---------- Ansible Notes ----------

Ansible –

Ansible is an open-source IT configuration management, orchestration, deployment tool.
Ansible used when we want to install some application on multiple servers then this tool is used.

Here default Ansible Inventory file is present **at /etc/ansible/hosts**
In this path the file path is given

If you want to create your own inventory file, then while giving commands give inventory file name with -i

Ansible-playbook -i inventory. Yml playbookname.yml

Inventory file syntax --

Create groups under these groups provide the servers Ip address
"Collection of hosts is called inventory"

Dynamic Inventory -

For this the scripts are available on ansible docs download that and paste in vi format with same name as
on docs. For this install boto3 application.

[webservers]
wb_1 Ip address
wb_2 Ip address

[sqlservers]
sq_1 Ip address
sq_2 Ip address

[db_web_sql_servers:children]
wb_1
sq_1

Here you can provide the servers means if you want only some of servers on which some tasks to be performed.
Here **children** keyword is important

Idempotency -

When we run same playbook multiple times it will not affect the process as the process is done in previous execution only.
If in that playbook a new task or module is added, then that part is only executed at running time.

To make a password encrypted -

mkpasswd --method=sha-512

By this they will ask for password and then they will give us encrypted format.

Playbook -

A Playbook is a simple yml file which consists of some modules or tasks that we want to perform on servers.

There are some standard modules which are used in yml file.

The Playbook consists of multiple plays and these plays consists multiple modules.

Sample Playbook Syntax -

```
-
    name: This is our third play.
    host: webserver
    task:
        -
            name: "We will create one file"
            command: touch /tmp/c.txt

        -
            name: "We will create one directory"
            command: touch /tmp/playbook3

        -
            name: "We will copy this file to /tmp/playbook3 directory"
            command: cp /tmp/c.txt /tmp/playbook3

-
    name: This is our third play.
    host: dbserver
    task:
        -
            name: "We will create one file"
            command: touch /tmp/d.sh
```

Variables –

Variables are used to store some important data.

Suppose you have to install many applications on same server then there are two methods

1) By mentioning service module many times for each application, it will become more lengthy
2) By using Variables, you can mention only application names and one service module which will fetch all application from Variables.

Example -

```
-
        name: This is our third play.
        host: webserver
        vars:
          serviceName:
                - apache2
                - nginx

        task:
           -
                name: "We will install application"
                service: name= {{ item }}
                with_items: {{ serviceName }}
```

Conditions -

We use condition when we need that some tasks are performed on specific conditions only

Loops -

We use Loops when we need to perform same tasks multiple time then we use loops.

Example -

```
        -
                name: This is our third play.
                host: webserver
                task:
                  -
                        name: "We will install application"
                        service: name= {{ item }} state=started
                        with_items:
                          - httpd
                          - nginx
                          - tomcat9
```

Modules -

1) Command
2) Copy
3) lineinfile
4) script
5) service
6) user
7) Register
8) when
9) apt
10) Debug

Many more modules are there check on Ansible-docs

If you have to use other files which contains variable in yml file use

vars_file:
- variable_1.yml
- variable_2.yml

If you have to use task.yml use

tasks:

include: task.yml

Ansible Role -

By this we will create a playbook but for this playbook all data is separated by files.
Tasks will be in task.yml file variables in variable.yml file
Means the file will be user friendly.

ansible-galaxy init rolename --> This command is used to create Role

Role's structure will be (tree rolename)

- default
- files
- handlers → Will consists of all tasks
- meta →
- tasks → Will consists of all tasks
- templets →
- tests →
- vars → Will consists of all variables

Ansible Asynchronous action –

Async → will try to execute the task in given time even if it does not complete then also it will go to next task
Poll → will check that above task is completed or not in background

Example -

```
-
    name: This play is for variable
    host: webserver
    task:
        -
            name: "Will create some variables"
            command: touch /etc/E.txt
            async: 60
            poll: 35

        -
            name: "Will create some variables"
            command: touch /etc/F.txt
```

Ansible Strategy -

Example -

```
    # This for when 2,3 servers are there

    -
        name: This play is for strategy
        host: webserver, sqlserver
        strategy: linear/free
        task:
            -
                name: "Will install apache2 here"
                apt: name= apache2 state= present"


            -
                name: "Will install tree command here"
                apt: name= tree state= present"
```

Here in webserver if the apache2 is not installed then it will install and then it will go on sqlserver to install same

After this the play will go to next task means first task will execute on both server and then 2 tasks will execute on both

But if we have 1000 server that time it will consume more time to avoid this will use strategy.

in /etc/ansible/ansible_cfg file one variable is there called forke by default its value is 5 means the task will perform on 5 severs then next

In playbook also we use this as above

In linear all tasks will perform on 1 server then on next

In free it will not be dependent on any server directly will install

This is when 1000 servers are there

```
-
    name: This play is for strategy
    host: webserver, sqlserver,1,2,3.....1000
    serial: 1
    # Value will show this task will perform at a time on how many servers means this
    playbook  will execute on 1 task then next server
    task:
        -
            name: "Will install apache2 here"
            apt: name= apache2 state= present"

        -
            name: "Will install tree command here"
            apt: name= tree state= present"
```

Ansible Error Handling -

By this in playbook if one task fails the playbook execution will not stop it will run for next task. Here if we know that this task will fail will add ignore errors = True by this this task will be skipped and next task will be executed.

Example -

```
-
    name: this playbook is for error handling.
    host: webserver, sqlserver
    task:
      -
        name: "This task will fail as we have to create one file inside one directory which
        is not present"
        command: touch /etc/tmp/New_Dir/New.txt
        ignore_errors: True

      -
        name: "This task will perform as we have to create a single file"
```

command: touch /etc/tmp/Run.txt

Here we will be checking that if one task fails the other means 2 task will not perform but if we want to
execute this task use error handling task after that task which will fail...

Ansible Vault -

Here we will encrypt some data/files which is confidential means the files having password id's
The following command will create a vault

ansible-vault encrypt abc.txt --output ABC.txt

Here abc.txt is the file which we want to encrypt, and the file will get as output will be ABC.txt
Now to decrypt these file use

ansible-vault view ABC.txt

To use the encrypted file as inventory use following

ansible-playbook -i ABC.txt playbookname.yml --ask-vault-pass

Ansible Lookup -

This used when the password of ssh connection and others password we have to store or use in our yml file then we use lookup. In company all these password things are given in csv file format so to use lookup

vars:
ansible-ssh-pass: "{{ lookup('csvfile', 'webserver file=filname.csv delimiter=,')}}"

Here

ansible-ssh-pass → variable name
csvfile → file format
webserver → server on which we done
file → file path of csv
delimiter → In this file multiple passwords are there to divide them used

Ansible-Facts –

Facts are nothing but the information about that playbook. To stop this add following in playbook under hosts –

Gather_facts=false

Examples –

Playbook_1.yml

```
-
   name: this is our first play.
   host: webserver
   task:
      -
        name: "Create a dummy file on wb1"
        command: touch /tmp/a.txt
```

Playbook_2.yml

```
-
  name: This is our second file.
  host: webserver
  task:
    -
      name: "Creating a directory"
      command: mkdir -p /tmp/new_dir

    -
      name: "Creating a file inside new directory"
      command: touch /tmp/new_dir/b.txt
```

Playbook_3.yml

```
-
  name: This is our third play.
  host: webserver
  task:
    -
      name: "We will create one file"
      command: touch /tmp/c.txt

    -
      name: "We will create one directory"
      command: touch /tmp/playbook3
```

```
    -
      name: "We will copy this file to /tmp/playbook3 directory"
      command: cp /tmp/c.txt /tmp/playbook3

-
  name: This is our third play.
  host: dbserver
  task:
    -
      name: "We will create one file"
      command: touch /tmp/d.sh
```

Playbook_4.yml

```
-
  name: This is our fourth play.
  host: webserver
  task:
    -
      name: "We will write one line in existing file"
      lineinfile: path= /tmp/c.txt line= " Hi Atul Misal"
```

Playbook_5.yml

```
-
  name: Service Module.
  host: webserver
  task:
    -
      name: "We will install one package"
      Service:
        name: negenix
        state: started
```

Strategies. Yml

```
# This for when 2,3 servers are there
-
  name: This play is for strategy
  host: webserver, sqlserver
  strategy: linear/free
  task:
   -
      name: "Will install apache2 here"
      apt: name= apache2 state= present"


   -
      name: "Will install tree command here"
      apt: name= tree state= present"


   # Here in webserver if the apache2 is not installed then it will install and
then it will go on sqlserver to install same
   # After this the play will go to next task means first task will execute on
both server and then 2 task will execute on both
   # But if we have 1000 server that time it will consume more time to avoid this
will use strategy.
   # in /etc/ansible/ansible_cfg file one variable is there called forke by
default its value is 5 means the task will perform on 5 severs then next
   # In playbook also we use this as above

   # In linear all tasks will perform on 1 server then on next
   # In free it will not be dependent on any server directly will install
#################################################################################
# This is when 1000 servers are there
-
  name: This play is for strategy
  host: webserver, sqlserver,1,2,3.........1000
  serial: 1    # Value will show this task will perform at a time on how many
servers means this playbook will execute on 1 task then next server
  task:
   -
      name: "Will install apache2 here"
      apt: name= apache2 state= present"

   -
      name: "Will install tree command here"
      apt: name= tree state= present"
```

Asynchronous. Yml

```
-
  name: This play is for variable
  host: webserver
  task:
   -
      name: "Will create some variables"
      command: touch /etc/E.txt
      async: 60
      poll: 35

   -
      name: "Will create some variables"
      command: touch /etc/F.txt

# async will try to excute the task in given time even if it not completes then
also it will go to next task

# poll will check that above task is completed or not in background
```

Condition_1.yml

```
-
  name: This play is for variable
  host: webserver
  vars:
    age = 18
  task:
   -
      name: "Will create some variables"
      command: touch /etc/E.txt
      when: age == 18
```

Condition_2.yml

```
-
  name: This play is for Condition
  host: webserver
  vars:
    age = 18
  task:
```

```
   -
      name: "Will create some variables"
      command: touch /etc/F.txt
      when: age == 18 and age == 19
```

Error_handling.yml

```
-
  name: this playbook is for error handling.
  host: webserver,sqlserver
  task:
    -
      name: "This task will fail as we have to create one file inside one
directory which is not present"
      command: touch /etc/tmp/New_Dir/New.txt
      ignore_errors: True

    -
      name: "This task will perform as we have to create a single file"
      command: touch /etc/tmp/Run.txt


#Here we will checking that if one task fails the other means 2 task will not
perform but if we want to
#excute this task use error handling task after that task which will
fail...
```

Include. Yml

```
-
  name: This play is for Include module.
  host: webserver
  vars_files:
    - vars.yml
  task:
    - include: task.yml
```

Vars.yml

```yaml
var1: var1
var2: var2
var3: var3
var4: var4
var5: var5
var6: var6
```

Task.yml

```yaml
-
  name: "Task 1"
  command: touch /etc/tmp/{{ var1 }}.txt

-
  name: "Task 2"
  command: touch /etc/tmp/{{ var2 }}.txt

-
  name: "Task 3"
  command: touch /etc/tmp/{{ var3 }}.txt

-
  name: "Task 4"
  command: touch /etc/tmp/{{ var4 }}.txt

-
  name: "Task 5"
  command: touch /etc/tmp/{{ var5 }}.txt
```

Variable_1.yml

```yaml
-
  name: This play is for variable
  host: webserver
  vars:
    ServicName: apache2
  task:
   -
      name: "Will create some variables"
      Service:
        name: {{ ServicName }}
        state: started
```