

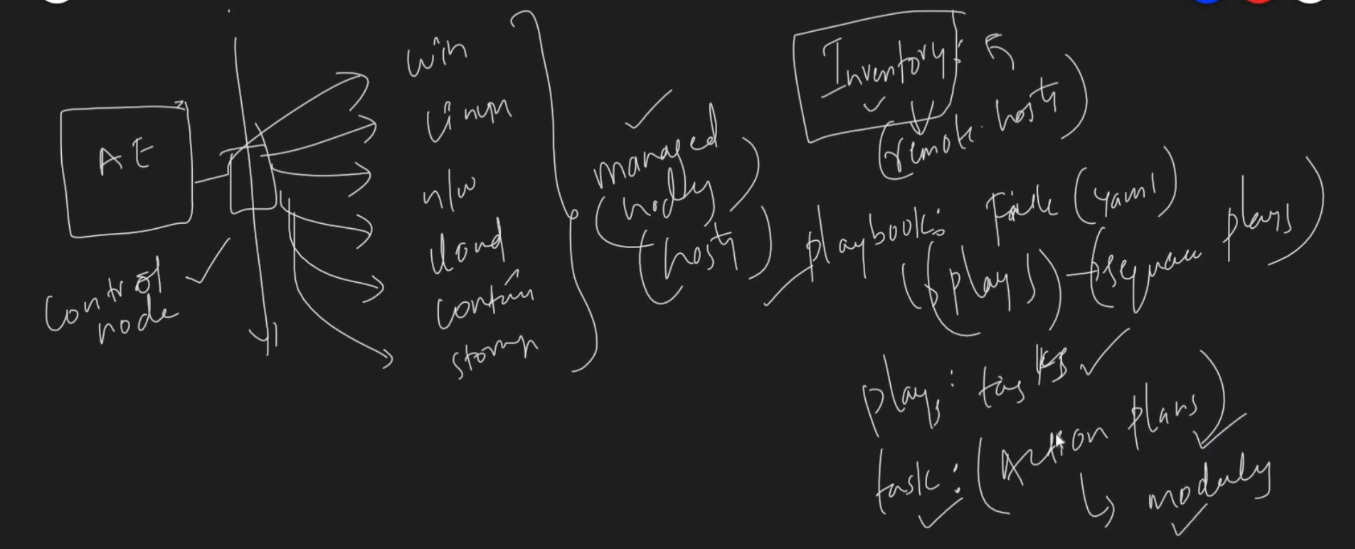
From above:

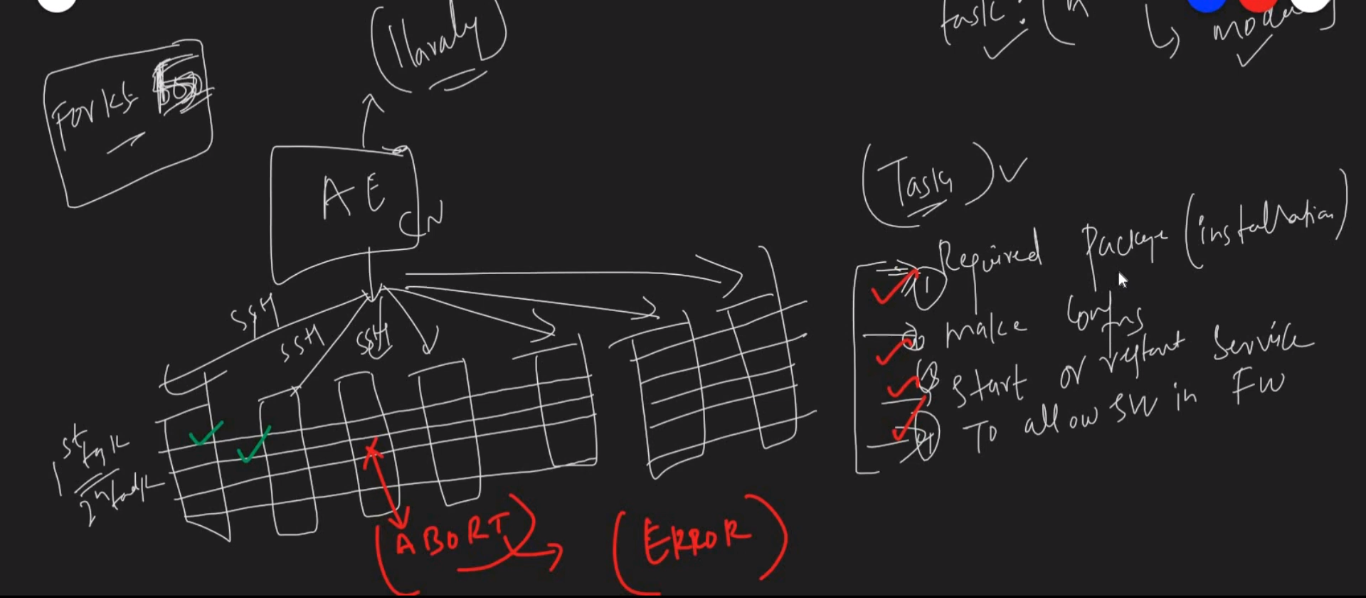
AE – Control node (ANSIBLE is agentless)

Clients – Managed nodes/Hosts

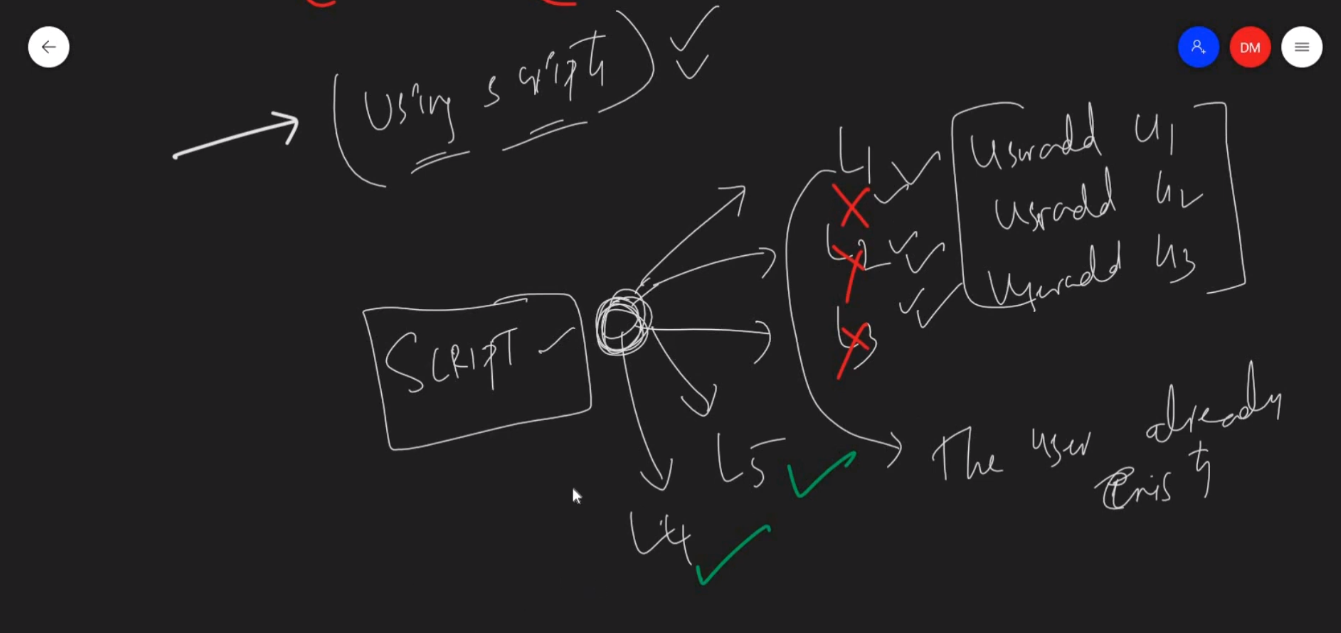
Here for running Ansible – In Linux Interpreter is python & for windows, it’s PowerShell.

* For OS like Linux/Win it’s ok, but if we wanna work with network devices, then Ansible can push commands directly and make configs.





* In Ansible, we can utilize ‘FORKS’, which is a feature of setting the minimum no of machines Ansible can run the plays or tasks simultaneously
* It is recommended, to keep fork value less, because, we can catch errors upfront before going for the other 100 nodes.
* Above pic we can see, that if Ansible fails at any step, it aborts there and does not go further.



**INSTALLATION OF ANSIBLE**

[https://www.linode.com/docs/guides/getting-started-with-Ansible/](https://www.linode.com/docs/guides/getting-started-with-ansible/)

Genetare the ssh key in the Control node and copy it to Ansible user in all managed nodes by below cmd:

Create the ansible user (say – ansibleuser) --enable password

ssh-copy-id username@IP/Hostname

eg: ssh-copy-id [root@203.0.113.0](mailto:root@203.0.113.0)

cd /etc/ssh & vi sshd.config 🡪 and make passwordVerification as uncomment

restart the service sshd.

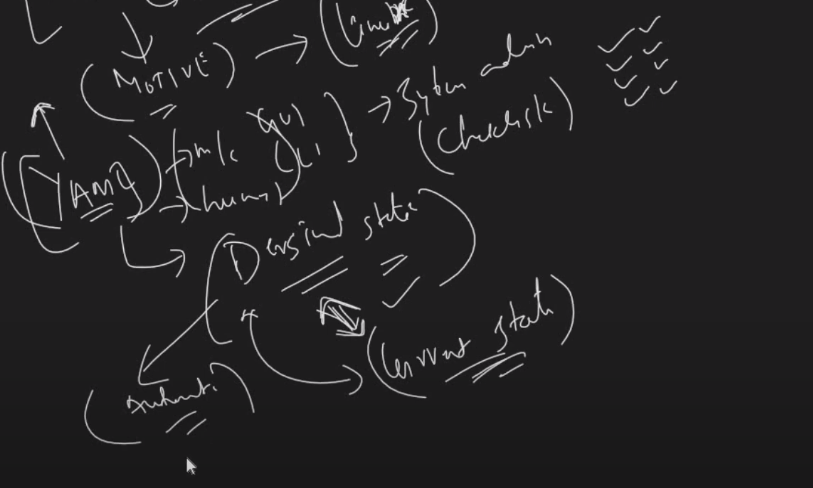
* Create 2 managed nodes, 1 Control Node 🡪 create ansible users [adduser <username>] and give them sudo access as below.
* visudo -- <user> ALL=(ALL) ALL -- <user> ALL=(ALL) NOPASSWD: ALL

sudo usermod -aG sudo <user> . – Add the line like above

* cd /etc/ssh/sshd\_config & add passwordauthentication yes (line)
* generate key for ansible user [ssh-keygen] and copy the key to managed nodes by below command [ssh-copy-id <public-IP>]
* Finally check connectivity as ssh <IP>.

**PART -2**

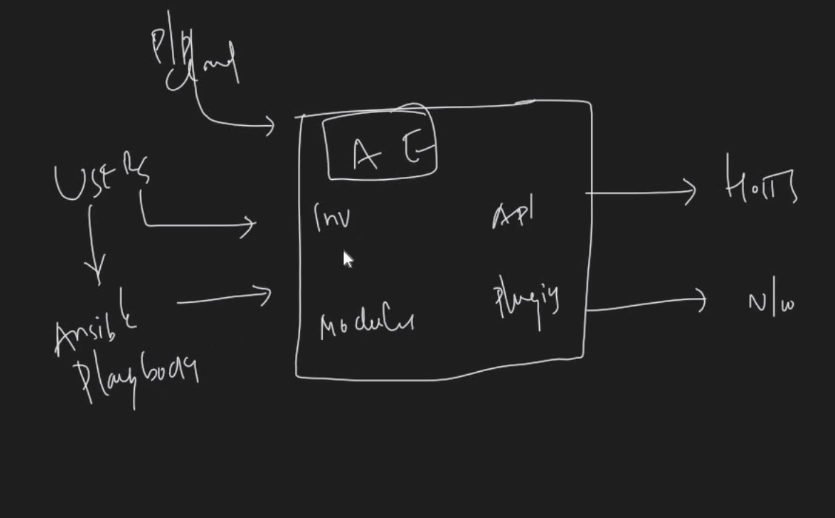
[**https://github.com/TrieTreeTechnologies/ansible-for-devops**](https://github.com/TrieTreeTechnologies/ansible-for-devops)

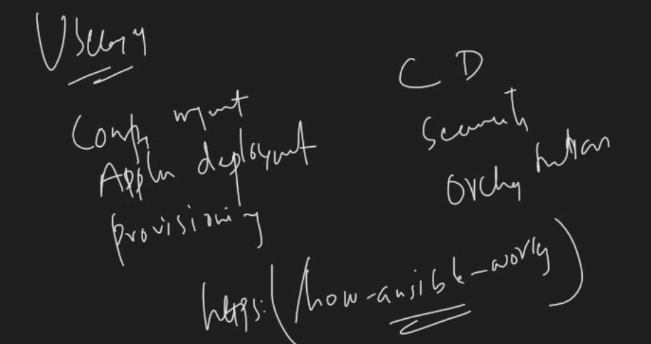


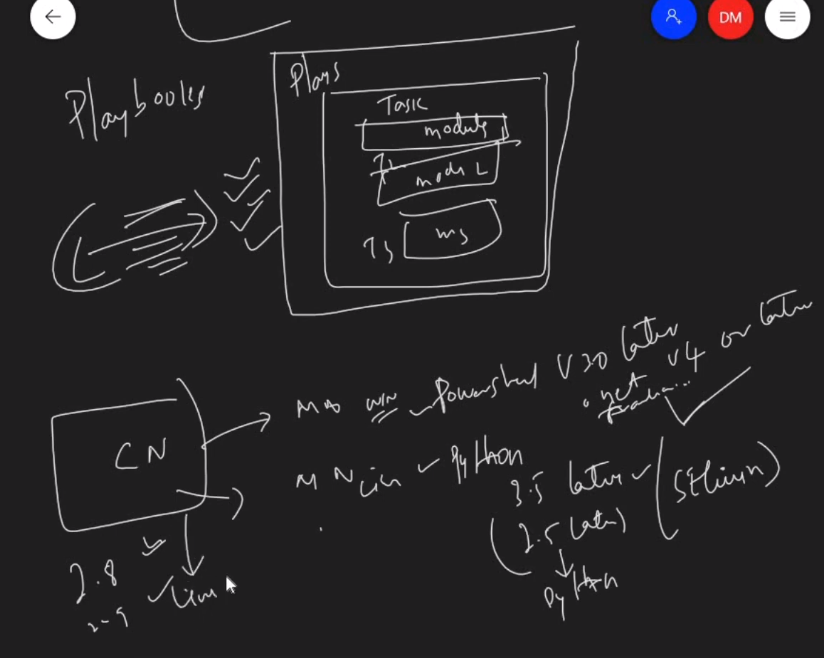
Ansible playbooks – written in YAML

For any Automation is called successful – Desired and current state should be same

The desired state defined in playbooks & current state is a state of a managed host

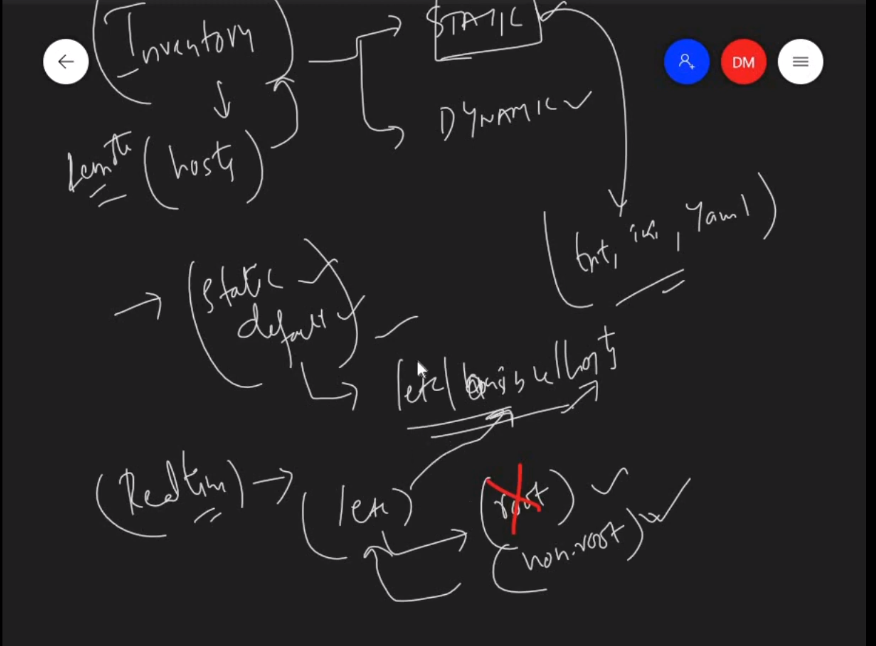
**ARCH of ANSIBLE:-**

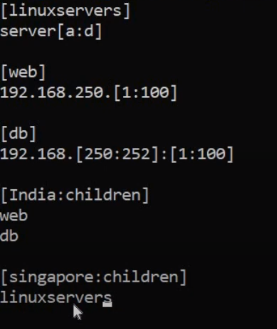




Playbook – Plays –Tasks (Task1,2,3,….,n) – Modules

Playbook(Plays(Tasks[1,2,..,n](modules)))





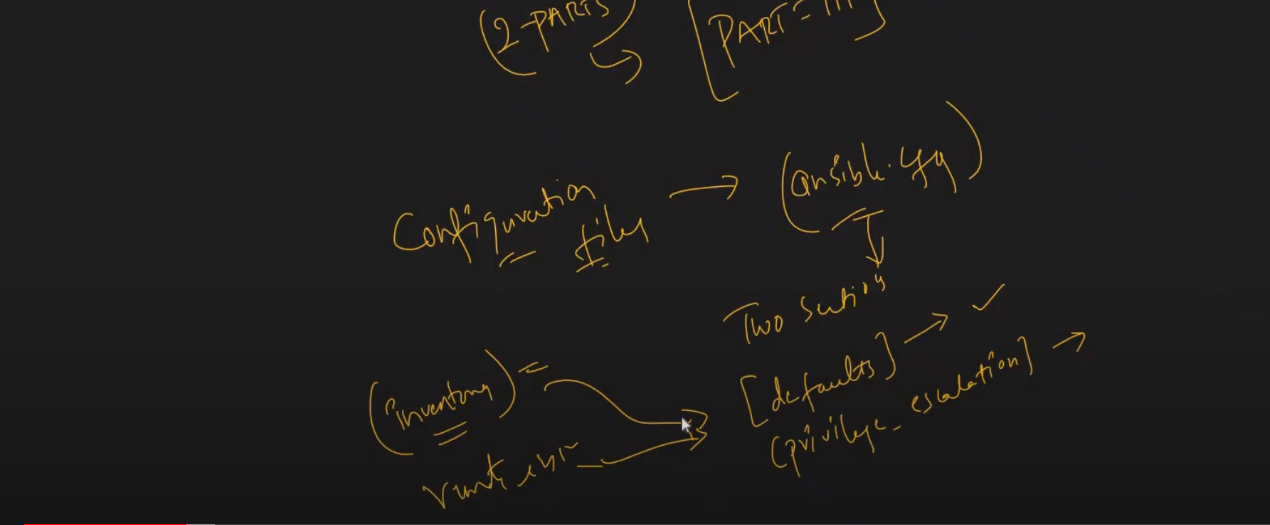
Ansible –list-hosts <inventory-group>

inventory-group = Linux servers/web/DB/India/Singapore/all/ungrouped

ansible-inventory –graph

ansible-inventory –graph –i inventory (This is used when we create a separate project-specific inventory, then, we can use this –I option to consider inventory in the current directory)

**PART-3**

****

Ansible –version – it gives, default inventory file, /etc/ansible/ansible.cfg

There are 2 sections in this – Default and privilege escalation

Since non-root users cannot access `etc` dir 🡪 we can maintain the cfg file under the user’s $HOME dir

* There is no project folder for ansible, then default config is /etc/ansible/ansible.cfg
* If we try to execute under project dir – then it takes default cfg as in users dir
* So, say, we created under /root/ansible\_test/ansible.cfg – then it takes it as default dir

[defaults]

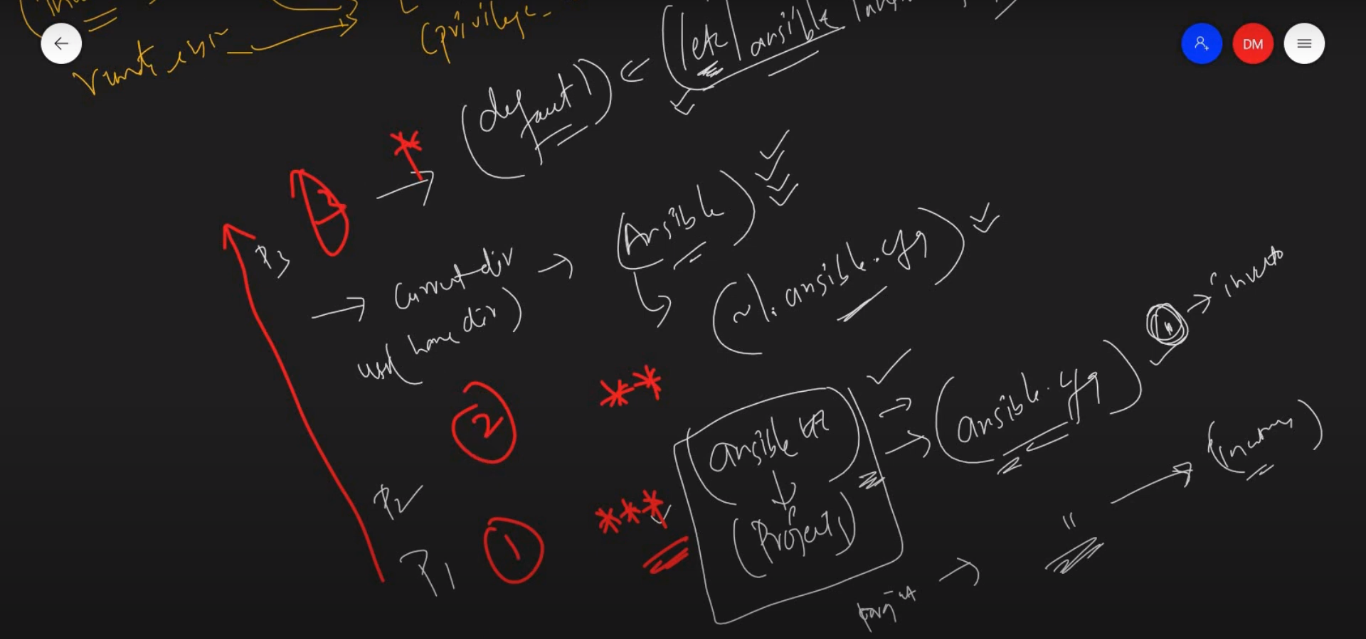
172.23.45.78

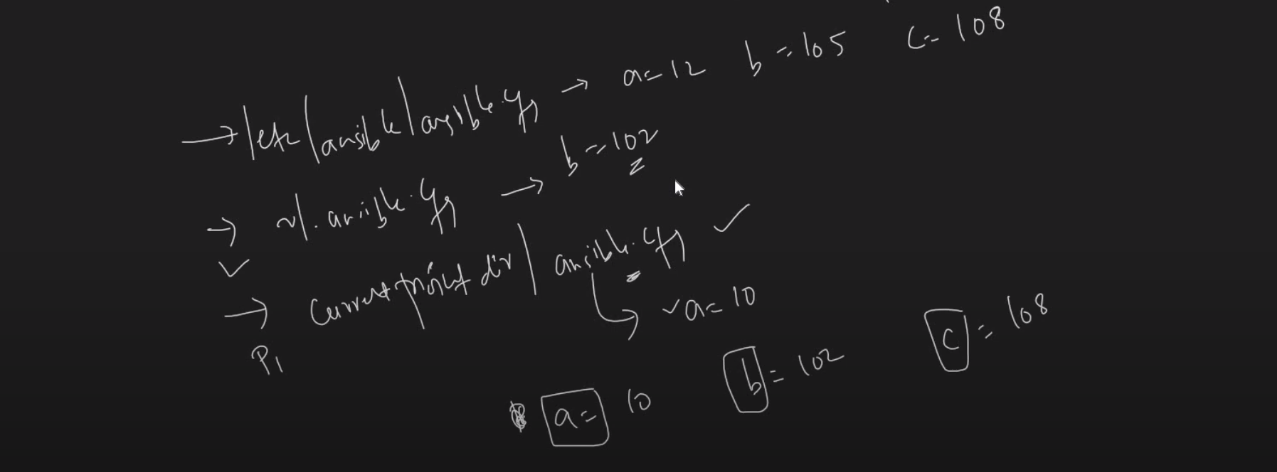
172.23…….

……

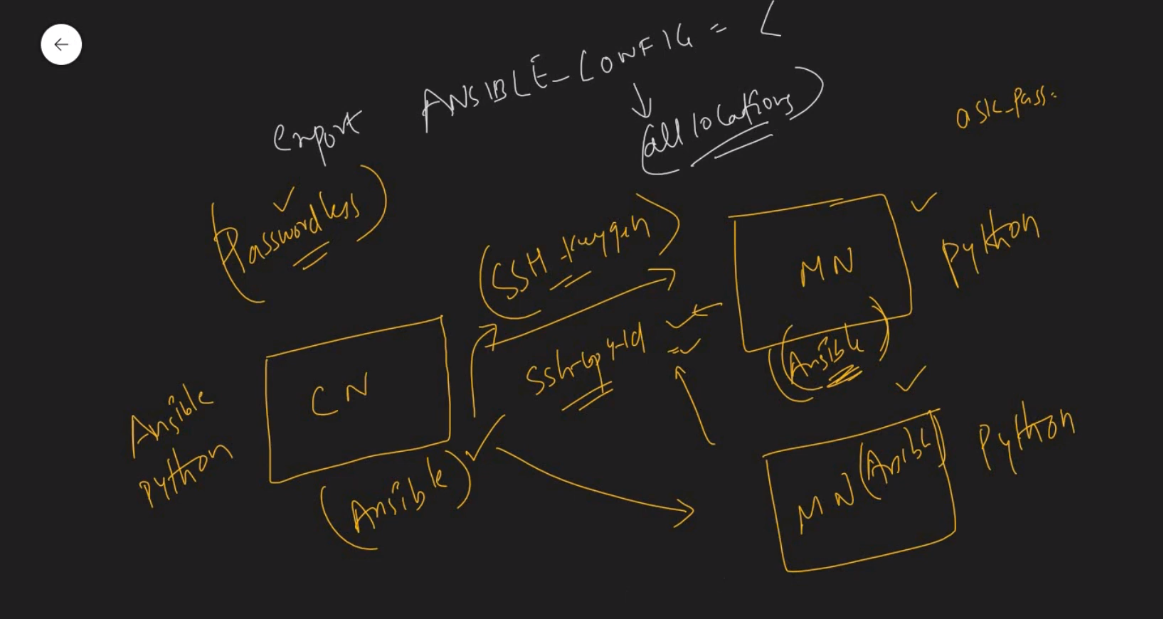
……

If we are executing ansible commands in specific folder/dir and it contains a cfg file like this, it considers it as default cfg file.

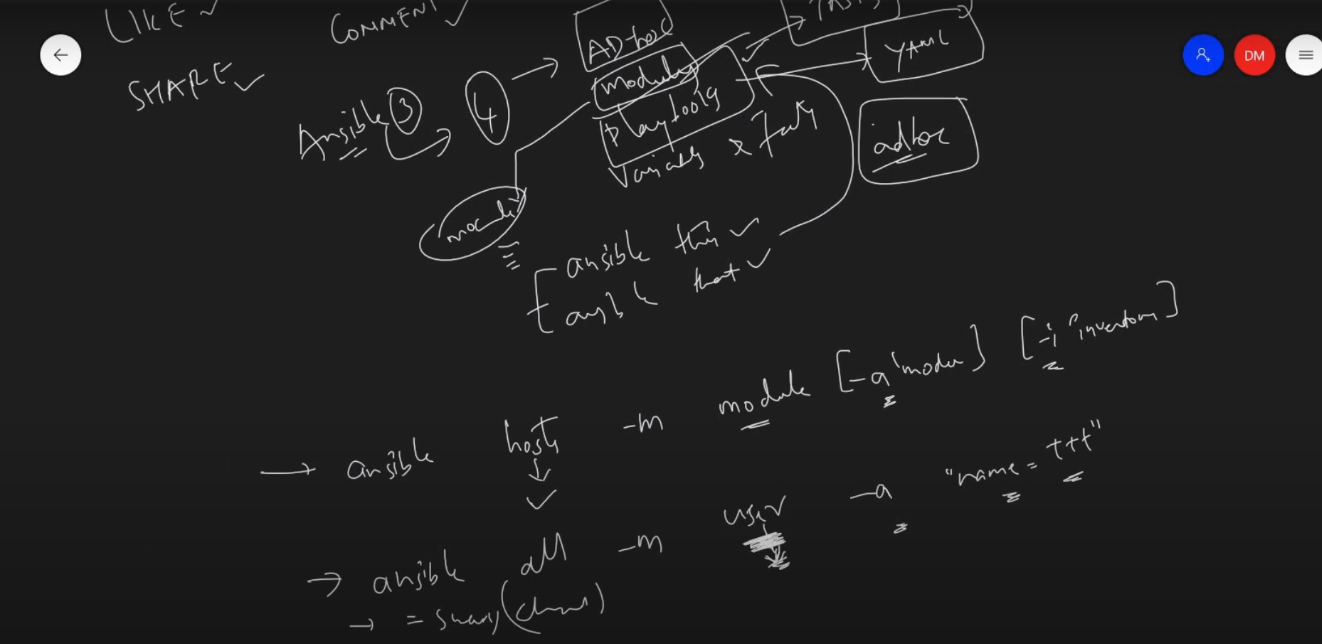




* Above says, if we try to call a , it takes from user dir and gives a=10 (since, it’s priority)
* B, it takes from 2nd priority folder, since we don’t have b in P1 folder
* C it takes from /etc/annsible/ansible.cfg
* We can also export cfg file location by   
  (export ANSIBLE\_CONFIG=/tmp/ansible.cfg) – it takes cfg from here.



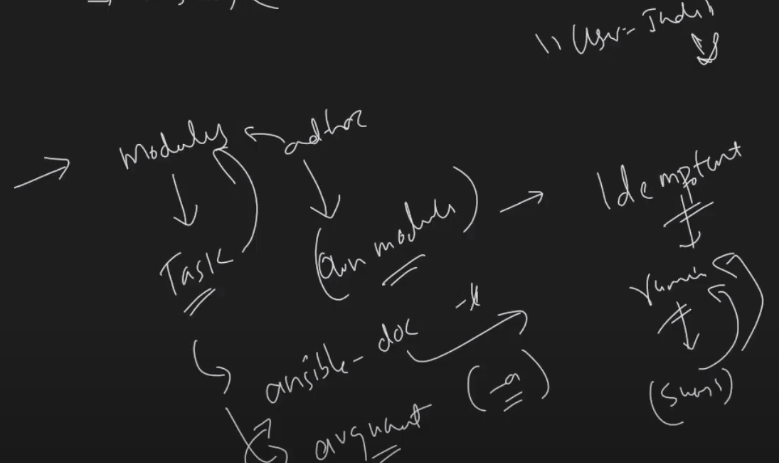
**PART-4**

****

**Adhoc commands** - Ansible ad hoc commands are one-liners designed to achieve a very specific task they are like quick snippets and your compact swiss army knife when you want to do a quick task across multiple machines.

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

Modules – Tasks, ad-hoc commands are defined thru modules, we can even write our own modules.



Eg:-

* ping –check python is there or not In managed hosts
* Service – checks service running/not
* Command – executes general commands
* Shell – {>>,|,^,\*,find} these characters can be sued thru shell module
* Copy – copied files to manages hosts
* Raw – run commands without python, below example is written to install python

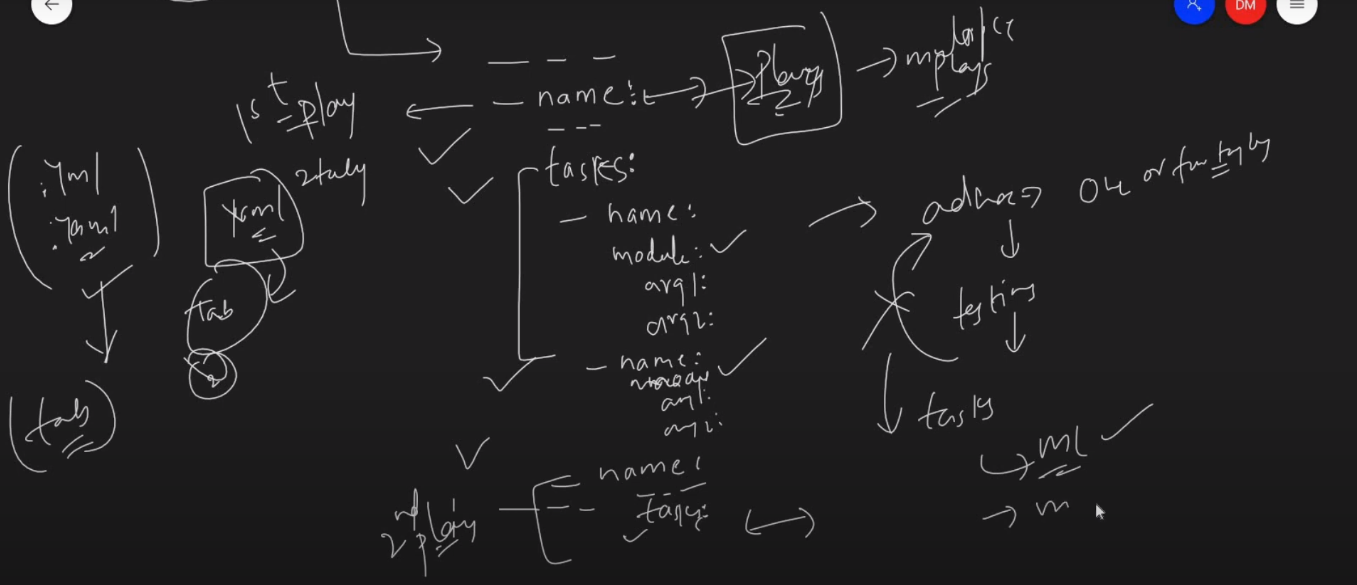
***[ansible -u root -i inventory hosts –ask-pass -m raw -a ‘yum install python3 –y’]***

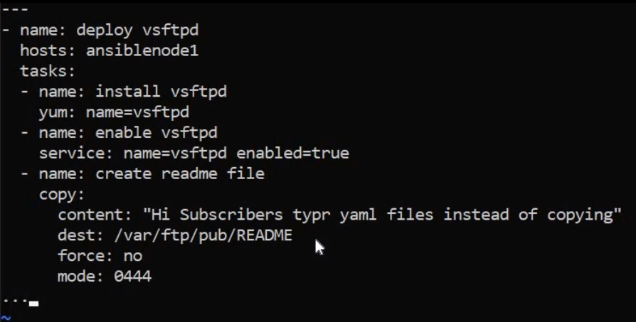
-u = become user ; --ask-pass = asking root password

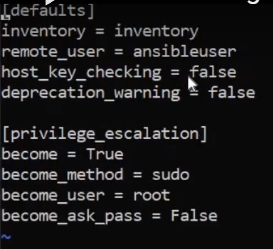
**Playbooks**: It is a metaphor representing the configuration files of Ansible.  It contains a list of tasks  (plays) in an order they should get executed against a set of hosts or a single host based on the configuration specified.  Playbooks are written in YAML, in an easy human-readable syntax

You can consider **ansible ad-hoc commands as shell commands** and a **playbook as a shell script.**

<https://www.middlewareinventory.com/blog/ansible-playbook-example/>







🡨 This one is ansible.cfg file, where I have defined my inbuilt specifications to ansible to run commands like to become root, ask pass etc..

Those also can be passed during running ad-hoc commands also.