Configuration Management with Ansible and Terraform

Course-End Project: Web Application Using Ansible

Objective

To create an automation script to deploy an application using Ansible and Jinja2 template.

Real-time scenario:

You have joined as a DevOps engineer in XYZ Pvt. Ltd. It is a platform where individuals can create their profile and start blogging on various topics. The application is ready to be hosted on a server. You are tasked with implementing an Ansible script to deploy this application on a remote Nginx server.

Tasks

The following tasks outline the process of deploying web application on a remote server:

1. Create an inventory file to define the remote server(s)

2. Write a YAML playbook with tasks for installing Nginx, copying web application files, deploying the Nginx configuration, and enabling the site

3. Create a directory for templates and a Jinja2 template for the Nginx configuration

4. Define variables in the playbook for application details and Nginx configuration

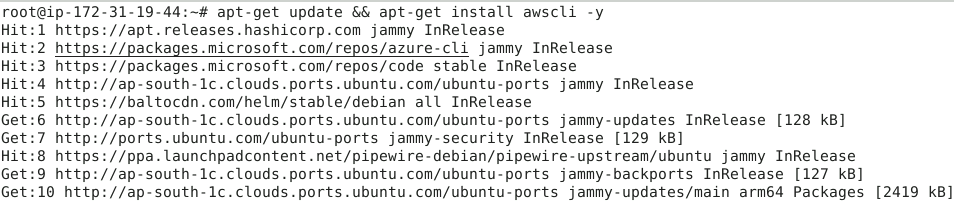
5. Include tasks in the playbook for installing Nginx, copying application files, deploying Nginx configuration, and enabling the Nginx site

6. Execute the playbook to deploy the web application on the remote server

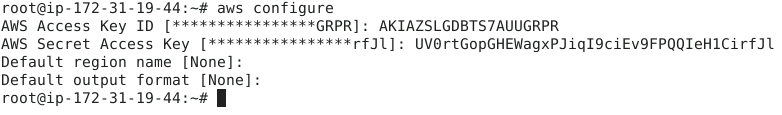
Solution

Step 1: Configure AWS CLI with access key and secret key to establish connection remotely

**# apt-get update && apt-get install awscli -y**



**# aws configure**



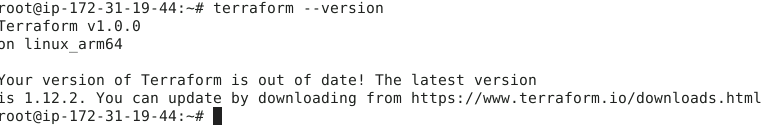
**Step 2: Install Terraform**

# wget -O - https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

# echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(grep -oP '(?<=UBUNTU\_CODENAME=).\*' /etc/os-release || lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

# sudo apt update && sudo apt install terraform

Check the terraform version by using the command “terraform –version”



**Step 3: Configure Terraform with new ssh key which will be used as key pair for launching VMs.**

# mkdir myproject

# cd myproject

# vim mykey.tf

provider "aws" {

region = "us-east-1"

}

resource "tls\_private\_key" "mykey" {

  algorithm = "RSA"

}

resource "aws\_key\_pair" "aws-key" {

  key\_name   = "web-key"

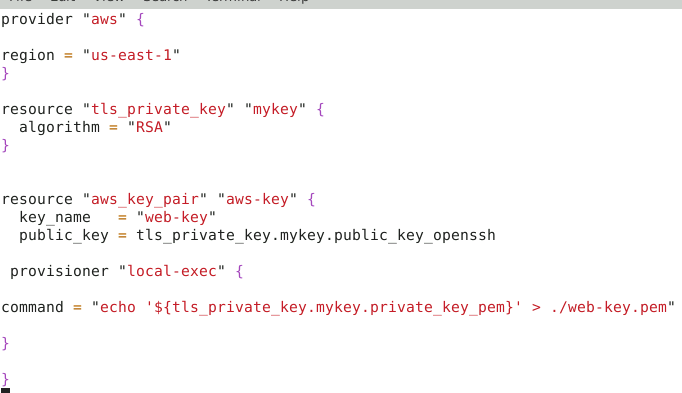
  public\_key = tls\_private\_key.mykey.public\_key\_openssh

 provisioner "local-exec" {

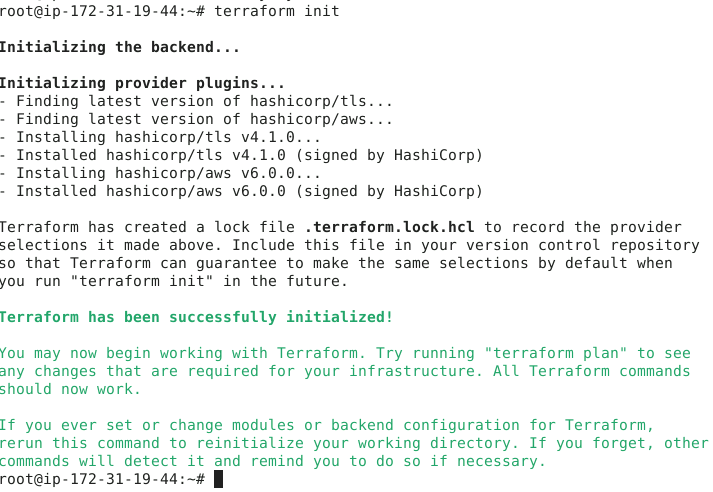
command = "echo '${tls\_private\_key.mykey.private\_key\_pem}' > ./web-key.pem"

}

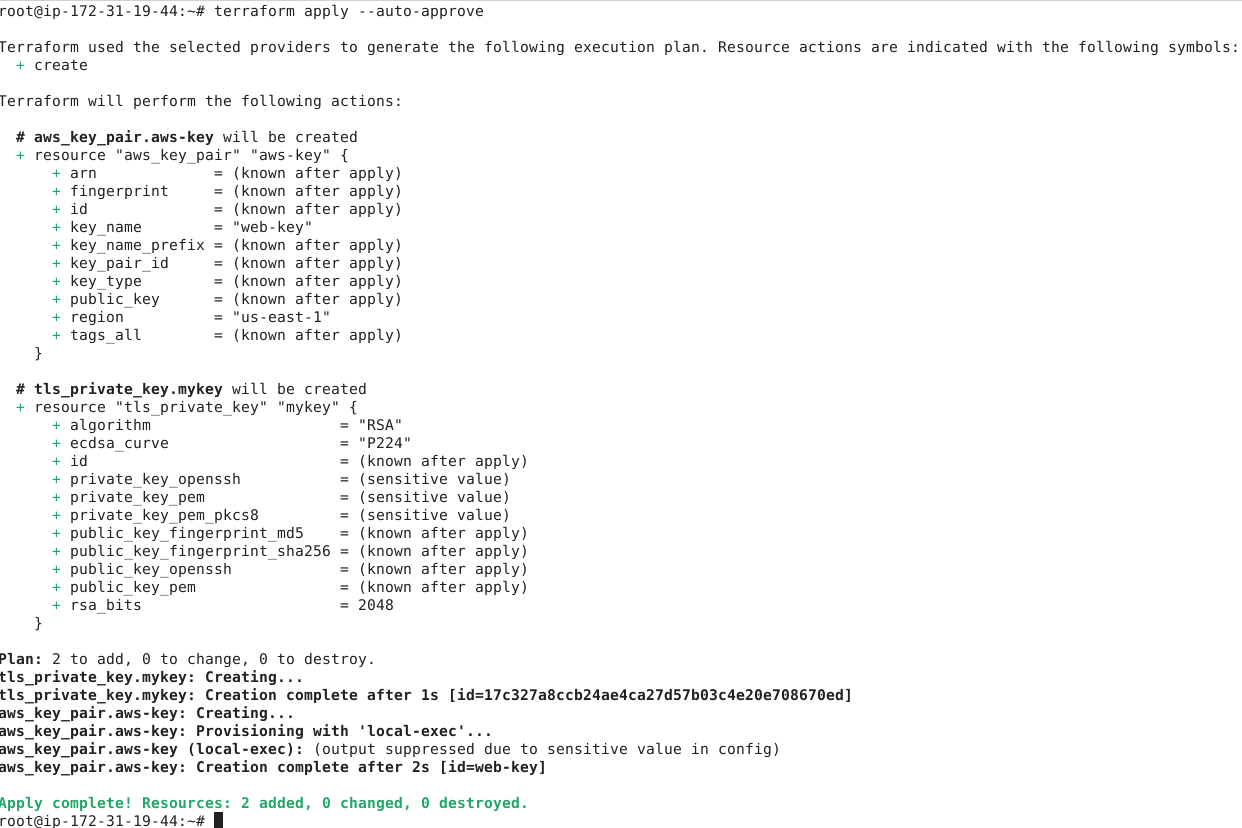
}



# Run the command “**terraform init**”



**#** Run the command **“terraform apply”**



**Step 4: Terraform script to provision and empty sandbox, add various setting to the sandbox like VPC, security group, route table, subnets, and key pair**

# vim main.tf

resource "aws\_vpc" "sl-vpc" {

cidr\_block = "10.0.0.0/16"

tags = {

Name = "sl-vpc"

}

}

resource "aws\_subnet" "sl-subnet" {

vpc\_id = aws\_vpc.sl-vpc.id

cidr\_block = "10.0.1.0/24"

depends\_on = [aws\_vpc.sl-vpc]

map\_public\_ip\_on\_launch = true

tags = {

Name = "sl-subnet"

}

}

resource "aws\_route\_table" "sl-route-table" {

vpc\_id = aws\_vpc.sl-vpc.id

tags = {

Name = "sl-route-table"

}

}

resource "aws\_route\_table\_association" "a" {

subnet\_id = aws\_subnet.sl-subnet.id

route\_table\_id = aws\_route\_table.sl-route-table.id

}

resource "aws\_internet\_gateway" "gw" {

vpc\_id = aws\_vpc.sl-vpc.id

depends\_on = [aws\_vpc.sl-vpc]

tags = {

Name = "sl-gw"

}

}

resource "aws\_route" "sl-route" {

route\_table\_id = aws\_route\_table.sl-route-table.id

destination\_cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.gw.id

}

variable "sg\_ports" {

type = list(number)

default = [22,443,80,8080]

}

resource "aws\_security\_group" "sl-sg" {

name = "sl-sg"

description = "Allow TLS inbound traffic and all outbound traffic"

vpc\_id = aws\_vpc.sl-vpc.id

dynamic "ingress" {

for\_each = var.sg\_ports

iterator = ports

content{

from\_port = ports.value

to\_port = ports.value

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

data "aws\_ami" "myami" {

most\_recent = true

owners = ["amazon"]

filter {

name = "name"

values = ["amzn2-ami-kernel-5.10-hvm-2.0.20250610.0-x86\_64-gp2"]

}

}

resource "aws\_instance" "myec2" {

ami = data.aws\_ami.myami.id

instance\_type = "t2.micro"

key\_name = "web-key"

subnet\_id = aws\_subnet.sl-subnet.id

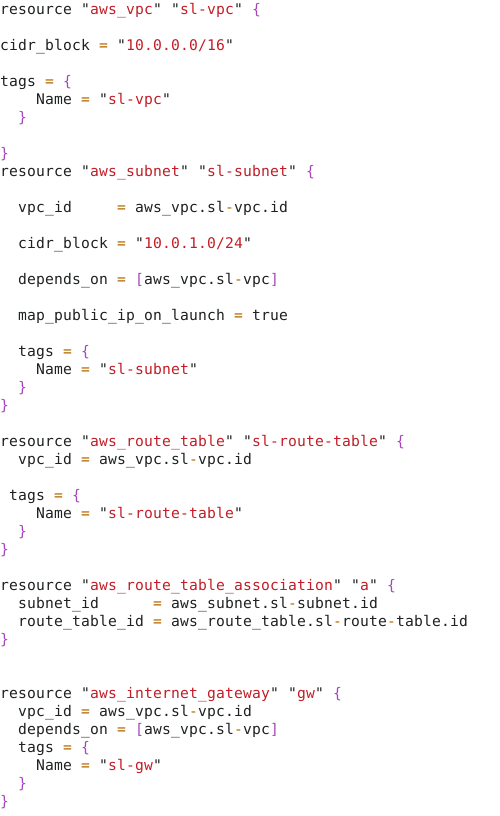
security\_groups = [aws\_security\_group.sl-sg.id]

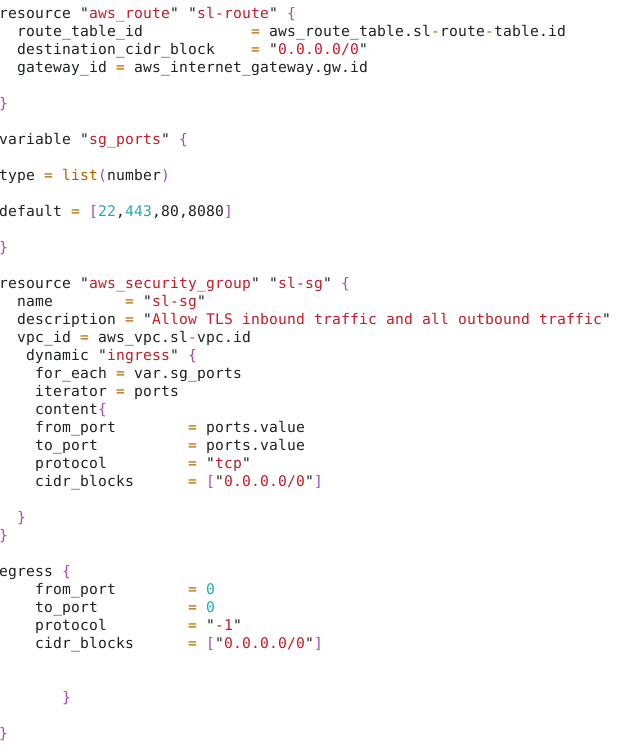
tags = {

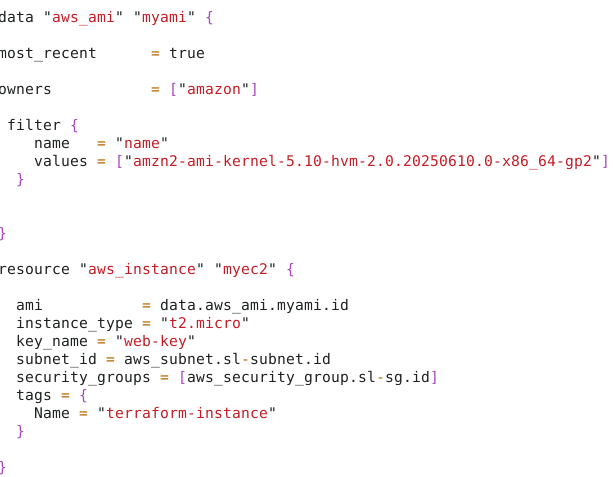
Name = "terraform-instance"

}

}

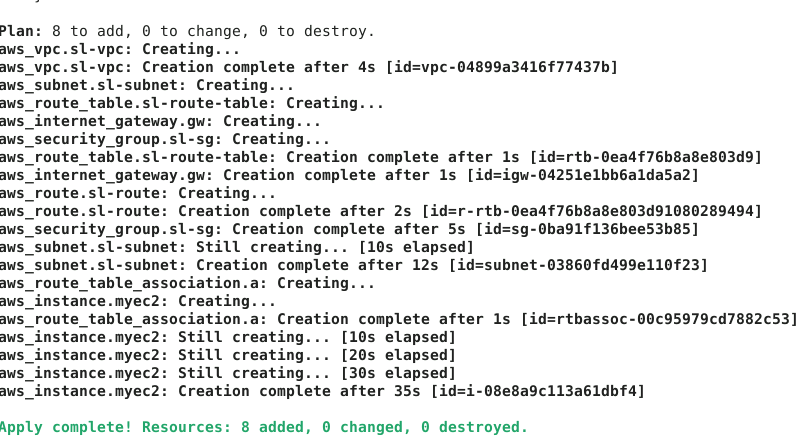




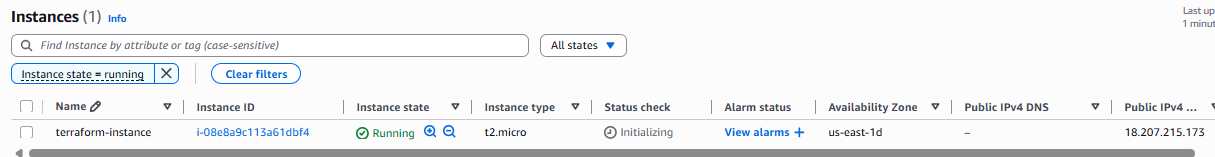


Save the file

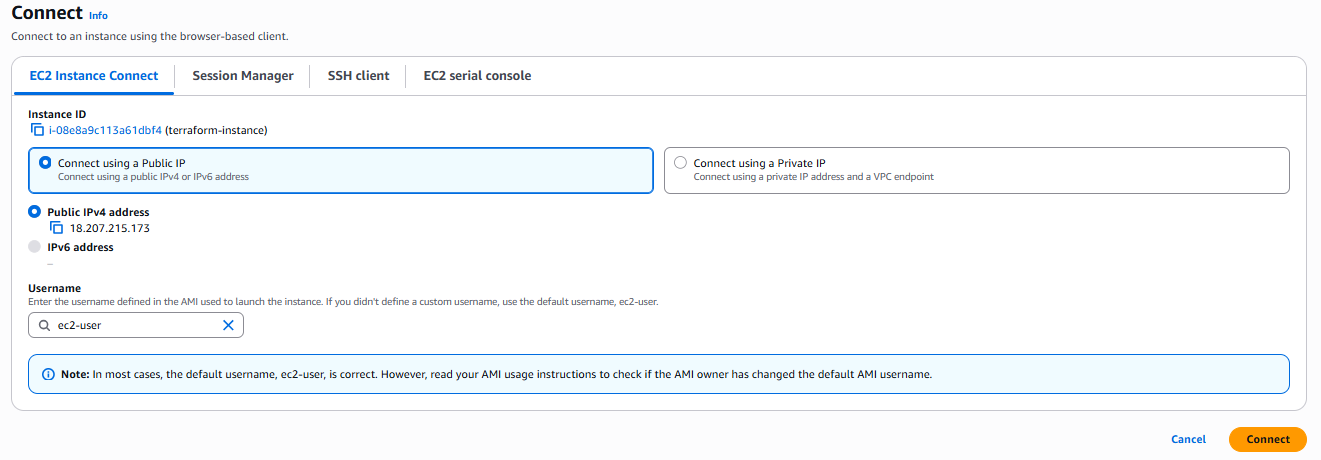
# Run the command “**terraform apply --auto-approve**”

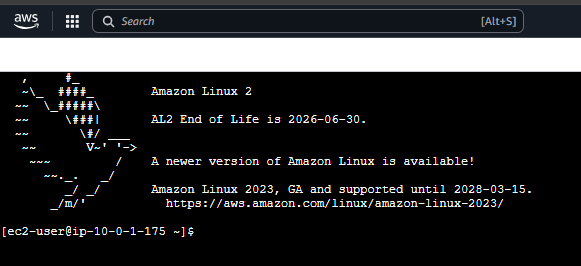


Login to AWS and go to EC2 instances.



Click on connect to connect the EC2 machine.





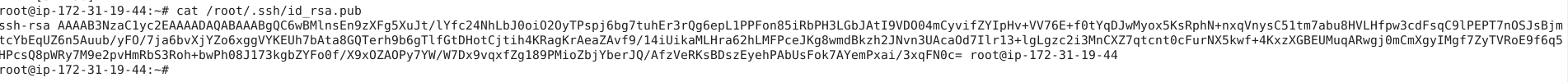
**Step 5: Ansible Setup for running the playbook on above created VM**

Generate ssh keys for root user on Lab terminal(Ansible Controller)

# ssh-keygen

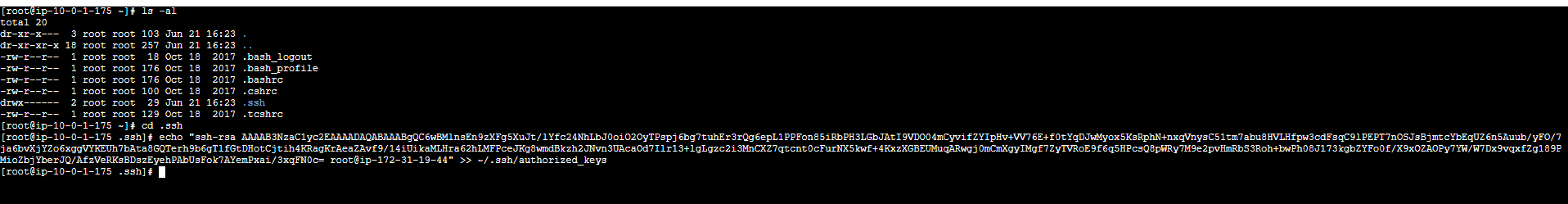


# cat /root/.ssh/id\_rsa.pub

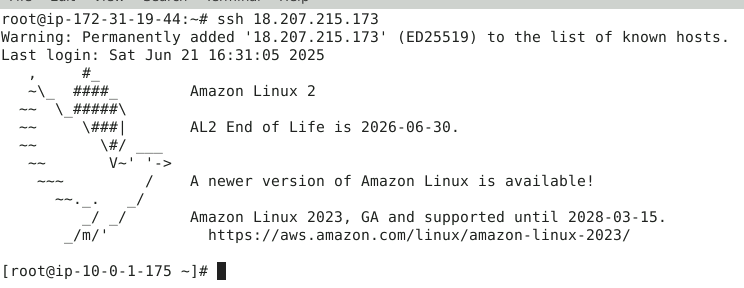


**GO to AWS EC2 server - Ansible worker**

**# cd .ssh**

**# echo "GIVE YOUR SSH PUBLIC KEY" >> ~/.ssh/authorized\_keys**

# Connect to the EC2 machine vis ssh in lab terminal



**Step 6: Create Ansible Inventory file with IP address of AWS EC2 server**

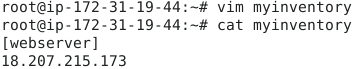
**In same directory where the terraform code is, create the ansible inventory**

# vim myinventory

[webserver]

<public ip of worker ec2 instance>

Save the file



**# pwd**

**Copy the path of the directory**

**# vim ansible.cfg**

[defaults]

inventory = /root/myproject/myinventory

**Save the file**

Run ping command to check the connection

Validate the setup :

# ansible webserver -m ping



**Step 7: Write the Ansible playbook.**

# vim playbook.yml

- name: run playbook using terraform

  hosts: webserver

  become: true

  tasks:

  - name: Install python3

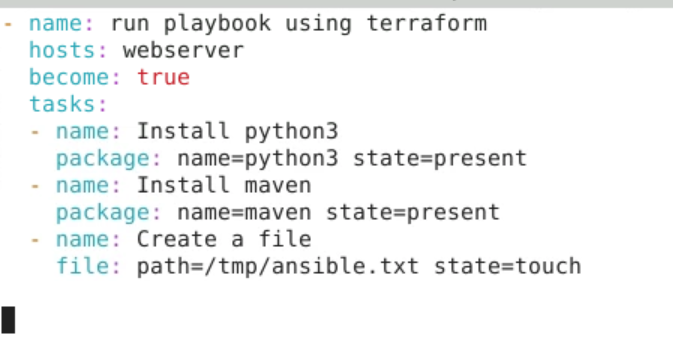
    package: name=python3 state=present

  - name: Install maven

    package: name=maven state=present

  - name: Create a file

    file: path=/tmp/ansible.txt state=touch



Save the file

**Step 8: Write Terraform code to run the playbook:**

# vim runplaybook.tf

resource "null\_resource" "run\_playbook" {

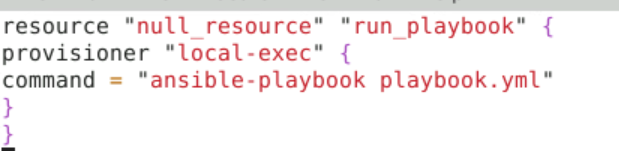
provisioner "local-exec" {

command = "ansible-playbook playbook.yml"

}

}

Save the file



**Step 9: Execute the playbook using terraform command**

# terraform init

# terraform apply -target null\_resource.run\_playbook

