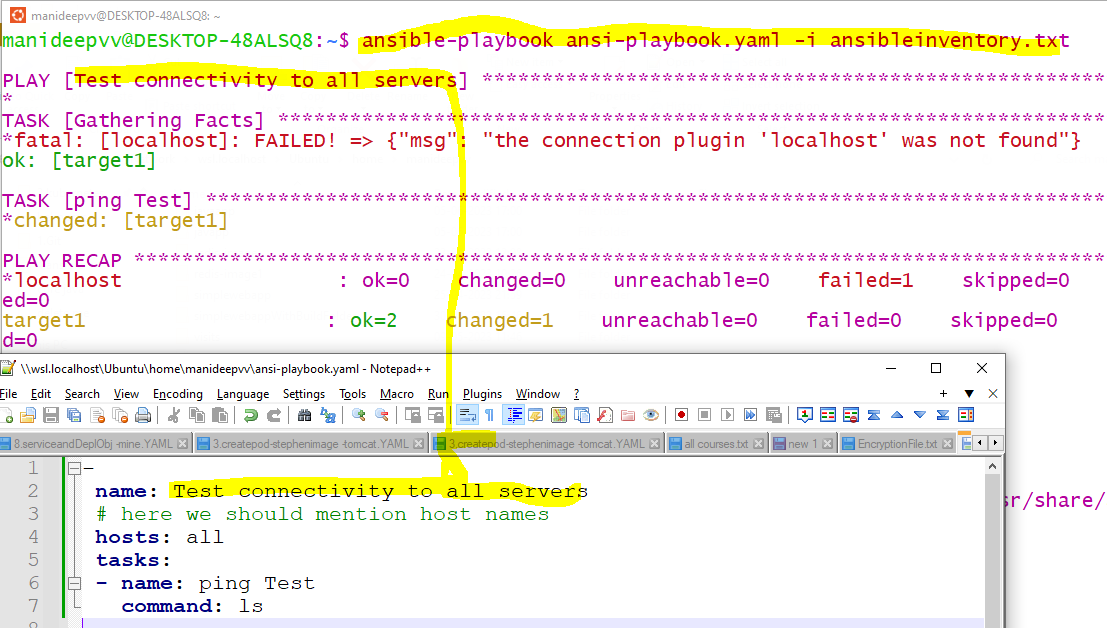
Way -1: providing inventory file

Command:- “ansible-playbook <playbook file name> -i <inventory-file-name>”

**Here providing inventory file name is mandatory**,

Because in ansible-playbook we will mention the hostname, but the actual host address will be present only in the inventory files

hosts: target1 target2 target3 hence giving inventory file is also mandatory



Way2: command without giving inventory file

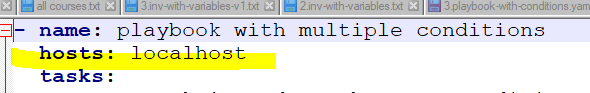
Generally, we don’t use this in production, we will use only in local

Cmd:-

ansible-playbook <play-book-name>

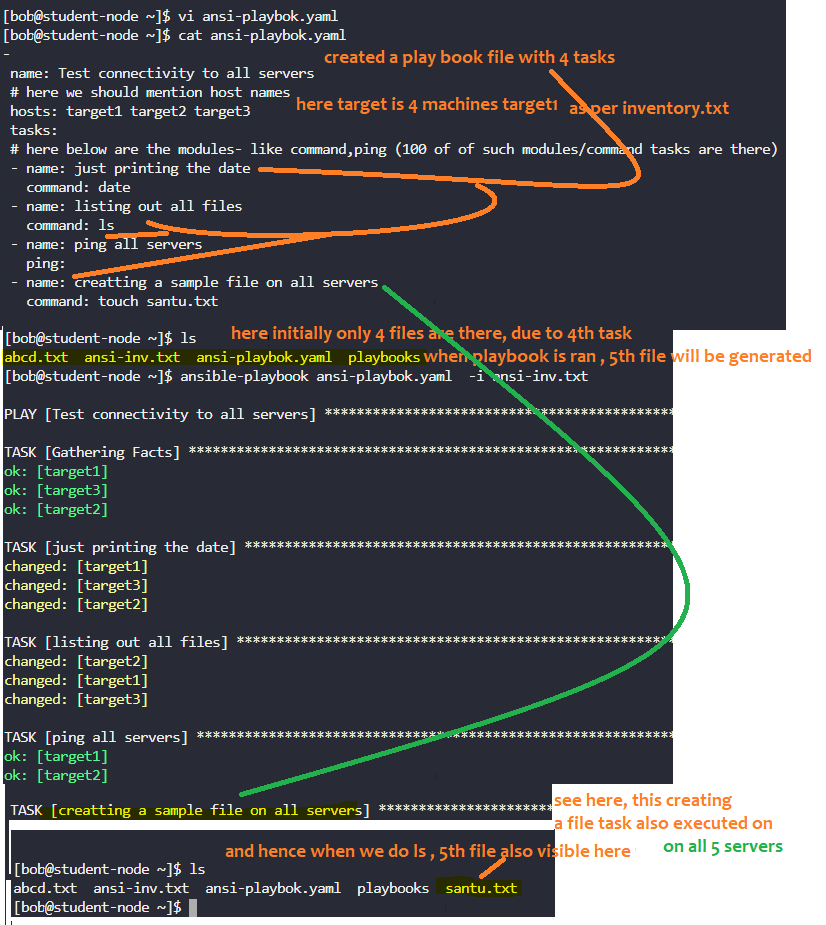
ansible-playbook 9.playbook-with-conditions-when.yaml

here I didn’t provide the inv file because inside playbook under hosts I didn’t used any alias name, I directly used host name as below

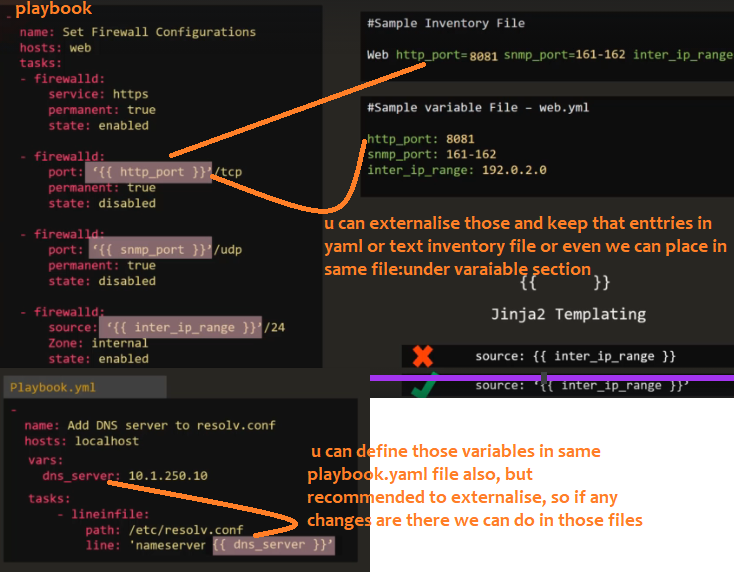


2. Exec a playbook with code

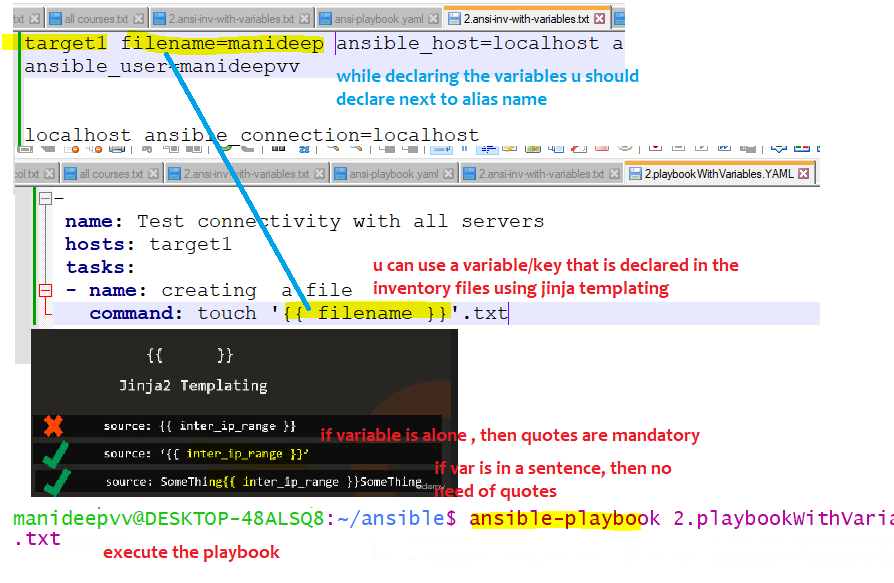




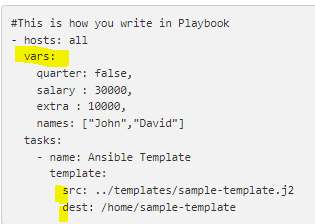
Variables inside a playbook







Here these keys are scoped to a server, same key can be present in another server also with different value



Conditions-when

We can add some conditions to execute a task, like “this task should be executed only when this condition is satisfied”

Usage of when, or, and conditions inside a playbook

Sample

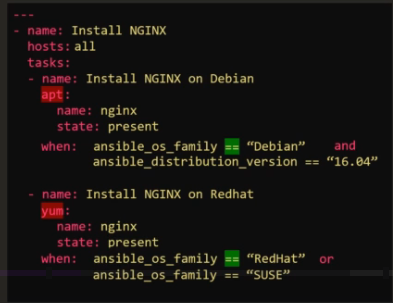
**When clause in Ansible is a raw jinja2 expression that defines the condition which will be evaluated for TRUE or FALSE.**

tasks:

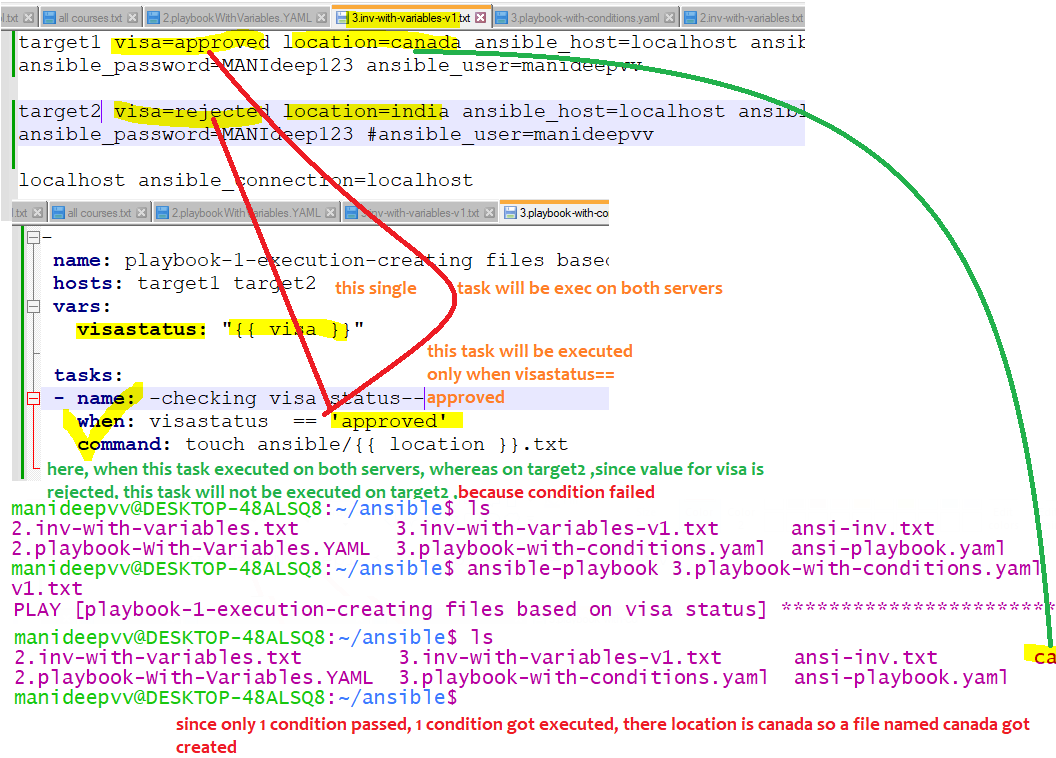
- name: "shutdown Debian flavored systems"

command: /sbin/shutdown -t now

when: ansible\_os\_family == "Debian" #here ansible\_os\_family is an inbuilt variable

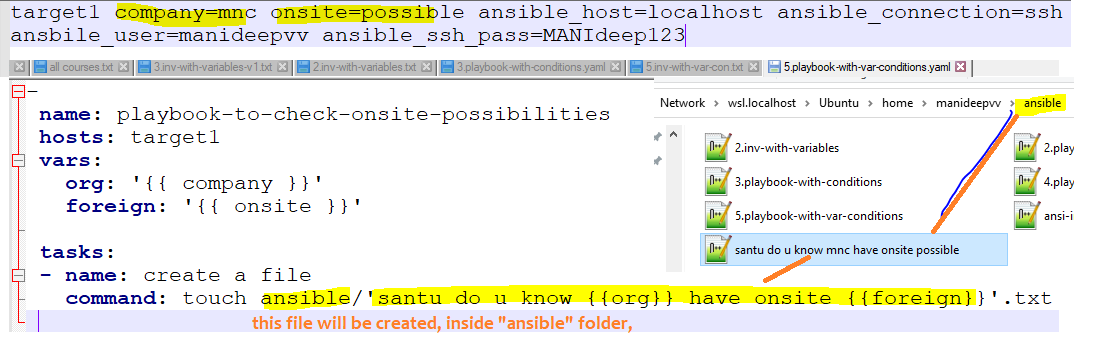


**Sample code with conditions**



Working files for variables-conditions





#### When condition-AND vs Or

tasks:

- name: "shutdown CentOS 6 and Debian 7 systems"

command: /sbin/shutdown -t now

when: (ansible\_distribution == "CentOS" and ansible\_distribution\_major\_version == "6") or

(ansible\_distribution == "Debian" and ansible\_distribution\_major\_version == "7")

* You can **define multiple conditions**, where all of them should be true to execute the tasks

tasks:

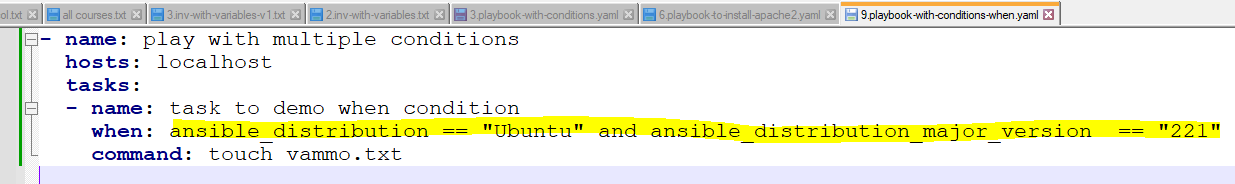
- name: "shut down CentOS 6 systems"

command: /sbin/shutdown -t now

when:

- ansible\_distribution == "CentOS"

- ansible\_distribution\_major\_version == "6"



AND & OR conditions demo

when: ansible\_distribution == "Ubuntu" and ansible\_distribution\_major\_version == "22"

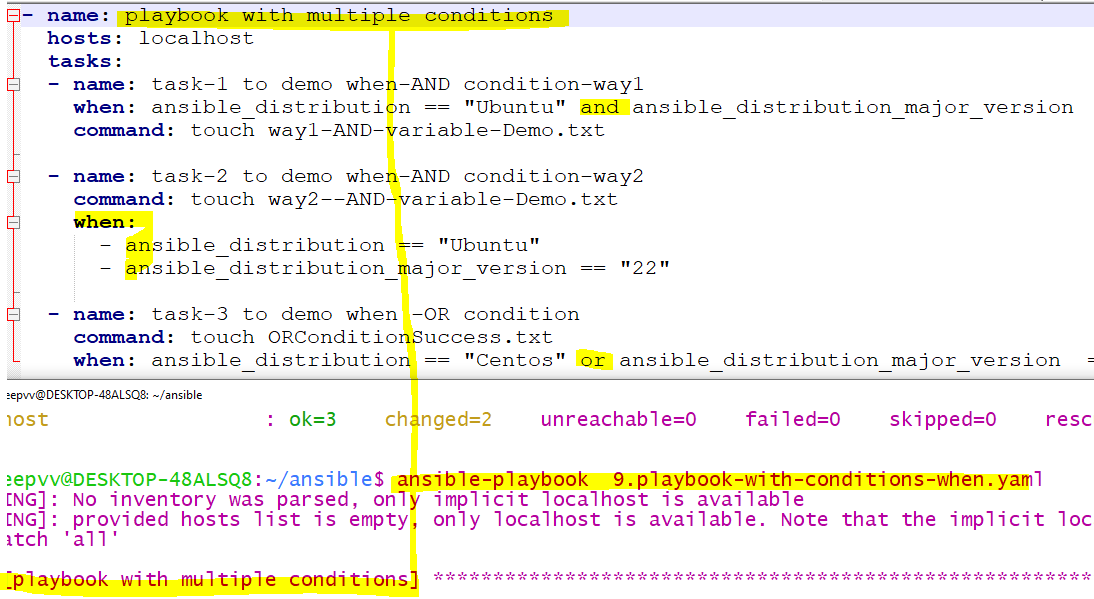
when:

- ansible\_distribution == "Ubuntu"

- ansible\_distribution\_major\_version == "22"

when: ansible\_distribution == "Centos" or ansible\_distribution\_major\_version == "22"





Example:3

This Playbook will add Java Packages to different systems (handling Ubuntu/Debian OS)

- name: debian | ubuntu | add java ppa repo

apt\_repository:

repo=ppa:webupd8team/java

state=present

become: yes

when: ansible\_distribution == 'Ubuntu'

- name: debian | ensure the webupd8 launchpad apt repository is present

apt\_repository:

repo="{{ item }} http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main"

update\_cache=yes

state=present

with\_items:

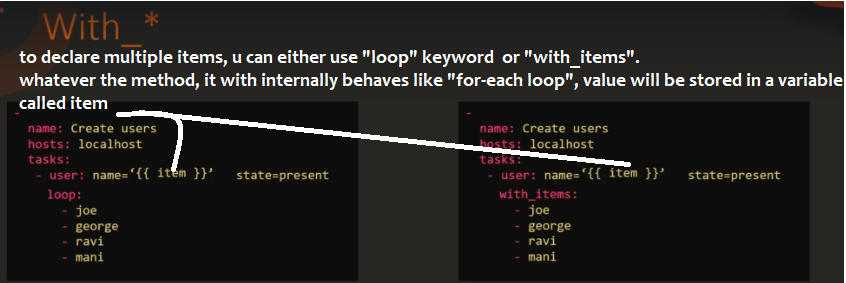
- deb

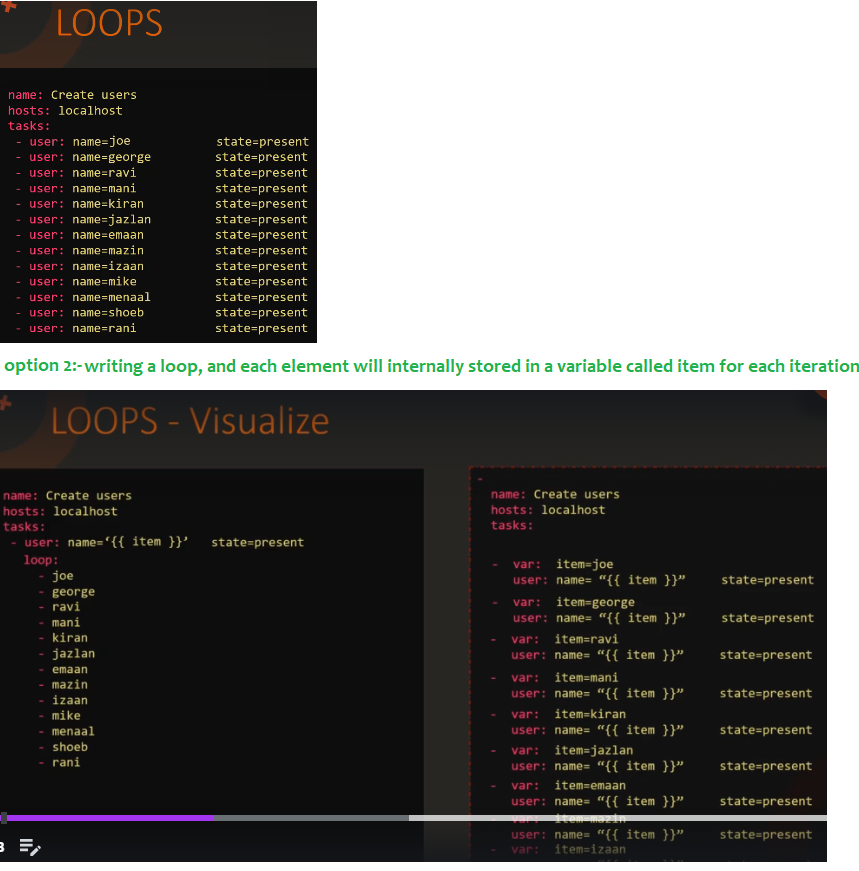
- deb-src

become: yes

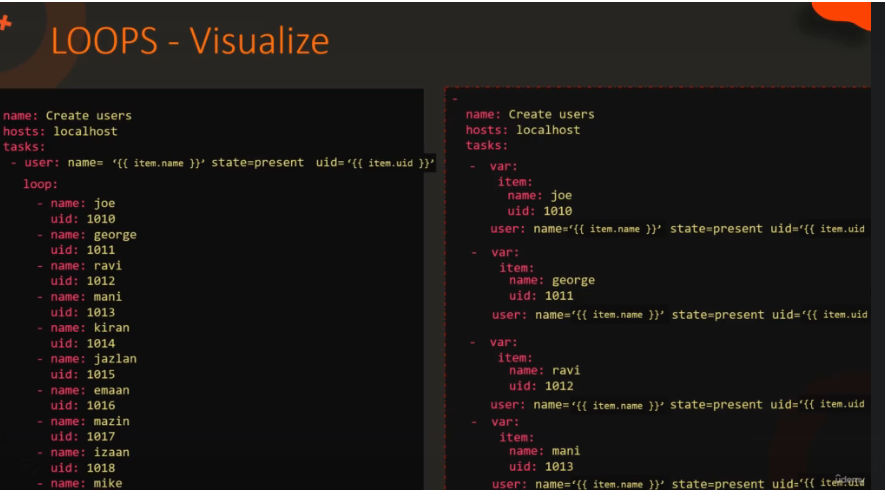
when: ansible\_distribution == 'Debian'

Loops with single variable





Loops with multiple variable



Playbook with shell module

---

- name: this play displays "hello world"

hosts: all

tasks:

- name: displaying "hello world"

shell: echo "hello world"

- name: displaying wishes for the day

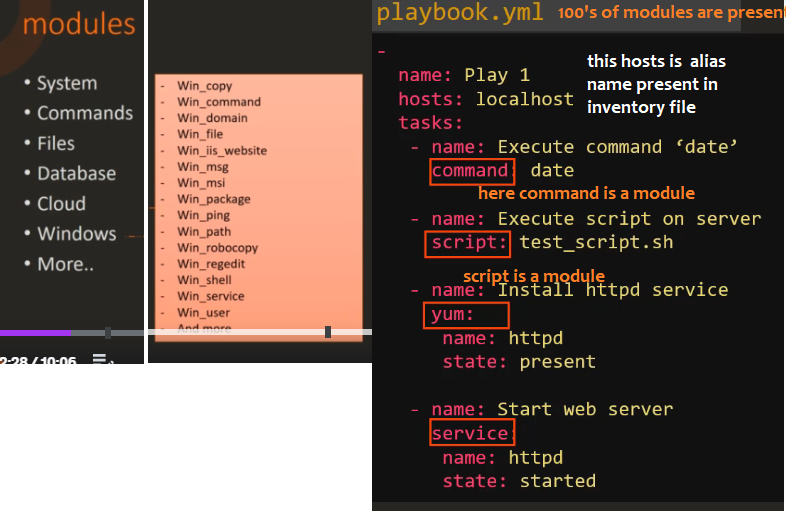
shell: echo "have a good day"

Modules

The different actions run by those tasks are called modules, 100’s of modules are present ex:- command,script,copy

**Modules are a piece of code that gets executed when you run playbook. You use them to describe the state you want the host to be in.**

**Each task in play is made of module and arguments.**



Things You Should Know About Modules

* Also referred as **task plugins** or **library plugins**
* **Default location** for Ansible modules is /usr/share/ansible
* Take arguments in **key=value** format (state=stopped)
* **Returns data in JSON format**
* **Modules should be idempotent**, meaning they should not do any changes if current state of system matches with the desired final state
* To access **list of all installed modules** using command line: ansible-doc -l
* To see the **documentation of particular module** using command line: ansible-doc yum where yum is the module name
* You can **run modules from the command line or include them in Playbook**.
* Ansible allows you to **write your own module** (this you will learn later in advanced courses of Ansible).

Let us now go through some standard modules: *apt*, *yum*, *shell*, *command*, and *template*.

Example modules like –script,

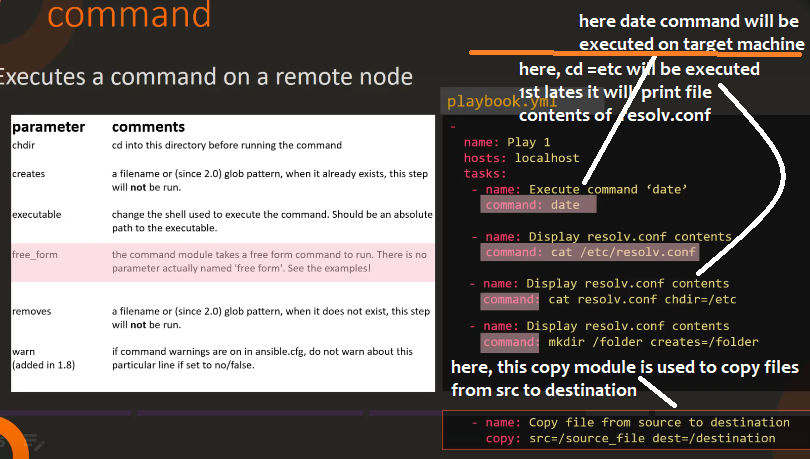
Command module

This is not idempotent, if u run playbook 100 times,ansible will execute command 100 times,

We can use command module to create some files and folders

ansible <host alias name> -i <inventory file name> -m command -a "any cmnd"

ansible target1 -i 3.inv-with-variables-v1.txt -m command -a "mkdir test"



In command module, the command name is followed by **arguments, which does not run on remote hosts through a shell(/bin/sh)**.

You **cannot use** various operations(|,<,> etc) and environment variable(#HOME).

* **Make a directory in remote host**: ansible host01 -i myhosts -m command -a "mkdir folder1"
* **Check files or folders in remote host**: ansible host01 -i myhosts -m command -a "ls"

*Now to check files or folders in****your terminal****use ls and observe the output. As you can see,****using command, you can execute tasks on remote host****.*

**Most of the Ansible modules are idempotent**.

Idempotent means lets say (server=started),means how many time u run it will ensure server is started, where as it wont start that many times

But command module does not exhibit this property as this **runs every time you run playbook**. Hence you will always find the changed status on running the same Playbook again and again.

Consider you wrote a task to copy a file to remote hosts using command module.

Ansible **command module will copy the same file the number of times you run the Playbook**.

Had this been idempotent; Ansible will not copy from the second time, as the file is already present (current state = desired state).

Command can be Idempotent

To overcome this, you can use creates or remove parameter, where you define the filename/pattern.

* creates: if filename **exists**, task will not run
* remove: if filename **does not exist**, task will not run

#This is how you write in Playbook

- name: copy a file, but do not copy if the file already exists

command: cp /home/dist/file1.txt /usr/someFolder/ creates=file1.txt

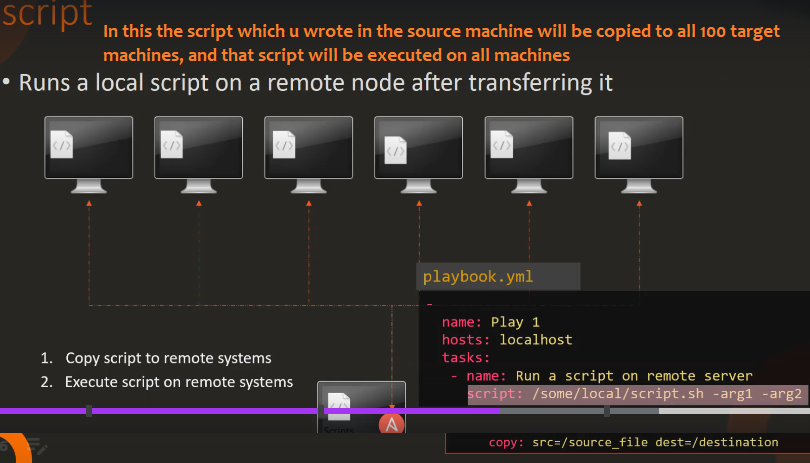
Command to move the files from one to another location

Sample playbook

Alert: this below code is not working, but just kept for reference



Script module



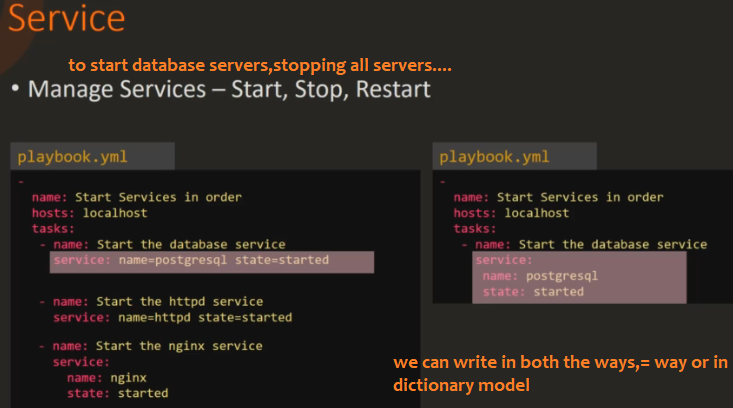
Service module

This is mainly used when u want to start and stop a software, like ssh server 



Let say , if u want to start tomcat server or ssh server or jboss server on all 10 remote machines, then simply execute a playbook

Which will have individual tasks to start server on the remote machine





Files module

1. creating directories and files

You can use **file module** to create files and directories, manage their permissions and ownership as shown:

ansible host01 -i myhosts -m file -a "dest=/tmp/test mode=644 state=directory"

* This will create **directory /tmp/test** on all the host01 of myhosts group
* mode defines permission of file/directory
* state can take value: file, directory, link, absent, etc

You can set the state to **absent** to delete a file or directory to delete it:

$ ansible host01 -i myhosts -m file -a "dest=/tmp/test state=absent"

APT module

This APT module or yum module is to install (may be download and install) new software , where to start those downloaded software’s like nginx,

Tomcat we need service module 

**APT (Advanced Package Tool) is a command-line tool used to easily manage (install, remove, search, etc.) packages on Ubuntu/Debian based Linux systems.**

**Debian Based OS**: Ubuntu, Kali Linux, SteamOS and much more.

$ ansible host01 -i myhosts -m apt -a "name=sudo state=latest"

#This is how you write in Playbook

- name: Upgrade sudo to the latest version

apt:`

name: sudo

state: latest

Example 2:- use APT module to install apache2 on all target servers

This is how your Playbook would look:

---

- name: install apache

hosts: all

sudo: yes

tasks:

- name: install apache2

apt: name=apache2 update\_cache=yes state=latest

You can define the same playbook as: whereas in above module under apt we defied dictionary as a line, here we will define as yaml fomat

---

- name: install apache

hosts: all

sudo: yes

tasks:

- name: install apache2

apt:

name: apache2

update\_cache: yes

state: latest

* observe **colon : is used while structuring arguments vertically**, whereas equal to sign = is used while structuring arguments horizontally

*Both ways of structuring your playbook is fine*.

* **This vertical structuring of arguments is not a list, as list starts with dash sign -**.
* **It is just like as keeping a dictionary inside apt**

Here **name, update\_cache and state are arguments of module apt**, hence they do not start with -.

Shell module

echo something

=======

ansible <host alias name> -i <inventory-file-name> -m shell -a "echo abcd"

ansible target1 -i 3.inv-with-variables-v1.txt -m shell -a "echo abcd"

here –m says module, -a means argument

In shell module, the command name is followed by arguments, which **runs** on remote hosts through a shell(/bin/sh).

You **can use** various operations(|,<,> etc) and environment variable(#HOME).

ansible host01 -i myhosts -m shell -a "echo $TERM": This displays the terminal name of host machine

#This is how you write in Playbook

- name: Execute the command in remote shell

shell: echo $TERM

### Sample Playbook 1 - "demo.yml"

---

- name: this play displays "hello world"

hosts: all

tasks:

- name: displaying "hello world"

shell: echo "hello world"

- name: displaying wishes for the day

shell: echo "have a good day"

Misc ansible commands

## Commands help

*To know more about each module you can try: ansible-doc ping*.

**ansible-console**: *This is a REPL using which you can run ad-hoc commands on chosen inventories*.

$ ansible-console <host-pattern> [-m module\_name] [-a args] [options]

* **ansible-pull**: *This inverts the default push architecture of Ansible into a pull architecture, which has near-limitless scaling potential*.

ansible-pull -U URL [options] [ <filename.yml> ]

* **ansible-doc**: *Displays data on modules installed in Ansible libraries*.

$ ansible-doc [-M module\_path] [-l] [-s] [module...]

* **ansible-vault**: *Using this you can****encrypt****any structured data file used by Ansible*.

$ ansible-vault [create|decrypt|edit|encrypt|rekey] [--help] [options] file\_name

* **ansible-galaxy**: *This is a shared****repository for Ansible roles****. This ansible-galaxy command can be utilized to manage these roles, or to create a skeleton framework for the roles to be uploaded to Galaxy.*

$ ansible-galaxy [delete|import|info|init|install|list|login|remove|search|setup] [--help] [options]

* a: This tells the **arguments** to pass to the module
* -m: Execute the **module**
* -b: Use **privilege escalation (become)**
* -i: The path to the **inventory**, which defaults to /etc/ansible/hosts
* --version: Show program **version number**
* --help: Shows help message

**An ad-hoc command is a single statement to complete a particular task. For example: consider you want to check if you could connect to your hosts.**

Enter the following command:

ansible group1 -i myhosts -m ping

The above statement is a **single task** to **ping** target host and **return pong** if the connection is successful.

* ansible is a **keyword** you need to write before running any ad-hoc command
* group1 is the group name of the **list of hosts**
* -m means **module**, this is followed by the **module name** ping, which will be executed to achieve the task

To know more about each module you can try: *ansible-doc ping*.

## Copy file to servers

You can use **copy module** to copy a file from your **control machine to host** as shown:

$ touch test.txt

* This will create a sample file which could be used to copy

$ ansible host01 -i myhosts -m copy -a "src=test.txt dest=/tmp/"

* copy the file test.txt **from your control machine** (where ansible is installed) **to all the hosts** defined in myhosts inventory group
* -a means **arguments** of that module (here copy module)
* src is **attribute of copy module** that defines the **source path** of file or directory **on control machine**

Similarly, to fetch a file from Host to your Control Machine, you can use ***fetch module***. You may use *ansible-doc fetch* to know about it.

Encryption-decryption

##### **Encrypting Your File**

As you just created a test.txt file, let us now encrypt the same using **ansible-vault keyword**.

* $ ansible-vault encrypt test.txt: encrypts the file.

This asks for a password to be set. Give a password and confirm it.

* ansible-vault edit test.txt: to edit the file and add some content.

This opens vi editor. Type some text, then save it(:wq)

* cat test.txt: to view the content inside.

Observe the output carefully

* ansible-vault decrypt test.txt: to decrypt the file, use the password set during encryption
* cat test.txt: now observe the output

loops

- name: debian | ubuntu | add java ppa repo

apt\_repository:

repo=ppa:webupd8team/java

state=present

become: yes

when: ansible\_distribution == 'Ubuntu'

- name: debian | ensure the webupd8 launchpad apt repository is present

apt\_repository:

repo="{{ item }} http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main"

update\_cache=yes

state=present

with\_items:

- deb

- deb-src

become: yes

when: ansible\_distribution == 'Debian'