**Ansible**

**Session 01–Starting ansible**

* If you want to do anything, OS to OS there are different commands available.
* If you will do things manually there are chances of human error.
  + It is also very slow.
* So, here we need some automation.
  + Here you can say automation with respect of configuration.
* There are lots of tools available in automation world, ansible is a new tool in this world, then why we need ansible?
  + All tools provide you automation, but ansible is a smart – intelligent tool.
  + All scripting language are imperative language.
    - Here you have to provide

1. What to do
2. How to do
   * Ansible works on declarative language.
     + Here you have to only provide
3. What to do
   * + Eg, ansible, ceph, puppet
   * Here we don’t have to provide os respective command.

* …..
  + - We can also get automation using script.
      * For this you have to use languages.

1. Perl
2. Bash shell
3. Shell scripting
4. Python – rubi
5. Lots more…
   * In company we have hydrogenous environment.
     + In company some servers are working on rhel 5, rhel 8, some are working on kali and ubuntu.
     + Here this scripting languages are working on imperative language.
       - You have to provide.
6. What to do
7. How to do
   1. It is a main problem, we don’t which command to use.
      * + If somehow you know all the commands of particular OS.
          - But when company launch new version, which have new commands your script fails.
8. Rhel 1 to 6
   1. service
9. Rhel 7 to 8
   1. Systemctl

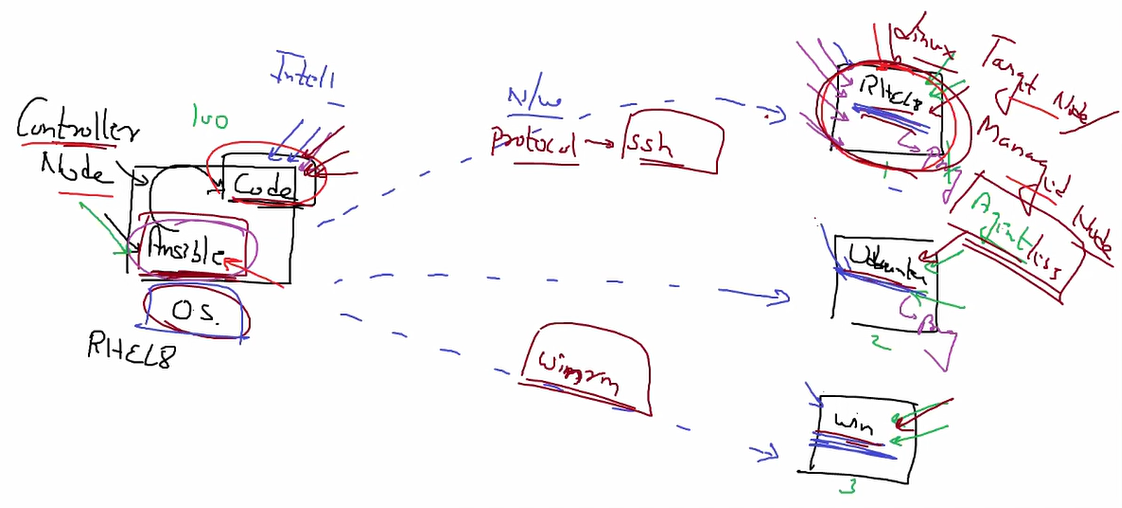
* Configuration management

1. Manual
2. Auto
   1. Imperative language
      1. What to do
      2. How to do
   2. Declarative language
      1. What to do

* Can you use ansible for provisioning OS?
  + Yes, we can.
  + But there are lots of mature tools for doing this.
* Ansible is build on top of python.

**Session 02 – CN-TN-state**

* Where we install ansible, is known as controller node.
  + Using this controller node, we can control target node / managed node.
    - Here you don’t have to install agent. They are agentless.
    - In Ansible competitors we have to install agents.

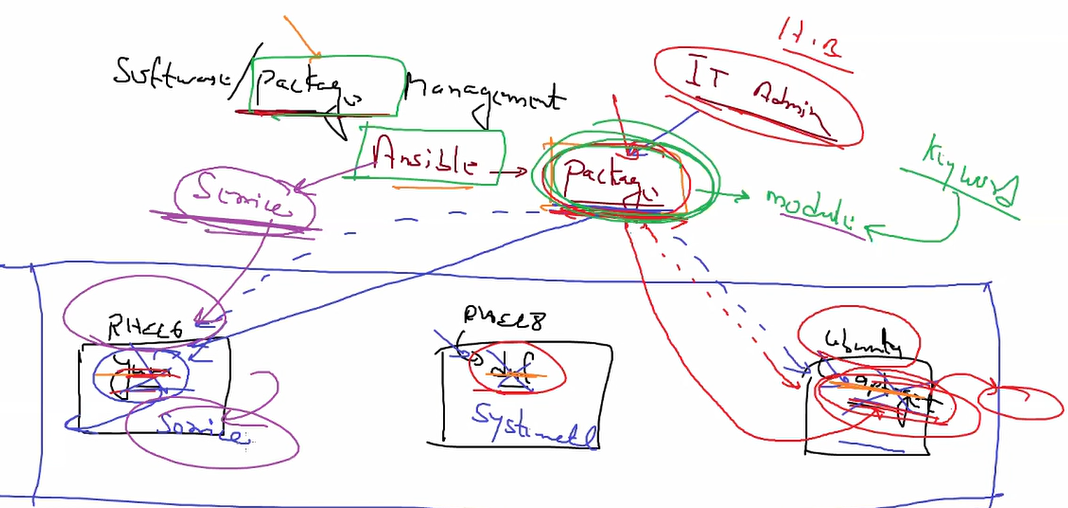


* Ansible is build on top of python, so you need python installed on the system.
  + Pip3 install ansible
* Here you have to provide lists of IP.
  + It is known as **inventory**.
    - Inventory contains.

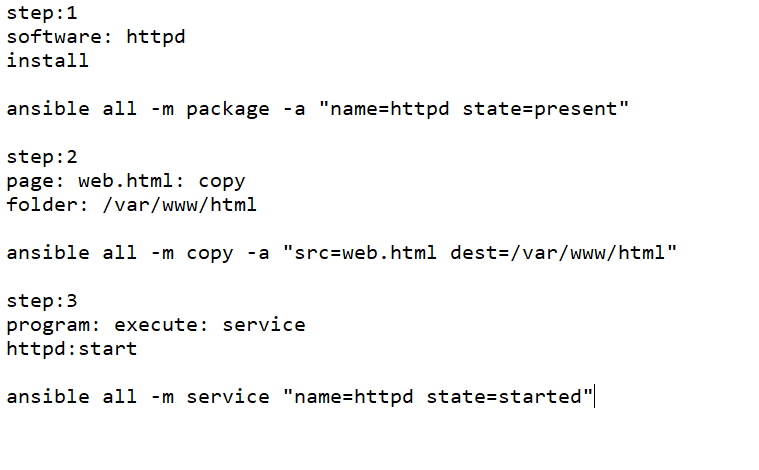
1. Ip
2. Protocol
3. User
4. Password
   * Ansible --version
   * Ansible all --list-hosts
     + It will give you list of hosts / target nodes.
     + It will go to the config file and search for inventory.
   * For creating inventory, you have to create a file.
     + For this you have to create configuration file.
     + Mkdir /etc/ansible
     + Vim /etc/ansible/ansible.cfg
       - [defaults]
       - Inventory= /root/ip.txt
     + Vim /root/ip.txt
       - 192.168.43.128 ansible\_user=root ansible\_ssh\_pass=redhat ansible\_connection=ssh
         * Here if you don’t mention ansible\_connection still it will work.
         * By default, it will use ssh.
     + Ansible all -m service -a “name=httpd state=started”
     + Ansible is also depended on sshpass.
       - So, you need to download this.
       - For this you have can download epel-release, which contain many software locations. After that you can use yum for download sshpass.
     + Yum install sshpass.
   * Ansible will go to target node through network, login there, find OS name and find OS respective command.
   * Ansible always check state of the service.
     + Like if webserver is in running state, then ansible will not start again this service.
     + Desired state = running
     + Current state
     + This thing is known as idempotency.

**Session 03 – ansible intelligence**

* Ansible all -m ping
  + Here ping is a module of ansible which will check connectivity with target node.
* Module
  + Ansible intelligence comes from this.
  + For package management we just have to know one thing package command.
  + Package command go to OS, and ask OS names so, he know which OS is this then he knows internally in rhel 6 this command, rhel 8 this command, ubuntu this command.
    - Package command will call apt-get command inside ubuntu and tell apt-get to install this software.
    - Package command will not be installing anything, he is just saying yum or dnf or apt-get to install this.
  + Ansible is intelligent because of this type of keywords (package), which are known as module.



* + Now we have one single command to run same kind of things in different OS.
    - One standard command.
  + So, you can say ansible have provides you one layer known as RAL (Resource abstraction layer).
    - Here,
      * resource = yum, dnf, apt-get
      * resource = Systemctl, service
* arguments
  + every module come with options known as **arguments**.
  + Here you can provide in this module what you want to do.
    - Name=httpd
    - State=present
  + Here if ansible successfully done his work, reply with rc=0.
* Here in ansible we are doing ssh on our behalf.
  + But in ssh we have to verify host key. Ansible cannot do this.
  + For this we can do one thing.
    - Vim /etc/ansible/ansible.cfg
      * Host\_key\_checking = false
  + Ansible
    - Won’t run anything – green colour
    - Installing – changed – yellow colour
    - Error – red colour
  + Controller node will also **work** as a target node.

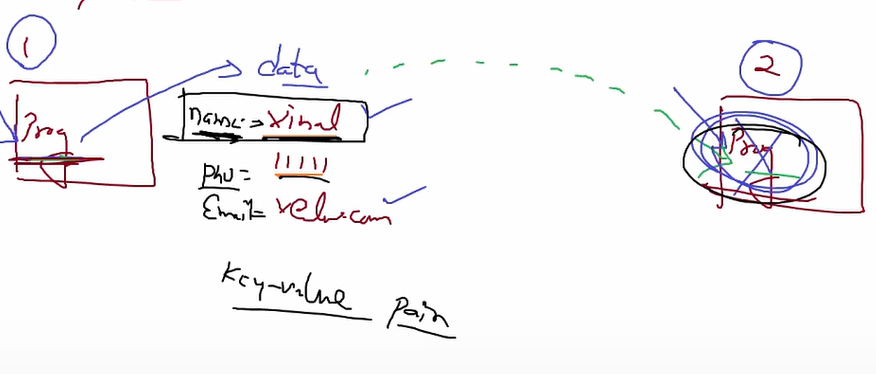


**Session 04 – playbook**

* In ansible there is 2 approach available.

1. Manual
   1. Known as ad-hoc.
   2. CLI
2. Auto
   1. For this you have to create a code inside a program file.
   2. This program files are known as playbook.
   3. 99% we use automatic way for doing things.

* In ansible we have to follow some syntax.
  + Ansible is declarative language, in this type of language we have to use key-value format.
  + For creating playbook, you have to usekey-value format.
* For transferring data between multiple programs, or you can say between OS, we need certain way so we can transfer data to between OS.



* So, here program 2 receiving data from program 1.
  + When name variable comes, it will retrieve information of it.
* Sometimes we have multiple values inside one variable.
  + For this you have to give them same indent started with (-).
* There are three types for key-value pair format.
  + JSON
  + XML
  + YAML
* Ansible support yaml language.
* Vim web.yml
  + All OS where you want to run a code, you have to mention in the top known as host.
    - Hosts: all
      * Hosts = key
      * All = value



* + Here this 2 hosts part are also known as play.
  + We can write multiple play so, hosts is also be indented.
  + For running this file, we have to use command ansible-playbook
    - Ansible-playbook web.yml



* + You can also write like this.
    - Sometimes we have many arguments, writing in single like will be harder.
  + Ansible-playbook --syntax-check <name.yml>
    - It will help you to get syntax error if you have.

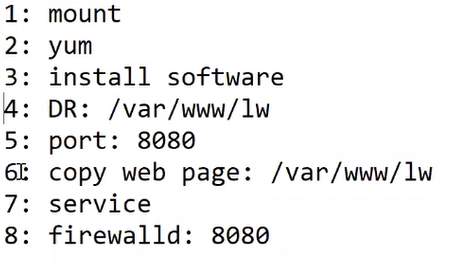
**Session 05 – webserver-firewall**

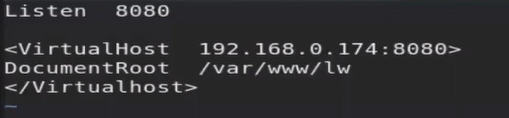
* If yum is not configured then sometimes ansible command will not work.
  + So, there is 2 way to do this, manual or we can use ansible for this.
* When you run ansible playbook it will work on silent mode.
  + If you want to see what they are doing behind the seen you can pass -v as a option in ansible playbook so, it will give you some highlights.
    - You can pass multiple -v so you will get more details.
* In linux we have ext4 format type like that in dvd / cdrom we have format type namely ISO9660.
* In yaml if you want to pass escape sequence you have to write this thing under double quotes.
  + Single quotes will not support parse.
* Installing yum, mounting dvd and configuring webserver from scratch.
  + Linux OS mostly comes with active firewall and they will not allow you to access webserver from outside.
  + For this you have to allow port 80 in the firewall.
    - For this thing we can also use ansible.
    - We also have to set immediate in firewall so we don’t have to restart system.
      * Firewall-cmd --list-port
        + List all the port that are allowed by firewall.

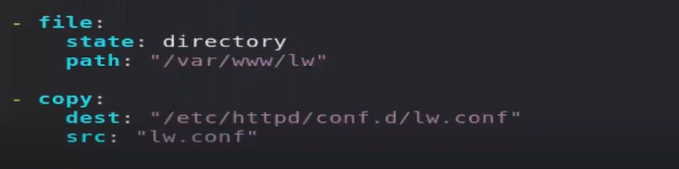


**Session 06 – Document root/variable**

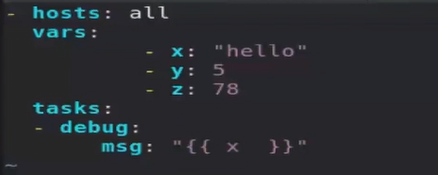
* Every service – tool have one document root, where we have to put our file.
* Eg, document root for apache webserver is /var/www/html.
* http work on port 80, we are going to change these 2 things using ansible.
  + If you want to change configuration for apache webserver, you have to change inside /etc/httpd/conf.d
  + Here any file you create with .conf extension it will work as configuration fie for apache.
    - You have to provide which port you want to listen.
    - You also have to provide document root (only if you are changing default root).
    - If you change anything in webserver configuration, you have to restart webserver.







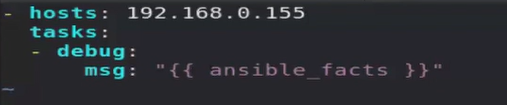
* + - Here you wan to mount so you have to create a directory.
      * Fir this we have to use file task.
  + Here sometimes we have to use same value at lots of place, and when you change here, you have to change everywhere.
  + For this we have to create a variable.
    - But instead of using lots of variable in same file, you can create a separate file for variable.
  + In ansible for printing something we have module called debug.
  + Vars keyword, using this you can tell ansible here we are creating variables.
  + For printing anything you have to use debug module.



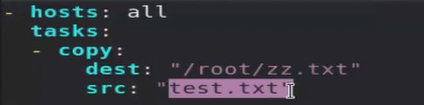
* + In ansible if you want to use variable between string (“”) then you have to use {{ }}.
    - It is known as string interpolation.
    - Ansible using jinja language, it comes from there.
  + Vars\_prompt  
    - name: dvd\_dir  
     private: no  
     prompt: “enter dvd mount point”
    - Now you can pass value of variable in your command prompt.
    - By default, when you use interact prompt it will not visible.
      * Here we have pass private == no, so it is visible.

**Session 07 – facts**

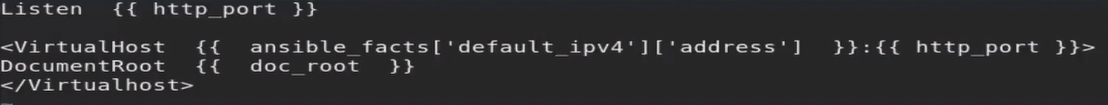
* Whenever you run playbook, facts will go and try to find all the information about your target node.
  + They will retrieve live information.
  + This information are known as facts.
    - Total no of free ram
    - kernel version
    - OS name & version
    - And lots more…
* After collecting this information, they will send this to Controller Node – playbook.
  + Sometimes we have some customize requirements.
    - Like install this hadoop master, if it requires more than 2 GB free ram.
  + Then using this fact they (ansible) can take live decision launch this cluster on that node or not.
  + If you want to perform any action in target node, you must have to include that OS IP in the inventory.
    - When you run playbook, it will first gather all the details and store inside ansible\_facts variable.



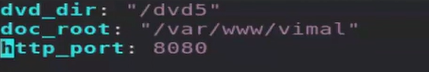
* + - If you run this file it will give you lots of information about that target node.
  + For retrieving facts we have special module called setup.
    - Ansible all -m setup
    - For customize facts we can pass arguments like.
      * Ansible all -m setup -a “filter=ansible\_system”
* You have to put your target node IP address here.
  + - Listen 8080
    - <virtualhost 192.168.43.128:8080>
    - Documentroot: /var/www/html
    - </virtualhost>
  + But here we don’t need to configure only one OS, here we are going to configure multiple (thousands of) system using ansible.
  + So, we cannot give one by one IP.
* If you want to send multiple content to the yaml, you have to use > symbol.
  + Multiline syntax.
* Copy module
  + When you use src inside copy module they will work as a static. They will pass complete file.

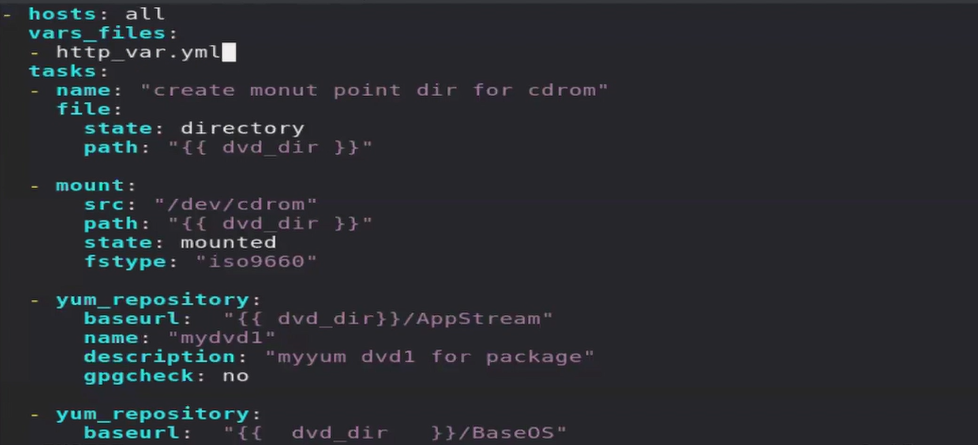


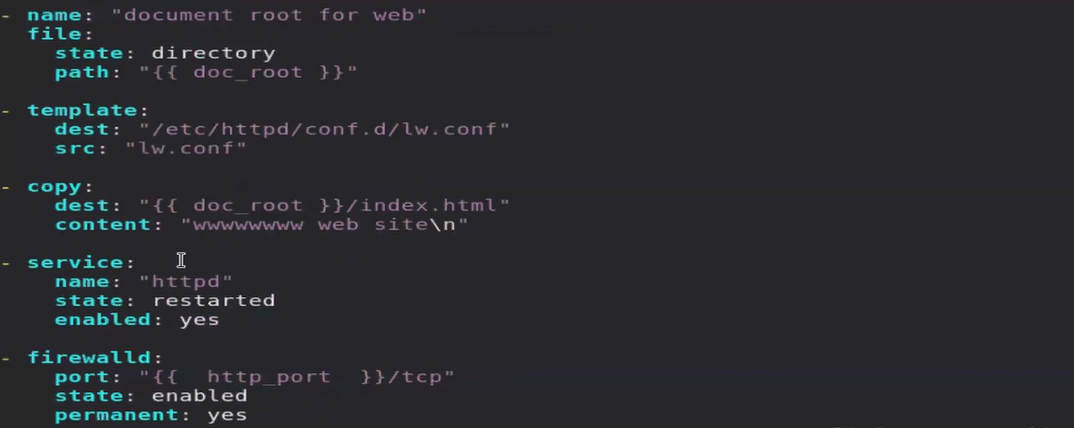
* + - Like here you are passing a file, but sometimes file contains some variable, but src argument will not parse this.
  + For this we have to use template module.
    - It is completely same as copy module.
    - Only difference is before uploading local src file to remote destination, whenever they find special symbol (variable) ( { { X} } )
      * This type of syntax of variable comes with jinja language.
    - Jinja language they replace value (variable value) and send file to remote location.
  + Here before uploading this file they replace value with the variable value.



* Here it will give you some default name to the task, instead of this you can give your name, so you can know what they are doing there.
* You can also give variable name inside different file so, when you need to change variable you just have to change ion that file.
  + It will not create mess with other file.



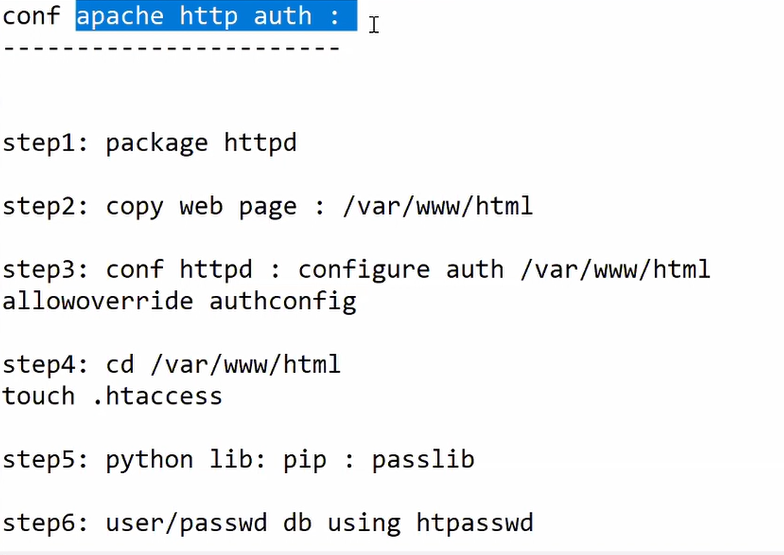




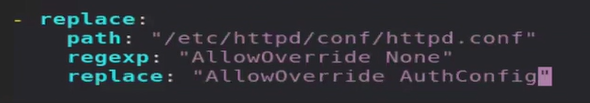
* There is one more thing SELinux only allow you to change port number to 8080, 80 & 81.
  + So, here we can simply disable selinux in target node manually at this moment.

**Session 08 – http Auth**

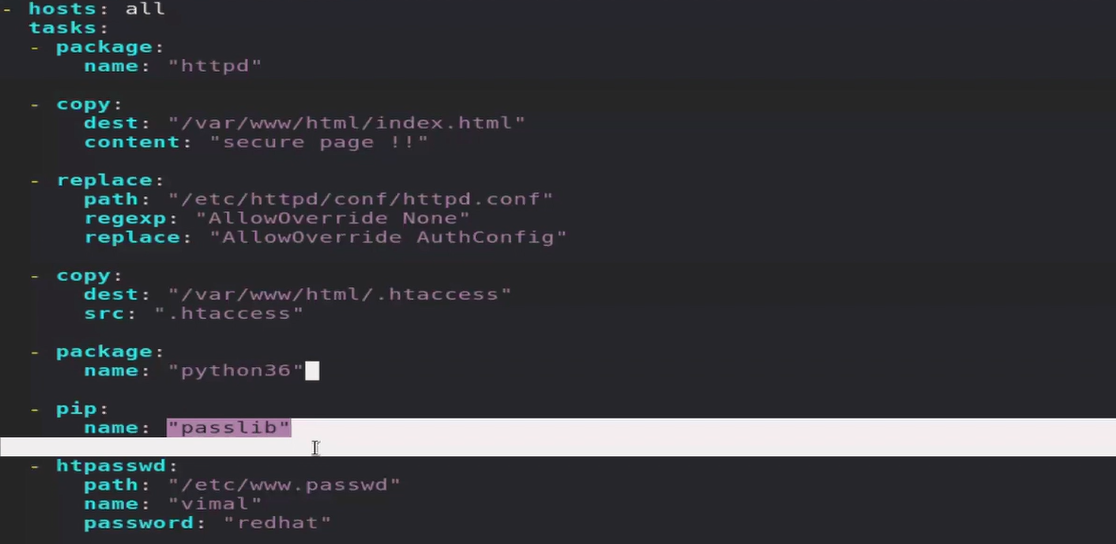
* When you come to facebook server, they might have said if someone come to this IP this facebook URL, I ask their browser to ask password.
* Anybody comes to my webserver using this http or https protocol, my webserver/ their browser will ask login password, this type of authentication is known as http Auth.
* Your browserbefore opening websitecheck your login password.
  + Here before able to see website we are asking a password, so how to send password to the server.
    - Here we are sending login and password inside http protocol.
* So, before giving /var/www/html file to client ask authentication.
* Main conf file for apache server is /etc/httpd/conf/httpd.conf.
  + Line no(134).
* Now we have to create database which contains login and password.
* For this we have to create one file.
* For this thing we have to use htpasswd command.
  + Htpasswd -c /etc/www.passwd akshit
    - C for creating new file. For second time, one more user (total 2 users).
      * Htpasswd /etc/www.passwd pal
  + But we have to tell to webserver that this is the configuration file.
  + we have special file for particular document root, so we have to create inside that document root.
  + Inside /var/www/html
    - Vim .htaccess
      * Here keywords are not case sensitive.
  + Authname is shown while asking password in browser.
  + We also have authtype = kerbaros
    - Systemctl restart httpd



* We can do this same thing with help of ansible.
* For this we have to replace that line with new line.
* Here first we have to find similar line/pattern(AllowOverride None)there so we have to use regex.
  + For apache allowoverride is not case sensitive but for regex we have to pass exact pattern.
* But ansible also have one module for this type of replacement which work as a regex.
  + Replace

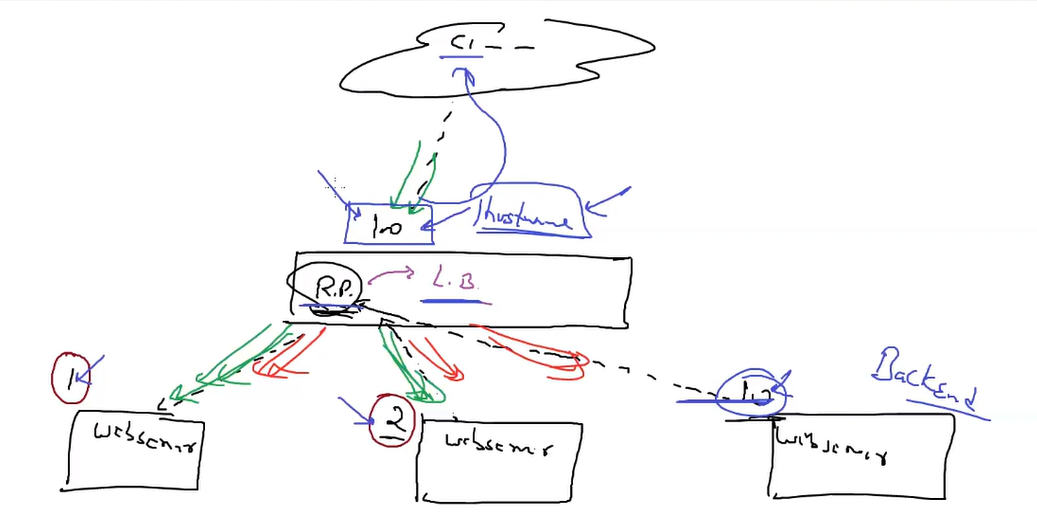


* Ansible-doc -l
  + It will give you all the modules that are available in your system.
  + This thing is available inside redhat only.
* Anisble-doc package
  + Here argument start with = is compulsory.
* We have copy .htaccess file, but we also have to create /etc/www.passwd
* Here we have to use htpasswd module, but it is depended on passlib library.
  + So, we have to download this inside target node.
    - Pip3 install passlib inside target node, so we need pip module in ansible.
  + Ansible doc pip
  + We also have to install pip if it is not available.
* Htpasswd module
* Pip module



**Session 09 – LB and Reverse proxy**

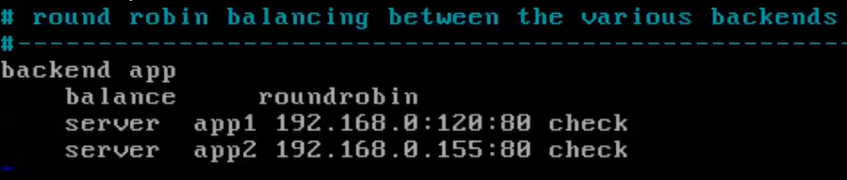
* We implement reverse proxy with load balancer.
* We have configured webserver.
* Now client come to the website, port 80 using internet.
* But every server has hardcoded limit.
  + In one point in time, only 100 clients can connect.
  + So, if 100 clients connected and now if some new clients come then these new clients get response server is busy or get timeout error.
  + But company never want to lose his clients and his reputation.
  + For solving this we can launch more webserver.
    - For launching this OS faster, we can use ansible.
    - But how client can know we have to go to this different system, different IP.
    - In company also we are terminating some servers, when there is less traffic, so we can save some cost.
    - So, client come to this IP it is not running and when it again launches IP may be changed.
  + For solving this we launch one more OS (IP 100).
    - This OS don’t have real website.
    - Any client come to this OS, they will allow as I am webserver and behind the seen on your behalf, they send request to webserver (webserver think that is the client), and webserver give reply to client (proxy server) and this proxy server reply to real client.
      * According to client IP 100 is server.
      * According to server IP 100 is client.
      * This thing is known as reverse proxy.
    - Here by this method, they are also giving you security.
      * In this client never hit to server they only
    - In webserver firewall only allow one IP of proxy server.
      * We isolated our servers.
  + Somebody behalf, somebody running this type of terms known as proxy.
  + Here all are going through IP address.
    - Here we can do one thing. We can guide proxy server to only send 100 clients to one server, then don’t send to same server, instead of this send to the different server (launch one more webserver) .
  + Reverse proxy also gives you one more thing, load balancer.
    - First client connects to first server second client to second server.
      * This thing also known as round robing.
    - So, any one server doesn’t get more load.



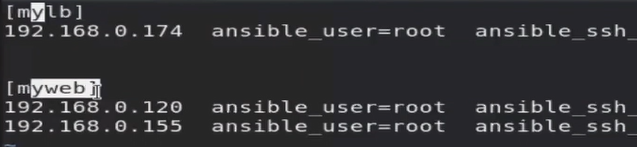
* + What you achieve

1. Reverse proxy
2. More security
3. Load balancer
4. And lots of other things.
   * Here reverse proxy is known as frontend.

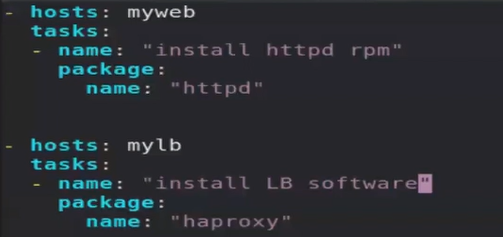
* For this type of proxy server, we have lots of tools available.
* One of the famous one is HAproxy.
* **Manually**
  + Install 2 webservers.
    - Step1: install haproxy
      * Yum install haproxy.
    - Step2: conf RP
      * /etc/haproxy/haproxy.cfg
        + Port number
        + Front end port number == bind
        + Backend server
      * Anything about backend server you can find in App block.



* + - * You have to update all your backend server here.
        + You can change name of backend server and update in both places.
    - Step3:
      * Stop firewall for that port.
      * Systemctl start haproxy.
      * Systemctl enable haproxy.
* Playbook
  + Here you want to configure 2 different things, load balancing and webserver.
  + For this you have to give group\_name.
    - Ansible all here all is also one group name.



* + - Now we can use this group name in hosts.

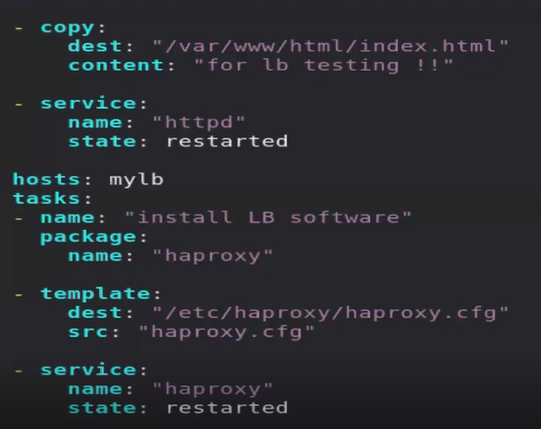


* + - Continuos …

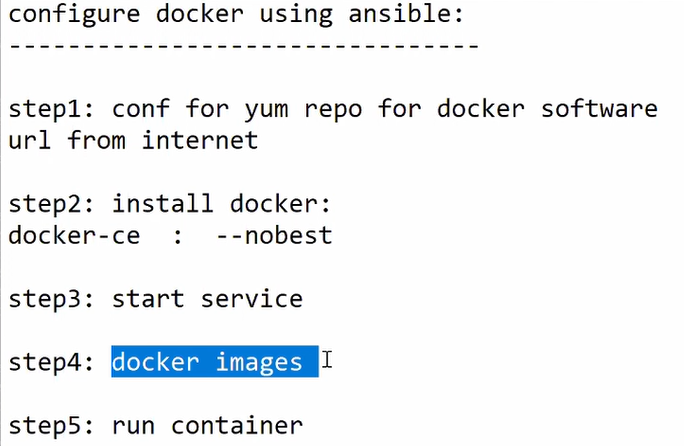
How we can automatically add new IP of webservers to the load balancer? So, it will automatically complete setup done.

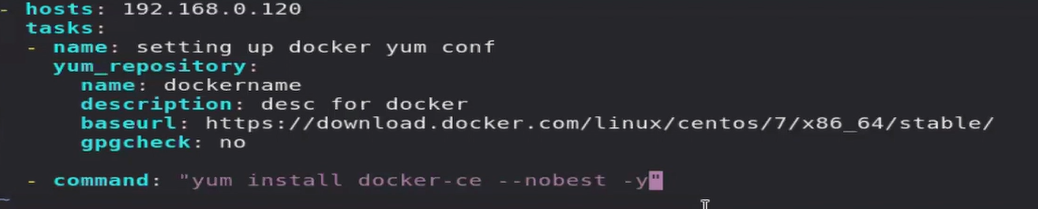
**Session 10 – LB continues docker**

* Copy config file inside controller node.
* Change above mentioned settings.



* But here we have to manually update LB entry for backend server.
* Docker
  + While installing docker you have to provide --nobest.
    - So, we have to change in our code.
  + Package module don’t come up with option to provide --nobest in ansible.
  + Here we have dedicatedmodule for yum packages.
    - But it will also not come up with --nobest as argument.
  + For this requirement we have OS specific commands.

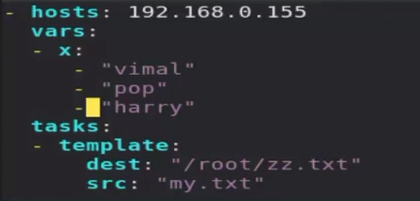




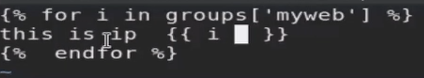
* + Command module
    - It will not provide you idempotency.

**Session 11 – Jinja language**

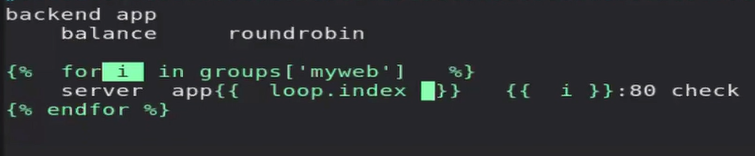
* Django and flask framework use jinja language behind the scene.
* In python lots of things using jinja as a language or framework.
* When you use template module, it will go to file or content and parse it.
  + If it finds any variable then it will replace value.
  + It will parse those things which will come inside { }.
    - Variable == {{X} }
    - If else = = {% %}
    - For = = {% %}
    - Comment == {# #}
* Eg,



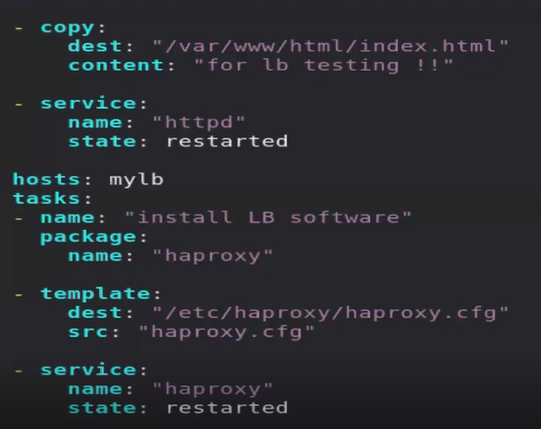
* + This is {{ x }}
* It is giving output like this, but we don’t want this type of output.
  + This is [‘vimal’, ’pop’, ’harry’]
* We want output like this,
  + This is vimal  
    this is pop  
    this is harry
  + So we can use loop here.
  + For iteration we have to use for loop.
* Group name of host is also work as a variable, stored inside group variable.



* + We have to close loop using endfor.



* + We upload this file using template module and template module change the variable value.
  + Here loop.index contains one counter.
    - It is pre created.
      * When loop run 1st time it will write 1, then 2….
  + Here we are using jinja language so, file is known as template.
  + So, it is good practice to store file with .j2 extension.
    - Haproxy.cfg.j2
    - It is jinja template.



**Session 12 –API // AWS instance launch**

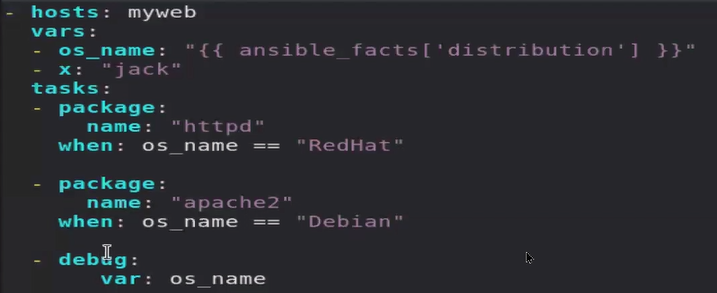
* Ansible is designed mainly for Configuration management. But it will also provide you some extra facilities.
* Ansible requirement comes after provisioning OS.
* We can do mainly 2 things in top of OS.

1. Provisioning
2. Configuration management

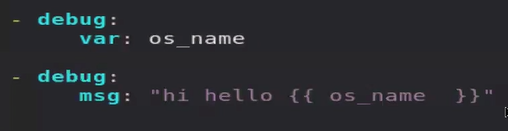
* Ansible is now days also support for provisioning.
  + But it is not mature.
    - Mature in the sense in will not able to solve your some of customize requirements.
* Python can be integrated with API.
* Ansible is built on top of python so we can also integrate API with ansible.
  + We have API available of almost all the tools, services.
  + Using ssh we can login to system and configure or run a program.
  + But using https we cannot do these things.
    - But using API we can run these things.
    - Here you are not running a program, on your behalf API will run these things for you.
* But in ansible you have to tell inventory, user id password and IP of the OS (API).
  + But AWS don’t provide you these things.
  + So here we have to tell ansible inventory is our own system (controller node).
    - Run playbook in controller node.
* For this we have module called ec2\_instance.
  + We need some libraries so; we can run this API.
  + We need boto library.
    - This library is for aws cloud.
  + Pip 3 install boto3.
* For launching instances, you have to follow these steps.
  + For authentication you have to provide accesskey, secret key.
  + For IP provide controller node IP, localhost.

**Session 13 –when//register//debug module**

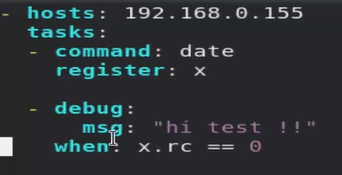
* Sometimes package name also different with OS.
* In redhat package name == httpd, but in ubuntu package name == apache 2.
  + So, for installing same package in different server we have to use if..else type of syntax.
  + For this we can use when syntax.

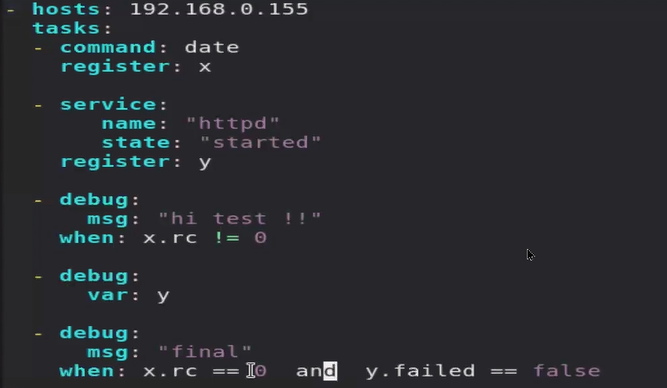


* Debug module
  + Using debug module, we can print anything.
  + We can pass var or msg as argument.
    - Msg, we can pass only string.
      * Here we have to use jinja language for printing variable value.
    - Var, we can print anything.



* We want to run next task only when previous task run, want previous task output in next task.
* For this we have to use key call register.
  + It will store output of the command / task.

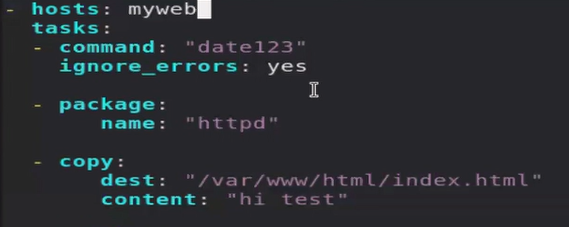


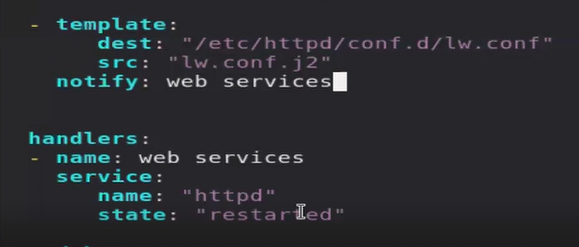


* Here in last line, we don’t use double quote for false, because it is Boolean.
  + It’s not a string.
* Here register and when is a keyword of tasks, so we have to give same indent as module.
  + They are not inside module.

**Session 14 – notify//handler**

* When one of the tasks fail, it will give you error and terminate next tasks.
  + Sometimes we don’t need this.
  + May be this task is not so much useful, but other tasks are critical.
* For this we have to use keyword called ignore\_errors: yes
  + Now if task don’t run it will give you error and says ignoring error but it will not terminateall other critical tasks.
* When you write a playbook, sometimes you have to change the config file and restart services.
  + But restart statedoesn’t come up with idempotency nature.
  + So, you haven’t changed anything, still it will restart our server which is not good.
  + We have changed config file using template module. So, we can tell template module whenever you change anythingnotify to this task.
    - This type of task is known as handlers.
  + Notify only call handler if that task is changed.





* Handler is not a keyword of tasks, it is separate keyword.
  + Here we can have multiple handlers, so we can use handler name inside notify.

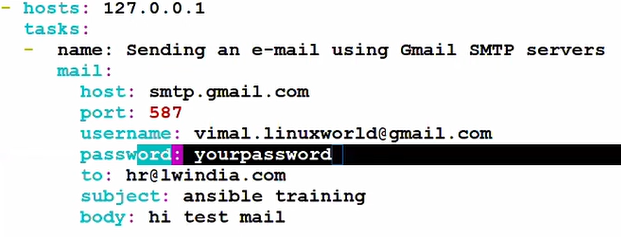
**Session 15 –exceptional handling**

* module
* Get\_url
  + Download files from internet and copy to the destination.
* For handling exception, we have to use block…rescue (try in other language).
  + Block will run code and handle exception, for exception it will give you alternative way.
    - It will try to run program and if it has any error it will run rescue operation.
      * It will rescue your program from crashing.





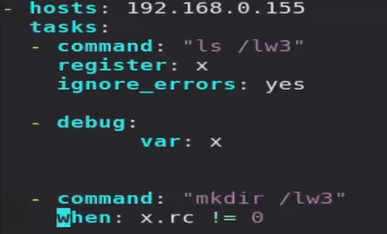
* + Try = block
  + Catch = rescue
  + Finally = always
* Mail module



* Here we don’t want to share our password directly like this.
  + So we need some security from ansible.
    - For this we can use vault concept.
* This module doesn’t have support for 2 factor authentication.

**Session 16 – vault//command//shell module**

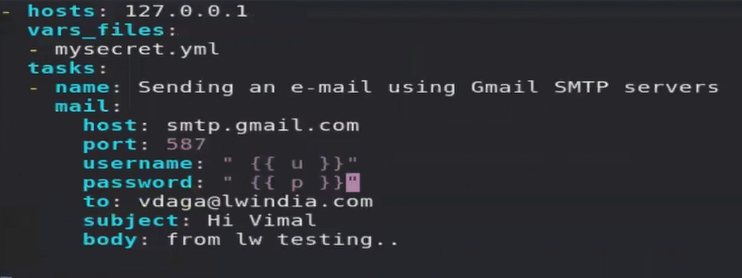
* When we run playbook in bigger environment, we are more focused into the result.
* And mostly we are focused in what changed, skipped or failed.
  + When we run command module it will always show you changed.
  + But that’s not our main point.
  + We are focused in some system restarted or packaged downloaded like that.
  + So, we can pass one keyword there in command module, so it will change the output, still It will not show you in output as changed.
    - **Changed\_when: false**



* This way we can make command module idempotency.
* Command module will not runshell respective things like pipe (|).
* For this kind of thing, we can use shell module.
* It is exactly same like command module, but it will support shell feature.
  + It works slow respect to command module, so use command module most of time.



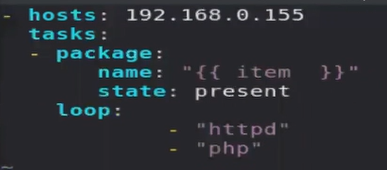
* Mail module continuos
* Here we are more concerned about password.
  + For this we can do one simple approach, we can store password in different file, and include file here.
  + Here we are directly connecting to mail server of google, instead of webApp of Gmail.
  + But by default, it is not allowed. So, you have to manually allow this in your account.
  + Search Gmail secure app -> Less secure app URL, and turn it on.

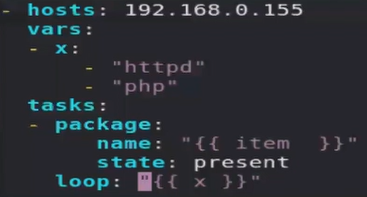


* We can also lock this file.
  + For this we have to use some key.
  + Instead of us nobody can open the file.
  + This is known as vault.
* Anisble-vault create mysecret.yml
  + It will use AES256 encryption.
  + For open this file,
    - Ansible-vault view mysecret.yml
    - Ansible-vault edit mysecret.yml

**Session 17 – loop//user\_creation**

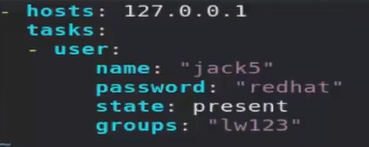
* If you want to install multiple packages.
* We can use package module multiple times, but it will consume more CPU time, multiple times CPU used.
* For this we can use loop concept.
* Here we have keyword loop.
  + In old version we have keyword called with\_items.





* In linux we have lots of things related to user, create, remove, group password.
  + We have different command for this.
  + Here we can also use ansible.
* Group of users in linux.
  + There are types of group.

1. Primary
2. Secondary
   * When you create a new user, group name with same username already created in the system.
     + It is known as primary group.
     + User belongs to his own group.
   * You can add user to group.
     + This group is known as secondary group.
   * For creating user = Useradd AK
   * For creating user with group = Useradd -G mygroup AK
   * For modifying group = Usermod -G mygroup AK
     + Cat /etc/passwd
       - Store user’s information.
     + Cat /etc/shadows
       - Store user password.
       - Here password stored using SHA512 hash.
     + Cat /etc/group
       - Store user belongs to which group.



* In redhat we cannotstore password in clear text format.
  + It will not work.
  + So, you have to convert password in SHA512 hash.

`

* User module
  + Here user module is used for creating user.
    - Using this you can also do customize changes in users and user groups and lots more…

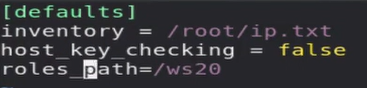
**Session 18 – Expert lecture**

**Session 19 – role//galaxy**

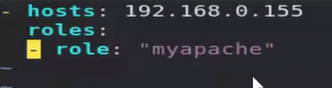
* Every program has some king of goal, eg, calculator, game.
  + It is created by multiple programmers.
  + It is known as package.
* In ansible this package is known asrole.
* Most of time we give name setup.yml to the main playbook.
* Many communities have created and uploaded role in the internet.
  + One of the famous communities is ansible galaxy.
  + Ansible-galaxy role initmyapache.
    - To create new role.
    - Here myapache is a name of a new role.
      * This will create a complete folder.
      * This folder will contains variable, handlers and all type of file.



* + Ansible-galaxy role list
    - Here you have to specify where is your role available.
  + Ansible-galaxy role list --roles-path /myrole
    - Myrole is a folder name.
    - To see how many roles, we have.
  + Role is only made for managing all yml file, variable file and all other things..
  + Here we are applying this role to the managed node.
    - Sometimes it is also known as orchestration.

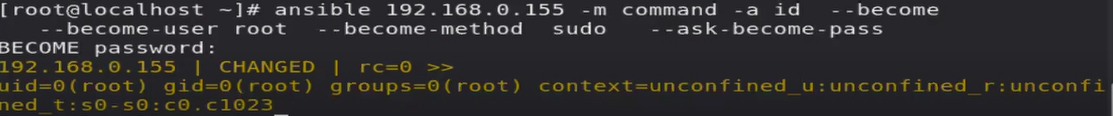


* + - Now you don’t have to provide path for role in any command.
* Ansible-playbook setup.yml --role-path /ws20
  + Here you can run a role same as you running playbook.

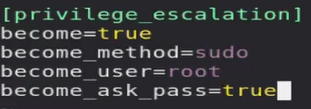


**Session 20 – Privilege Escalation**

* By default,root user has all privileges/permissions.
* If anyone can get access to this account, it will may harm your OS.
  + For this root account in disabled in the industry most of time.
  + But without root you cannot do most of things in the linux.
* But if you want to give some extra power to normal user. This thing is known as privilege escalation.
  + For this we have to use concept called sudo.
  + Here normal user will run command, with power of the root.
  + For this you have to edit in one file, /etc/sudoers.
* Running an adhoc using normal user (privilege escalation).



* We have to add this line in every steps.
  + So, if you want to always runsame thing, make this default.
  + For making this default we have to use configuration file of ansible.



* + - Add this thing in your configuration file.
* Most of times instead of password-based authentication, we use public key-based authentication.
  + Most of time password-based authentication is disabled.
  + Ssh-keygen
    - This command is used to create a new public-private keypair.
  + Upload public key to remote login.
  + For this we have to use command called ssh-copy-id.
  + This will automatically pick public key from your laptop, and authorize you.
  + For the first time it will ask you password.
    - After login and uploading file we can disable password-based authentication.

**Session 21 – Practical implementation//Expert lecture**

**Session 22 – Dynamic inventory**

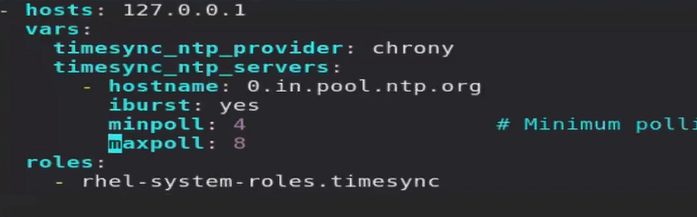
* In bigger environment we have multiple OS, OS will restart reboot and every time IP will be changed.
  + Every time go to OS & manually update IP in the inventory file is not possible.
* Instead of giving full path of inventory you can give folder name.
  + All the file belongs to that folder will be work as a inventory.
  + You can check this with ansible all --list-hosts
    - Ansible only support .py, yml, json extension or file without extension.
    - Ansible is not supporting extension .txt as a inventory file.
  + If we use name.py as one of the inventory ansible will first execute it.
  + So here we can create some code of python which can go to somewhere on the fly and fetch a IP and give to ansible.
  + For this we have to use some proper format, which is supported by ansible.
  + Static inventory using python
    - <https://github.com/vimallinuxworld13/ansible_dynamic_inventory/blob/master/hosts.py>
  + Dynamic inventory using python.
    - <https://github.com/ansible/ansible/tree/stable-2.9/contrib/inventory>
* Here we are going to use pre created dynamic inventory for Ec2.
* Here we have to provide our credentials.
  + For this we can edit the file, but it can break our code, and we cannot able to debug.
  + So you can pass as a variable inside shell.
  + Here python (or any other language) is going to use shell variable.
  + So we have to use export keyword.
    - Export AWS\_REGION=’ap-south-1’
  + Now when you run ansible all --list-hosts, it will give you IP of instances running on AWS.
* Here we cannot give group name to the dynamic inventory, here tags are used.
  + Ansible tag\_Country\_IN --list-hosts
  + Here tag\_country\_IN is a group name.
    - Eg, ansible all --list-hosts.

**Session 23 – system\_roles//chrony//group\_vars//host\_vars//lineinfile**

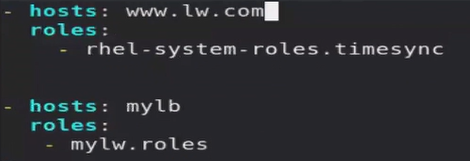
* By default there are three types of roles available in ansible.

1. Create own role
2. Use ansible galaxy
3. System roles.
   1. Redhat has also pre created some of the roles.
   2. They are already available in your system, base OS.
   3. They are known as system roles.

* Yum install rhel-system-roles
  + By this software ansible system role will come in your system.
* All the roles will give you one file (readme) so you can know how to use this particular role.
* There are some servers which provide you time, those servers are known as **time server**.
  + All the client sync time with this server, for this they are using **NTP protocol**.
  + One of the famous servers is chronyd.
  + It is a common service, so ansible has pre created one role for this.
* Create this yaml file to run a role for chrony.
  + This yaml file code is also available in README.
    - They will also provide you some example of yaml file.



* Sometimes in this setup.yaml file we have to provide lots of variables.
  + So instead of doing this we can create separate file for variables.
  + For this variable we have dedicated folder name in same folder where we run our main(setup.yml) code.
  + We have to use folder called **group\_vars**.
    - In this you have to create a folder with group name.
    - You can create separate variable for separate groups.
  + Now when you run playbook, ansible will go inside this folder and search for folder name same as group name.
    - Eg, myweb, mylb.
  + If they find any of the folder, they will include all the variable files we have in the myweb folder inside group\_vars.
    - Here you can write any random name for variable files inside group folder.
* If you want to give variable for IP address.
  + Here instead of IP you have to give dns name.
  + For this we have directory called **host\_vars**.



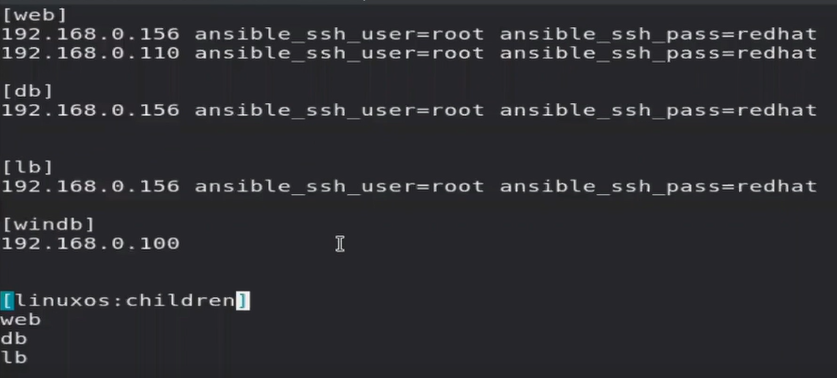
* + Now our playbook is so much simple.
* If you have customized requirement, like if you find this pattern then replace this with this.
  + Then we have one intelligent module for this.
  + Lineinfile:  
     path: /etc/httpd/conf/httpd.conf  
     regexp: “[Ll]isten 80”  
     line: “Listen 81”
  + By default they are case sensitive.
    - So we can use regex something like [Ll]isten.
  + You can use this in hadoop hdfs-site.xml file also.
    - Like after configuration and before configuration tag end, write property, value and all.
* What is the difference between replace and lineinfile module?
  + Replace module will replace a word, lineinfile module will replace a complete line.
  + Replace module will check all things word by word, while lineinfile will check line by line.

**Extra lecture 1**

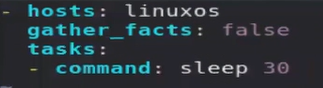
* Ansible-galaxy role search apache
* Ansible-galaxy role install geerlingguy.apache
  + Download role for apache from galaxy.
  + They will download and extract it to the /etc/download/roles.
* If you want to include task file inside main task file.
  + Include\_tasks: “file\_name”
* Revision of command, shell, exceptional handling.
  + Revision of ignore\_errors, when.

**Extra lecture 2**

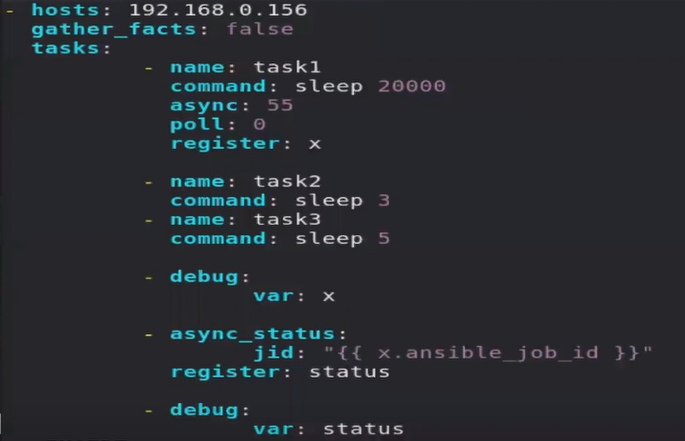
* When you do something serially it is also known as sync.
* By default, nature of ansible is they do one task at a time.
  + They will go to one single IP, then second IP of same group, then other hosts.
* Sometimes you want to run some modules on all the linux OS, but not in the windows.
  + Here you can use concept call **children**.



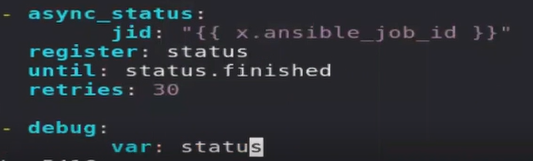
* + Ansible linuxos --list-hosts
* Netstat -nct
  + It will show you real time connection.
  + Here t stands for TCP, so you can only see TCP connections.
  + You can use this concept to check serial behavior of ansible.



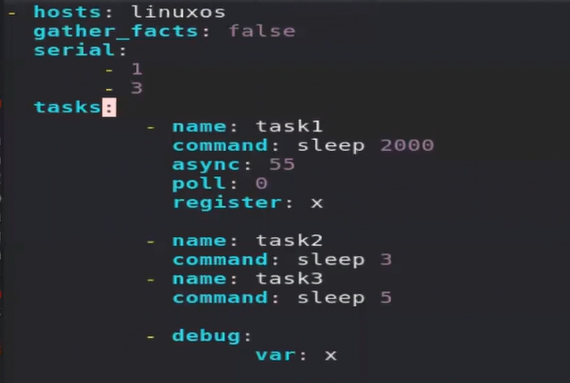
* For making our code parallel, we have to use one keyword called forks inside ansible configuration file.
  + By default, value of fork is 5.
  + So, they can run 5 things parallelly.
* Ansible-config dump
  + It is an internal keyword for ansible configuration file.
  + It will give you keywords for ansible configuration file with default value.
* Ansible-config list
  + It will give you manual of keywords.
    - You can use ansible-config list | grep fork
* Till the task complete ansible will keep on monitoring/checking this task.
  + It is also known as keep on Polling task.
  + In simple term ansible will keep on syncing with the task.
  + You can say ansible is syncing with managed node.
* Here ansible has multiple forks, so he can run same tasks parallelly on multiple nodes, but he cannot run multiple tasks parallelly.
* If you want to run multiple tasks parallel then you have to use keyword called, **async**.
  + Here if given time task is not completed then they will give you error.
  + Here we have to use one more keyword called **poll**, it will go to managed node, in some seconds and check task is completed or not..
  + Suppose, async = 40, poll = 2,
    - Poll will go to managed node every 2 second until managed node task is completed.
    - But it will go for only up to 40 seconds (20 times), after this they won’t go there.
  + Once this task completed, they will go for second task.
* But our main focus is to run multiple tasks parallelly.
* Here we can use poll, if we change **poll = 0**.
  + It is a special number.
  + Now controller node will not sync with managed node, they will no go to managed node to check whether task is completed or not.
* In this case task1 will run in the background, and ansible go for the second task.
  + Ansible send instruction to managed node do this, and come to the second task.
  + Once other tasks completed ansible playbook will complete and give you result.
  + But in this case, we don’t know our async task is completed successfully or not.
* Ansible will give job unique ID to every async task.
  + We can use this to track async task.
  + For this we have module called async\_status.



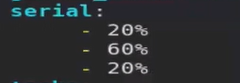
* Here in status in will give you error, because task is not yet finished.
* So we can give argument until: status.finished
  + By default, it will retry 3 times.
  + But our task is not yet complete so it will still give you error.
  + We can use argument retries: 30
  + Now they will try 30 times.



* Sometimes we create a batch of computers.
* For this we have to use **serial keyword**.
* First, they will work with some OS, then they will work with other remaining OS.



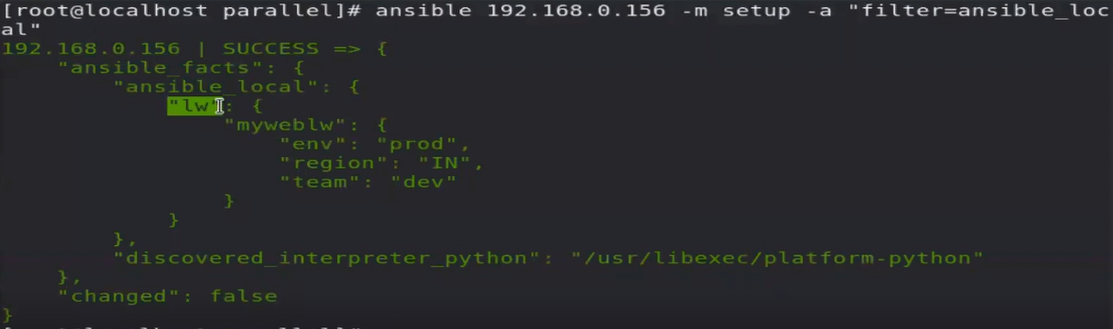
* + You can also give them in percentage wise.



* + We can use this concept in rollout strategy also.
* **Custom facts**
  + We can create customize facts for particular node.
    - Eg, if ENV=prod, then do this is this system.
  + For this thing we have one specific location.
    - Mkdir /etc/ansible/facts.d
  + Here you can create file with any name, but extension should be fact.
    - This file would be created in the managed node.
      * We can use playbook for this thing also.
    - **Vim lw.fact**



* + In controller node,
    - Custom facts are stored inside ansibl local keyword.
    - Ansible all -m setup -a “filter=ansible\_local”



* + File name is a main key here.
* **Session 01 – starting ansible**
  + OS – OS different command
  + Manual -> human error
  + Automation script
  + Configure management
  + Imperative – declarative language
  + Hydrogenous environment
* **Session 02 – CN/TN state**
  + Controller node – target node / managed node
  + Agentless
  + Installation
  + Inventory
  + Configuration
  + Current state
  + Idempotency
  + Installing epel-release
    - sshpass
* **Session 03 – ansible intelligence module**
  + Ping module in ansible
  + Module
    - Ansible intelligence reason
  + Arguments
  + Ssh
    - Host\_key\_checking=false
  + Ansible colors
  + Configuring webserver
* **Session 04 – playbook** 
  + Revision of first three lecture
  + Ansible
    - Manual
      * Known as ad-hoc
      * Ansible all command to running this.
    - Automatic
      * Known as playbook.
      * Ansible-playbook <name.yml>
  + Yaml format
  + Ansible-playbook --check-syntax <name.yml>
* **Session 05 – webserver firewall**
  + Cdrom format iso9660
  + Configure webserver from scratch
  + Firewall allow port 80
    - Firewall-cmd --list-port
* **Session 06 – document root variable**
  + Document root
    - /etc/httpd/conf.d
    - Restart webserver
  + Variable
    - Jinja language
    - Vars\_prompt
    - {{ }}
* **Session 07 – facts** 
  + Facts
    - -m setup
  + Template module
    - Same as copy module
    - But they parse data before sending
  + Separate file for variable
    - Jinja language
  + Selinux allow only port 8080 80 & 81 for webserver.
* **Session 08 – http auth**
  + Browser ask login before showing page
  + /etc/httpd/conf/httpd.conf
    - Line 134 – AuthConfig
  + Htpasswd-c /etc/www.passwd pal
    - C for first time new file creation.
  + Vim .htaccess
    - Create database for auth inside same document root.
  + Replace module
  + Pip module
  + Htpasswd module
* **Session 09 – LB and reverse proxy**
  + Hardcoded limit in server
    - Only 100 clients can connect.
  + HAProxy
    - You achieve
      * Reverse proxy
      * More security load balancer
  + /etc/haproxy/haproxy.cfg
* **Session 10 –LB continues Docker**
  + Playbook for LB
  + Docker
    - Pass --nobest
  + Command module
    - OS respective command.
* **Session 11 – jinja** 
  + When you use template module, it will go to file or content and parse it.
  + syntax of jinja language
    - Variable = = { {X} }
    - If else = = {% %}
    - For = = {% %}
    - Comment == {# #}
  + Loop.index
    - Pre created counter inside for loop.
  + Group variable.
  + Jinja used, file == template, extension= .j2.
* **Session 12 – API//aws instance launch**
  + Provisioning
  + Configure management
    - Ansible is not proper mature for provisioning.
  + Aws API
  + Ec2\_instance module
    - Boto library.
* **Session 13 – when//register//debug module**
  + When
    - If…else type condition
  + Register
    - Both are keywords of tasks (not module).
  + Debug module
    - Var
    - Msg
* **Session 14 –notify//handler**
  + Notify
  + Handler
* **Session 15 –exception handling**
  + Exception handling
    - Try = block
    - Catch = rescue
    - Finally = always
  + Mail module
    - Issue in mail module
* **Session 16 – vault//command module**
  + keyword
    - Changed\_when: false
  + Command module
    - Not idempotency
  + Shell module
  + Mail module
    - Vault
      * Anisble-vault create mysecret.yml
      * Ansible-vault view mysecret.yml
      * Ansible-vault edit mysecret.yml
    - Lock file
* **Session 17 – loop//user\_creation**
  + Loop
  + Usercreate
  + User group
    - Primary
    - Secondary
  + SHA512
  + Using loop for creating user.
* **Session 18 – Expert Lecture**
* **Session 19 – Role//galaxy**
  + In ansible role == package
    - Ansible galaxy, download role.
  + Ansible-galaxyrole init<role\_name>
  + Ansible-galaxy role list
    - Give path of role inside ansible config file.
  + Orchestration
* **Session 20 – Privilege Escalation**
  + Root user disabled
  + User-password disabled
    - Key based authentication
  + Privilege\_escalation inside ansible config file
  + Ssh-keygen
    - Public key – private key
  + Ssh-copy-id
* **Session 21 – expert lecture practical implementation**
* **Session 22 – dynamic inventory**
  + IP change in bigger industry
  + Multiple inventories, same folder
    - Instead of full path folder path
    - Ansible configuration, Inventory location
    - .txt not supported.
    - . py, json and many extensions are supported.
  + Dynamic inventory for ec2
    - Providing credentials
    - Export keyword
      * Shell variable accessed by python program
      * Export AWS\_REGION=’ap-south-1’
    - Python3 hosts.py –lists
  + Ansible tag\_Country\_IN --list-hosts
* **Session 23 – system\_role//chrony//group\_var//host\_var//lineinfile\_module**
  + System roles
    - Comes with software available in dvd.
    - Yum install rhel-system-roles
  + Using pre created system roles
  + Time server – centralized
    - NTP protocol
    - ReadME file
    - Demo playbook
  + Group\_vars
    - Same folder where you are running main(setup.yml) file.
    - Create a folder inside with same name as group name.
  + Host\_vars
    - Work with DNS instead of group name.
  + Lineinfile module
    - Same as replace module
    - Check and replace line by line
* Extra lecture – 1
  + 29/12/20
  + Ansible-galaxy role search apache
  + Ansible-galaxy role install geerlingguy.apache
  + Include\_tasks
* Extra lecture – 2
  + 29/12/20
  + Sync
  + Children
  + Netstat -nct
    - Provide you real time connection.
  + Forks
  + Ansible-config dump
  + Ansible-config list
  + Async
  + Poll
  + Async\_status module
  + until: status.finished
  + retries: 30
    - default: 3
  + custom\_facts
    - ansible\_local
  + serial keyword

**Modules**

1. Package
   1. To download – install packages
      1. Name: “httpd”
      2. State: present
2. Service
   1. To start – stop service.
      1. Name: “httpd”
      2. State: “started”
3. Copy
   1. To copy files/folders
      1. Dest: “/root/Desktop”
      2. Src: “/root/xyz.txt ”
      3. Content: “Hii hello”
4. Template
   1. Same as copy module, here before copying file, it will do string interpolation.
      1. Dest: “/root/Desktop”
      2. Src: “/root/xyz.txt ”
      3. Content: “my name is {{ X }}”
5. Firewalld
   1. To work with firewall
      1. Port: “80/tcp”
      2. State: enabled
      3. Permanent: yes
      4. Immediate: yes
6. File
   1. To create a file/ folder
      1. State: directory
      2. Path: /var/www/html
7. Mount
   1. To mount something.
      1. Src: “/dev/cdrom”
      2. Path: “/dvd1”
      3. State: mounted
      4. Fstype: “iso9660”
         1. For cdrom
8. Yum\_repository
   1. To configure yum
      1. Baseurl: “/dvd1/AppStream”
      2. Name: “mydvd1”
      3. Description: “My yum dvd1”
      4. Gpgcheck: no
9. Debug
   1. To print
      1. Msg: “hi hello”
         1. You can print string only.
            1. For pass variable we have to write inside curly braces.
      2. Var: ansible\_facts[ansible\_distribution]
         1. You can print anything.
10. Replace
    1. Path: “/etc/httpd/conf/httpd.conf”
    2. Regexp: “AllowOverride None”
    3. Replace: “AllowOverrideAuthConfig”
11. Lineinfile
    1. it is same as replace module.
    2. But it will check on line wise and replace a complete line.
       1. path: /etc/httpd/conf/httpd.conf
       2. regexp: “[Ll]isten 80”
       3. replace: “Listen 81”
12. Command: “mkdir AK”
    1. Write OS respective commands.
       1. Date
       2. Cal
13. Shell: “ls | grep A”
    1. Same as command module.
    2. it is slow, but it will support shell related things like pipe ( | ).
14. Pip
    1. To install python libraries
       1. Name: “passlib”
15. Htpasswd
    1. Use for http password. Here you have to provide username password for httpAuth.
       1. Path: “/etc/www.passwd”
       2. Name: ”vimal”
       3. Password: “redhat”
16. Get\_url
    1. Use for downloading something from internet.
       1. url: <https://sameple.com>
       2. dest: “/var/www/html”
17. mail
    1. to send mail to someone
       1. host: smtp.gmail.com
       2. port: 587
       3. username: akshit@gmail.com
       4. password: mypassword
       5. to: pal@gmail.com
       6. subject: ansible training
       7. body: done successfully
18. user
    1. to create// work related to user.
       1. Name: “akshit”
       2. Password: “redhat”
       3. State: present
       4. Groups: “lw123”
19. Async\_status
    1. To track async tasks.
       1. Jid: “ {{ x.ansible\_job\_id }}”
20. Ec2\_instance
    1. To launch ec2 instance
       1. Region: ap-south-1
       2. Image\_id: ami-08f36247gehdsf
       3. Instance\_type: t2.micro
       4. Vpc\_subnet\_id: subnet-5yu678
       5. Security\_group: sg-56789fvbnmy
       6. Key\_name: mykey
       7. Name: myos
       8. State: present
       9. Aws\_access\_key: <myAccessKey>
       10. Aws\_secret\_key: <mySecretKey>

**Keywords**

1. Host\_key\_checking: false
   1. Inside configuration file, default.
2. Vars:
   1. To create variables inside file.
      1. x:
      2. y:
3. Vars\_prompt
   1. To ask variable value from prompt.
      1. name: dvd\_dir
      2. private: no
      3. prompt: “enter dvd mount point”
4. Vars\_files
   1. To include variable files from another different folder/location.
      1. Name.yml
5. Register: X
   1. It will save output in a variable.
6. When
   1. Module will be only run if condition satisfied.
      1. Os\_name == “RedHat”
7. Ignore\_errors: yes
   1. It will ignore error.
   2. Sometimes something is not much important, we can ignore if it will not work.
8. Notify
   * 1. Some command will always run, it will not support idempotency nature.
     2. We can use handlers here.
     3. Eg, service restarted.
     4. We can put this inside handler.
   1. Handlers
      1. Handlers is not a part of tasks.
9. Block
   * 1. Sometimes we may have exception in our code.
     2. We can use this three to improve our code performance.
   1. Rescue
      1. It is same like, try…catch…finally.
   2. always
10. changed when: false
    1. in bigger environment we always use final result of playbook.
    2. Command module always give something changed, but its not main focus, so we can use this, then it will not show something has been changed in command module.
11. loop
    1. it will used as a loop inside ansible yml file.
12. Include\_tasks: “<file\_name>”
    1. To include other task file inside main file.
13. Until: status.finished
    1. Until this finished it will retry.
    2. By default they retry 3 times
14. Retires: 30
    1. Now they will retry 30 times.
15. Serial
    1. Run tasks batch wise.
    2. First run in some managed nodes, after that run in other managed nodes.
    3. Eg, first run in 2 IP, then run in other 3 IP’s.

jinja language

* Variable = = { {X} }
* If else = = {% %}
* For = = {% %}
* Comment == {# #}