AKSHIT
10/22/23

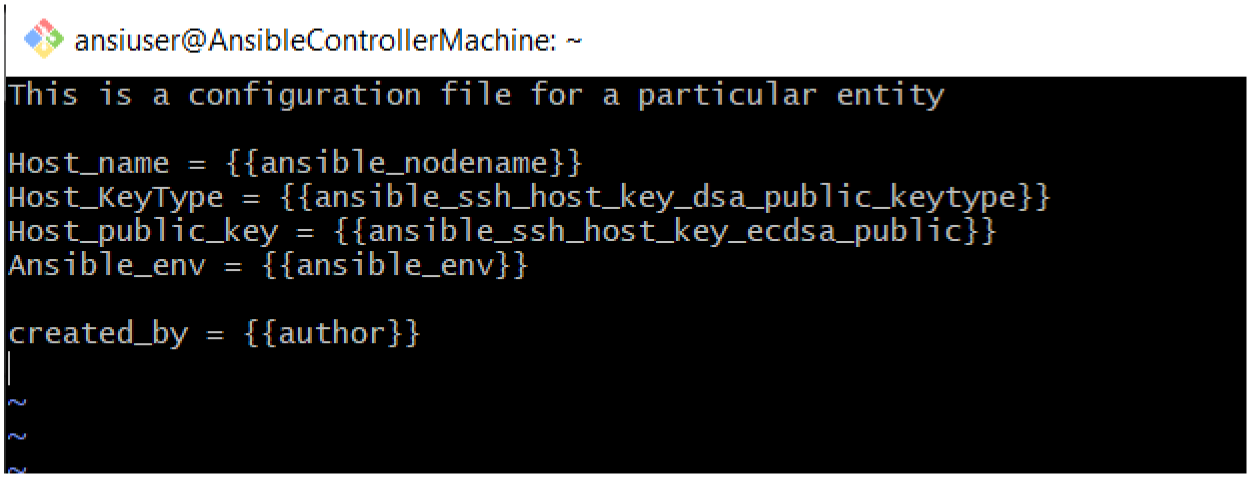
Jinja2 Template, for-each loop, Ansible Roles, Dynamic Inventory, Ansible Vault

Links:-
Dynamic Inventory EC2 Plugin:
[amazon.aws.aws_ec2 inventory – EC2 inventory source — Ansible Documentation]

As we know, that the fact variable contains the information about the host servers. Their name is generic and are fixed globally. Their value changes and modifies based on the configuration of the host servers. We can define a generic configuration based on those fact variables and store it in the jinja2 template. So that it can be deployed on all the host servers at once.

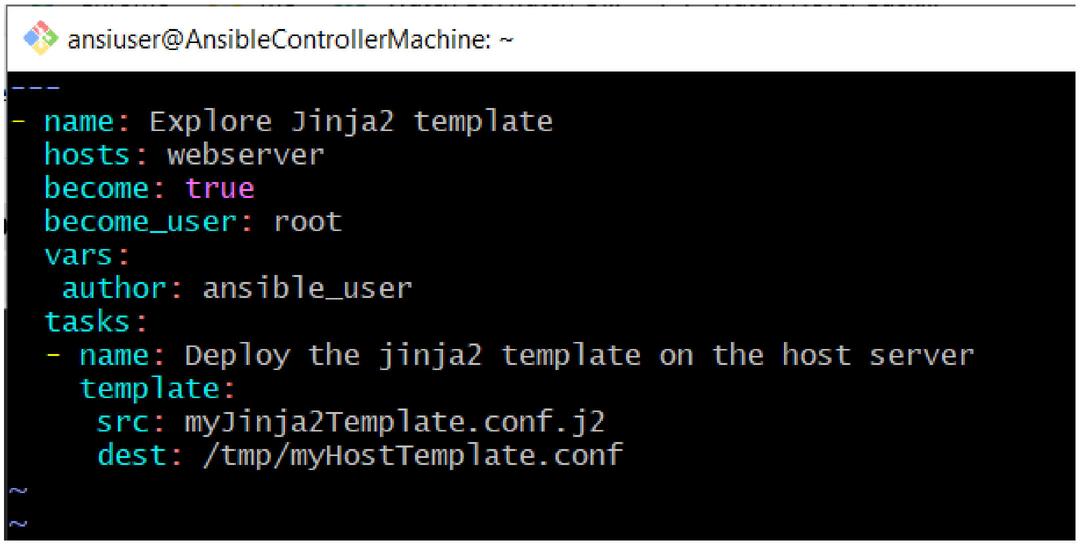
1. Jinja2 template- It is the file we create with .j2 extension. We can define any fact variables in this Jinja2 template assigning each fact variable a custom variable name, according to our preferences. Thereafter, we can deploy that jinja2 file in the host servers through template module in the playbook.
   After that, executing a playbook will deploy the template on the host machines. We can check it by seeing the worker node config details from the host server in our controller machine console window.

➔ This is a jinja2 template file where we have added the few of fact variables, who's values will be different for every host server.
  Here, we have given each fact variable its unique custom variable which will be assigned a value in the host servers by these fact variables.
  The {{author}} is locally defined variable which we will define in the playbook. It is not the fact variable.


ansiuser@AnsibleControllerMachine: ~

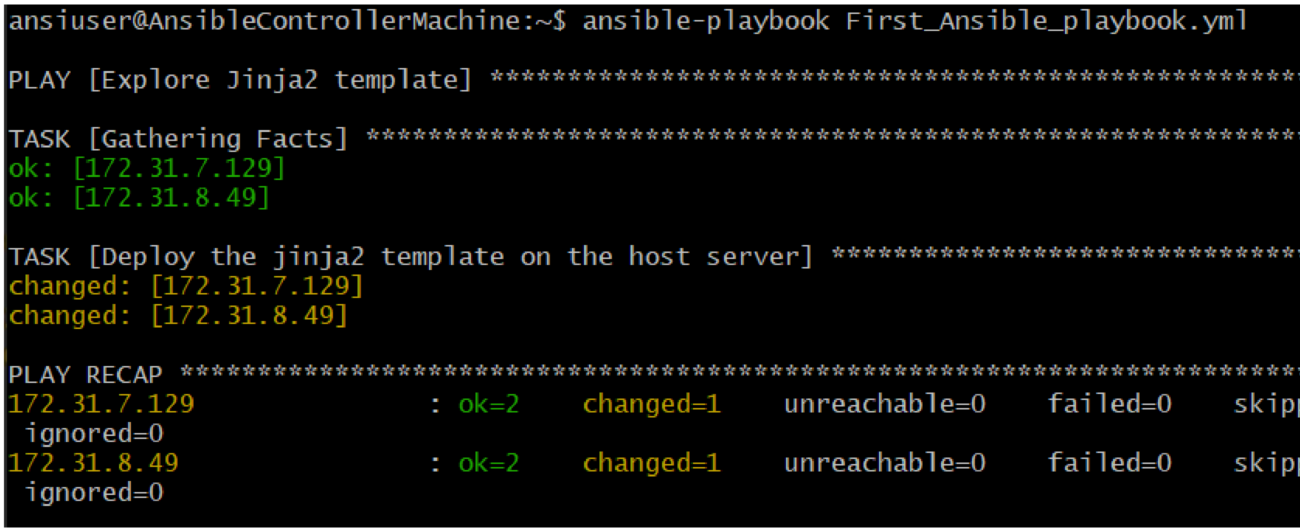
```
This is a configuration file for a particular entity

Host_name = {{ansible_nodename}}
Host_KeyType = {{ansible_ssh_host_key_dsa_public_keytype}}
Host_public_key = {{ansible_ssh_host_key_ecdsa_public}}
Ansible_env = {{ansible_env}}

created_by = {{author}}

~
~
```

➔ In the playbook, we have defined the author variable. We used the template module, in which we defined the source of the jinja2 file and the destination of the host server where we want this file to be deployed.
Please note*
→ While giving the destination path, we give the file name in which we want our template to be deployed
→ While giving the destination path, we don't use the .j2 extension in the file.


ansiuser@AnsibleControllerMachine: ~

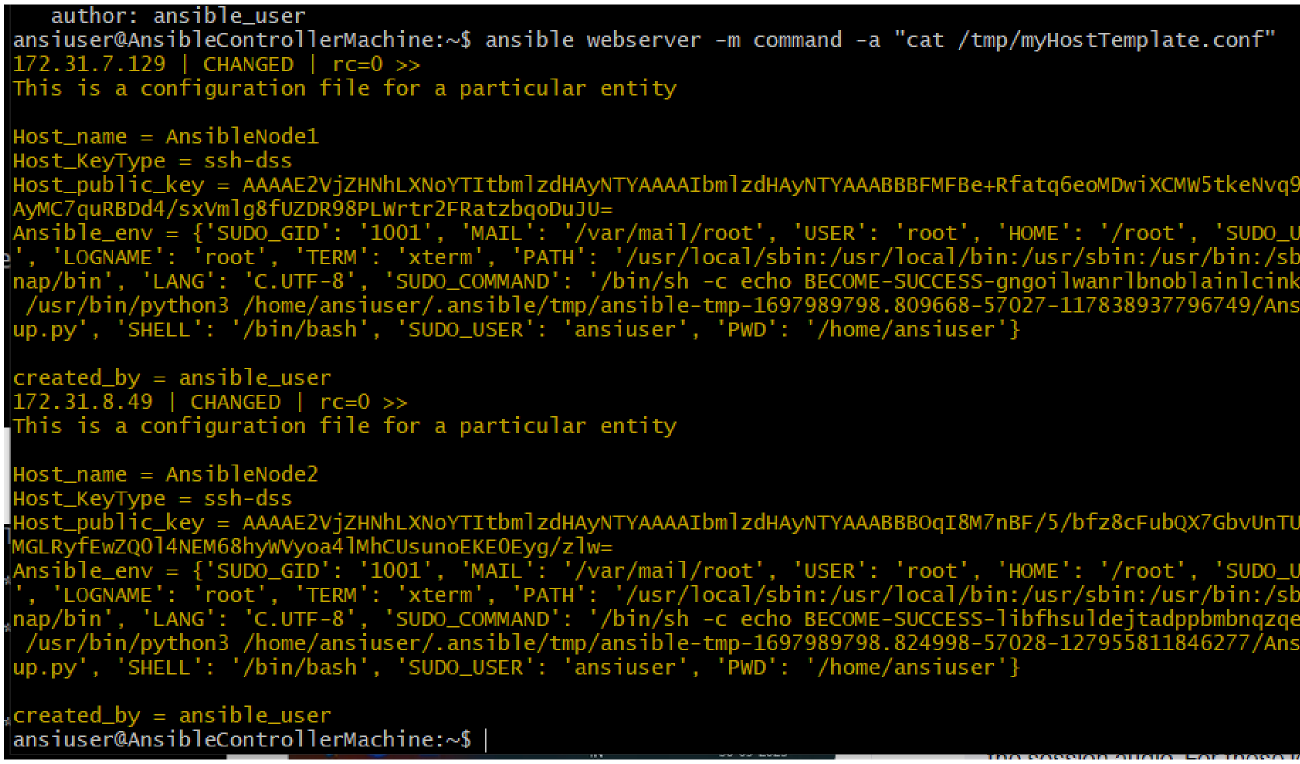
```
---
- name: Explore Jinja2 template
  hosts: webserver
  become: true
  become_user: root
  vars:
   author: ansible_user
  tasks:
  - name: Deploy the jinja2 template on the host server
    template:
     src: myJinja2Template.conf.j2
     dest: /tmp/myHostTemplate.conf
~
~
```

→ The jinja2 template has been successfully deployed



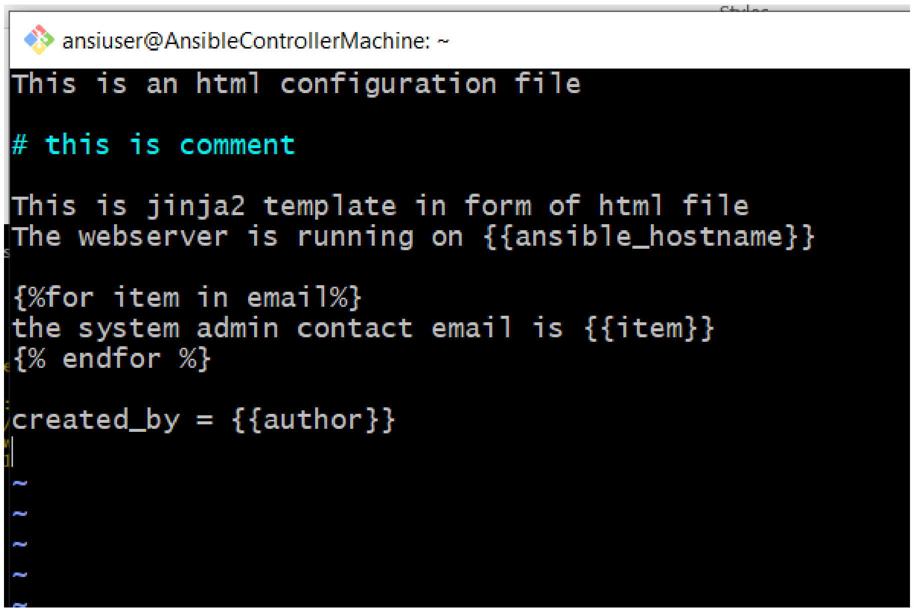
```
ansiuser@AnsibleControllerMachine:~$ ansible-playbook First_Ansible_playbook.yml

PLAY [Explore Jinja2 template] ****************************************************

TASK [Gathering Facts] ************************************************************
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [Deploy the jinja2 template on the host server] ******************************
changed: [172.31.7.129]
changed: [172.31.8.49]

PLAY RECAP ************************************************************************
172.31.7.129               : ok=2    changed=1    unreachable=0    failed=0    skipp
 ignored=0
172.31.8.49                : ok=2    changed=1    unreachable=0    failed=0    skipp
 ignored=0
```

➔ Using the ansible ad-hoc command to show the output of the node servers jinja2 file from the controller machine.
We can see that the every variable we assigned for every particular fact variable, has its value assigned according to the configuration of the host server



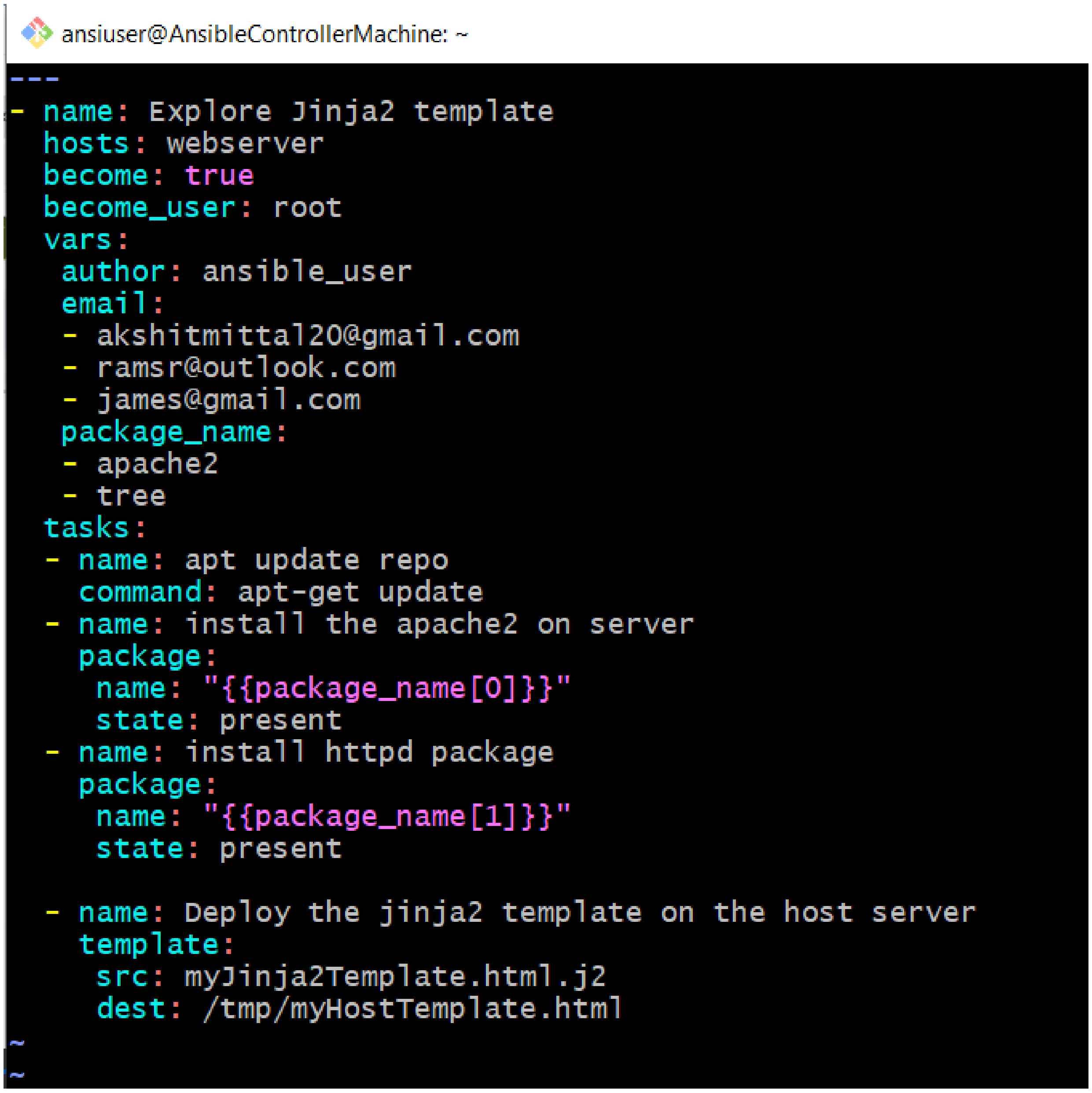
```
   author: ansible_user
ansiuser@AnsibleControllerMachine:~$ ansible webserver -m command -a "cat /tmp/myHostTemplate.conf"
172.31.7.129 | CHANGED | rc=0 >>
This is a configuration file for a particular entity

Host_name = AnsibleNode1
Host_KeyType = ssh-dss
Host_public_key = AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFMFBe+Rfatq6eoMDwiXCMW5tkeNvq9
AyMC7quRBDd4/sxVmlg8fUZDR98PLWrtr2FRatzbqoDuJU=
Ansible_env = {'SUDO_GID': '1001', 'MAIL': '/var/mail/root', 'USER': 'root', 'HOME': '/root', 'SUDO_U
', 'LOGNAME': 'root', 'TERM': 'xterm', 'PATH': '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sb
nap/bin', 'LANG': 'C.UTF-8', 'SUDO_COMMAND': '/bin/sh -c echo BECOME-SUCCESS-gngoilwanrlbnoblainlcink
 /usr/bin/python3 /home/ansiuser/.ansible/tmp/ansible-tmp-1697989798.809668-57027-117838937796749/Ans
up.py', 'SHELL': '/bin/bash', 'SUDO_USER': 'ansiuser', 'PWD': '/home/ansiuser'}

created_by = ansible_user
172.31.8.49 | CHANGED | rc=0 >>
This is a configuration file for a particular entity

Host_name = AnsibleNode2
Host_KeyType = ssh-dss
Host_public_key = AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOqI8M7nBF/5/bfz8cFubQX7GbvUnTU
MGLRyfEwZQ0l4NEM68hyWVyoa4lMhCUsunoEKE0Eyg/zlw=
Ansible_env = {'SUDO_GID': '1001', 'MAIL': '/var/mail/root', 'USER': 'root', 'HOME': '/root', 'SUDO_U
', 'LOGNAME': 'root', 'TERM': 'xterm', 'PATH': '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sb
nap/bin', 'LANG': 'C.UTF-8', 'SUDO_COMMAND': '/bin/sh -c echo BECOME-SUCCESS-libfhsuldejtadppbmbnqzqe
 /usr/bin/python3 /home/ansiuser/.ansible/tmp/ansible-tmp-1697989798.824998-57028-127955811846277/Ans
up.py', 'SHELL': '/bin/bash', 'SUDO_USER': 'ansiuser', 'PWD': '/home/ansiuser'}

created_by = ansible_user
ansiuser@AnsibleControllerMachine:~$ |
```

➔ Using for loop in jinja2 template file.
→ Creating another jinja2 template file with .html.j2 extension. For loop starts and ends with the same syntax and item is the fixed argument for any loop and email is the loop list defined in the variable.



```
This is an html configuration file

# this is comment

This is jinja2 template in form of html file
The webserver is running on {{ansible_hostname}}

{%for item in email%}
the system admin contact email is {{item}}
{% endfor %}

created_by = {{author}}

~
~
~
~
~
```

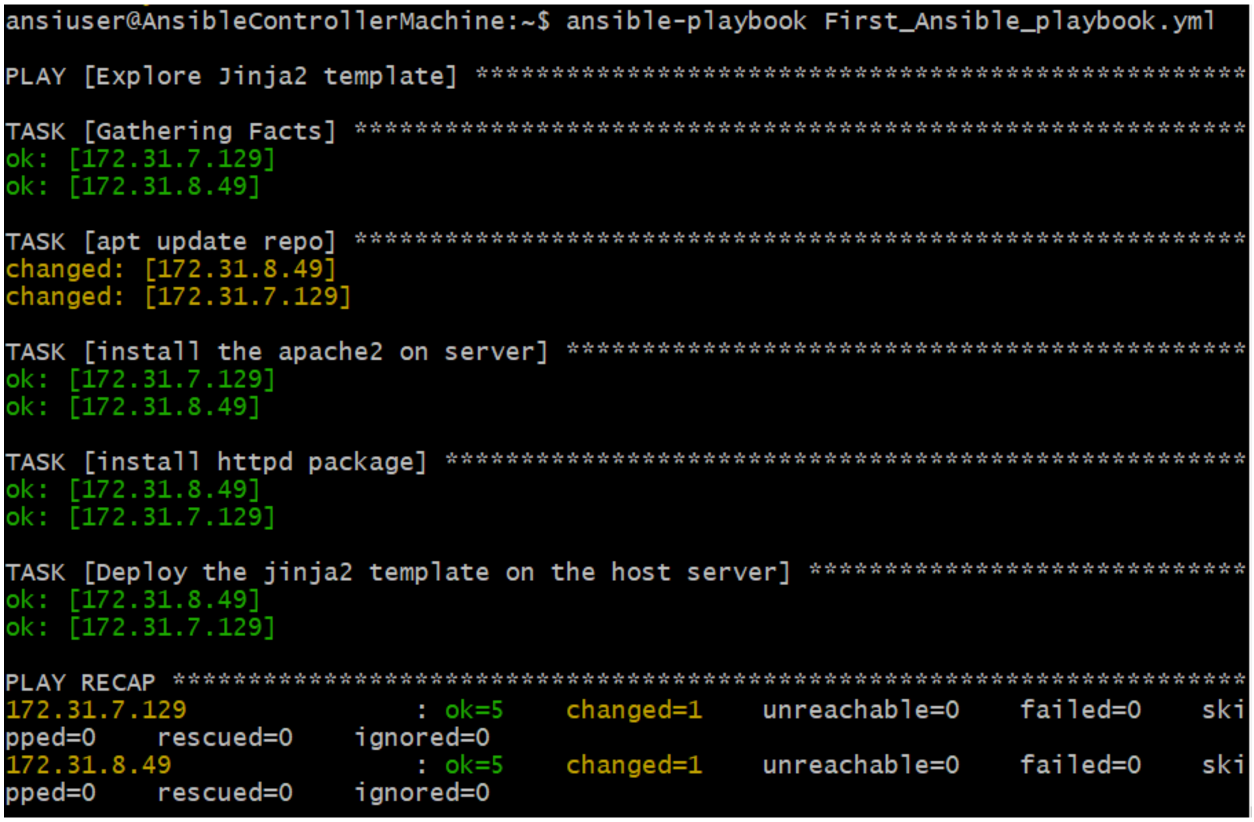
Variable with index number can also be called like this- Package name[index number]



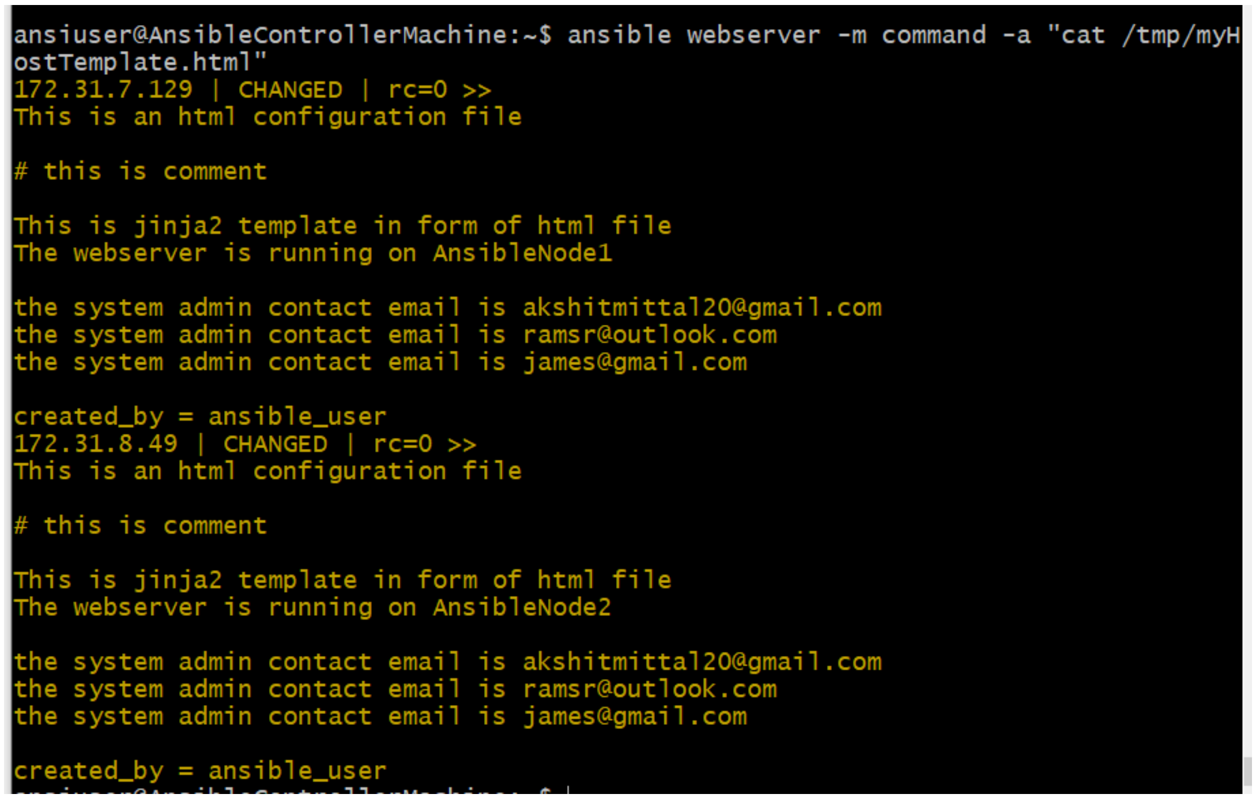
ansiuser@AnsibleControllerMachine: ~

```
---
- name: Explore Jinja2 template
  hosts: webserver
  become: true
  become_user: root
  vars:
   author: ansible_user
   email:
   - akshitmittal20@gmail.com
   - ramsr@outlook.com
   - james@gmail.com
   package_name:
   - apache2
   - tree
  tasks:
  - name: apt update repo
    command: apt-get update
  - name: install the apache2 on server
    package:
     name: "{{package_name[0]}}"
     state: present
  - name: install httpd package
    package:
     name: "{{package_name[1]}}"
     state: present

  - name: Deploy the jinja2 template on the host server
    template:
     src: myJinja2Template.html.j2
     dest: /tmp/myHostTemplate.html
~
~
```

The playbook will be executing successfully and the jinja2 template file will be visible in the host servers.



```
ansiuser@AnsibleControllerMachine:~$ ansible-playbook First_Ansible_playbook.yml

PLAY [Explore Jinja2 template] ************************************************

TASK [Gathering Facts] ********************************************************
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [apt update repo] ********************************************************
changed: [172.31.8.49]
changed: [172.31.7.129]

TASK [install the apache2 on server] ******************************************
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [install httpd package] **************************************************
ok: [172.31.8.49]
ok: [172.31.7.129]

TASK [Deploy the jinja2 template on the host server] **************************
ok: [172.31.8.49]
ok: [172.31.7.129]

PLAY RECAP ********************************************************************
172.31.7.129               : ok=5    changed=1    unreachable=0    failed=0    ski
pped=0    rescued=0    ignored=0
172.31.8.49                : ok=5    changed=1    unreachable=0    failed=0    ski
pped=0    rescued=0    ignored=0
```



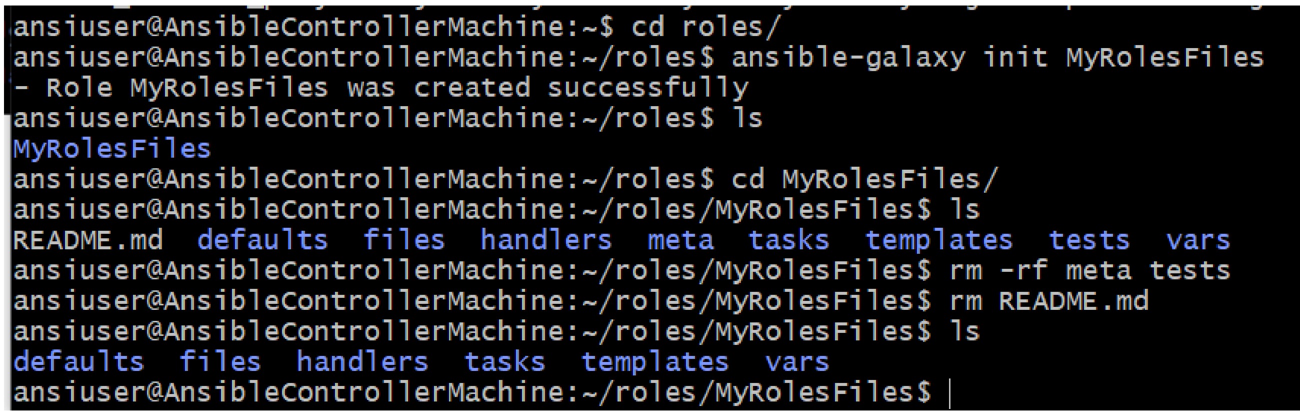
```
ansiuser@AnsibleControllerMachine:~$ ansible webserver -m command -a "cat /tmp/myH
ostTemplate.html"
172.31.7.129 | CHANGED | rc=0 >>
This is an html configuration file

# this is comment

This is jinja2 template in form of html file
The webserver is running on AnsibleNode1

the system admin contact email is akshitmittal20@gmail.com
the system admin contact email is ramsr@outlook.com
the system admin contact email is james@gmail.com

created_by = ansible_user
172.31.8.49 | CHANGED | rc=0 >>
This is an html configuration file

# this is comment

This is jinja2 template in form of html file
The webserver is running on AnsibleNode2

the system admin contact email is akshitmittal20@gmail.com
the system admin contact email is ramsr@outlook.com
the system admin contact email is james@gmail.com

created_by = ansible_user
```

Please note*
The apt module with update-cache argument
and the command module with apt-get update argument
Does the same tasks.

2. Roles

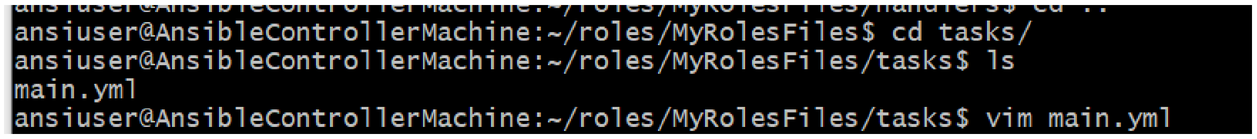
➔ We define various sections of playbook like handlers, tasks, templates, variables and so on, in the separate files into the roles directory and call the roles directory into our main playbook outside that directory.
Ansible provides the initialization of roles using ansible-galaxy. What it does is create the files automatically into the role directory and we just need to fill the information in them. And those files will automatically be recognized by the playbook when executed.
Lets see how it works
➔ Make directory named Roles in the ansiuser directory where you are going to define the playbook

➔ Execute #ansible-galaxy init apache - This command in the roles directory. It will automatically create the different files for each section of the playbook. Delete the files which you don't need. And start filling the data in each file

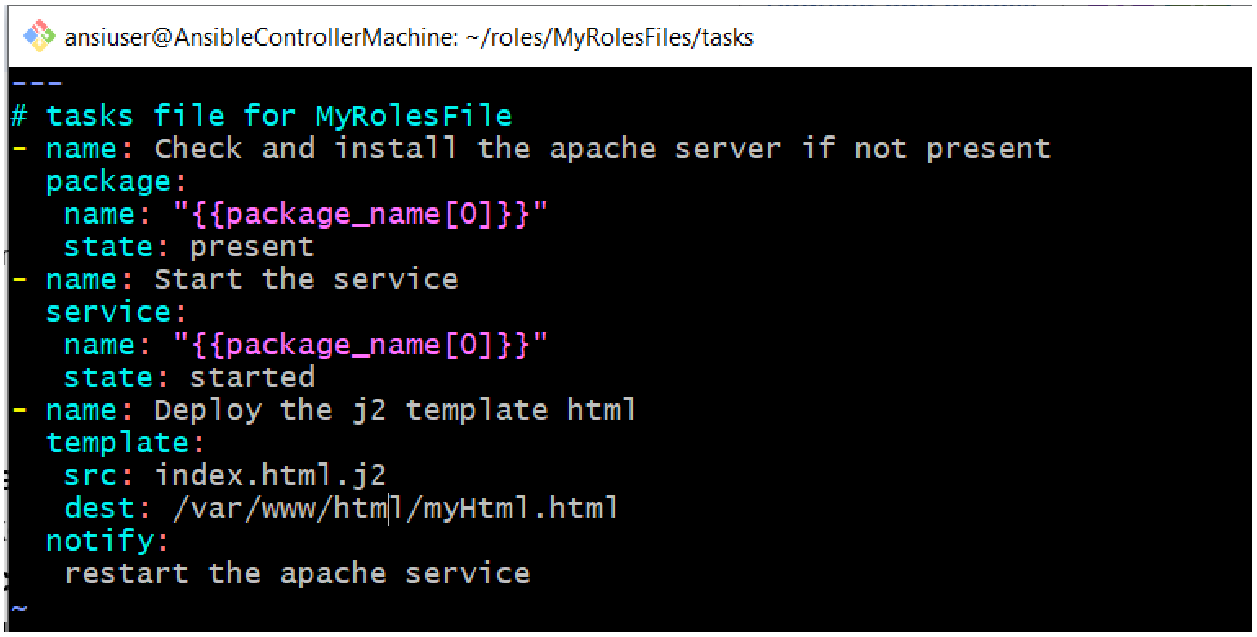


```
ansiuser@AnsibleControllerMachine:~$ cd roles/
ansiuser@AnsibleControllerMachine:~/roles$ ansible-galaxy init MyRolesFiles
- Role MyRolesFiles was created successfully
ansiuser@AnsibleControllerMachine:~/roles$ ls
MyRolesFiles
ansiuser@AnsibleControllerMachine:~/roles$ cd MyRolesFiles/
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ ls
README.md  defaults  files  handlers  meta  tasks  templates  tests  vars
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ rm -rf meta tests
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ rm README.md
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ ls
defaults  files  handlers  tasks  templates  vars
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$
```

➔ Create a task in the task directory in main.yml file



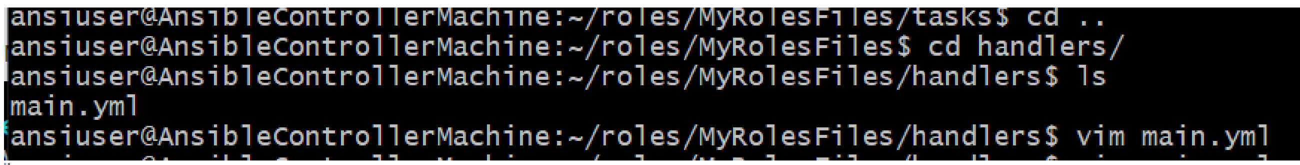
```
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ cd tasks/
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/tasks$ ls
main.yml
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/tasks$ vim main.yml
```



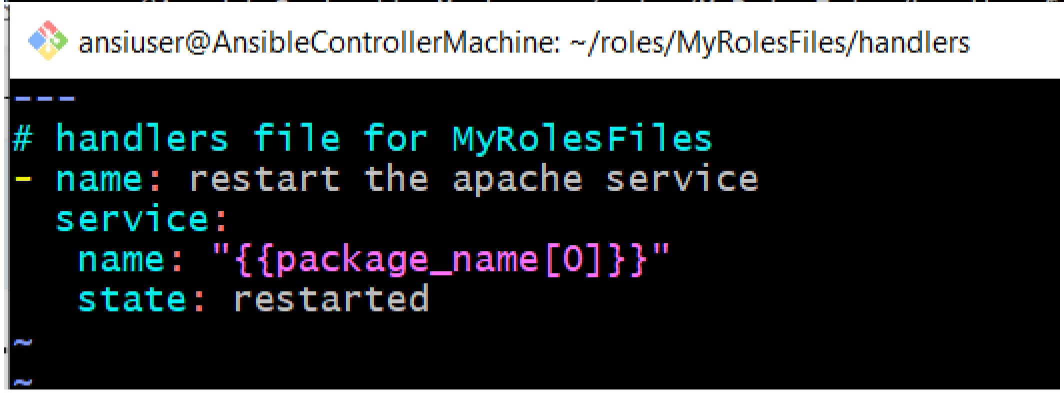
ansiuser@AnsibleControllerMachine: ~/roles/MyRolesFiles/tasks

```
---
# tasks file for MyRolesFile
- name: Check and install the apache server if not present
  package:
   name: "{{package_name[0]}}"
   state: present
- name: Start the service
  service:
   name: "{{package_name[0]}}"
   state: started
- name: Deploy the j2 template html
  template:
   src: index.html.j2
   dest: /var/www/html/myHtml.html
  notify:
   restart the apache service
~
```

➔ Let's fill the handlers section



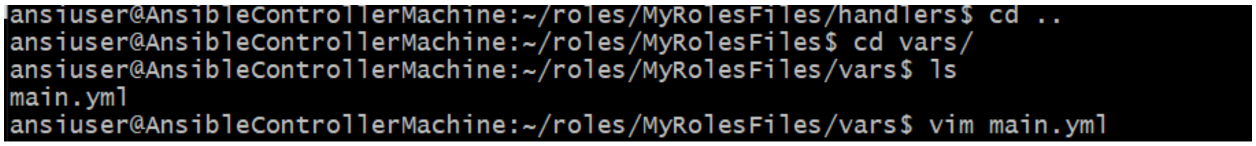
```
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/tasks$ cd ..
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ cd handlers/
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/handlers$ ls
main.yml
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/handlers$ vim main.yml
```



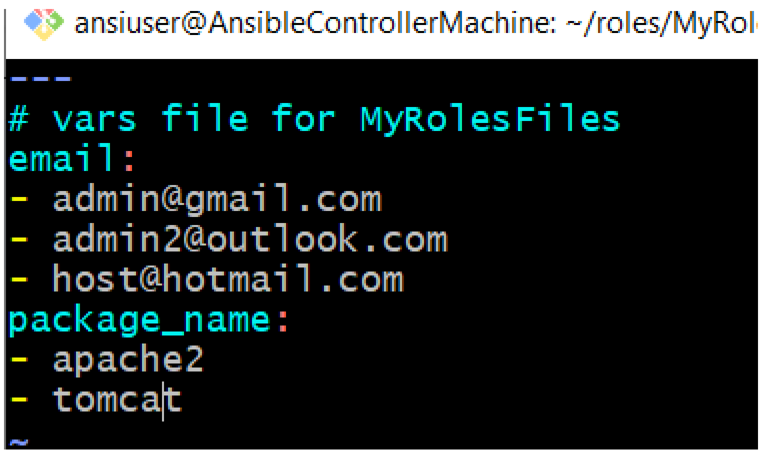
ansiuser@AnsibleControllerMachine: ~/roles/MyRolesFiles/handlers

```
---
# handlers file for MyRolesFiles
- name: restart the apache service
  service:
   name: "{{package_name[0]}}"
   state: restarted
~
~
```

➔ Lets fill the Variable files now



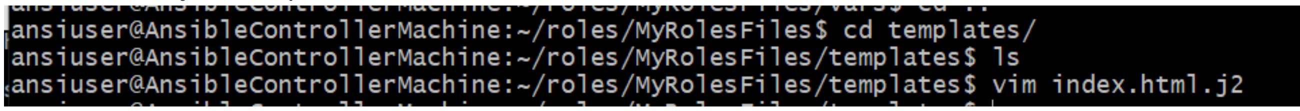
```
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/handlers$ cd ..
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ cd vars/
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/vars$ ls
main.yml
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/vars$ vim main.yml
```



ansiuser@AnsibleControllerMachine: ~/roles/MyRol

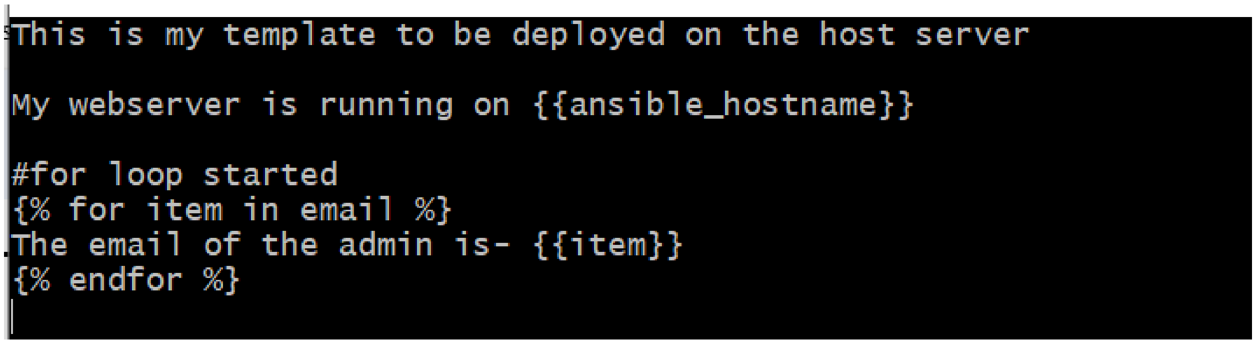
```
---
# vars file for MyRolesFiles
email:
- admin@gmail.com
- admin2@outlook.com
- host@hotmail.com
package_name:
- apache2
- tomcat
~
```

➔ Let's write Jinja 2 template



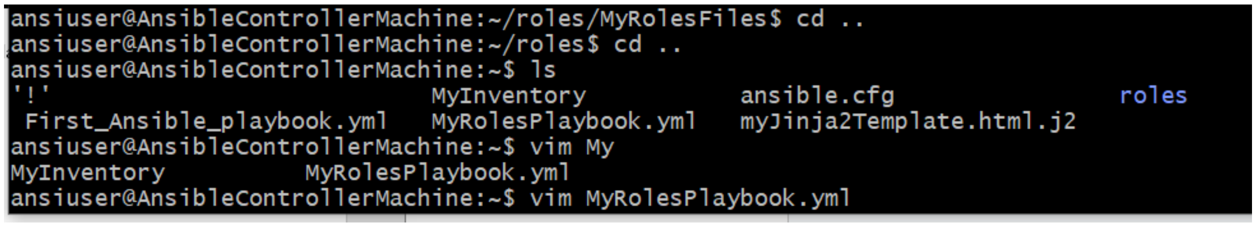
```
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ cd templates/
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/templates$ ls
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles/templates$ vim index.html.j2
```

Here, email is the list of variables above

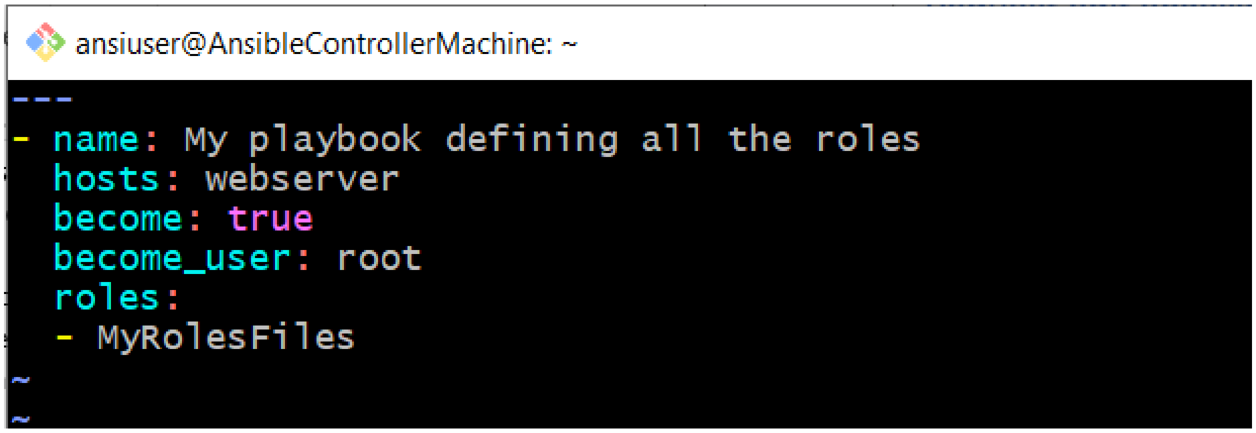


```
This is my template to be deployed on the host server

My webserver is running on {{ansible_hostname}}

#for loop started
{% for item in email %}
The email of the admin is- {{item}}
{% endfor %}
```

➔ Now come out of this roles directory and write the roles playbook

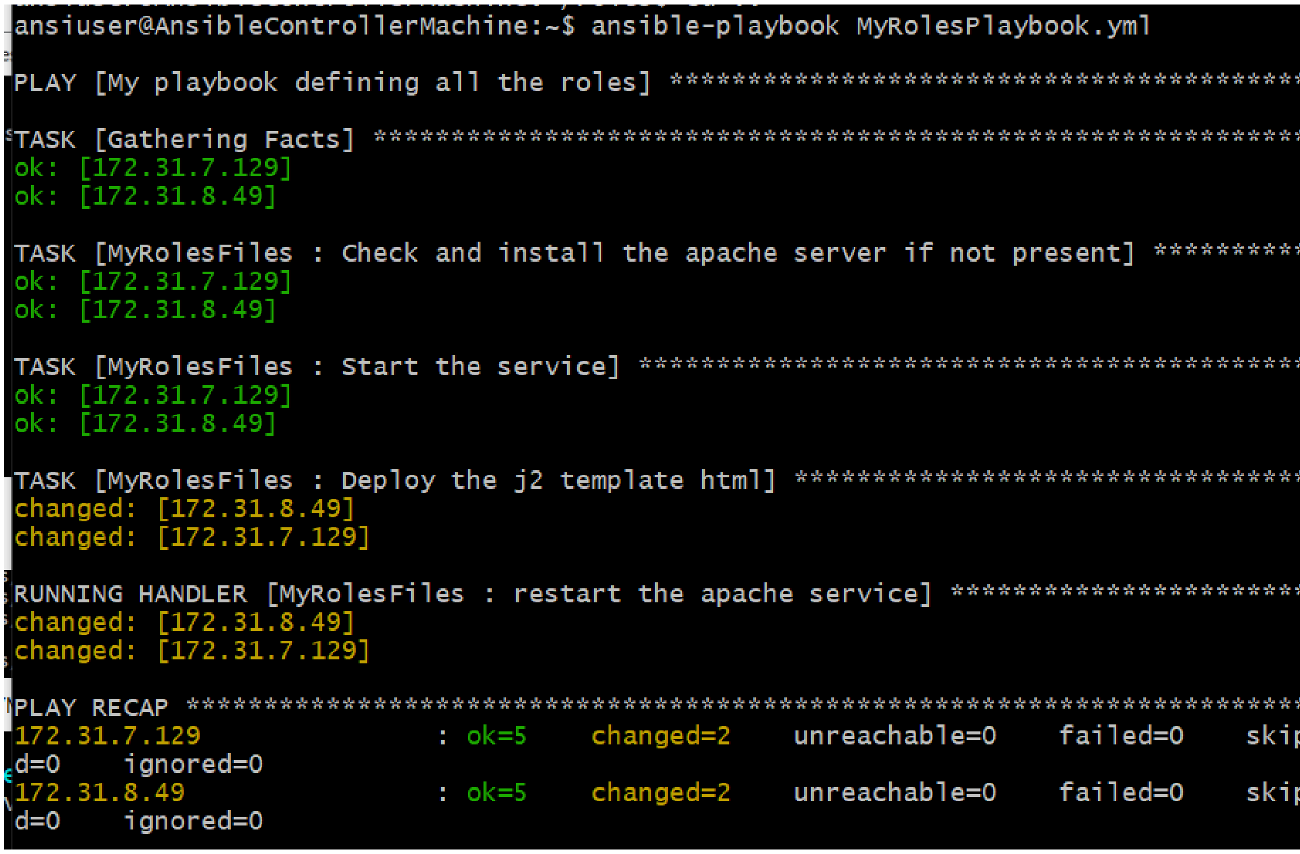


```
ansiuser@AnsibleControllerMachine:~/roles/MyRolesFiles$ cd ..
ansiuser@AnsibleControllerMachine:~/roles$ cd ..
ansiuser@AnsibleControllerMachine:~$ ls
'!'                             MyInventory          ansible.cfg                  roles
 First_Ansible_playbook.yml    MyRolesPlaybook.yml  myJinja2Template.html.j2
ansiuser@AnsibleControllerMachine:~$ vim My
MyInventory           MyRolesPlaybook.yml
ansiuser@AnsibleControllerMachine:~$ vim MyRolesPlaybook.yml
```



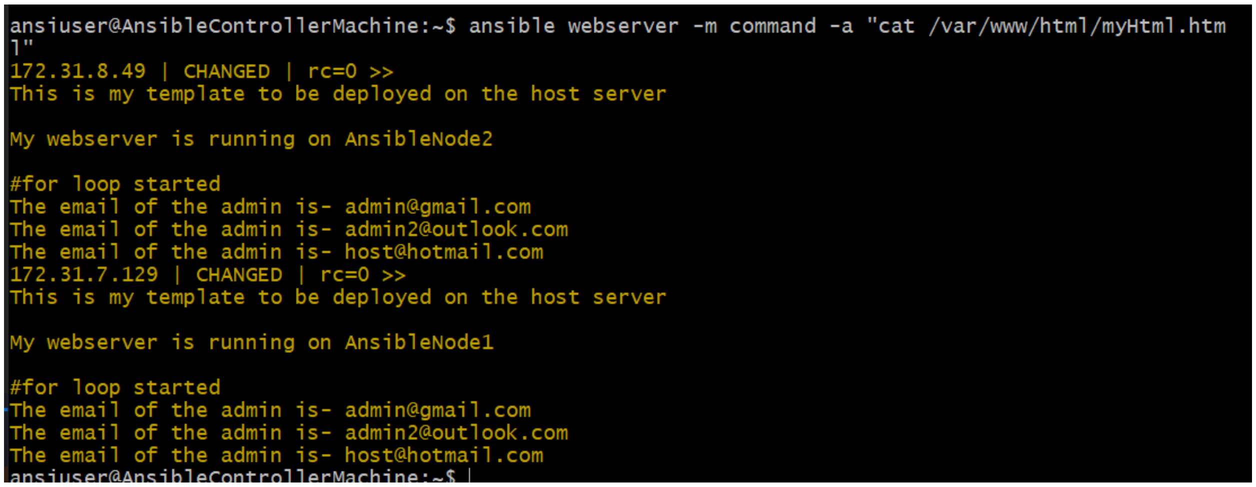
ansiuser@AnsibleControllerMachine: ~

```
---
- name: My playbook defining all the roles
  hosts: webserver
  become: true
  become_user: root
  roles:
  - MyRolesFiles
~
~
```



```
ansiuser@AnsibleControllerMachine:~$ ansible-playbook MyRolesPlaybook.yml

PLAY [My playbook defining all the roles] ******************************************

TASK [Gathering Facts] ************************************************************
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [MyRolesFiles : Check and install the apache server if not present] **********
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [MyRolesFiles : Start the service] *******************************************
ok: [172.31.7.129]
ok: [172.31.8.49]

TASK [MyRolesFiles : Deploy the j2 template html] *********************************
changed: [172.31.8.49]
changed: [172.31.7.129]

RUNNING HANDLER [MyRolesFiles : restart the apache service] ***********************
changed: [172.31.8.49]
changed: [172.31.7.129]

PLAY RECAP ************************************************************************
172.31.7.129               : ok=5    changed=2    unreachable=0    failed=0    skip
d=0    ignored=0
172.31.8.49                : ok=5    changed=2    unreachable=0    failed=0    skip
d=0    ignored=0
```

➔ We can see that the template was successfully deployed on both the servers and it is visible on host servers.

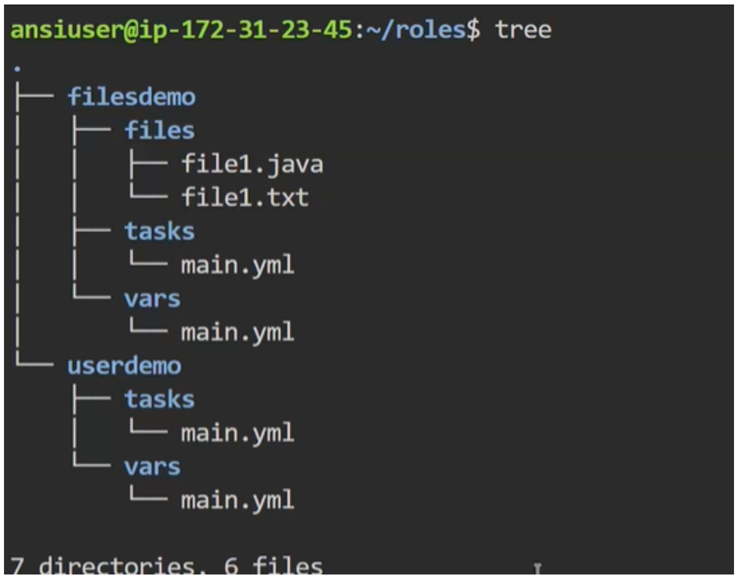


```
ansiuser@AnsibleControllerMachine:~$ ansible webserver -m command -a "cat /var/www/html/myHtml.htm
l"
172.31.8.49 | CHANGED | rc=0 >>
This is my template to be deployed on the host server

My webserver is running on AnsibleNode2

#for loop started
The email of the admin is- admin@gmail.com
The email of the admin is- admin2@outlook.com
The email of the admin is- host@hotmail.com
172.31.7.129 | CHANGED | rc=0 >>
This is my template to be deployed on the host server

My webserver is running on AnsibleNode1

#for loop started
The email of the admin is- admin@gmail.com
The email of the admin is- admin2@outlook.com
The email of the admin is- host@hotmail.com
ansiuser@AnsibleControllerMachine:~$
```

*please note-
If the package is not being able to install even after updating the apt repository, that

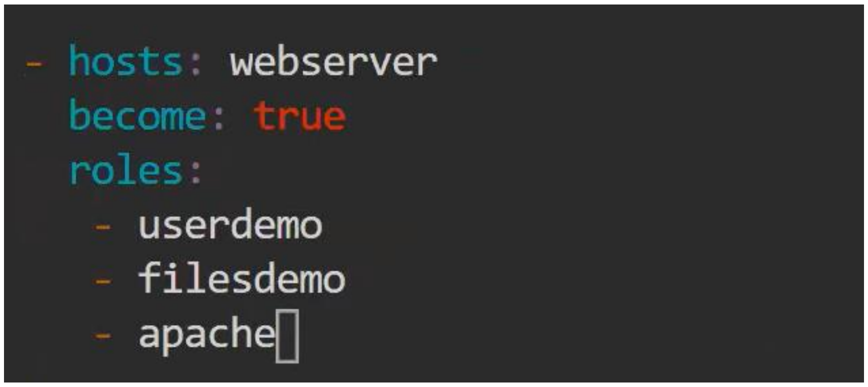
means it is needed to be downloaded on the repository first. There we use yum_repository module.
Also,
We can define multiple tasks file in the task directory in the roles and call it in the main task file.
You can refer to this Github repo url to be clear with the concept
https://github.com/akshitmittal20/AnsibleRolesDemo.git

- You can install tree package to visualize your file structure



```
ansiuser@ip-172-31-23-45:~/roles$ tree
.
├── filesdemo
│   ├── files
│   │   ├── file1.java
│   │   └── file1.txt
│   ├── tasks
│   │   └── main.yml
│   └── vars
│       └── main.yml
└── userdemo
    ├── tasks
    │   └── main.yml
    └── vars
        └── main.yml

7 directories, 6 files
```

- You can create Multiple roles files in the directory and define all the roles in the main playbook file to execute every role at once.

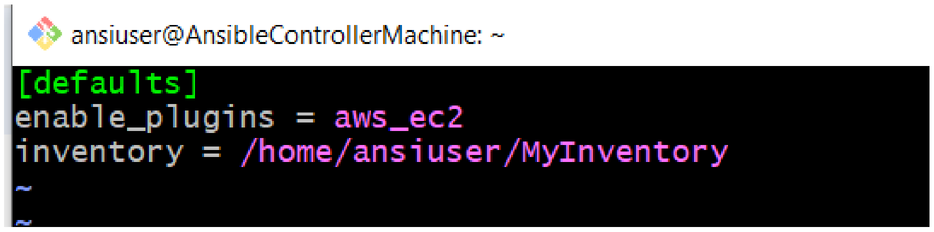


```
- hosts: webserver
  become: true
  roles:
   - userdemo
   - filesdemo
   - apache
```

3. Dynamic Inventory
   [amazon.aws.aws_ec2 inventory – EC2 inventory source — Ansible Documentation]

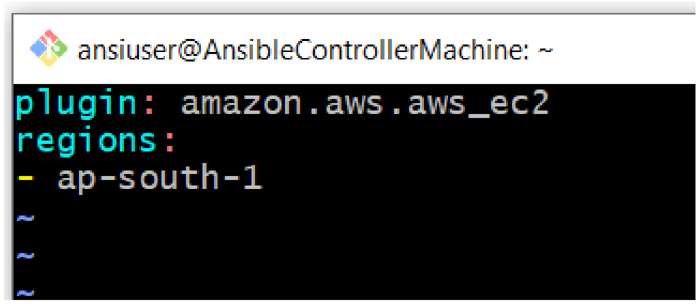
➔ We need to install python, boto3, botocore on our controller machine to execute dynamic inventory from AWS. Along with that we need to install aws package on our controller machine via ansible galaxy command
# ansible-galaxy collection install amazon.aws
Install boto3 and botocore
# sudo apt install python3-pip -y
# pip3 install boto3

➔ After that, you need to update the ansible.cfg file with the plugin details.



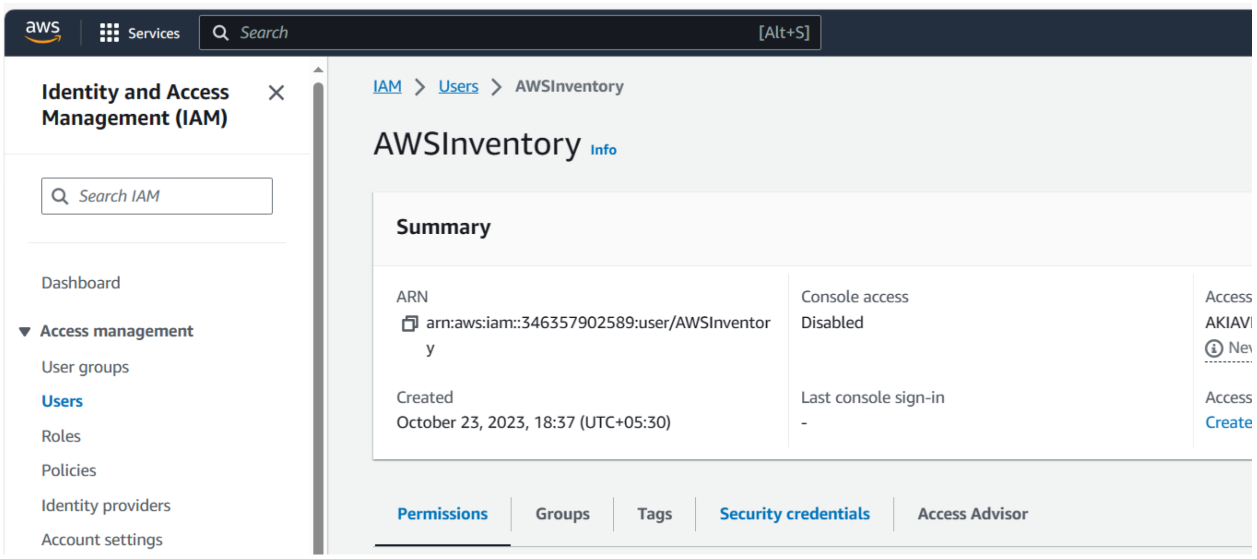
```
ansiuser@AnsibleControllerMachine: ~
[defaults]
enable_plugins = aws_ec2
inventory = /home/ansiuser/MyInventory
~
~
```

➔ Create AWS inventory File named - # vim aws_ec2.yml



```
ansiuser@AnsibleControllerMachine: ~
plugin: amazon.aws.aws_ec2
regions:
- ap-south-1
~
~
~
```

➔ Now, you will need to create Access Key and Secret Keys via IAM roles on AWS to be called on the controller machine.


IAM > Users > AWSInventory

AWSInventory Info

Summary

ARN
arn:aws:iam::346357902589:user/AWSInventory

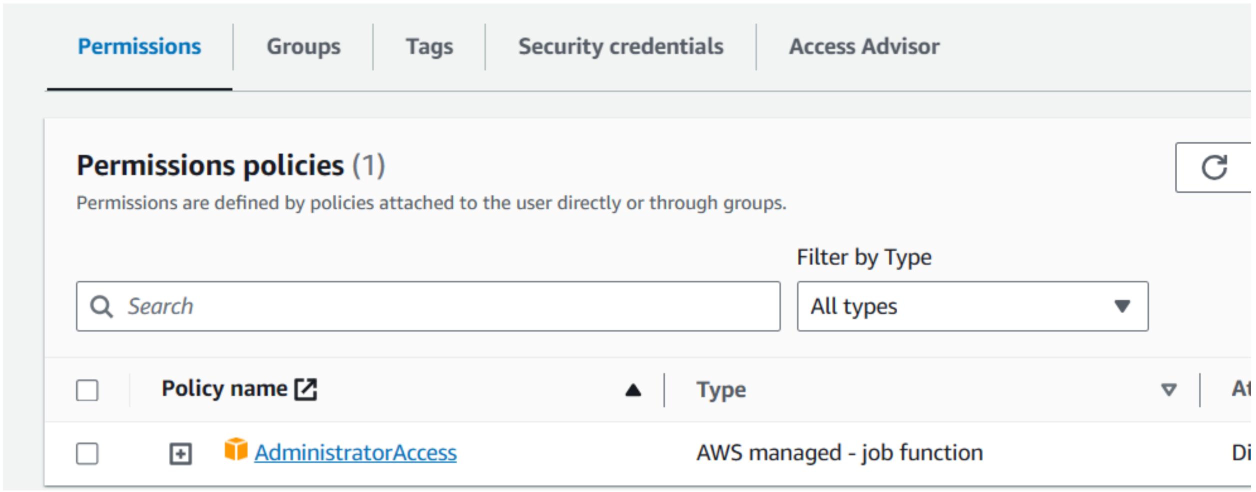
Created
October 23, 2023, 18:37 (UTC+05:30)

Console access
Disabled

Last console sign-in
-

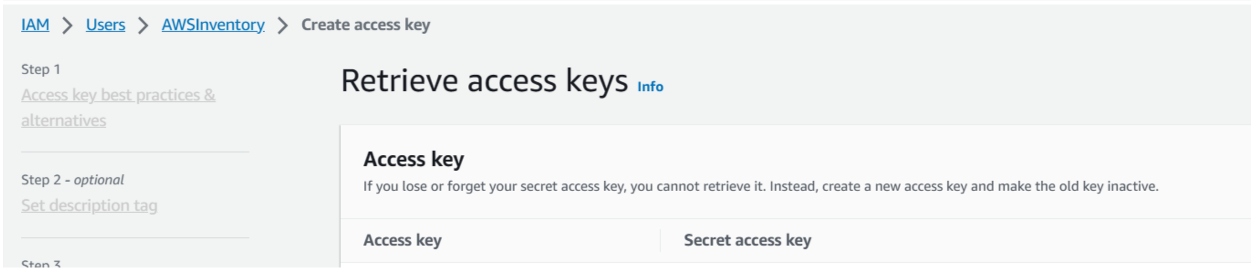
Access
AKIAVI

Access
Create


Permissions | Groups | Tags | Security credentials | Access Advisor

Permissions policies (1)
Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type
AdministratorAccess	AWS managed - job function


IAM > Users > AWSInventory > Create access key

Step 1
Access key best practices & alternatives

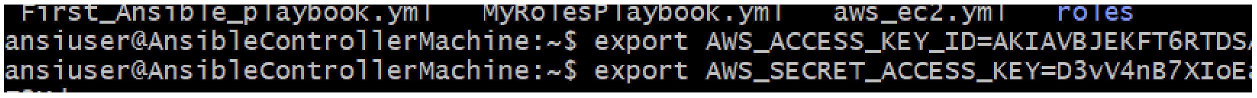
Step 2 - optional
Set description tag

Retrieve access keys Info

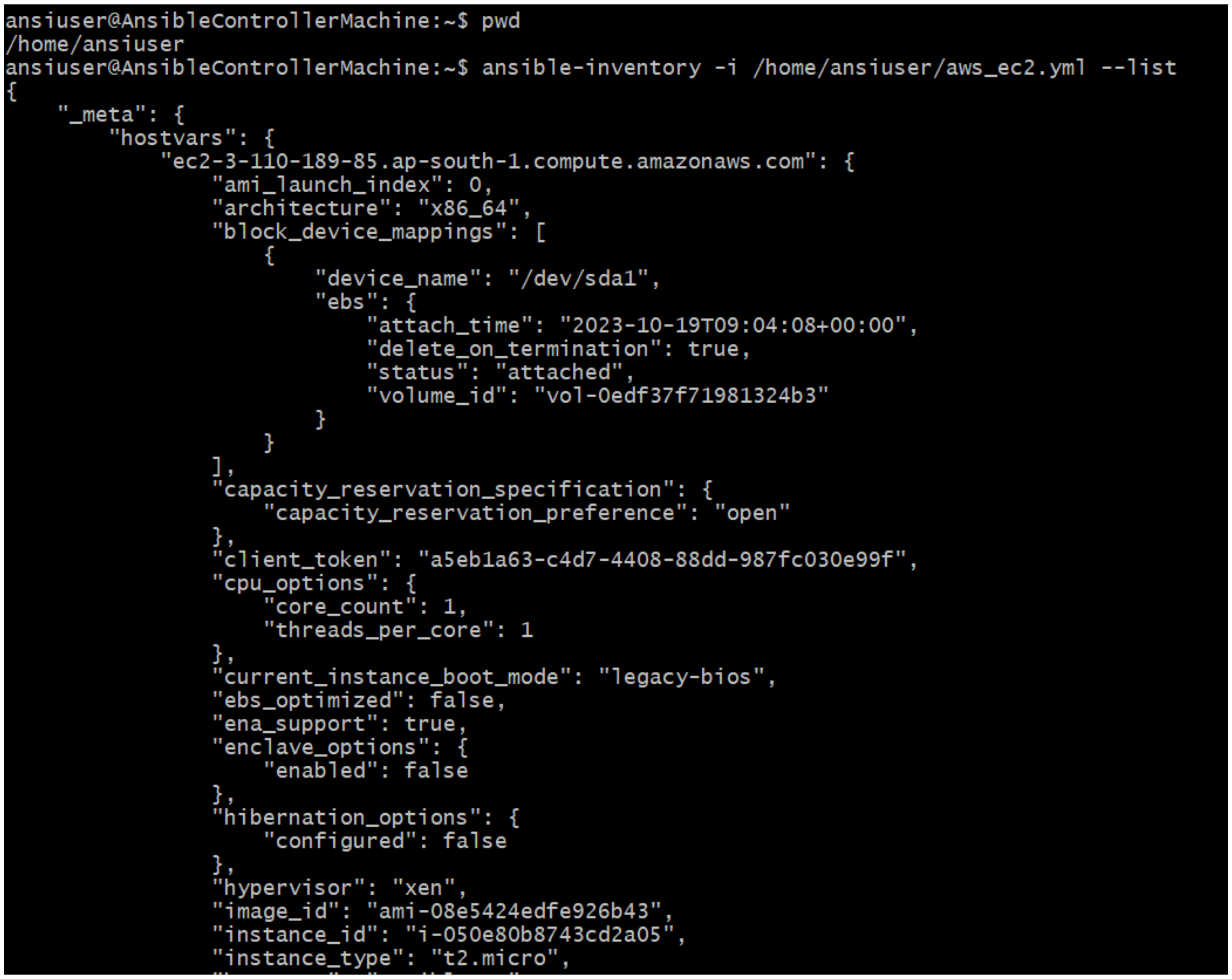
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key | Secret access key

➔ Copy the access and secret key and call in the AWS directory in the console via command #export AWS_ACCESS_KEY_ID= , #export AWS_SECRET_ACCESS_KEY_ID=

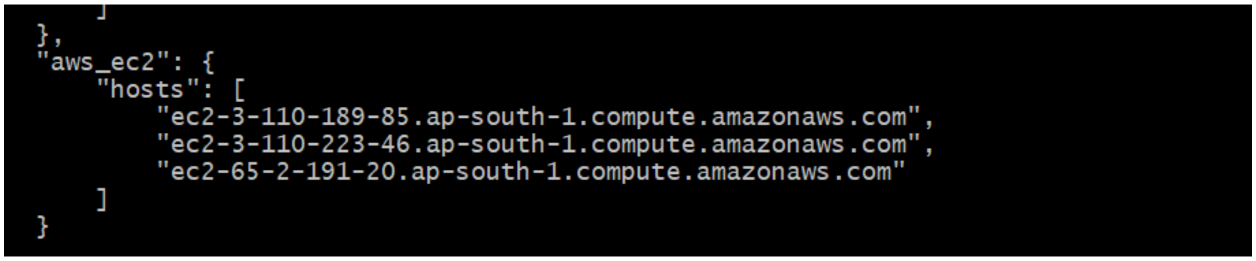

```
 First_Ansible_playbook.yml   MyRolesPlaybook.yml   aws_ec2.yml   roles
ansiuser@AnsibleControllerMachine:~$ export AWS_ACCESS_KEY_ID=AKIAVBJEKFT6RTDS
ansiuser@AnsibleControllerMachine:~$ export AWS_SECRET_ACCESS_KEY=D3vV4nB7XIoE
```

➔ Now if you run the command for ansible inventory giving the EC2 yaml file path, you will be able to see the details of your AWS EC2 machines present in the region you specified in the yaml file.



```
ansiuser@AnsibleControllerMachine:~$ pwd
/home/ansiuser
ansiuser@AnsibleControllerMachine:~$ ansible-inventory -i /home/ansiuser/aws_ec2.yml --list
{
    "_meta": {
        "hostvars": {
            "ec2-3-110-189-85.ap-south-1.compute.amazonaws.com": {
                "ami_launch_index": 0,
                "architecture": "x86_64",
                "block_device_mappings": [
                    {
                        "device_name": "/dev/sda1",
                        "ebs": {
                            "attach_time": "2023-10-19T09:04:08+00:00",
                            "delete_on_termination": true,
                            "status": "attached",
                            "volume_id": "vol-0edf37f71981324b3"
                        }
                    }
                ],
                "capacity_reservation_specification": {
                    "capacity_reservation_preference": "open"
                },
                "client_token": "a5eb1a63-c4d7-4408-88dd-987fc030e99f",
                "cpu_options": {
                    "core_count": 1,
                    "threads_per_core": 1
                },
                "current_instance_boot_mode": "legacy-bios",
                "ebs_optimized": false,
                "ena_support": true,
                "enclave_options": {
                    "enabled": false
                },
                "hibernation_options": {
                    "configured": false
                },
                "hypervisor": "xen",
                "image_id": "ami-08e5424edfe926b43",
                "instance_id": "i-050e80b8743cd2a05",
                "instance_type": "t2.micro",
```

You will be able to see the EC2 instances information under section:



```
        ]
    },
    "aws_ec2": {
        "hosts": [
            "ec2-3-110-189-85.ap-south-1.compute.amazonaws.com",
            "ec2-3-110-223-46.ap-south-1.compute.amazonaws.com",
            "ec2-65-2-191-20.ap-south-1.compute.amazonaws.com"
        ]
    }
```

➔ Now, In order to execute any modules on the inventory, you have to perform the SSH connection

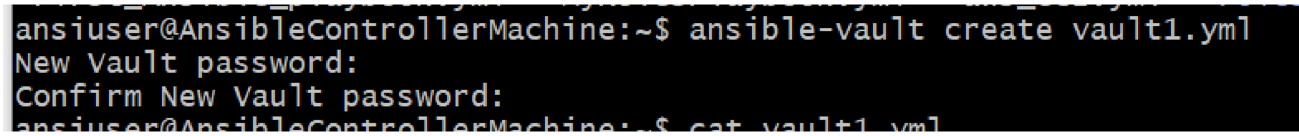
Perform all 3 steps of executing ssh connection on your ansiuser as we have done previously
# vim /etc/ssh/sshd_config
# vim /etc/sudoers
# systemctl restart sshd

➔ Generate the ssh key on controller machine and perform the same steps of copying ssh key on host servers via public IP
# ssh-copy-id -i ansiuser@publicipOfEc2
It will be successfully connected to the inventory ec2 machines and you will be able to execute modules on them.

4. Ansible Vault

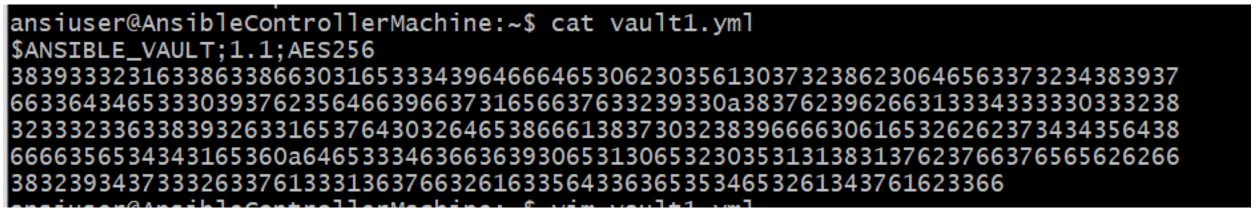
➔ Used for transforming data into the encrypted data with password and then decrypting it when entered the password.

Creating the file with vault function



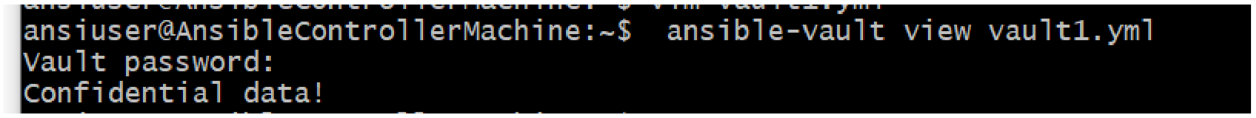
```
ansiuser@AnsibleControllerMachine:~$ ansible-vault create vault1.yml
New Vault password:
Confirm New Vault password:
```

Now any data, that you have entered in the file will be only visible in the encrypted format



```
ansiuser@AnsibleControllerMachine:~$ cat vault1.yml
$ANSIBLE_VAULT;1.1;AES256
383933323163386338663031653334396466664653062303561303732386230646563373234383937
66336434653330393762356466396637316566376332393330a383762396266313334333330333238
32333233633839326331653764303264653866613837303238396666306165326262373434356438
6666356534343165360a646533346366363930653130653230353131383137623766376565626266
38323934373332633761333136376632616335643363653534653261343761623366
```

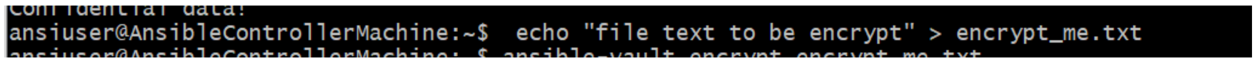
➔ To view the data in decrypted format. Enter the password giving the view command



```
ansiuser@AnsibleControllerMachine:~$  ansible-vault view vault1.yml
Vault password:
Confirmential data!
```

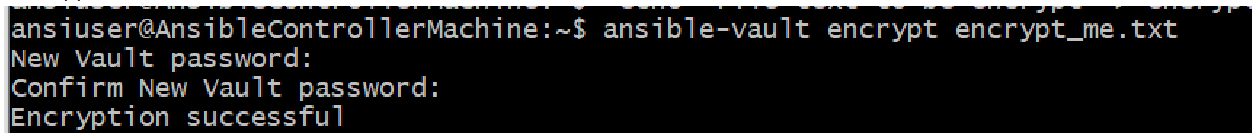
➔ Encrypting an existing file

Create a file

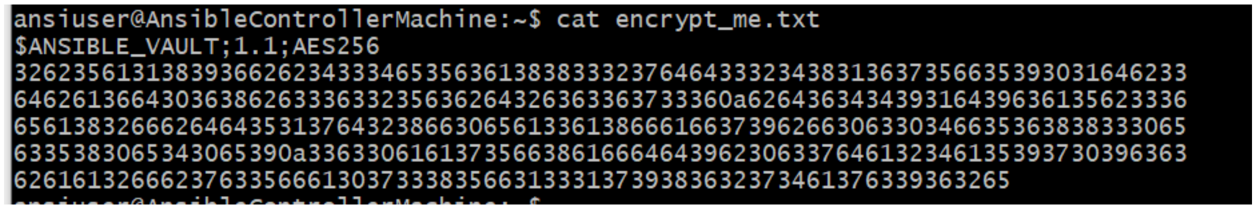


```
ansiuser@AnsibleControllerMachine:~$  echo "file text to be encrypt" > encrypt_me.txt
```

Encrypt the file

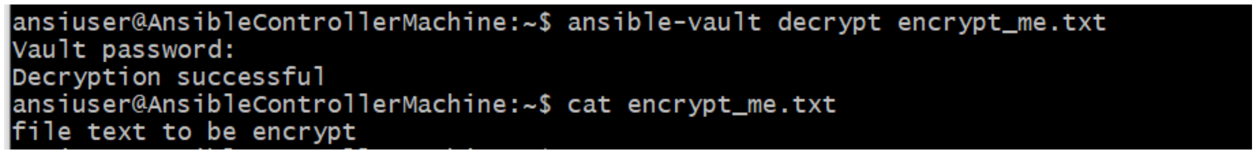


```
ansiuser@AnsibleControllerMachine:~$ ansible-vault encrypt encrypt_me.txt
New Vault password:
Confirm New Vault password:
Encryption successful
```



```
ansiuser@AnsibleControllerMachine:~$ cat encrypt_me.txt
$ANSIBLE_VAULT;1.1;AES256
326235613138393662623433334653563613838333237646433323438313637356635393031646233
64626136643036386263336332356362643263633637333360a626436343439316439636135623336
65613832666264643531376432386630656133613866616637396266306330346635363838333065
6335383065343065390a336330616137356638616664643962306337646132346135393730396363
62616132666237633566613037333835663133313739383632373461376339363265
```

➔ Decrypt the file



```
ansiuser@AnsibleControllerMachine:~$ ansible-vault decrypt encrypt_me.txt
Vault password:
Decryption successful
ansiuser@AnsibleControllerMachine:~$ cat encrypt_me.txt
file text to be encrypt
```

➔ Edit an existing Encrypted file
# ansible-vault edit vault1.yml

➔ Change the password of encrypted file
# ansible-vault rekey vault1.yml

➔ Store the vault password in a file and pass the file name in command
# echo "password" > my_passwd_file
# ansible-vault view --vault-password-file my_passwd_file encrypt_me.txt