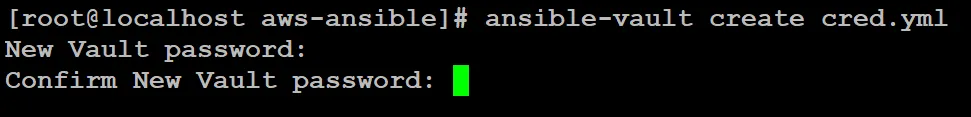
# Configuring Web Server on EC2 Instance Using Ansible

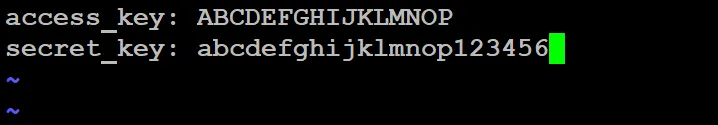
1. Provision EC2 instance through ansible.
2. Retrieve the IP Address of instance using a dynamic inventory concept.
3. Configure the webserver through ansible.
4. Create a role for the webserver to customize the Instance and deploy the webpage to the root directory.

Create a credential

ansible-vault create cred.yml



**open the VI editor. Inside the editor write these below mentioned script.**

.

## Configuring “ansible.cfg” file :

We have to **have the AWS key-pair in “pem” format.**

**Next, create one folder inside your workspace which we gonna use to configure our Ansible Roles. I gave the name of the folder “roles”.**

vim /etc/ansible/ansible.cfg[defaults]

host\_key\_checking = false

remote\_user = ec2-user

ask\_pass = false

private\_key\_file = /root/aws-ansible/webkey.pem

roles\_path = /root/aws-ansible/roles[privilege\_escalation]

become = true

become\_method = sudo

become\_user = root

become\_ask\_pass = false

# Creation of Ansible Playbook :

## Installation of boto3 python library :

Boto.yml

|  |
| --- |
| - hosts: localhost |

|  |
| --- |
| gather\_facts: no |

|  |
| --- |
| vars\_files: |

|  |
| --- |
| - cred.yml |

|  |
| --- |
| tasks: |

|  |
| --- |
| - name: |

|  |
| --- |
| pip: |

|  |
| --- |
| name: boto3 |

state: present

* Here I selected the **Host Group to localhost as I am going to run this script on my own controller node.** The value of **“gather\_facts”** is no because I don't want to collect information about my localhost system.
* Next, **in “vars\_files” I have given “cred.yml”** because from this file Ansible gonna pick up the values of aws access key and secret key.
* Next, **to configure anything on AWS using Ansible, we need “boto3” python library** as this library gonna help us to reach the API of AWS.

## Security Group :

[ansible-sg.yml](https://gist.github.com/raktim00/09ff5cd2d4644d56ca473d5d9fb62ab3" \l "file-ansible-sg-yml)

- name: Creating Security Group for WebServer on AWS

ec2\_group:

name: WebSG

description: Security Group for Web Server allowing port for http and ssh

region: ap-south-1

aws\_access\_key: "{{ access\_key }}"

aws\_secret\_key: "{{ secret\_key }}"

rules:

- proto: tcp

from\_port: 22

to\_port: 22

cidr\_ip: 0.0.0.0/0

- proto: tcp

from\_port: 80

to\_port: 80

cidr\_ip: 0.0.0.0/0

rules\_egress:

- proto: all

cidr\_ip: 0.0.0.0/0

* As we can see, this script is **creating one AWS Security Group called “WebSG” on the region “ap-south-1”**. To, make things simple I just **hard-coded these values** but we can easily use variables to take input from the user.
* Next in **“aws\_access\_key”** and **“aws\_security\_key”** I have mentioned the variable names mentioned in **“cred.yml”** file.

## EC2 Instance :

[ansible-ec2.yml](https://gist.github.com/raktim00/7b99a515e3bfc226d20aacee3540c26d" \l "file-ansible-ec2-yml)

- name: launching ec2 instance for webserver

ec2:

key\_name: webkey

instance\_type: t2.micro

image: ami-0e306788ff2473ccb

wait: true

group: WebSG

count: 1

vpc\_subnet\_id: subnet-6dfdc705

assign\_public\_ip: yes

region: ap-south-1

state: present

aws\_access\_key: "{{ access\_key }}"

aws\_secret\_key: "{{ secret\_key }}"

instance\_tags:

Name: webserver

register: ec2

## Dynamic Inventory :

[ansible-ssh-role.ym](https://gist.github.com/raktim00/199576c412f04d6b365b2e94273ccc5c" \l "file-ansible-ssh-role-yml)l

- name: Add new instance to host group

add\_host:

hostname: "{{ item.public\_ip }}"

groupname: webserver

loop: "{{ ec2.instances }}"

- name: Wait for SSH to come up

wait\_for:

host: "{{ item.public\_dns\_name }}"

port: 22

state: started

loop: "{{ ec2.instances }}"

- hosts: webserver

gather\_facts: no

tasks:

- name: running role

include\_role:

name: httpdserver

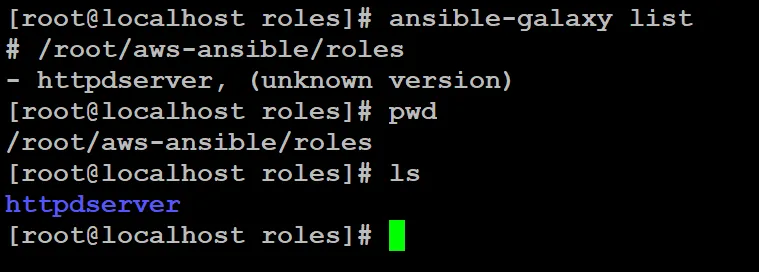
* Here, at first, we use one module called **“add\_host”**. This module helps us to **add IP dynamically in a temporary inventory variable**. Here each time my Ansible Playbook will run, it will store the metadata of the Instances in a variable called “ec2”.  
  **Next using for loop, we gonna fetch the Instances one after another** in case we have more than one Instance and **then put their public IP in a variable called “hostname”.**  
  **Finally, this “hostname” variable gonna store the IPs and create one host group named “webserver”.**
* Next, I used one module called **“wait\_for”,** which helps to **look if my instance is ready to do SSH.** Here using the **public DNS of our Instance we are checking if the SSH service has started on port no 22 or not.** Once the Instance is ready to do SSH then it will go on next play.
* If you noticed one thing till the **“wait\_for”,** each and every play we run on the localhost. But now we are using “webserver”. **This is the same name as the host group that we are creating temporarily using a dynamic inventory concept.**
* **In “webserver” host group we have run one Ansible Role called “httpdserver” using “include\_role” keyword.**

## Creating Role :

For creating Role you need to **go inside the “Role” directory that you have specified in the ansible.cfg file**. In our case, we have created one directory called “roles” inside our workspace and mentioned that inside our “ansible.cfg” file.

**So, go inside the folder called “roles” in your Workspace and run this below mentioned command and role will be created.**

ansible-galaxy init httpdserver

.

## Creating Task inside Role :

Next, we need to write **the tasks which will configure HTTPD server** on our EC2 Instance.

For the go inside the **“httpdserver”** folder and then go inside the **“tasks”** folder. There you will find one file called **“main.yml”**. Edit that file and write the below-mentioned script.

[main.yml](https://gist.github.com/raktim00/82a7e937a6bd602e3916a8cd6b11e132" \l "file-main-yml)

# tasks file for httpdserver

- name: install httpd

package:

name: httpd

state: present

- name: copy code

get\_url:

url: https://raw.githubusercontent.com/raktim00/DevOpsHW/master/index.html

dest: "/var/www/html/index.html"

- name: start httpd

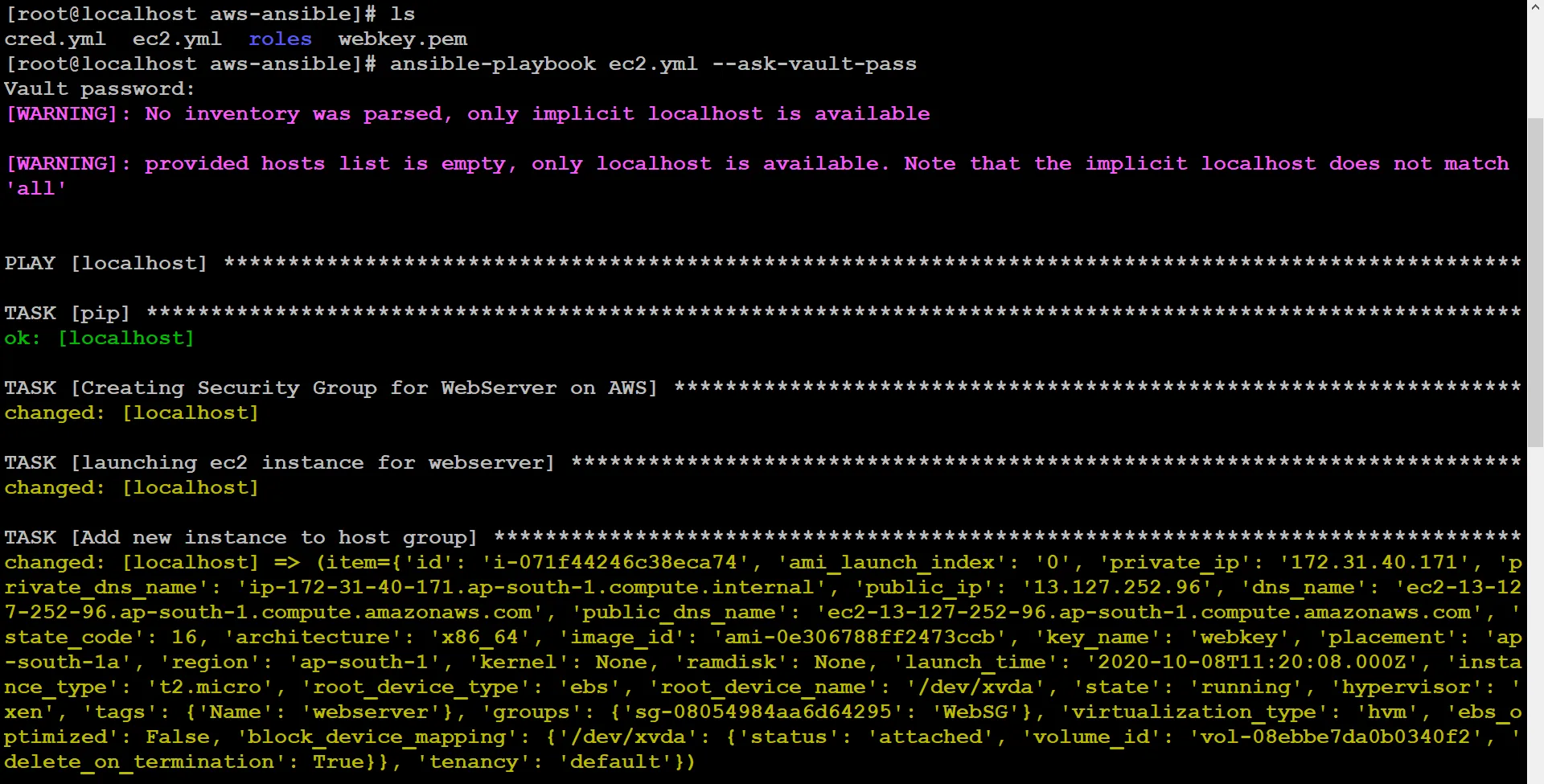
service:

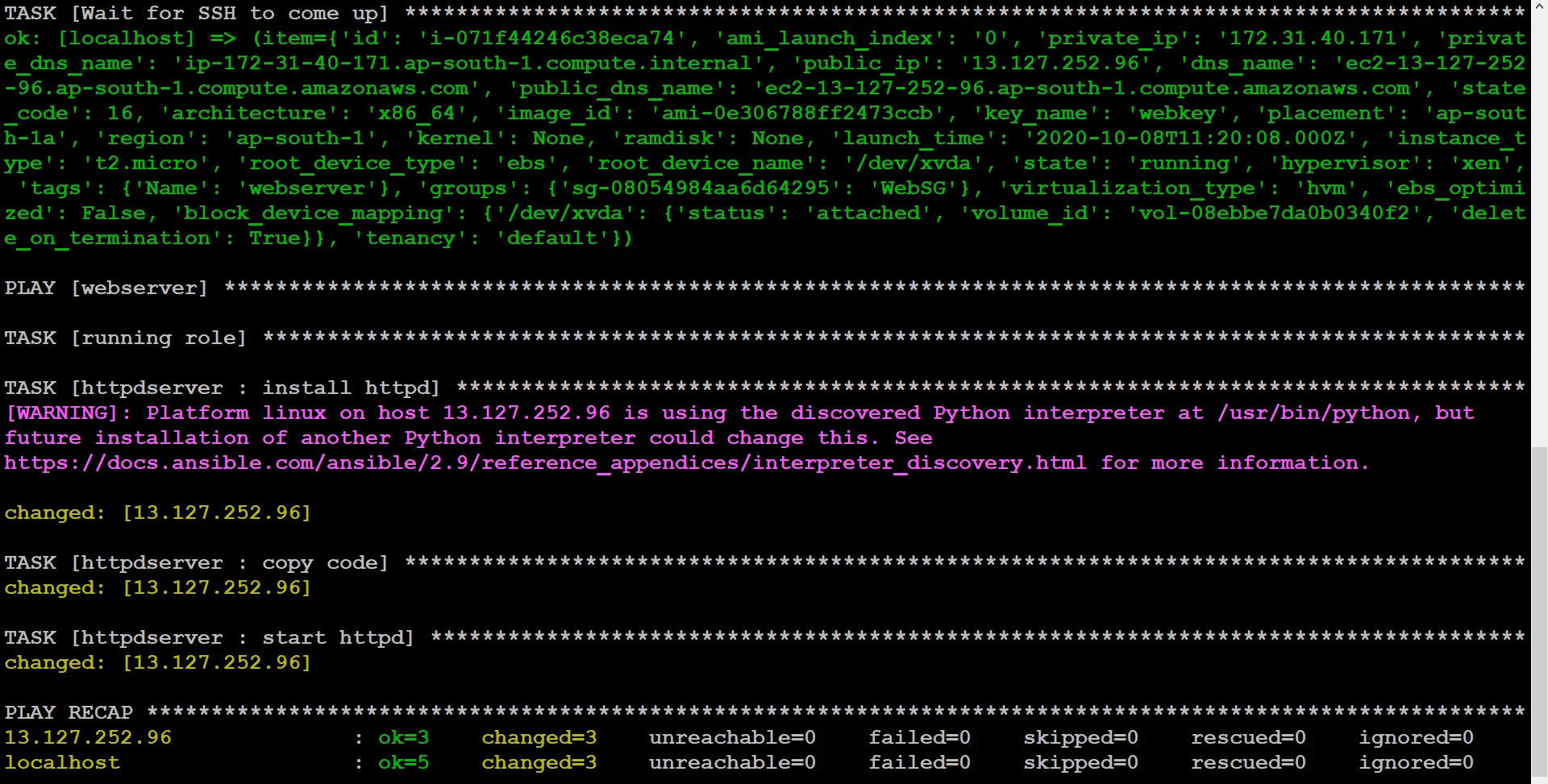
name: httpd

state: started

# Let's Deploy :

ansible-playbook ec2.yml --ask-vault-pass



.

Output

