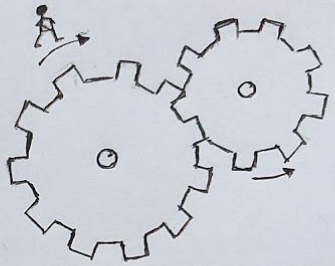

ANSIBLE

Day 1:

Install
Run a program
} Configuration.

Installing, Running an application (program) Mannually at twice, thrice is OK. But if the same steps or different programs needs to be runned and install again and again then there we need to automate that part.

AUTOMATION



If you want to ~~out~~ configure some thing. write a script and use that script every time.

If you write a code in python, perl, r they work on the concept of Imperative.

If you don't know os respective command. Its impossible to do configuration.

Imperative Language → 1.) What to do
2.) How to do.

Declarative Language → 1.) What to do
(Inbuilt Intelligence) → You are Declaring something!
eg. Ansible, puppet, chef

Write once Run anywhere (WORA)

Day 2:

ANSIBLE

- Ansible is tool / software for configuration management of operating system.

`rpm -q ansible` → check Ansible is installed or not.

`yum install ansible` → Installing Ansible

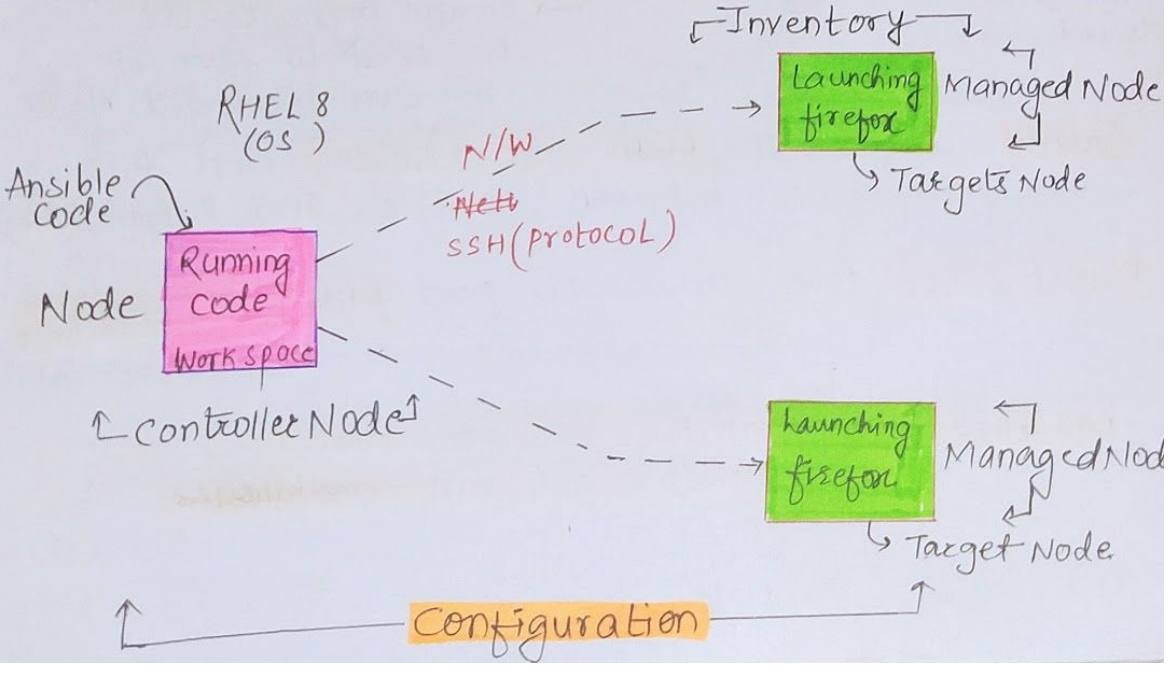
Ansible doesn't comes under redhat 8 version. (DVD file).

Python is the base language for Ansible.
therefore We have to use pip command to install yum.

`pip install ansible` → Installing Ansible via pip.

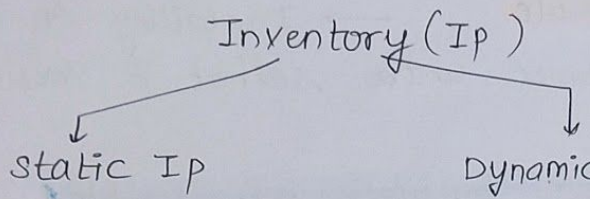
The system on which you run the code (Redhat 8) is known as **controller node**.

`ansible --version` → check the version



- Ansible doesn't go to other system and do the process.
- Ansible will only call or we can say it will recognise which command to run or to call.
- Managed Node Ip is also known as Inventory Ip.

`ansible all --lists-host` → will check all hosts.



Inventory is the database of Ip of the Node (managed node)

Ansible configuration file :-

`gedit /etc/ansible/ansible.cfg`

↳ Ansible will get all info from above file

inside ansible.cfg.

```
[defaults]
inventory = /etc/myhosts.txt
```

section — [defaults]
Keyword — inventory

↳ inside myhosts.txt We need to give Ip of our managed Node

Ansible works on push mechanism and this is the difference between Ansible and puppet, chef.

Puppet, chef tools works on ~~root~~ pull mechanism.

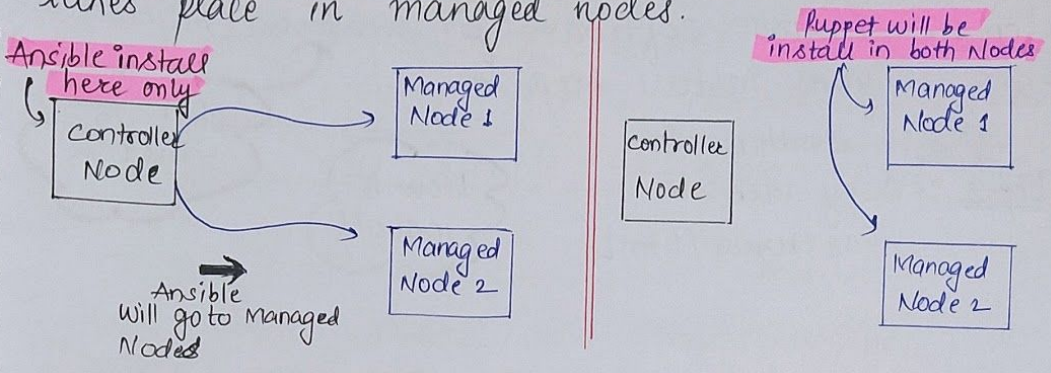
---

inside myhosts.txt file

```
192.168.0.102 ansible_ssh_user = root
              ansible_ssh_pass = redhat@123
```

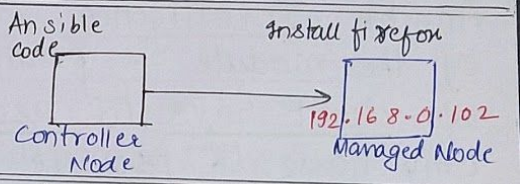
Also install sshpass ~~(Yum install sshpass)~~

- Ansible will be downloaded only in controller node. ~~Ath.~~ and ansible will be pushing the code and go to inventory (managed node) and do the automation. Therefore Ansible is Agent less.
- In case of puppet and cheff, you need to install puppet, chef to all the managed nodes and we have to send codes to managed nodes from controller node and the automation will takes place in managed nodes.



Case 1:- In Ansible Case 2 :- In Puppet

```
ansible all --list-hosts
  hosts (1):
    192.168.0.102
```



Now we have only one managed Node ~~controller~~ connected with controller node.

```
ansible all -m package -a "name=firefox state=present"
```
(module, Attribute, name of software, state)

Ansible is so smart that they only run which is our desire, if our desire exists already then it will not run. Because Ansible Works on Idempotent.

Ansible will install firefox on 192.168.0.2
But you need to provide user name and password

Ansible is agentless. you only need to install install Ansible into controller node ~~(master)~~ and not in the Managed Node.

This seperates Ansible from competitors like puppet and chef.

Ansible works on ssh protocol. If the ~~the~~ your target is linux.

But if your target is window system then the protocol is WINRM

Ansible is seamless. If don't need to change code for different systems for the same requirement.

```
ansible all -m service "name=httpd state=started"
```

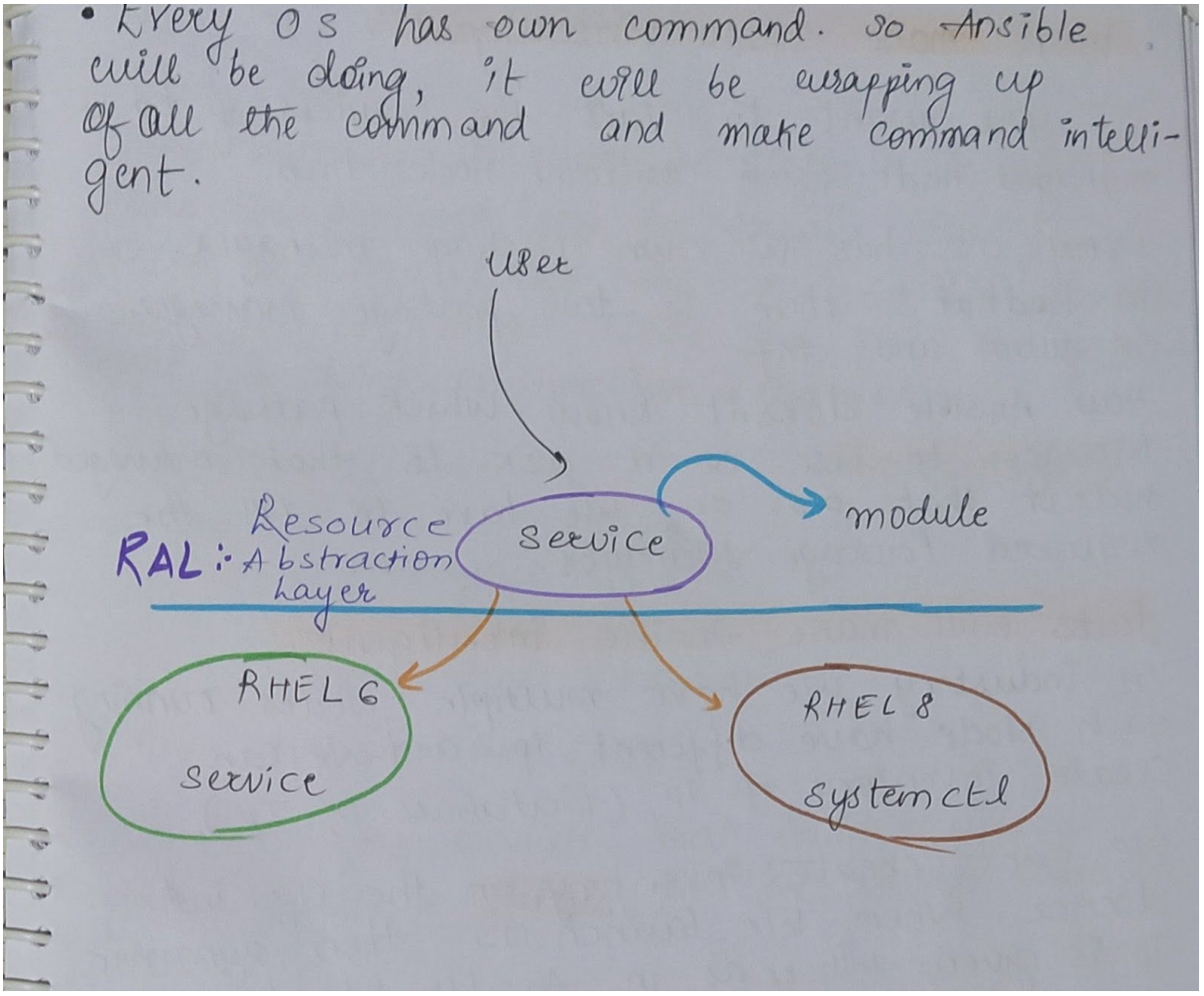
- all → all hosts
- -m → module
- service → want to install, run stop service
- name → software Name
- state → state mode

you can set your connection as per your requirement.

Bydefault is ⟶ smart

for ssh ~~&~~ ⟶ ssh or paramike

Day 3:



Date: 24/11/2020
Day: 4

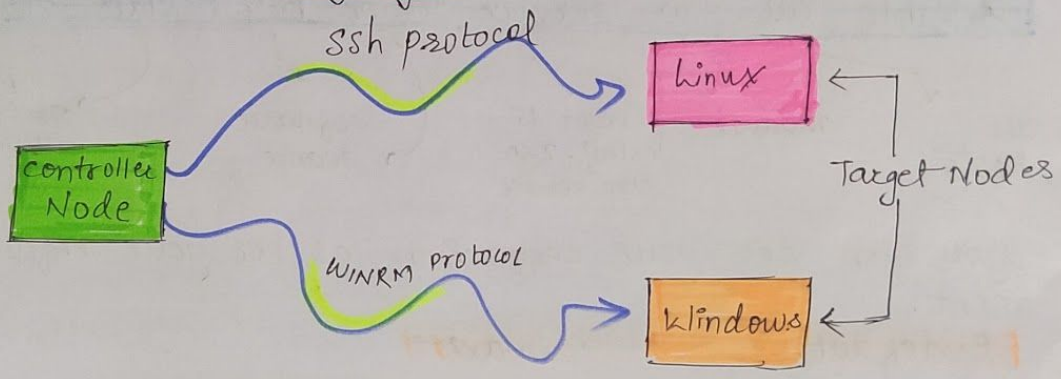
- Configure Management (CM)

Mannual (ad-hoc)
Manually writting the commands
CLI way

Imperative language

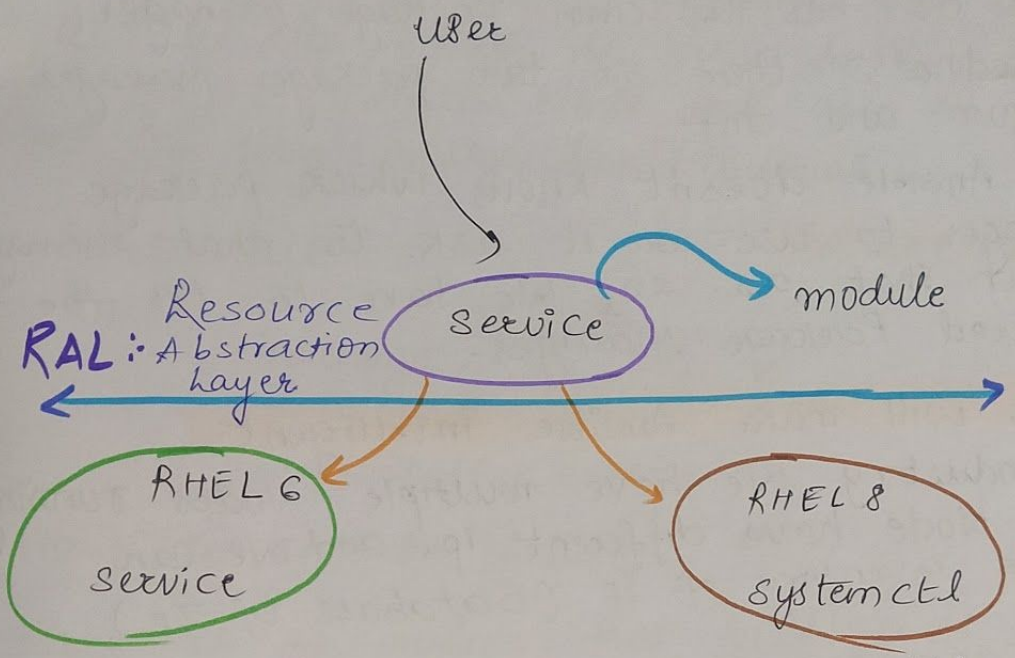
Declarative language
Gives you Intelligence

- Ansible is just a tool that provides yous Declarative language.



- Ansible is the intelligent tool used for Configure Management.
- Intelligency of Ansible comes from modules.
- To avoid ad-hoc way we can use a better way i.e programming file → Playbook (in Ansible world)

- Every OS has own command. so Ansible what will be doing, it will be wrapping up of all the command and make command intelligent.



- In the above abstraction layer Ansible hides all the modules of the target nodes.

Ansible → Playbook

Puppet → Manifest file

Chef → Recepies

IMPLEMENTATION OF PLAYBOOKS

- Key value formats → Json, XML, YAML ← Ansible supports YAML
- Ansible's playbook are to be written in YAML.

  Key value pair is also known as serde.

  serde → serializer and deserializer.

If you want to exchange any information then the data must be in same format from sending side to recieving side.

lists in yaml :-

Key → email:

- "v@gmail.com"
- "j@lw.com"
- "p@lw.com"

} lists

→ same indent.

To check the yaml code there are some tools. One of the famous tool is lint tool.

search Yaml lint.com.

Ymal language starts with --- (three hypens) and ends with ---

CONFIGURATION OF APACHE WEBSERVER

```
- hosts : "Ip1" , "Ip 2"
  tasks :
    - package :
        name : "httpd"
        state : present
    - copy :
        src : "/web.html"
        dest : "/var/www/html"
    - service :
        name : "httpd"
        state : started
```

```
ansible-playbook web.yml
```
→ Will run the playbook

```
ansible-playbook --syntax-check web.yml
```
→ will check the syntax

```
ansible all -m file -a "path=/dvd state=directory"
```

Whenever you adhoc command the give lot of info after running the code. Where as playbook do it silently.

```
ansible-playbook -v web.yml
```
→ Verbosity.

you can go 4 or 5 level of verbosity

```
ansible-playbook -vvvv web.yml
```

Date: 25/11/2020
Day:5

To

yum should be configured in the target Nodes for configuration.

yum configuration.

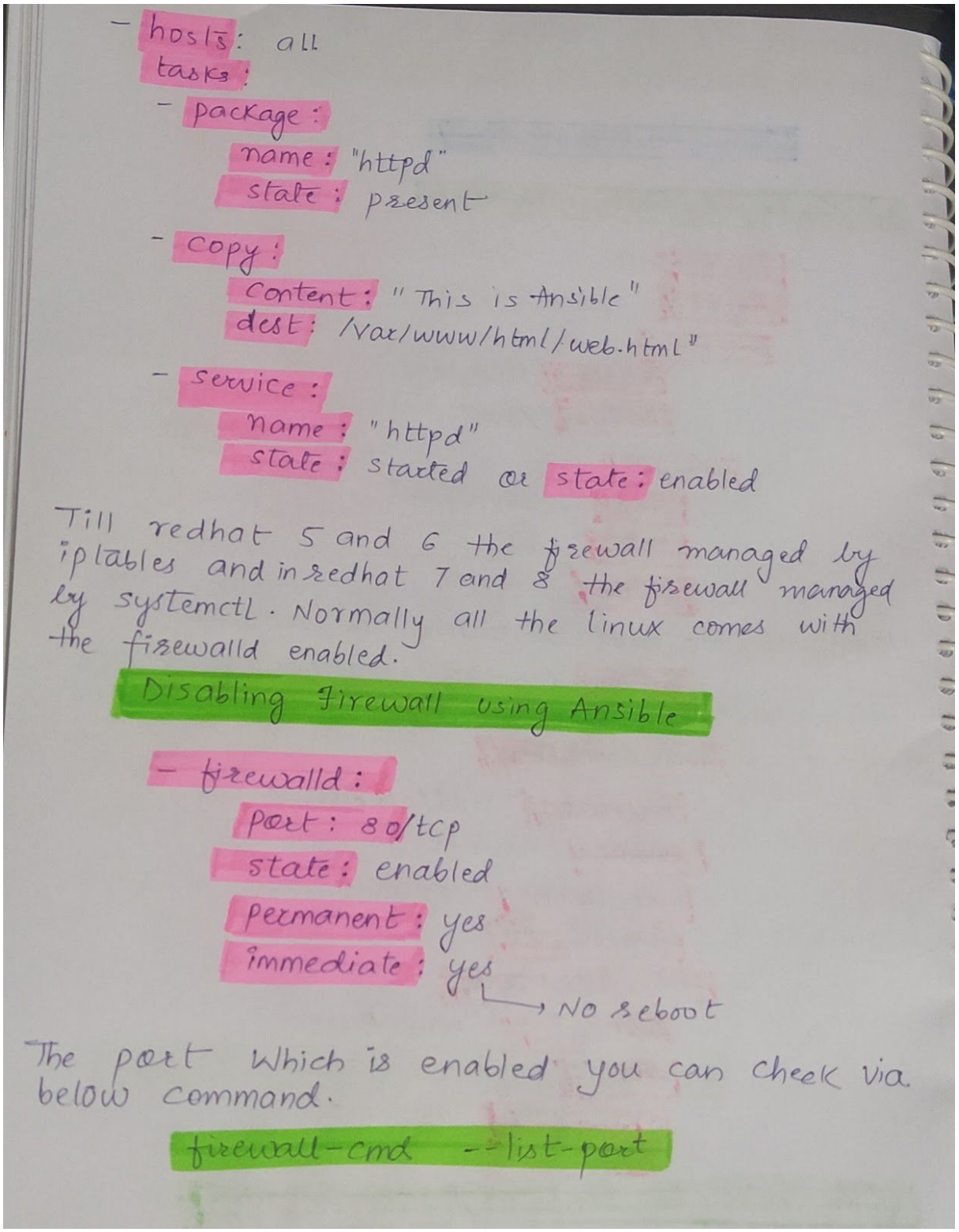
1.) Create a folder anywhere.

```
- hosts: all
  tasks:
    - file:
        state: directory
        path: "/dvd"
    - mount:
        src: "/dev/cdrom"        or "/dev/sr0"
        Path: "/dvd"
        state: mounted
```

Name of the dvd.iso format is iso9660.

```
        fstype: "iso9660"

    - yum_repository:
        baseurl: "/dvd/AppStream"
        name: "mydvd1"
        description: "my yum1"
        gpgcheck: no
    - yum_repository:
        baseurl: "/dvd/BaseOS"
        name: "mydvd2"
        description: "my yum2"
        gpgcheck: no
```

yum will be configured successfully.



Date: 26/11/2020
Day: 6

```
netstat -tnlp | grep http
```
→ see the http port.

Changing the documentation root of apache webserver

```
cd /etc/httpd/conf.d/
```
→ configuration file of Apache.

Creat a file.

```
Vi lw.conf
```

```
Listen ip:8080
<Virtualhost>
  documentroot /var/www/lw/
</Virtualhost>
```

CUSTOMIZING DOCUMENTATION ROOT OF APACHE USING ANSIBLE

```
- file:
    state: directory
    Path : "/var/www/lw"
- copy:
    src : "/root/lw.conf"
    dest: "/var/www/lw"

- service:
    Name: "httpd"
    state: restarted

- firewalld:
    port : 8080/tcp
    state: enabled
    permanent: yes
    immediate: yes
```

VARIABLES IN ANSIBLE

```
- hosts: "Ip"
  vars:
    -x : "This is Variable"
  tasks:
  - name: "Print the x"
    debug: "This "{{ x }}"
      msg: "{{ x }}"

  - name: "Printing os Name"
    debug:
      msg: "This os is {{ ansible_distribution }}"
```

Note: ansible_distribution is the pre-defined Keyword.

debug:- is the keyword to print the value.

msg:- is the parameter of debug that will print the value of the debug.

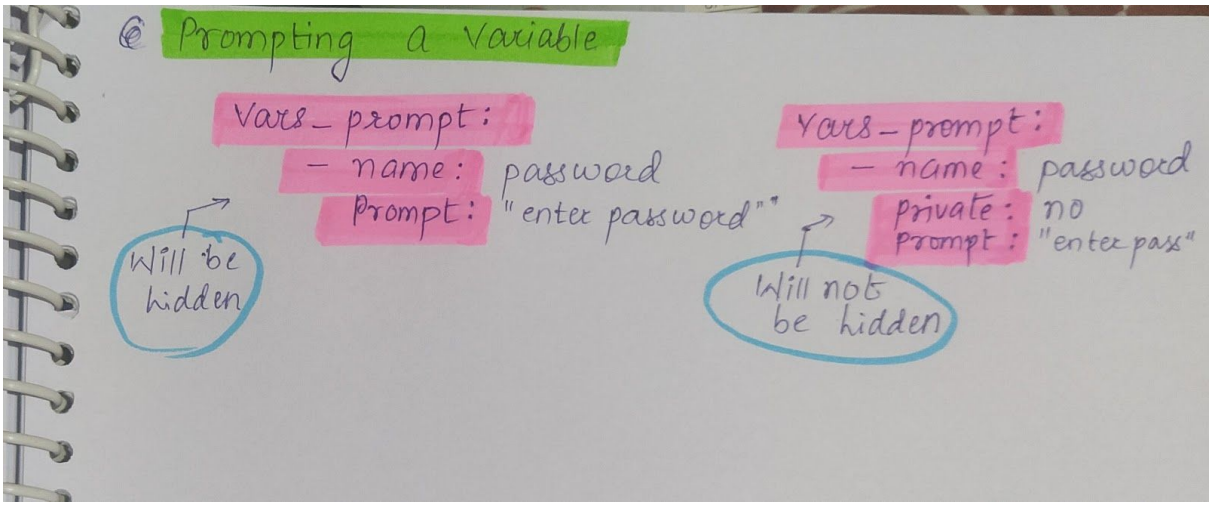
Vars:- is the module from which you can create variables. the -x is the variable and is in the list of vars.

Note: the vars modules must be above of task module.

We use **Jinga template** to use the Variable.

Note : `"{{ spacebar x spacebar }}"`

- Press spacebar
- Variable
- press spacebar



CHANGING DOCUMENTATION ROOT OF APACHE WEBSERVER

```
cd /etc/httpd/conf.d/

vi lw.conf

Listen 8080
<virtualhost ip:8080>
documentroot /var/www/lw/
</virtualhost>
```

```
   CUSTOMIZING DOCUMENTATION ROOT OF APACHE WEBSERVER USING
ANSIBLE
- hosts: all
  vars_prompt:
  - name: "dvd"
    prompt: "Enter Location where you want to create directoty, eg. /dvd"
    private: no
  - name: "document_root"
    prompt: "Enter Location where you want to create directoty, eg. /var/www/lw"
    private: no
  - name: "httpd_configuration"
    prompt: "Enter Location where you want to copy httpd_configuration file"
    private: no
  - name: "port_no"
    prompt: "Enter port no"
    private: no
  vars:
    - dvd_location: "/dev/cdrom"

- tasks:
  - name: "Creating Directory for Mounting"
    file:
      state: directory
      path: "{{ dvd }}"
  - name: "Mounting The dvd"
    mount:
      src: "{{ dvd_location }}"
      path: "{{ dvd }}"
      state: mounted
      fstype: "iso9660"
  - name: "Configuring AppStream"
    yum_repository:
      baseurl: "file:///{{ dvd }}/AppStream"
      name: "dvd1"
      description: "appstream"
      gpgcheck: no
  - name: "Configuring BaseOS"
    yum_repository:
      baseurl: "file:///{{ dvd }}/BaseOS"
      name: "dvd2"
      description: "baseos"
      gpgcheck: no
  - name: "Customizing the apache webserver"
    file:
      state: directory
      path: "{{ document_root }}"
  - name: "Copying the files"
    copy:
```

```
    src: "{{ httpd_configuration }}"
    dest: "{{ document_root }}"
  - name: "Starting the httpd Service"
    service:
      name: "httpd"
      state: restarted
  - name: "setting the firewall"
    firewalld:
      port: "{{ port_no }}/tcp”
      state: enabled
      permanent: yes
      immediate: yes
```

Date: 25/11/2020
Day:7

Prompting a Variable

```
vars_prompt:
  - name: password
    prompt: "enter password""
```

Will be hidden

```
vars_prompt:
  - name: password
    private: no
    prompt: "enter pass"
```

Will not be hidden

ANSIBLE FACTS

Facts:- Gathering information about the target Node is known as facts.

When you run a ansible playbook, the ansible will first gather facts (information) about the target Node. The information will help ansible to configure the target Node more effeciently.

```
- hosts: Ip
  tasks:
    - debug:
        msg: "{{ ansible_facts }}"

    - debug:
        msg: "{{ ansible_facts['system'] }}"
```

> setup is a seperate module that will gather all the facts of a particular hosts.

```
ansible Ip -m setup
```
→ will show all the facts.

```
ansible Ip -m setup -a "filter=ansible_architecture"
```
→ will filter the facts.

```
ansible Ip -m setup | less
```

Hu

```
- hosts: all
  tasks:
   - debug:
      msg: "{{ ansible_facts* }}"
   - copy:
      dest: "/root/zz.txt"
      content: >
          This is Ansible \n
          using content parameter \n
          from copy module.
```

---

copy module the content in a static way.

Template module is same as copy module but you can enter the value of variable in the content.

```
- hosts: all
  ~~tasks:~~
  vars:
   - x: "This is x"
  tasks:
   - template:
      dest: "/root/zz.txt"
      content: >
          The value of x is: {{ x }} \n
```

templates

Actual Name of ansible_facts is system.

∴ {{ansible_facts["system"]}}

{{ansible_facts[~~ansible~~"default-ipv4"]["address"]}}

{{ansible_facts["default_ipv4"]["address"]}}

→ fetching Ip address from facts

Listen 8080

<Virtualhost {{ ansible_facts['default_ipv4']['address'] }} :8080>

documentroot /var/www/lw

</Virtualhost>

Now, we have to use template module instead of copy module.

```
- template :
    src : "/root/lw.conf"
    dest : "/etc/httpd/conf.d/lw.conf"
```

Date: 01/12/2020
Day:8

HTTP Authentication

Webserver will check the login name and password sending login name and password in HTTP protocol.

cd /etc/httpd/conf/ ~~conf.d~~ httpd.conf

Steps to do HTTP Authentication.

step 1: package httpd install.

step 2: copy Webpage : /var/www/html

step 3: configure auth /var/www/html.
Replace allowoverride authconfig

step 4: create a file .htaccess at /var/www/html

step 5: install python's pip 3.

step 6: install python lib pip : passlib

step 7: create a user and its password inside /var/www. With name .passwd.

step 8: Restart the httpd service.

Here you can search the doc or module using command.

ansible-doc -l | grep package_name

ansible-doc package_name → detailed example

Requirements

ansible-doc pip → pip module

ansible-doc htpasswd → htpasswd module.

Webserver will asks user name and password. from the url of the Webserver.

We need to create own database for http authentication.

you will not have http authentication file inside /var/www/html.

Main conf file for the apache ~~os~~ is

Vi /etc/httpd/conf/httpd.conf.

inside this httpd.conf the setting done is none.
so you need to change that.

you need to write AuthConfig. AuthConfig will enable the Authentication configuration of http.

so replace AllowOverride None ⟹ AllowOverride AuthConfig

A and C is capital.

Cat line 134

you need to create login name and password. and the name of the file where you create username and password is .htpasswd.

inside ~~/etc~~ /etc/httpd/conf.d

Run below comman

htpasswd -c /etc/www.passwd ~~filename~~ username

-c will create file

then type password and /etc/www.passwd they create user and password in encrypted format.

if you want to create one more user then don't pass -c option.

htpasswd /etc/www.passwd username

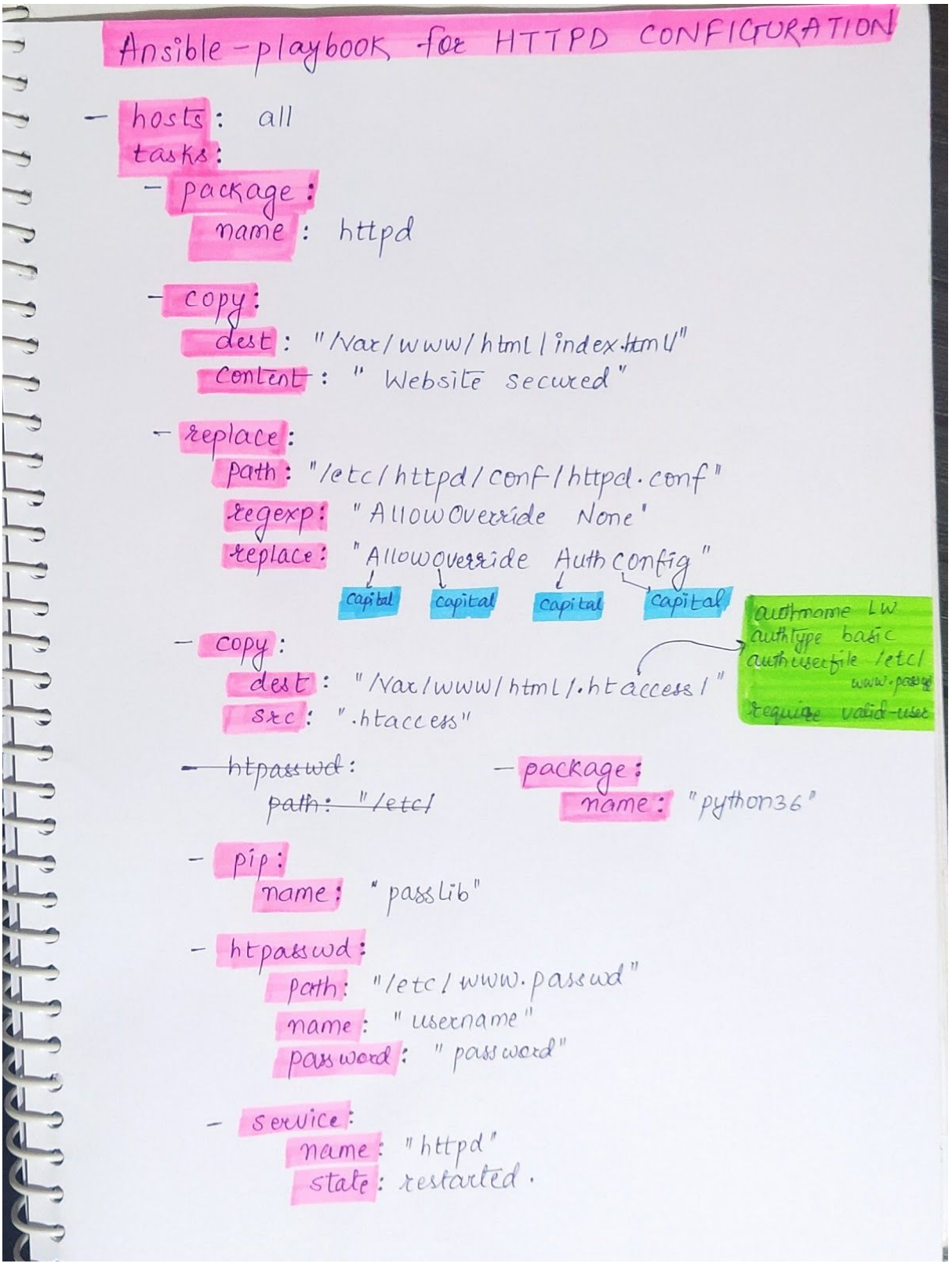
- Our Webserver doesn't know where you have created username and password.
- There is mentioned in apache configuration file that to know where you have created username and password we need to create ~~.htpasswd~~ .htaccess inside documentation root (/var/www/html).

Vi .htaccess

inside .htaccess you need to write below code.

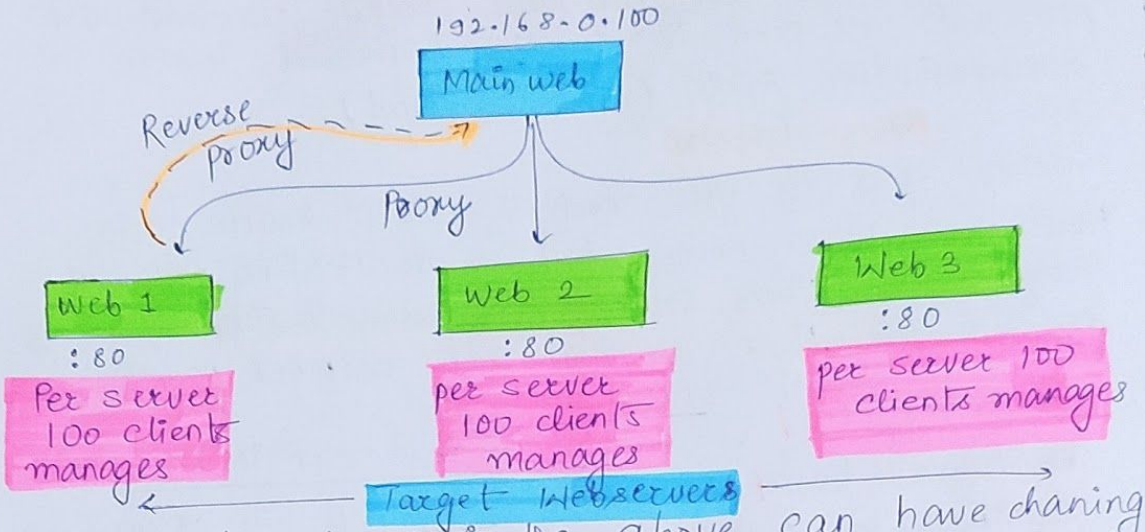
How your folder know which authentication you are going to use. There is a basic authentication specially designed for login username and password page.

```
authname LW          ← popup message
authtype basic       ← type authentication is basic
authuserfile /etc/www-passwd   ← set the location of your file
require valid-user   → all user are valid here.
```



Date: 02/12/2020
Day:9

CONFIGURING LOAD BALANCERS USING ANSIBLE



- All the targets ⊙ in the above can have chaning the Ip. so the application in the webserver will not be driven from one static Ip. so this is a challenging task.
- Also if your application's webserver can be crashes, terminated. so to avoid this challenging issues. all the webservers will have to send the requests to one Master Webserver. One that server will have one single Ip. ~~test~~ let say 192.168.0.100.

If any client comes to 192.16.0.100 then it will recreate the request to one of the target Servers.

Then the target server will reply to Master Webserver. This is known as reverse proxy.

In this case, the client will never hit the web-server. or We can say that client never goes to the target webseers.

- Target Webservers will never goes to the public world. only the main webservers will be connected to the target Webservers.
- To configure the proxy, we have different products in the market. eg. nginx, haproxy.

Reverse proxy is known as front end.

The port on the main webservers work is known as front end port.

The Backend servers port will have to tell to the reverse proxy. (HAproxy)

HAPROXY is just a product name. concept is known as reverse proxy.

Load Balancer works on the concept of Round Robin.

Steps to configure Haproxy:

Step 1:- Install Haproxy.

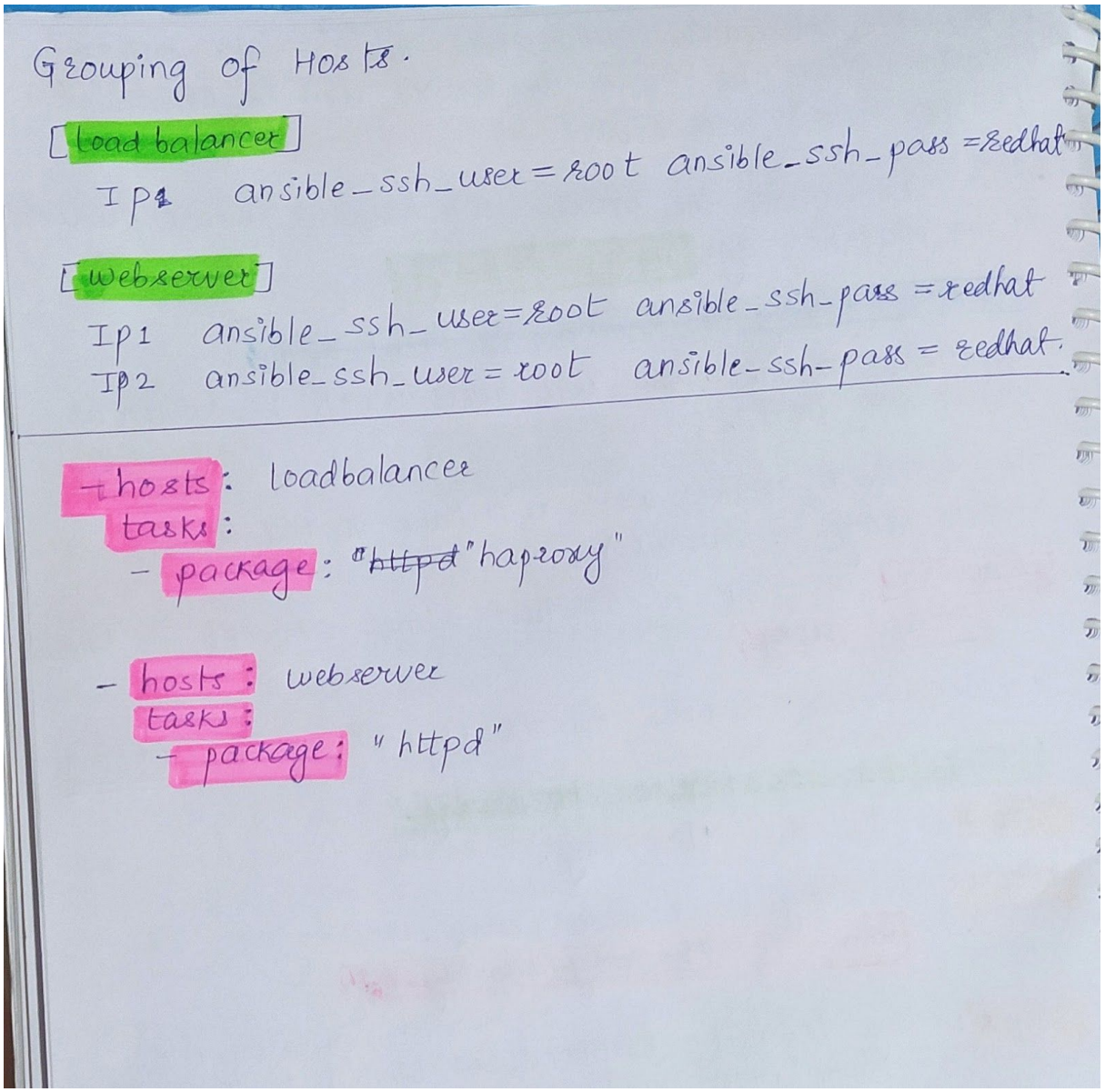
Step 2:- configure reverse proxy.
Vim /etc/haproxy/haproxy.cfg.
↳ configuration file.

Step 3:- By default frontend is Binded with 5000 port.
change your port.

Step 4:- By default ~~fro~~ backend have app block.
you need to give your server's Ip to backend app.

```
backend app
    balance roundrobin
    server app1 Ip1:80 check
    server app2 Ip2:80 check
```

Step 5:- Start the HAPROXY
Systemctl start haproxy



Date: 03/12/2020
Day:10

Grouping of Hosts.

[loadbalancer]

Ip1 ansible_ssh_user=root ansible_ssh_pass=redhat

[webserver]

Ip1 ansible_ssh_user=root ansible_ssh_pass=redhat
Ip2 ansible_ssh_user=root ansible_ssh_pass=redhat

```
- hosts: loadbalancer
  tasks:
    - package: "~~httpd~~" "haproxy"

- hosts: webserver
  tasks:
    - package: "httpd"
```

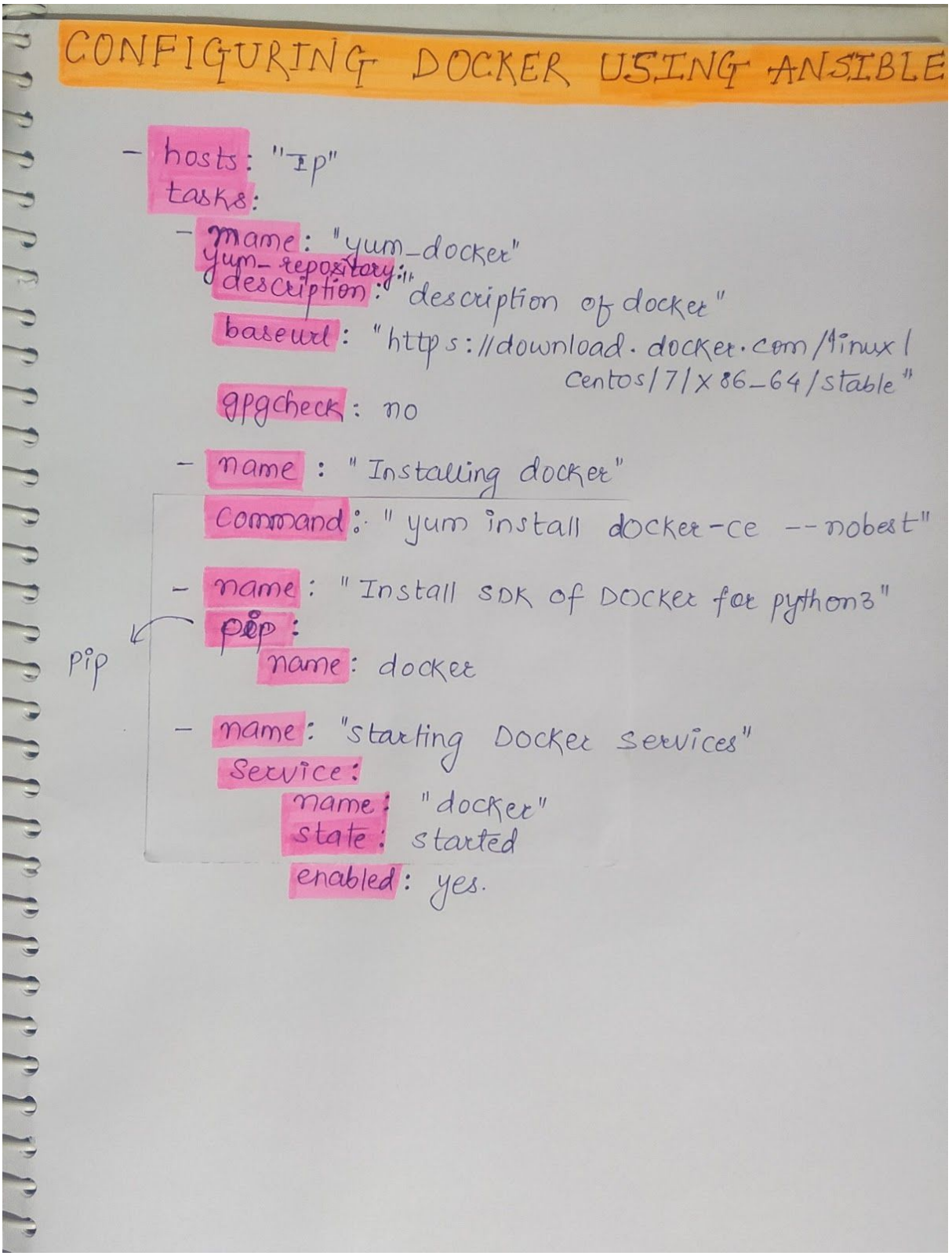
copy the httpd.conf file.

```
cp /etc/httpd/httpd.conf /root/
```

and Now we can copy the httpd.conf with our own editings into target node from controller node.

```
- hosts: "loadbalancer"
  tasks:
    - package: "haproxy"

    - template:
        src: "/root/httpd.conf"
        dest: "/etc/httpd/"
    - service:
        ~~stat~~ name: "haproxy"
          state: "restarted"
```

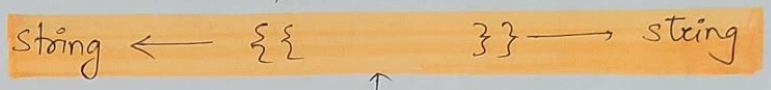
Note: If you have disconnect error, then disable Selinux in controller node.



Date: 04/12/2020
Day:11

JINJA IN ANSIBLE

- copy module will not parse the data which it copies whereas template module does.
- for copy module, every thing is a string. Whereas, we need to tell the Template that which are the variables, loops, conditions, etc inside the file.



What's inside this, template module processes it and give the value of it.

Behind the template module, they ~~you~~ use {{ }} # etc, keywords. and this keywords comes from a language known as jinja.

Jinja is a framework or a library in python.

One variable contains multiple items.

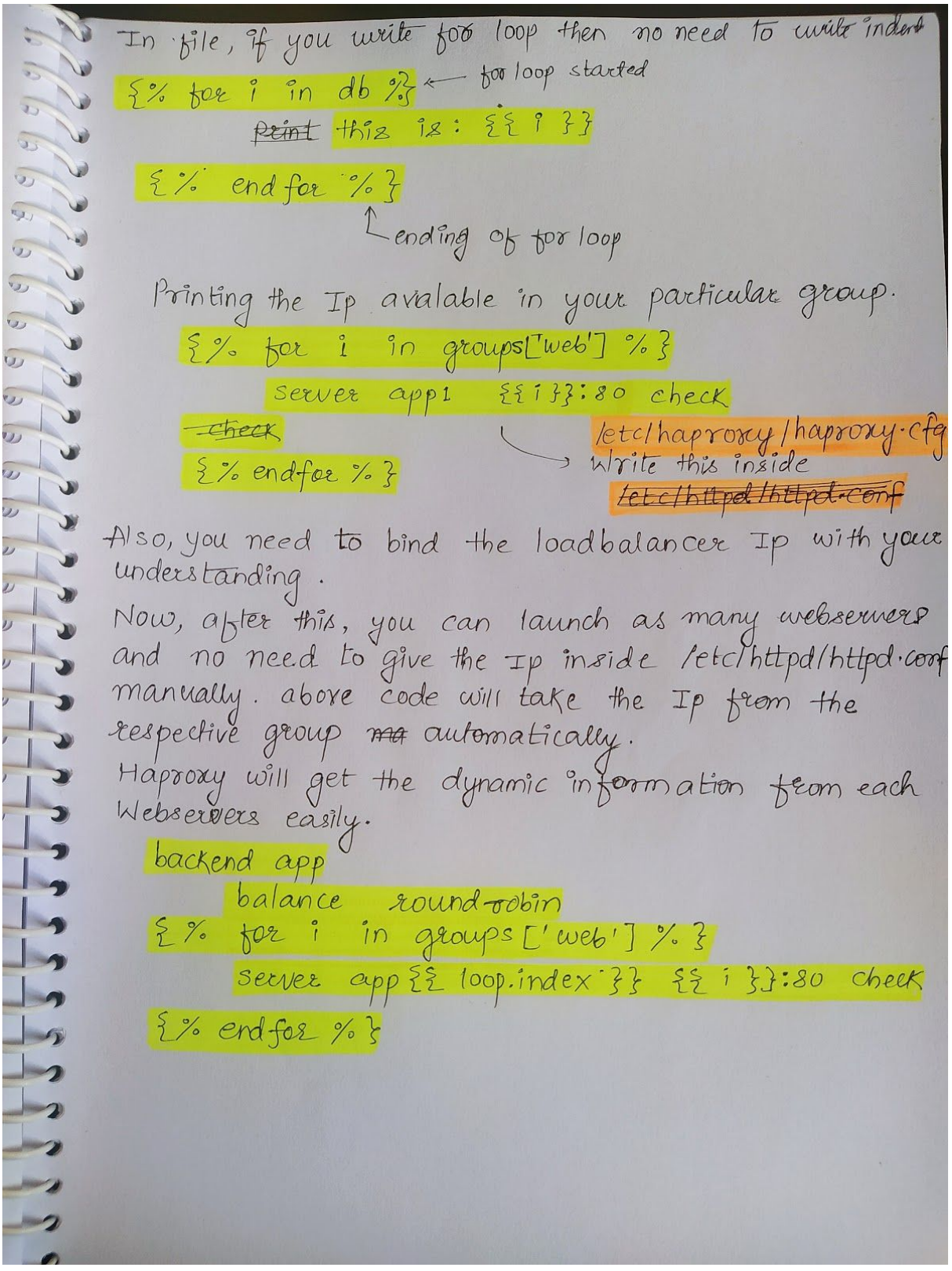
```
- hosts: x
  vars:
    - x:
        - "pop"
        - "harry"
        - "Jack"
```

Here, x contains three items. x is a list.

For loop in python,

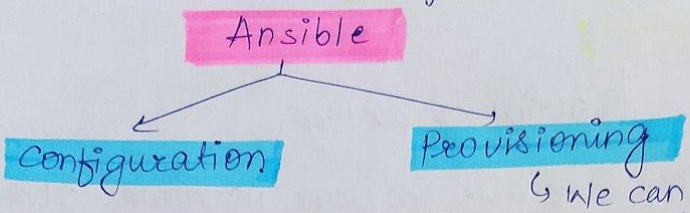
```
db = ['a', 'b', 'c', 'd']

for x in db:
    print("This is: ", x)
```

```
>>> This is: a
    This is: b
    This is: c
    This is: d
```



Date: 09/12/2020
Day:12

- Ansible is only meant for configuration management tool.
- Ansible can also used for provisioning the OS.

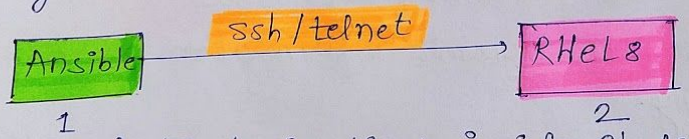


- Terraform and Ansible Works together. Terraform do provisining and Ansible do configuration.

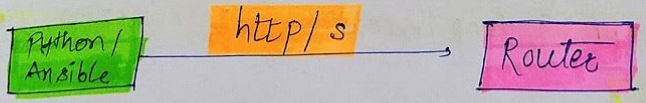
↳ We can use terraform, foreman for provisioning.

LAUNCHING AWS EC2 INSTANCE USING ANSIBLE

- API can be integrate with any of the programming language.
- Ansible can be integrated with the API to make the configuration. so we need to install a library. Boto, Boto3 is the library that will helps Ansible to integrate with API.

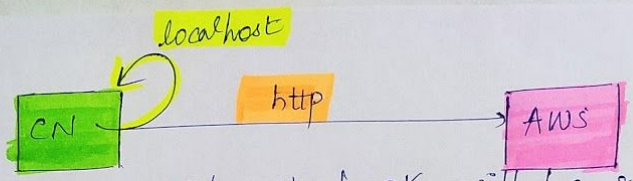


We need to login first inside Rhel8 through Ansible to do any configuration.



We won't login inside Router. so using interface program we can do the configuration.
Boto, Boto3 does you for the same.

```
yum install python3
pip3 install boto
pip3 install boto3
```



your ansible playbook will be running inside your own host. And ~~for~~ using the http protocol we can use the API of Aws to ~~do~~ the provisioning.

```
- hosts: "localhost"
  tasks:
    - name: "Launching ec2 instance"
      ec2:
          key_name: "keyname"
          instance_type: "t2.micro"
          image: "Image_id"
          wait: yes
          count: 1
          instance_tags:
              Name: "os_name"
      vpc_subnet_id: "subnet_id"
          assign_public_ip: yes
      ~~region: "ap-south-1~~
          state: present
          region: "ap-south-1"
          group_id = "security_group_id"
          aws_access_key: "access_key"
          aws_secret_key: "secret_key"
```

Note:- All the above parameter must be in the same indent. so, vpc_subnet_id is in ~~wrong~~ indent. it should be in same indent.

Date: 10/12/2020
Day:13

CONDITIONS IN ANSIBLE

- We can manipulate our playbooks using conditions.
- conditions denoted by "when" keyword.

- you had a requirement such that you need to configure apache webserver in Redhad OS and Ubuntu as respectively.

  Software http → in redhad. OS.

  Software apache2 → in ubuntu OS.

We can do this with the help of facts.

```
ansible all -m setup
```

→ showing facts of OS

the Name of os ansible's facts stores in ansible distribution.

```
- hosts: Ip
  vars:
    - x: "{{ ansible_distribution }}"
  tasks:
    - package:
        name: "httpd"
        state: present
      When x == "Redhat"
```

↳ or ansible_facts[ansible_distribut

case sensative. capital R, capital

when x == redhat Package task will run.

If x != "redhat" then ansible will skip the abov tasks.

Difference between var and msg in debug

```
- debug:
    var: "{{ ansible_facts['ansible_distribution'] }}"
- debug
    msg: "my os name is {{ ansible_facts['ansible_distribution'] }}"
```

apache2 installing for ubuntu os

```
- hosts: all
  Var: - x: "{{ ansible_facts['distribution'] }}"
  tasks:
    - package:
        name: httpd
        state: present
      When x == 'Debian'
```

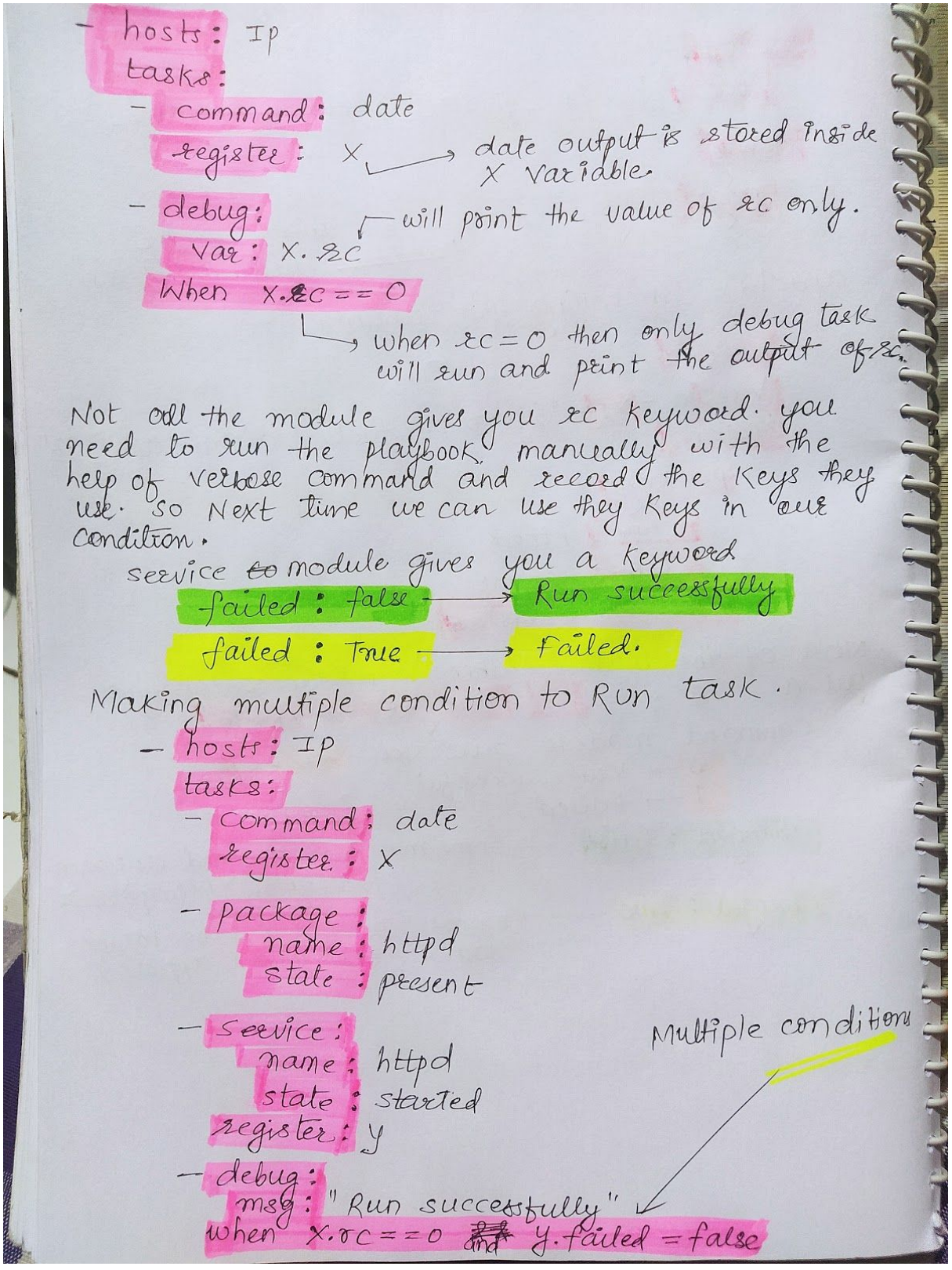
Most of the module When run succefully gives you a code. i.e rc, Changed: true/false.

command module gives you rc: 0/1

0 → Run successful
1 → failed.

changed: false → means not changed anything facility of idempotance

Changed: True → means changed on the target node. (yellow colour appears)



Here in multiple condition,

When X.rc ==0 and y.failed == false

Date: 11/12/2020

Day:14

EXCEPTION HANDLING IN ANSIBLE

- If you stuck by a error in a playbook then the entire tasks fails.
- To avoid this situation Ansible provides you some key words to ignore these and move on next task.

```
ignore_errors: yes
```

eg.

```
- hosts: all
  tasks:
    - package:
        name: "httpd"
    - copy:
        src: "path"
        dest: "give wrong path"
      ignore_errors: yes
    - service:
        name: "httpd"
        state: restarted
```

(1) Here playbook will throw a error

(2) This will ignore the copy task

(3) the httpd will be restarted.

In the above playbook we are restarting the httpd everytime the playbook runs. so restart is not a good practice. By restarting the service the conf file loads on the Ram and also it will remove the caches. some times caches will plays a vital Role for searching errors. So we need to set a playbook such that if the change is there in the copy module or in the content of the file then only it will restart the service.

- so to tackle this problem Ansible provides you with a module name handlers.
- handlers will handle you playbook's module.
  So below code is written such that if you changed the content then it will restart the service. handler comes with the keyword notify. notify will helps handler to notify or which handler you want to run. There can be many handlers so you need to notify those handler.
- We need to give handlers a name. so all the handlers in a single playbook easily ~~not~~ can be understood.

```
- hosts: "Ip"
  task:
    - name: "Installing httpd"
      package:
        name: "httpd"
    - name: "copying the pages"
      ~~scm~~ src: "path"
        dest: "path"
      notify: web_server  → playbook will get notified

  handlers:
    - name: web_server
      service:
        name: "httpd"
        state: restarted  → This will run and restart the service.
```

Handlers in Ansible

Date: 15/12/2020
Day:15

When you run the ignore_errors module then they skips the tasks. Sometimes its not good to skip the tasks.

```
ansible-doc uri
```

```
- uri
    url=" Enter your url"
```

```
- get_url:
    url: "enter your url"
    dest: "/destination/"
```

If your system has internet connectivity then only runs this. If not the fails.

When the ansible doesn't understand the module they gives exception.

```
route del -net 0.0.0.0
```

↳ Internet will be disconnected.

Simplest way to handle the error is ignore_errors. it will skip the errors/tasks. If your the respective tasks is dependent then there you don't have to use ignore_error.

BLOCKS AND RESCUE

Blocks and rescue is another way to handle exceptions in Ansible.

Blocks → This will have a code/tasks in its blocks. If any respective tasks from the respective Block the it will send message to ~~Resuce~~ Rescue block. and Rescue will print the message.

```
- hosts: 'Ip'
  tasks:
    - block:
      - package:
          name: "httpd"
          state: present

      - get_url:
          url: "enter your url"
          dest: "/path"

      - service:
          name: "httpd"
          state: ~~re~~ started
    rescue:
      - debug:
          msg: "exception handled".

    ~~fino~~
    always:
      - debug:
          msg: "This will always run"
```

- block:
  ⋮
  rescue:
  always:

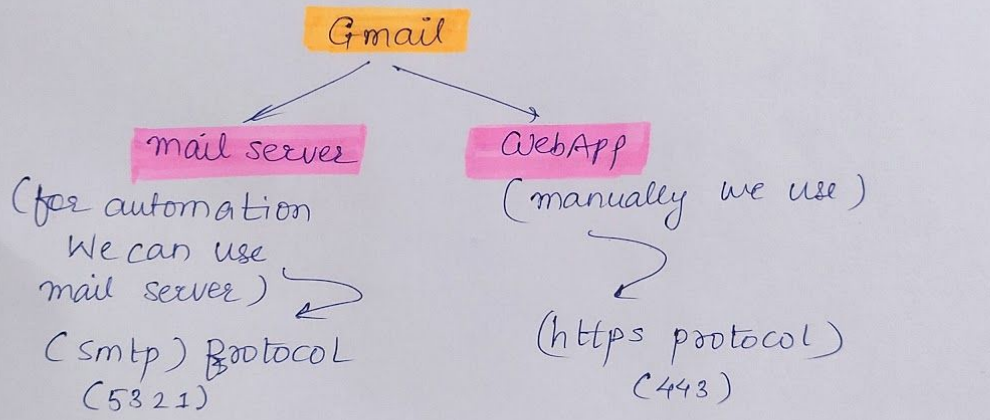
first block

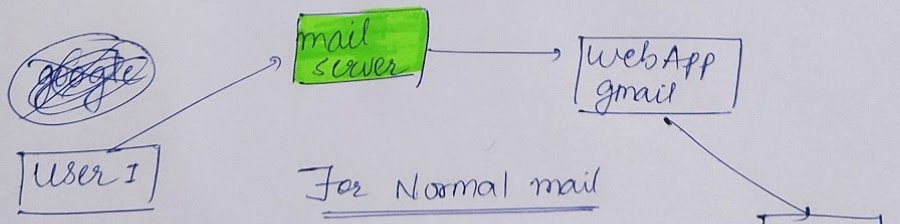
- block:
  rescue:
  always

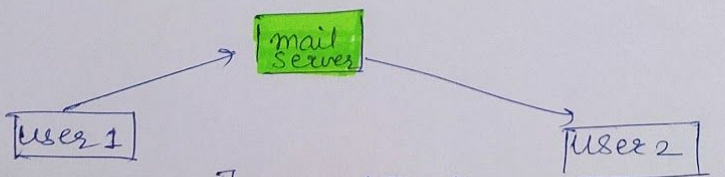
second block.

`ansible-doc mail`

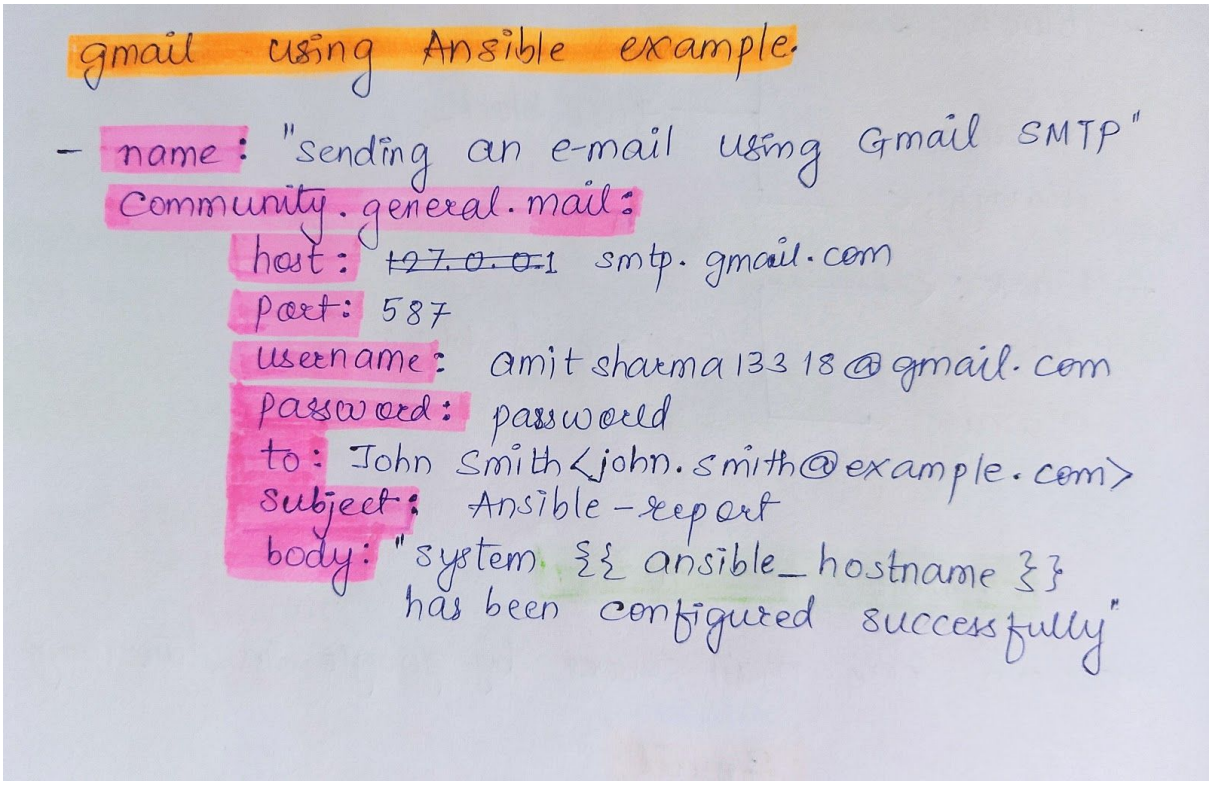
We can use mail server by google in two ways







For automation mail



Date: 17/12/2020
Day:16

gmail using Ansible example.

```
- name: "sending an e-mail using Gmail SMTP"
  community.general.mail:
      host: ~~127.0.0.1~~ smtp.gmail.com
      port: 587
      username: amitsharma13318@gmail.com
      password: password
      to: John Smith <john.smith@example.com>
      subject: Ansible-report
      body: "system {{ ansible_hostname }}
             has been configured successfully"
```

CREATING YOUR OWN IDEMPOTANCE NATURE

```
- hosts: ip
  tasks:
    - package:
        name: "httpd"
        state: present
    - command: "date"
      changed_when: false.
```

```
- command: "mkdir /w1"
  when: false
```

this will not run as when: false.

`ls -l /lw3` >>> No such directory

`echo $?` >>> 2

`ls -l /lw2` >>> 0

`echo $?` >>> 0

If the value of exit code is not zero then the directory does not exists.

```
- hosts: "IP"
  tasks:
    - command: "ls /lw3"
      register: x
      ignore_errors: yes
    - debug:
        var: x
    - command: "mkdir /lw3"
      When: x.rc != 0
      Changed_when: false
```

this command will run like idempotance

↳ or

```
- command: "mkdir /lw3"
  When: x.rc != 0
  When: false
```

↳ this task will be skipped

```
cat /etc/passwd | wc -l
>>> 50
```

↳ 50 lines ⟶ 50 users

Command module will only support commands of the systems. But commands module doesn't support shell symbols of shell.

Therefore we have shell module, so we can use the shell symbol inside shell module.

The shell module is ~~compare~~ slow as compare to Command module.

```
- shell: " cat /etc/passwd | wc -l "
  register: no_of_users
- debug:
    Var: no_of_users
```

VAULTS IN ANSIBLE

Ansible ~~valut~~ Vault is a feature of ansible that allows you to keep sensative data such as password or keys in encrypted files, rather than as plaintext in playbook or roles. These vault files can then be distributed or placed in source control.

CREATING VAULT :-

```
ansible-valut create --vault-id password@prompt secret.yml
```

The above command will create a vault to secret.yml ~~when~~ the secret.yml file will be given inside the main playbook.

```
- hosts: "Ip"
  vars_files: secret.yml
  ↓ tasks
```

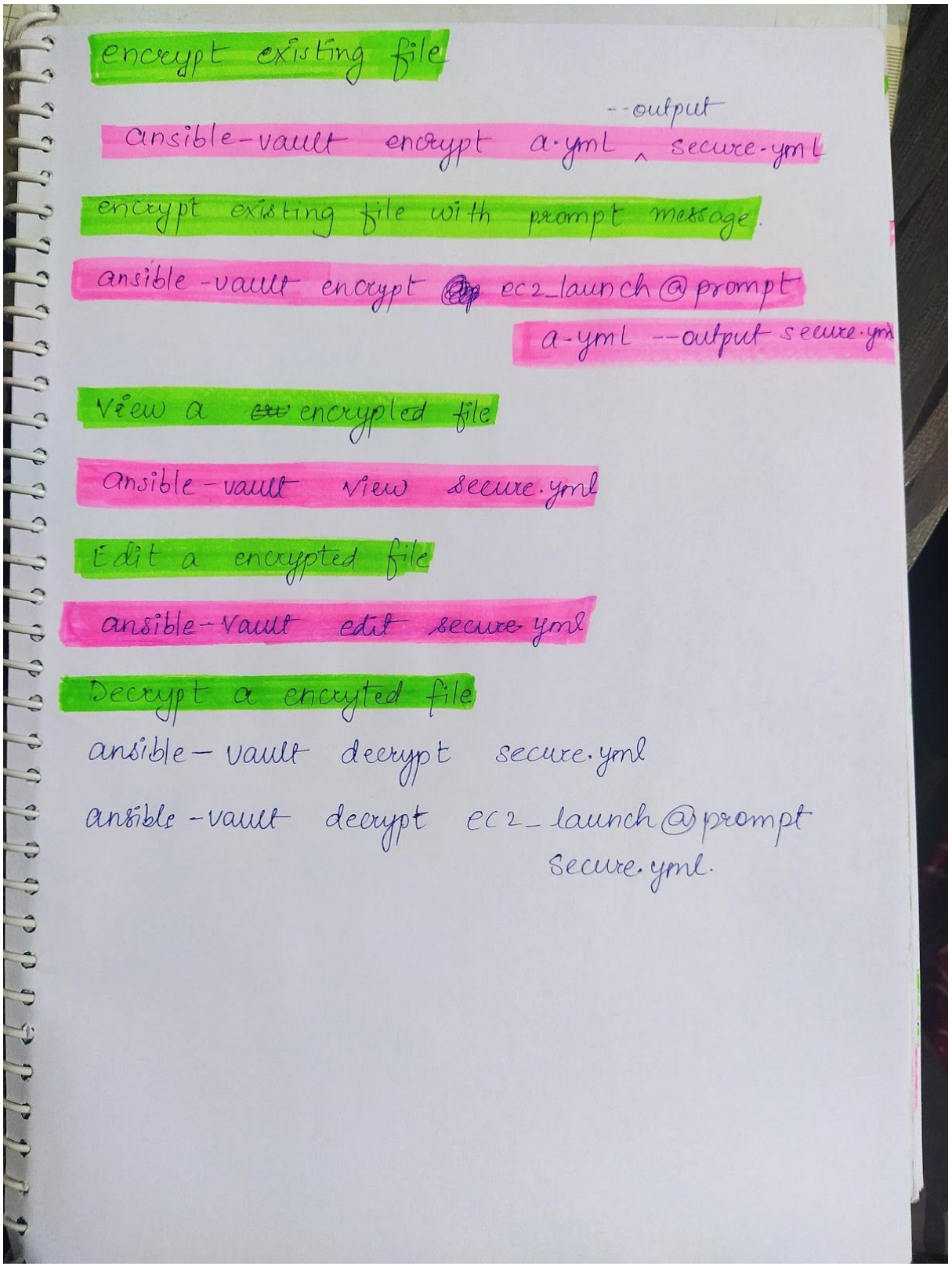
Whenever you run the above playbook you need to specify the vault file you created.

```
ansible-playbook --ask-vault-pass playbookname.yml
```

By running the above playbook the option of prompt you have passed while creating a vault it will prompt here lik below.

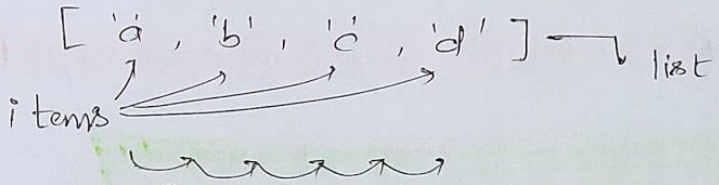
```
password:
```

after getting the password then only the whole playbook will be executed.



Date: 18/12/2020
Day:17

LOOPS IN ANSIBLE




```
P = ["httpd", "php", "nginx"]
```

temporary variable

```
for i in p:
    print(i)
```

In Ansible the pre created temporary variable known as items.

In Ansible we have loop keyword except for from python.

loop execution in Ansible.

```
- hosts: all
  tasks:
    - package:
        name: "{{ items }}"
        state: present
      loop:
        - "httpd"
        - "php"
        - "nginx"
```

The loops values will go inside {{ item }} and the loop will run three times as the values in the loop is 3.

in ansible previous ~~pers~~ version the loop keyword was not there. Instead of loop we were having with_items

Or you can create a variable that will contain multiple softwares. and loop will be executing that variables.

```
- hosts: all
  vars:
    - x:
      - "httpd"
      - "php"
  tasks:- "ngimx"
  -package:
      - name: "{{ item }}"
        state: present
    loop: "{{ x }}"
  -debug:
      var: x.[0]
```

→ will point 0th item from the x list.

`groupadd group_name` ⟶ creating group.

`useradd -G group_name user_name`

Adding users to respective groups

`usermod -G group_name user_name`

Modifying another user to the same group

Creating users using Ansible.

```
- hosts: "Ip"
  tasks:
    - user:
        name: "Jack1"
        password: "redhat"
        state: present
```

When you create a user a seperate group is created for that user known as primary group. But if you want to add a user in different group that is known as secondary group

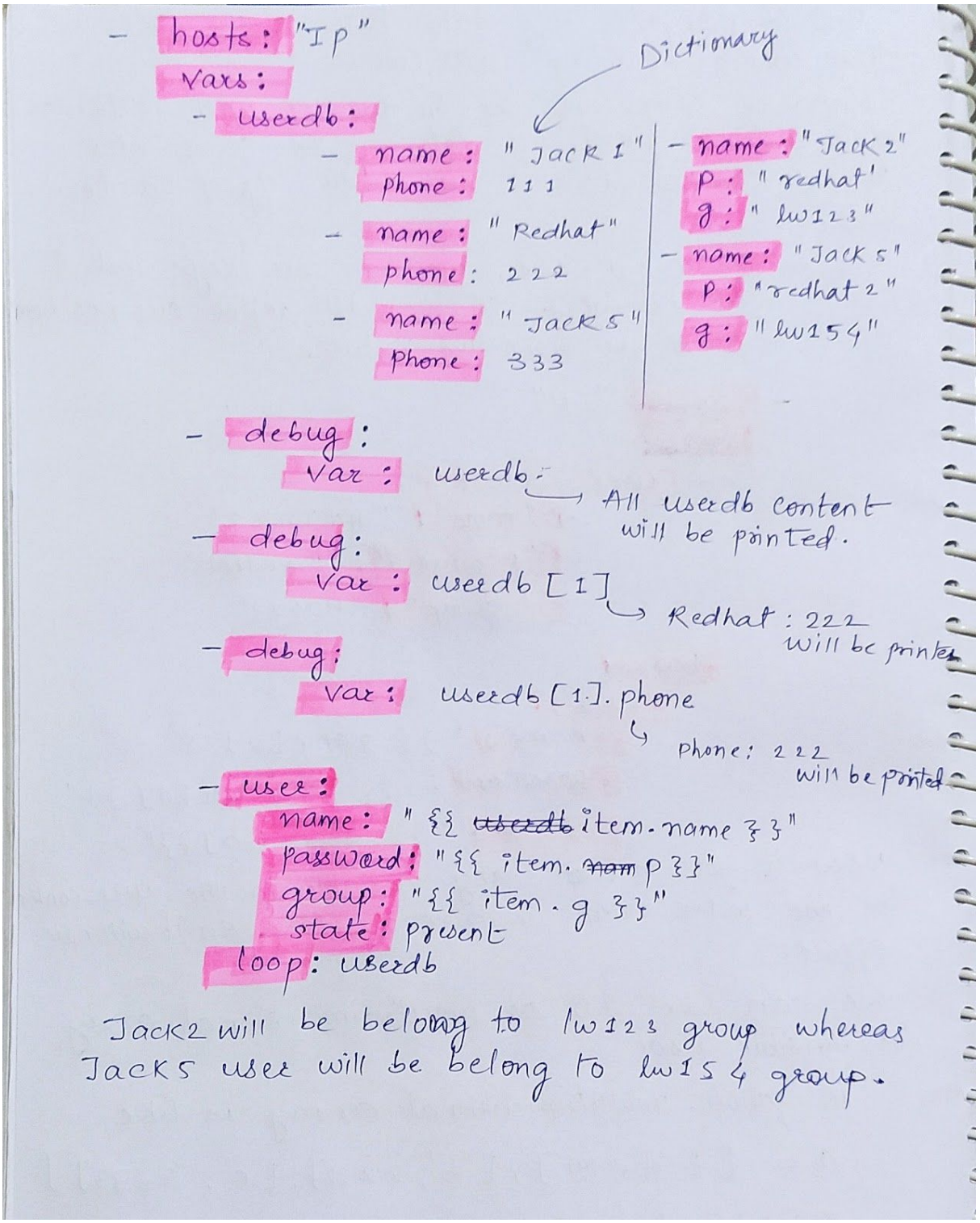
```
- groups: "lw123"
```

↓

user "jack1" will now added to ~~neutyce~~ group lw123.

Adding multiple users to the group

```
- hosts: "Ip"
  vars:
  - u1: "Jack6"
    - "redhat"
    - "lw123"
  tasks:
    - user:
        name: "{{ u1[0] }}"
        password: "{{ u1[1] }}"
        group: "{{ u1[2] }}"
        state: present
      loop: "{{ u1 }}"
```



Date: 23/12/2020

Day:18

Simplifying Playbooks With Roles

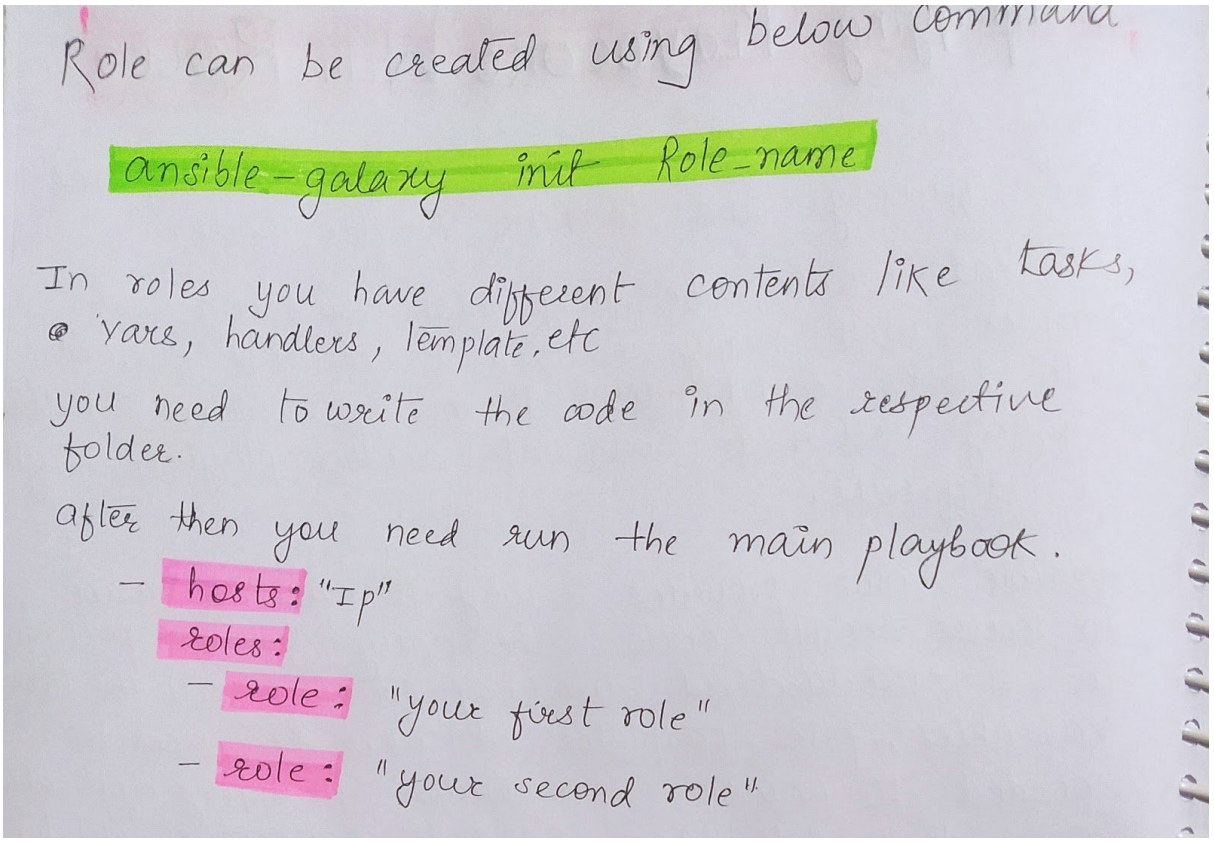
using Ansible Roles We can develop playbooks more quickly and to Reuse Ansible code.

In real words, the play might be long and complex, with many included or imported files. and with tasks and handlers to manage various situation. copying all that code into another playbook might be difficult.

Ansible roles provides a way to make it easier to reuse Ansible code generically. you can package, in a standardized directory structure, all the tasks, variables, files, templates, and other resources needed to provision infrastructure or deploy application. Copy that Role from project to project simply by copying the directory. you can then simply call that role from a play to execute it.

Ansible roles have the following benefits :-

- Role group content, allowing easy sharing of code with others.
- Roles can be written that defines the essential elements of a system type : web server, database server, Git repository, or other purpose.
- Roles make larger projects more manageble
- Roles can be developed in parallel by different administrators.



Date: 24/12/2020
Day:20

PREVILEGE ESCALATION IN ANSIBLE

Any user that has all the power is also known as **privilege user.**

The command which works in root user or privilege user will not run in the Normal user user you created.

If you want that user will have all the power then it is known as **privilege escalation.**

sudo gives you the facility of privilege escalation

root account have **id = 0**

When you run "id" in the terminal then you will get your respective user id.

But when you run the "id" command with the sudo then your normal user will become root and id of the Normal user changes to uid = 0 which is the id of root.

When the Normal user changes ids id ~~after~~ with running sudo then this concept is known as becoming.

become = True.

↳ your user will changes to root user.

To run your command from the root user you need to become root.

~~become~~ **become_user = root**

Also to becoming root you need to run the command with the power of sudo. So,

become_method = sudo.

So in ansible configuration file (ansible.cfg) you will have a keyword name privilege escalation.

```
[privilege_escalation]
become = true
become_user = root
become_method = sudo
become_ask_pass = false
```

so every time you run the playbook then the privilege escalation inside configuration file will run. and the command will run from the root user using sudo (substitute user domain) whenever you run ~~passwo~~ sudo then it will ask password for the root use. so the above command ask_pass or become_ask_pass = false. so it will not ask the password.

you can run the privilege escalation's parameter using Ansible adhoc command.

```
ansible "Ip" -m package -a "name = httpd" state = present" -- become user root -- become method sudo --ask-become-pass.
```

Now the above command will ask password and make the necessary configuration.

So its a tedious task to ~~run~~ write the privilege escalation's parameter again and again. so we will ~~help~~ ~~the~~ take the help of ansible configuration file.

become_ask_pass = false will run only when the user have given NOPASSWD : ALL in the sudoers file.

Normally, the password authentication in Aws linux ec2 instance is disabled by default. so do that you can generate a ssh key and copy the ssk key id to respective user.

`ssh-keygen` ⟶ create a ssh key

`ssh-copy-id` ⟶ will copy the ssh key id

`ssh-copy-id user@Ip`

This is known as passwordless authentication. Because when you login to your respective user where you have copy the ssh key id it will not ask password. you have already provided that.

Also you need to do ssh password authentication enable.

`Vi /etc/sshd/ssh_config`

In the above you need to do passward Authentication yes

`password Authentication = yes`

the start or restart the sshd service.

`Sevice sshd restart`

ANSIBLE GALAXY

> ansible-galaxy role search apache

> ansible-galaxy role search

↳ this will search all the list of role.

> ansible-galaxy install geerlingguy.apache

(geerlingguy: accountname, apache: rolename)

This command will install apache role from geerlingguy account/user.

Newer command to install role

> ansible-galaxy role install geerlingguy.apache

If you want to include other tasks,

- include_tasks: "task_name.yml"

To include vars;

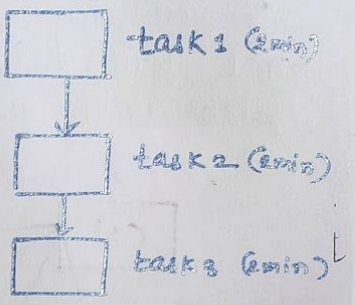
- include_vars: "vars.yml"

ASYNC - BATCH - PARALLELISM

ASYNC BATCH PARALLELISM

> By default Ansible playbooks runs the task in synchrous way.

> Ansible runs the task serially or We can say in **synchronous Way**.



> So, Ansible Waits if the tasks is yet to completes.

> Also if your host is multiple let say you have 4 Ip's in host. The Ansible will install the software or configure the tasks in all 4 Ip's or system.

> This is known as **sync.** (sereally)

> So parallelism is a concept, which helps to runs the task in a parallel way.

> By default, ansible playbook runs on sync

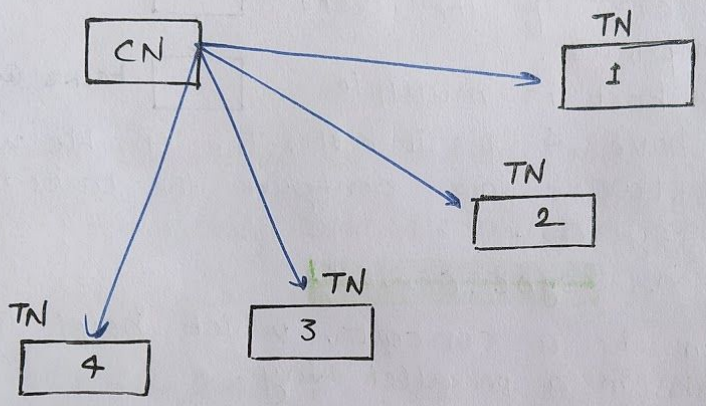
We can place the groups inside the main groups the groups inside the main groups are known as **children**.

```
[web]
ip 1
ip 2
[db]
ip 3
[lb]
ip4
[winos]
ip5

[linuxos:children]   <- main group
web
db
lb                   <- childrens
```

netstat -nct

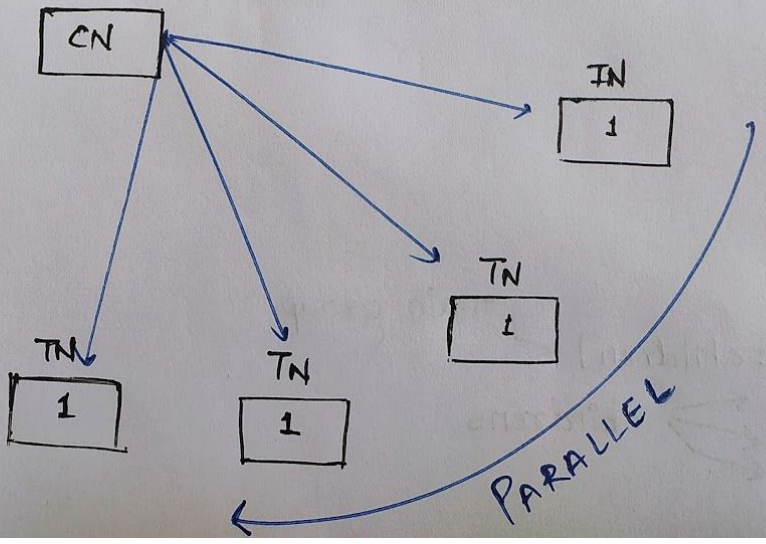
You can check on the realtime on which of the system is your connected using ssh.

If you do ssh with certain Ip then netstat -nct will helps you to see on the real time which system with ~~por~~ which protocol the system is connected.



RUNNING PLAYBOOK IN SERIAL WAY (DEFAULT)

you can ~~sheet~~ check by running netstat -nct after running the playbook in the controller Node. If one Ip comes then it will be proved that Ansible runs the playbook in serial manner.



If you want to establish parallel configuration then you need to tell how many nodes you want to keep in parallel.

By default in ansible.cfg file we have a keyword which defines the parallel scenario. The name of the keyword is forks.

By default the value of the forks is 1

Creating a connection to the managed node is known as forks.

By default the value of forks is 1 therefore Ansible runs the playbook in a serial manner.

```
ansible-config dump
```

The above command will shows you how much keyword supported by ansible in ansible configuration. and should be in which section eg. [defaults], [privilege_escalation], etc.

```
ansible-config dump | grep i forks
```

↳ insensative.

```
DEFAULT_FORKS(default) = 5
```

By default the value of forks will be 5

If you want to update in the configuration file,

```
[defaults]
inventory = path
host_key_checking = false
roles
forks = 2
```

↳ Now 2 host will be configured parallelly.

ansible-config list | grep -i fork

>> DEFAULT_FORKS:

description: Maximum number of forks Ansible will use to execute tasks.

- { name: ANSIBLE_FORKS }
- { key: forks, section: defaults }

name: number of tasks forks

→ section is default.

ansible-config list

→ will shows you all the commands/keywords written to be in which section of ansible configuration file.

By default Ansible is syncing with the task. Means they are monitoring, polling, checking the tasks. We can say Ansible is sync the tasks.

```
- hosts: localhost
  tasks:
    - name: task 1
      command: sleep 10
    - name: task 2
      command: sleep 5
```

→ Ansible is polling/monitoring the task till the task1 completes.

task1 completes then task2 will run.

If your task1 is independent of task2 then you can trigger the tasks independently. means task1 and task2 will run simultaneously. This is known as Async. (In Asynchronous manner).

If you want to run the tasks in parallel the we can go with Async.

In Async you have to set the timer of the respective tasks.

```
- hosts: localhost
  tasks:
    - name: task1
      command: sleep 20
      async: 50
      poll: 2
    - name: task 2
      command: sleep 5
```

As you can see in the above code, task 1 will run upto 20 s. We have set the timer of the task i.e 50 s. Till 50 s the task1 will be having a chance to run. And the poll after every 2 s. keeps on checking the task 1 is it completed or not. If it is not completed after 50 s then Ansible will fail that task 1.

We have set poll: 2 s therefore, it will be coming again and again to check the task 1 without going or ruhing task 2. Therefore task 1 will run first then only task2 will run. But this occurs due to poll, that its coming again and again. if we set poll: 0 s then task 2 will not wait for task 1 to complete .and runs the task2 immediately.

```
- hosts: localhost
  tasks:
    - name: task 1
      command: sleep 20000
      async: 50000
      poll: 0 ----> Not wait to complete task 1.
    - name: task 2
      command: sleep 20
```

If you giving poll: 0 then Async time will not matter Poll will skill the task1 it not run at the respective time.

We can make async per task basis. We can also track the task Wether the task is completed the program or have to done the configuration or not.
Before closing the playbook, ansible will give the output of previous task which was ignored by async.

We can use job_id or we can say ansible_job_id

To track respective task We have a module in ansible known as async_status.

async_status will show you the started, finished, failed status of the tasks.

```
- hosts: localhost 'Ip'
  tasks:
  - name: task 1
    command: sleep 2000
    async: 55
    poll: 0
    register: X  ——→ output register in X
  - name: task 2
    command: sleep 3
  - name: task 3
    command: sleep 5
  - debug:
      var: X
  - async_status:  ——→ predefined keyword
      jid: "{{ X.ansible_job_id }}"
    register: status  ——→ output registered in status
    until: status.finished  ↳ until the task is finished till then ansible will check the status
    retries: 30  ↳ No. of retries till 30 retry the task1 will take
```

Ret
retries belongs to async_status not with async, poll.

If you had a requirement to install or configure something first in all the target nodes and then you can run something to make it useful. like this If you had a requirement then, Ansible had a keyword for this, i.e serial. like, if you want to install, configure, copy some files, software then in serial you can give the numbers ~~of tasks~~ of the hosts to be in the serial.

like,

```
- hosts: Ip
  serial:
    - 1  ⟶ only one host will be taken
    - 2  ⟶ two hosts will be taken
    - 3  ⟶ Three hosts will be taken.
  tasks:
```

Now, what you ever wants to configure, install copy, etc then it will run in the first host i.e only one Ip and for all its respective tasks. Similarly, it will take two hosts next time and three hosts at the third time.

group of 2, 3 are known as Batch.
Batch of 2 Ip's, Batch of 3 Ip's.

CUSTOM FACTS :-

you can create your own facts known as custom facts.

you need to create a facts.d folder inside /etc/ansible/facts.d

In this folder you can create your file, with your own name eg. lw.fact. ensure the extension of the file must be .fact.

Inside the file you can give your information about the facts.

```
[webserver]
team = dev
region = IN
env = prod
```

Now when you run the facts command then the above custom facts will be also included.
Make sure that you need to give the file name with .fact extension. this will also include the path of the custom facts.

~~ansible_facts.lw.webserver~~ ⟶ ~~path of your custom facts~~.

the file lw.facts will be created inside the managed node inside /etc/ansible/facts.d/lw.facts

```
ansible ip -m setup -a "filter=ansible_local"
```

the location of your custom facts will be

ansible_facts.ansible_local.lw.webserver

you can use your custom facts for your ease deployment or configuration in the target node.

Date: 23/12/2020
Day:21

DYNAMIC INVENTORY

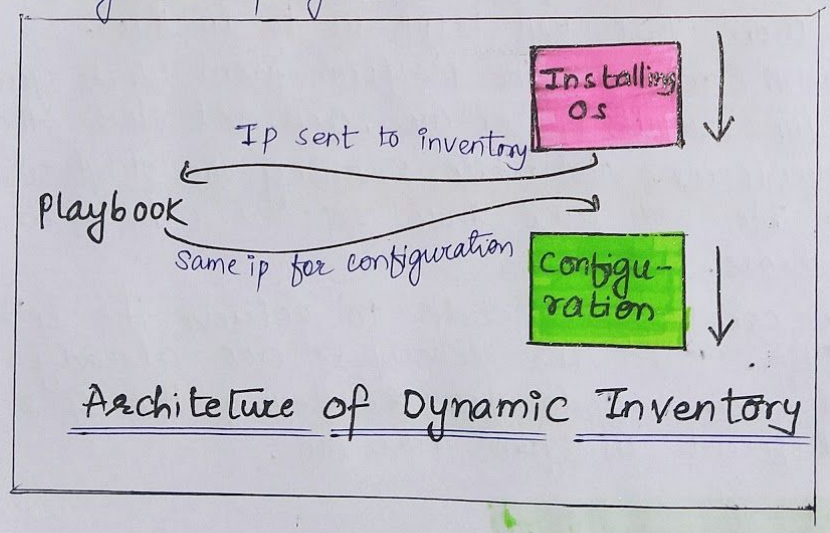
Having a static inventory will never helps us.

Static Inventory:- The inventory in which you needs to put the ip address manually is known as static inventory.

Dynamic Inventory:- The inventory in which the Ip updates automatically / dynamically without human interruptions.

static inventory will not gonna helps you much in complex environment where there you need to install and uninstall the particular OS.

In Dynamic inventory, the Ip will be fetched dynamically from the inventory / hosts file while running the playbook.



Architecture of Dynamic Inventory

If you want to create any inventory then you need to create the file.
But for multiple inventory file you can create a directory. eg. /mydb.

While creating multiple files for multiple hosts the create files with extension of .txt in the directory /mydb.

Ansible supports the file with following extension.

- a ⟶ No extension
- a.yml ⟶ Yaml
- a.json ⟶ Json
- a.py ⟶ python.

In python, we have scanning tools like Nmap. this will helps python to run the code. Same tool We need some scanning, Ip's and ~~take~~ retrieve them into inventories files. We have a boto software from python that will helps us to do this. Boto will goes to the platform from where you wants the ~~output~~ Ip like Aws, Gcp. and stores them into inventories. And while Running the playbook ~~you~~ ansible will take those Ip's as a host and runs them.

you can create your code to retrieve the Ip's from Aws. But the developers are already created some code for retrieving the Ip's from different platform like Aws.

~~for~~ ~~yum~~ pip3 install boto

pip3 install boto3

you Need to install boto and boto3

Steps to achieve Dynamic Inventory.

1.) Install wget (yum install wget)

2.) Install the Dynamic Inventory scripts i.e ec2.py and ec2.ini in /mydb and update /mydb in .cfg.

3.) As you have python3 in your system therefore you need to update the shebang in ec2.py file.

#!/usr/bin/env/python ⟶ #!/usr/bin/python3

4.) Give the region and secret key or access key in ec2.ini file.

```
regions = ap-south-1
aws_access_key_id = ''
aws_secret_access_key =
```

5.) Now you need to export the below command in the terminal itself.

```
export AWS_REGION = 'ap-south-1'
export AWS_ACCESS_KEY = ' '
export AWS_ACCESS_SECRET_KEY = ' '
```

6.) Now run the ansible all --lists-host

7.) If the code is not showing any hosts and if the error at 172 line in ec2.py then go inside ec2.py and comment the code of 172 line.

8.) ansible all --list-hosts. Now it will work

Link for the Dynamic Inventory article. Read each line to achieve better outcome.
ANSIBLE DYNAMIC INVENTORY

Date: 04/01/2021
Day:22

SYSTEMS ROLES

- When you create a Role the Variable have a seperate file. So you can add the variables in the Vars folder.
- The motto of the Roles is to plug and play either you create or download.
- Redhat has its own pre-created Roles. So you can you that system Rdes.

`ansible-galaxy list`
↳ list of all roles in your system.

`yum install rhel-system-roles`
Install this software to get the system roles.

`cd /usr/share/doc/rhel-system-roles/`
↳ this folder will created after downloading rhel-system-roles.

`cd /usr/share/ansible/roles`
↳ this is the default path of system roles.
So you need to update the roles path.
If you don't update the roles path then ansible will take the path of roles automatically.

`cd /usr/share/doc/rhel-system-roles/`
this will provide you document of respective roles.

`kdump` `network` `postfix` `selinux` `timesync`
this all are the precreated docs. of the system roles.

products available to configure your target node as ntp clients are ntp, chronyd software.

Redhat provide the system roles for configuring ntp clients. name of the role is timesync.

Group_Vars & Host_Vars

Previously We ~~are~~ Were doing like creating a variables file and inside vars_files: module. so We need to update if the file name changed and the playbook looks bulky.

So We can customize and manages it.

create a folder group_vars. This will have your group of variables.

Inside group_vars. you can create a seperate space to create all the respective variable in it.

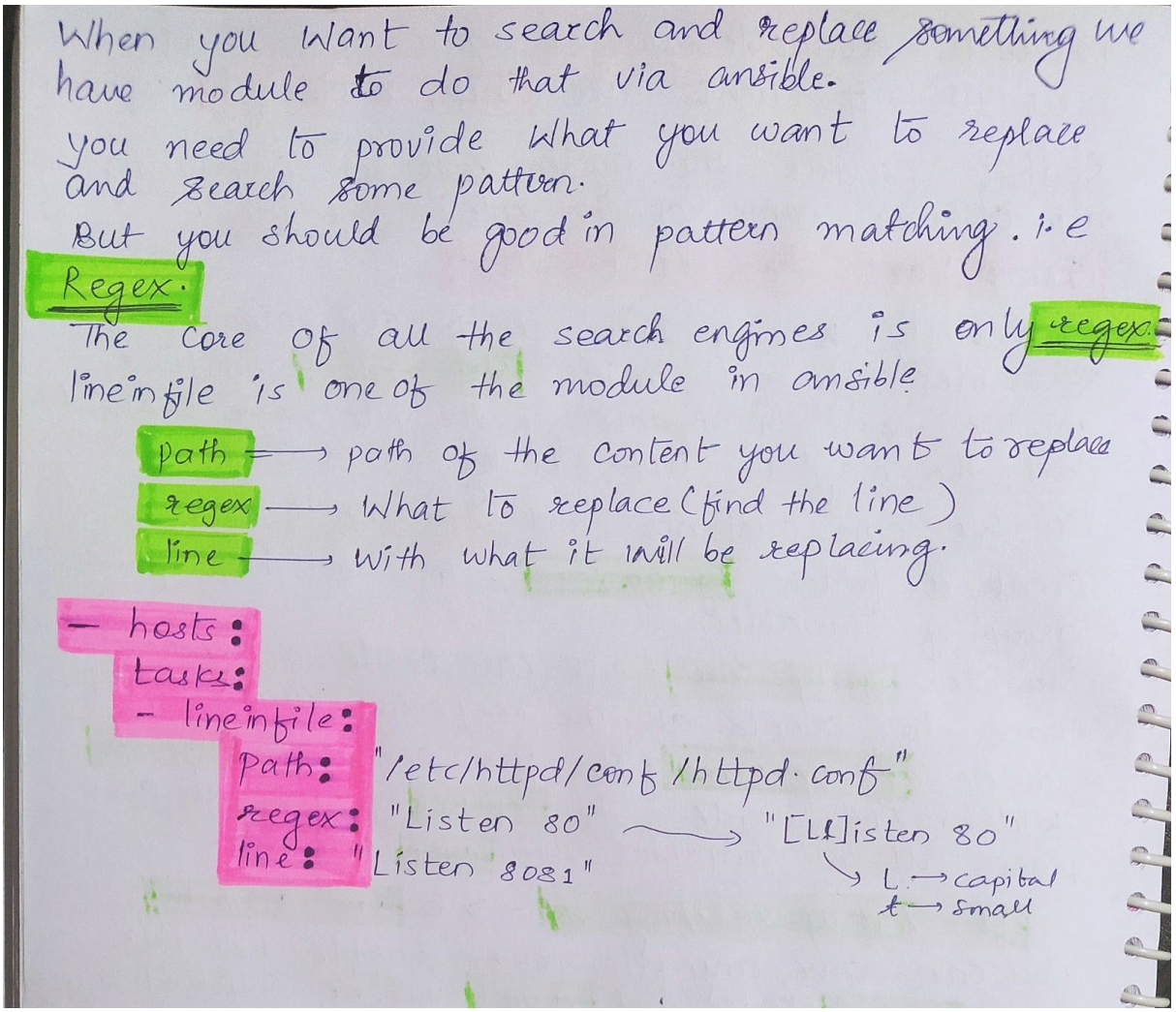
inside group_vars folder you can create myweb folder. And inside the myweb folder you can place all the variables with .yml extension.

cd /group_vars/myweb/ → ~~ls~~ vim a.yml

you can give any file name. Ansible will pick the myweb folder and a.yml will be automatically included.

```
- hosts: myweb
  roles:
    - rhel-system-roles.timesync
```

→ No vars_files it will automatically load the files from myweb.



Thank you all for your support. Hope my notes you had find interesting in Ansible Journey 🔥🔥☺☺

Connect me on LinkedIn 🔥🔥

https://www.linkedin.com/in/amit-sharma-35439016a