Turning a Playbook Into a Role

Muhannad Sinan

Turning a Playbook Into a Role

Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

Assignment

The playbook given below creates a web server (apache2) and updates the web server with some sample html file to display the working of the server.

This below playbook is created as one big monolithic playbook and cannot be reused.

We will use roles to break this playbook into multiple playbooks such that these playbooks are reusable and easy to managed.

```
---

- hosts: all
  become: true
  vars:
    doc_root: /var/www/example
  tasks:
    - name: Update apt
      apt: update_cache=yes

    - name: Install Apache
      apt: name=apache2 state=latest

    - name: Create custom document root
      file: path={{ doc_root }} state=directory owner=www-data group=www-data

    - name: Set up HTML file
      copy: src=index.html dest={{ doc_root }}/index.html owner=www-data
group=www-data mode=0644

    - name: Set up Apache virtual host file
      template: src=vhost.tpl dest=/etc/apache2/sites-available/000-
default.conf
```

```
        notify: restart apache

    handlers:
      - name: restart apache
        service: name=apache2 state=restarted
```

Make sure that the web server runs with the below playbook.

To test the correct installation of use below sample `index.html`

```
<html>

<head><title>Configuration Management Hands On</title></head>
<h1>This server was provisioned using <strong>Ansible</strong></h1>

</html>
```

To configure the files use below sample template file:-

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
DocumentRoot {{ doc_root }}
<Directory {{ doc_root }}>

AllowOverride All
Require all granted
</Directory>
</VirtualHost>
```

Step1:

Creating three EC2 instances, one Linux server as an Ansible master and two Ubuntu servers using this CloudFormation template:

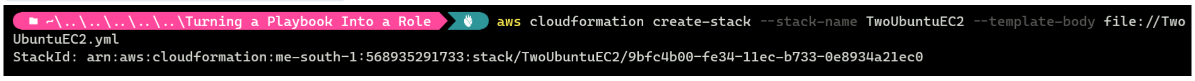
`TwoUbuntuEC2.yml`

```
AWSTemplateFormatVersion: "2010-09-09"
Metadata:
  License: Apache-2.0
Description: "OPTIONAL Assignment 2 - Turning a Playbook Into a Role"
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the
instance
```

```
    Type: AWS::EC2::KeyPair::KeyName
    Default: main
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t3.micro
    AllowedValues: [t3.nano, t3.micro, t3.small, t3.medium]
    ConstraintDescription: must be a valid EC2 instance type.
  SSHLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: (\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  HTTPLocation:
    Description: The IP address range that can be used to SSH to the EC2 instances
    Type: String
    MinLength: 9
    MaxLength: 18
    Default: 0.0.0.0/0
    AllowedPattern: (\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
  LatestAmzLinuxAmiId:
    Type: "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>"
    Default: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  LatestUbuntuAmiId:
    Type:  'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default:
'/aws/service/canonical/ubuntu/server/20.04/stable/current/amd64/hvm/ebs-gp2/ami-
id'
  InstanceCount:
    Description: Number of EC2 instances (must be between 1 and 5).
    Type: Number
    Default: 1
    MinValue: 1
    MaxValue: 3
    ConstraintDescription: Must be a number between 1 and 5.

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: ansbile-master-sg
      GroupDescription: Enable SSH access via port 22
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref "SSHLocation"

  NodeSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: ansbile-node-sg
      GroupDescription: Enable SSH access via port 22 and HTTP access via port 80
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: !Ref "SSHLocation"
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref "HTTPLocation"

  EC2InstanceMaster:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref "InstanceType"
      SecurityGroups: [!Ref "MasterSecurityGroup"]
      KeyName: !Ref "KeyName"
      ImageId: !Ref "LatestAmzLinuxAmiId"
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          sudo yum update -y
          sudo amazon-linux-extras install ansible2 -y
      Tags:
        - Key: Name
          Value: ansible-master
  EC2InstanceNode1:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref "InstanceType"
      SecurityGroups: [!Ref "NodeSecurityGroup"]
      KeyName: !Ref "KeyName"
      ImageId: !Ref "LatestUbuntuAmiId"
      Tags:
        - Key: Name
          Value: Node1
  EC2InstanceNode2:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref "InstanceType"
      SecurityGroups: [!Ref "NodeSecurityGroup"]
      KeyName: !Ref "KeyName"
      ImageId: !Ref "LatestUbuntuAmiId"
```

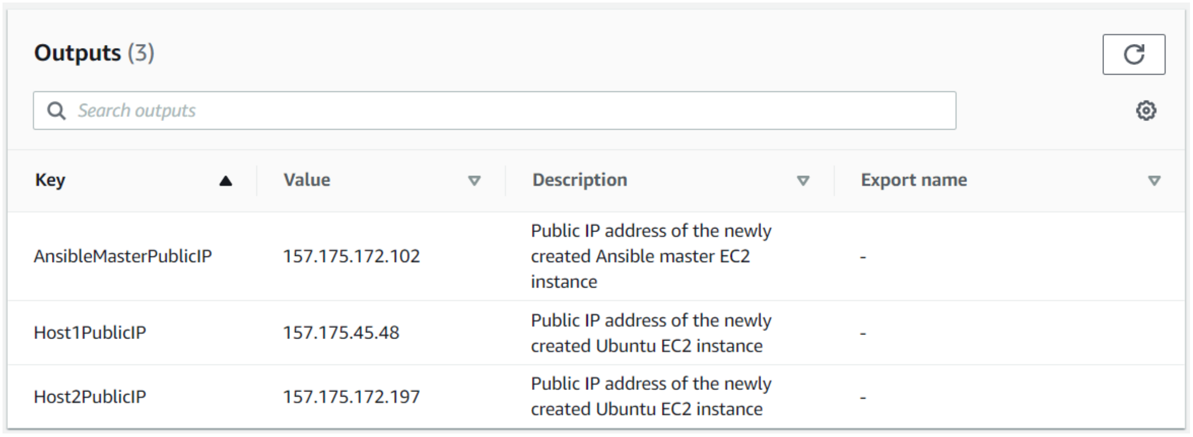
```yaml
      Tags:
        - Key: Name
          Value: Node2

Outputs:
  AnsibleMasterPublicIP:
    Description: Public IP address of the newly created Ansible master EC2 instance
    Value: !GetAtt [EC2InstanceMaster, PublicIp]

  Host1PublicIP:
    Description: Public IP address of the newly created Ubuntu EC2 instance
    Value: !GetAtt [EC2InstanceNode1, PublicIp]

  Host2PublicIP:
    Description: Public IP address of the newly created Ubuntu EC2 instance
    Value: !GetAtt [EC2InstanceNode2, PublicIp]
```

Creating the CloudFormation stack using aws cli:
`aws cloudformation create-stack --stack-name TwoUbuntuEC2 --template-body file://TwoUbuntuEC2.yml`



```
~\..\..\..\..\..\Turning a Playbook Into a Role  aws cloudformation create-stack --stack-name TwoUbuntuEC2 --template-body file://Two
UbuntuEC2.yml
StackId: arn:aws:cloudformation:me-south-1:568935291733:stack/TwoUbuntuEC2/9bfc4b00-fe34-11ec-b733-0e8934a21ec0
```

The outputs is:

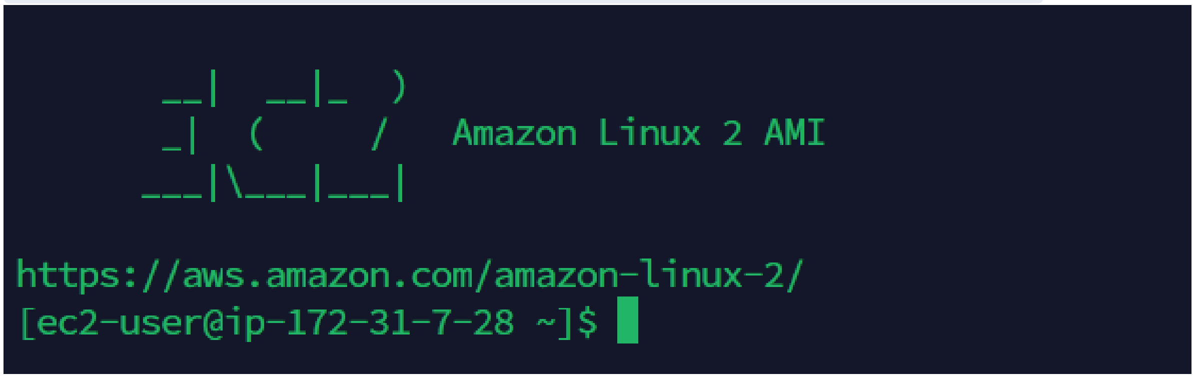


Outputs (3)

Key	Value	Description	Export name
AnsibleMasterPublicIP	157.175.172.102	Public IP address of the newly created Ansible master EC2 instance	-
Host1PublicIP	157.175.45.48	Public IP address of the newly created Ubuntu EC2 instance	-
Host2PublicIP	157.175.172.197	Public IP address of the newly created Ubuntu EC2 instance	-

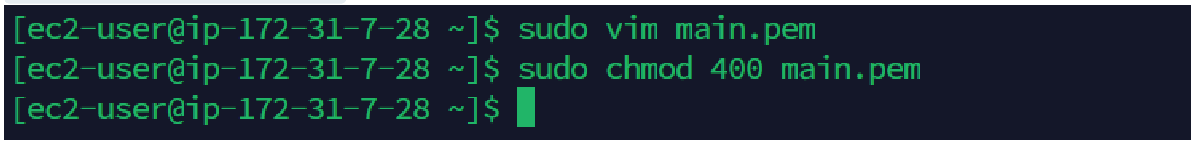
Step2:

Connect to `ansible-master` server using ssh protocol:
`ssh -i "main.pem" root@ec2-157-175-172-102.me-south-1.compute.amazonaws.com`



```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-7-28 ~]$
```

Add my private ssh key to the Ansible master server to this directory `/home/ec2-user/main.pem` and change the permissions to 400:
`sudo vim main.pem`
`sudo chmod 400 main.pem`

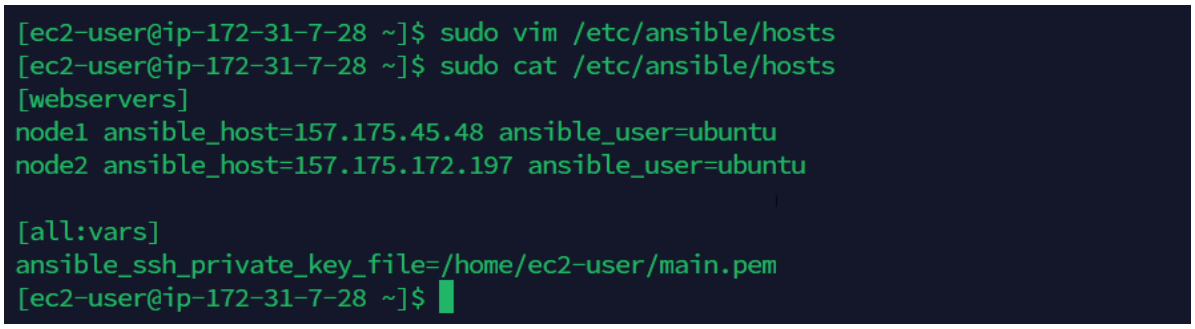


```
[ec2-user@ip-172-31-7-28 ~]$ sudo vim main.pem
[ec2-user@ip-172-31-7-28 ~]$ sudo chmod 400 main.pem
[ec2-user@ip-172-31-7-28 ~]$
```

Step3:

Creating an inventory file in the path `/etc/ansible/hosts`
`sudo vim /etc/ansible/hosts`
with the following content:

```
[webservers]
node1 ansible_host=157.175.45.48 ansible_user=ubuntu
node2 ansible_host=157.175.172.197 ansible_user=ubuntu

[all:vars]
ansible_ssh_private_key_file=/home/ec2-user/main.pem
```



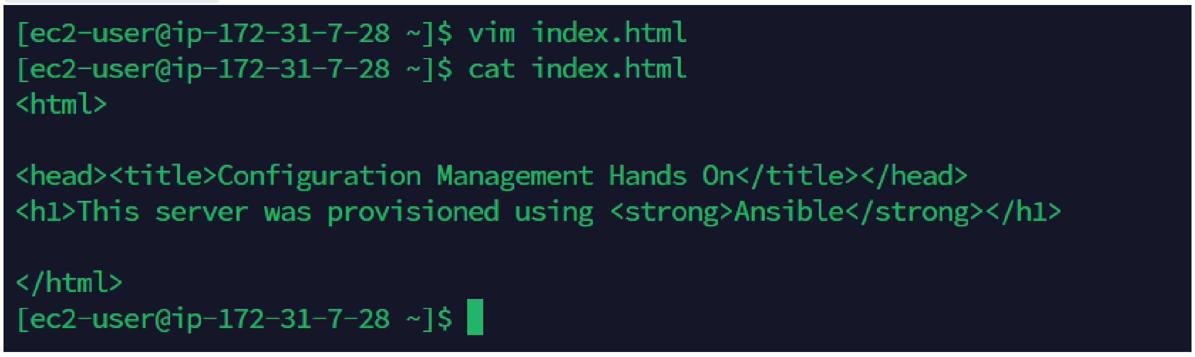
```
[ec2-user@ip-172-31-7-28 ~]$ sudo vim /etc/ansible/hosts
[ec2-user@ip-172-31-7-28 ~]$ sudo cat /etc/ansible/hosts
[webservers]
node1 ansible_host=157.175.45.48 ansible_user=ubuntu
node2 ansible_host=157.175.172.197 ansible_user=ubuntu

[all:vars]
ansible_ssh_private_key_file=/home/ec2-user/main.pem
[ec2-user@ip-172-31-7-28 ~]$
```

Step4:

Creating an html file `index.html` :

```
<html>


<head><title>Configuration Management Hands On</title></head>
<h1>This server was provisioned using <strong>Ansible</strong></h1>


</html>
```

`vim index.html`



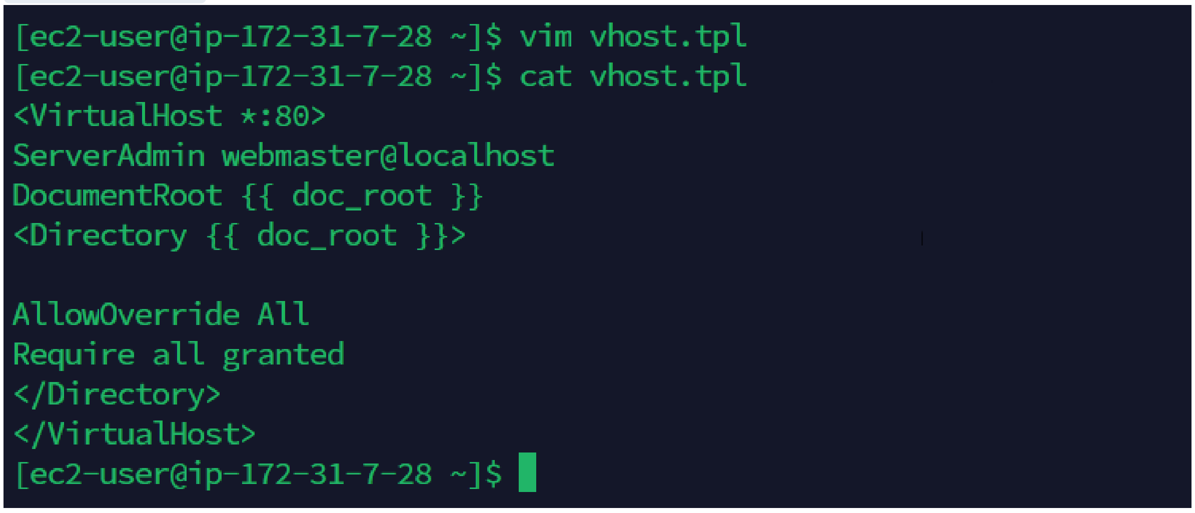
```
[ec2-user@ip-172-31-7-28 ~]$ vim index.html
[ec2-user@ip-172-31-7-28 ~]$ cat index.html
<html>

<head><title>Configuration Management Hands On</title></head>
<h1>This server was provisioned using <strong>Ansible</strong></h1>

</html>
[ec2-user@ip-172-31-7-28 ~]$
```

Step5:

Creating a configure template file `vhost.tpl` :

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
DocumentRoot {{ doc_root }}
<Directory {{ doc_root }}>

AllowOverride All
Require all granted
</Directory>
</VirtualHost>
```

`vim vhost.tpl`

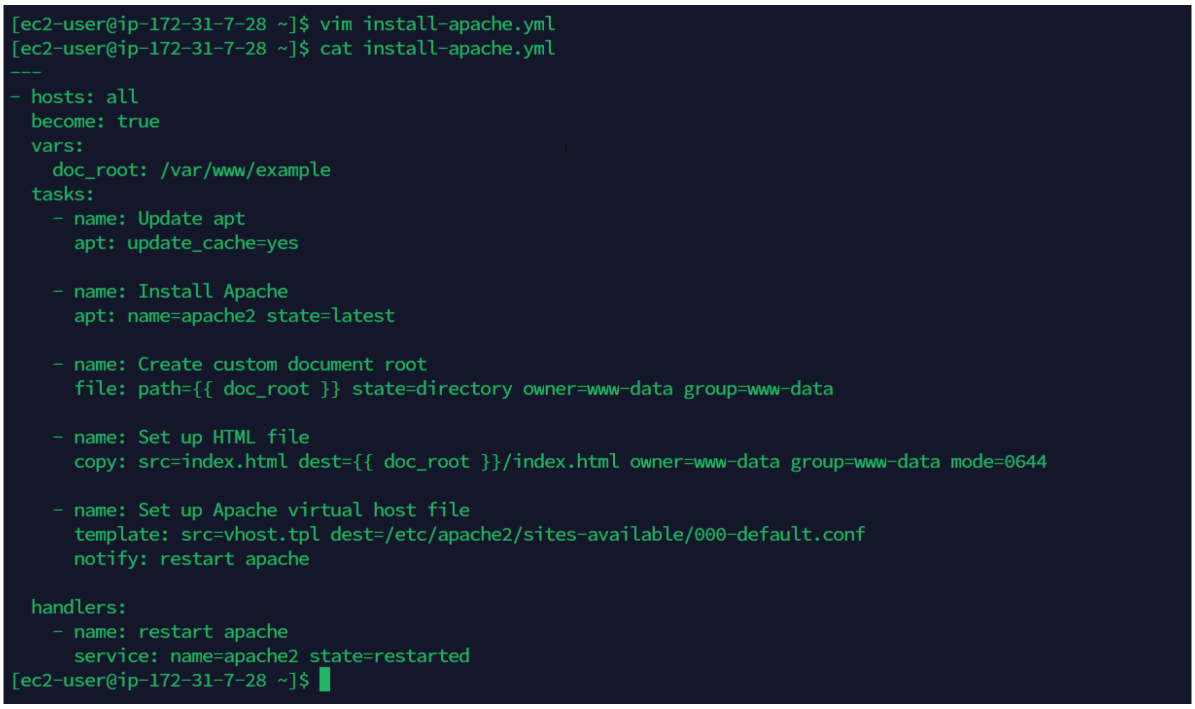


```
[ec2-user@ip-172-31-7-28 ~]$ vim vhost.tpl
[ec2-user@ip-172-31-7-28 ~]$ cat vhost.tpl
<VirtualHost *:80>
ServerAdmin webmaster@localhost
DocumentRoot {{ doc_root }}
<Directory {{ doc_root }}>

AllowOverride All
Require all granted
</Directory>
</VirtualHost>
[ec2-user@ip-172-31-7-28 ~]$
```

Step6:

Writing a playbook to install apache on both of the hosts
`vim install-apache.yml`

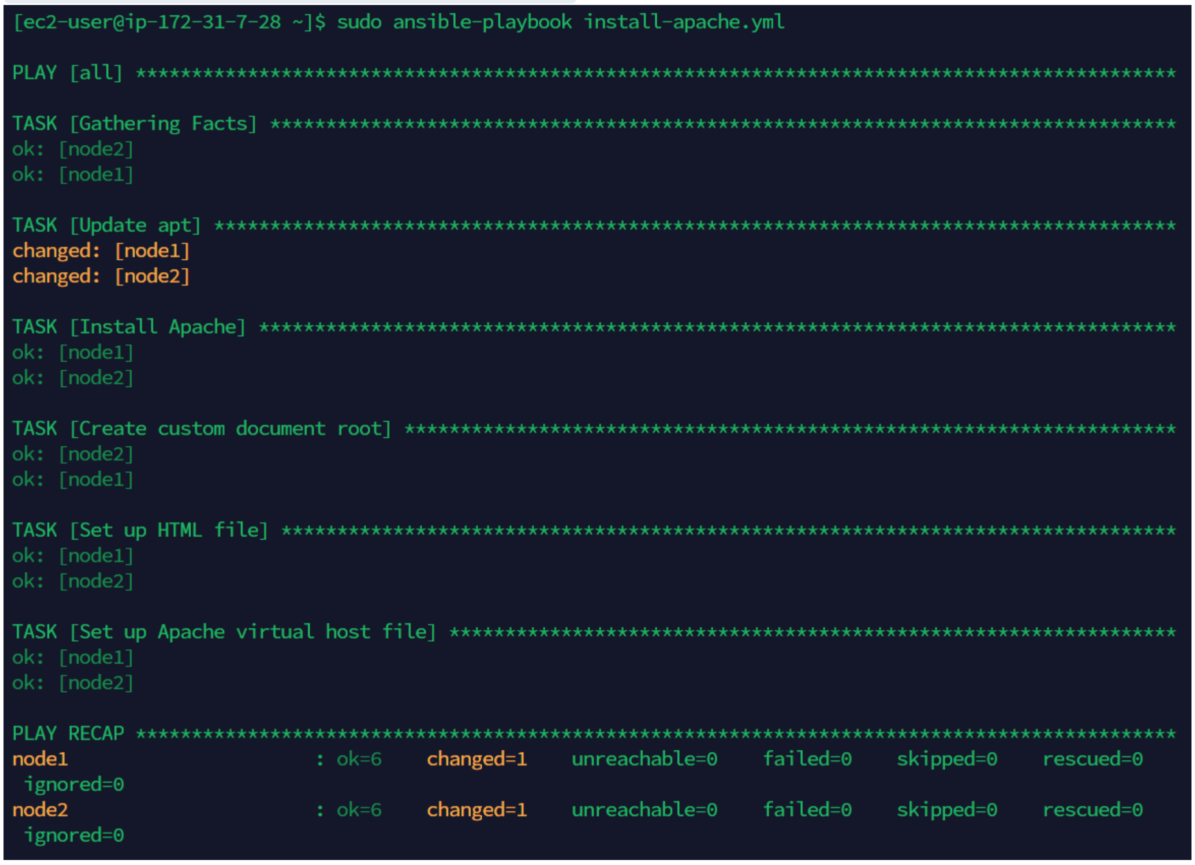
```
---
- hosts: all
  become: true
  vars:
    doc_root: /var/www/example
  tasks:
    - name: Update apt
      apt: update_cache=yes

    - name: Install Apache
      apt: name=apache2 state=latest

    - name: Create custom document root
      file: path={{ doc_root }} state=directory owner=www-data group=www-data

    - name: Set up HTML file
      copy: src=index.html dest={{ doc_root }}/index.html owner=www-data group=www-
data mode=0644

    - name: Set up Apache virtual host file
      template: src=vhost.tpl dest=/etc/apache2/sites-available/000-default.conf
      notify: restart apache

  handlers:
    - name: restart apache
      service: name=apache2 state=restarted
```



```
[ec2-user@ip-172-31-7-28 ~]$ vim install-apache.yml
[ec2-user@ip-172-31-7-28 ~]$ cat install-apache.yml
---
- hosts: all
  become: true
  vars:
    doc_root: /var/www/example
  tasks:
    - name: Update apt
      apt: update_cache=yes

    - name: Install Apache
      apt: name=apache2 state=latest

    - name: Create custom document root
      file: path={{ doc_root }} state=directory owner=www-data group=www-data

    - name: Set up HTML file
      copy: src=index.html dest={{ doc_root }}/index.html owner=www-data group=www-data mode=0644

    - name: Set up Apache virtual host file
      template: src=vhost.tpl dest=/etc/apache2/sites-available/000-default.conf
      notify: restart apache

  handlers:
    - name: restart apache
      service: name=apache2 state=restarted
[ec2-user@ip-172-31-7-28 ~]$
```

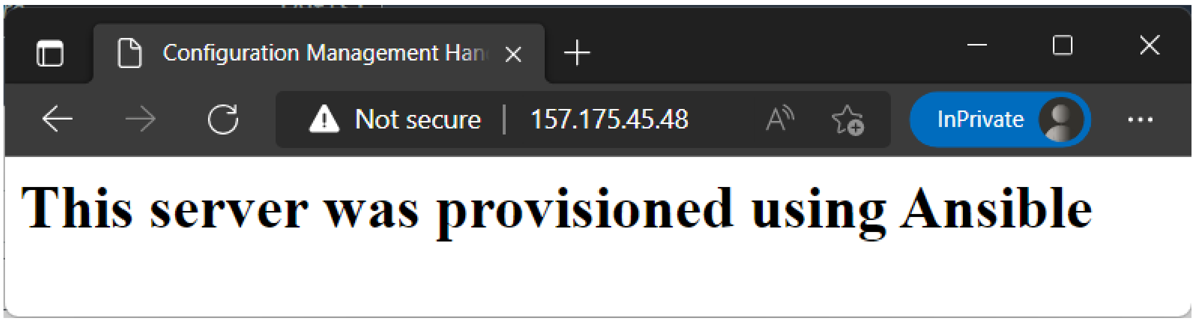
Run the following command in the folder where `install-apache.yml` file is saved

`sudo ansible-playbook install-apache.yml`



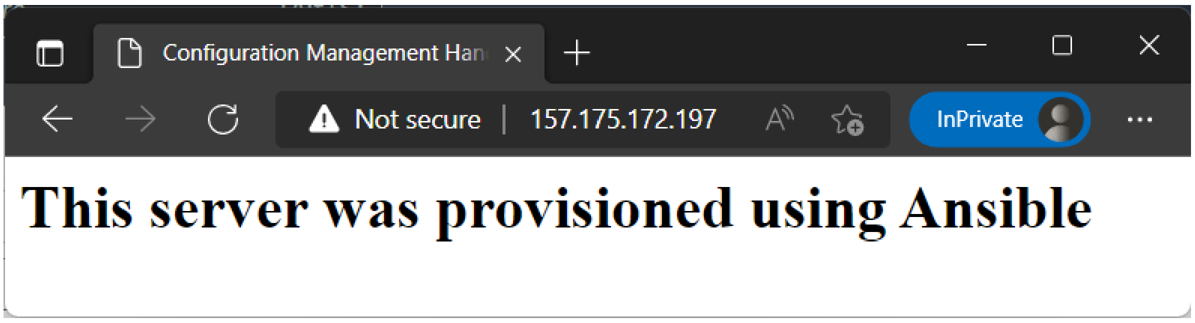
```
[ec2-user@ip-172-31-7-28 ~]$ sudo ansible-playbook install-apache.yml

PLAY [all] *****************************************************************************************

TASK [Gathering Facts] *****************************************************************************
ok: [node2]
ok: [node1]

TASK [Update apt] **********************************************************************************
changed: [node1]
changed: [node2]

TASK [Install Apache] ******************************************************************************
ok: [node1]
ok: [node2]

TASK [Create custom document root] *****************************************************************
ok: [node2]
ok: [node1]

TASK [Set up HTML file] ****************************************************************************
ok: [node1]
ok: [node2]

TASK [Set up Apache virtual host file] *************************************************************
ok: [node1]
ok: [node2]

PLAY RECAP *****************************************************************************************
node1                      : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
 ignored=0
node2                      : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
 ignored=0
```

Node1



This server was provisioned using Ansible

Node2



This server was provisioned using Ansible