Configuration Management with Ansible and Terraform

Course-End Project: Ansible playbook for deploying html code and configurations on NGINX server

Objective

To automate the provisioning of infrastructure and configuring it with integration of Terraform and Ansible.

Real-time scenario:

Royal Hotel is a globally leading chain of hotels. Recently, as part of scaling up operations, they aim to automate every operation in the hotel. For this, multiple applications are onboarded within all the hotel’s main server. To keep these applications up and running and to scale them appropriately, they need fully managed virtual machines on AWS. They want to have an automated provisioned infrastructure through which they can create a new developer VM and manage some developer configurations on that server.

Tasks

The following tasks outline the process of provisioning and infrastructure using Terraform and parameterised Jenkins job:

1. Configure Terraform with new ssh key which will be used as key pair for launching VMs.

2. Configure AWS CLI with access key and secret key to establish connection remotely

3. Write Terraform script to provision and empty sandbox.

4. Add various setting to the sandbox like VPC, security group, route table, subnets, and key pair.

5. Create Ansible playbook which will be invoked by Terraform for configuration management operations.

6. Executing Terraform script with the created keys to establish connection and configure the provisioned VM.

Solution

Step1: Create Inventory file to store worker details

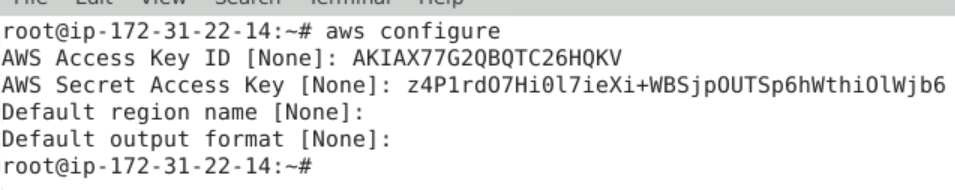
# cd

# ssh-keygen

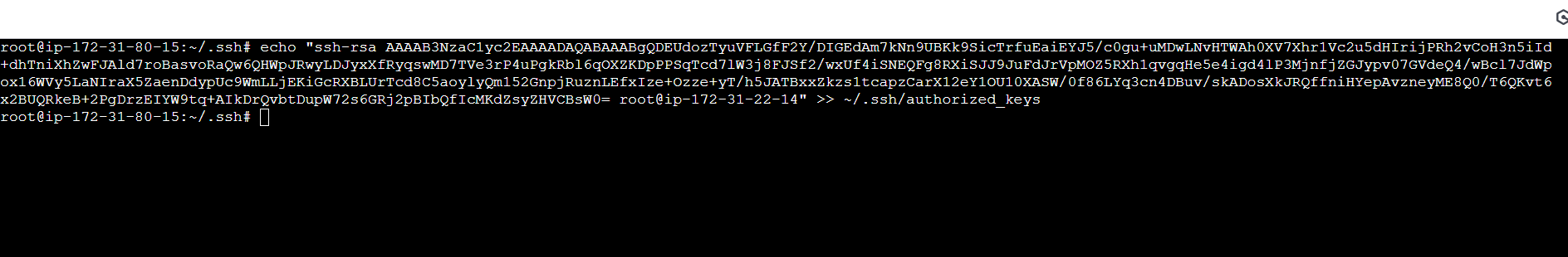
#  cat /root/.ssh/id\_rsa.pub



Step-2: Configure AWS.



Step-3: Create worker node in AWS EC2 and Copy the public key in .ssh folder



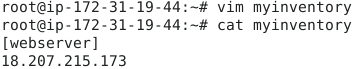
Step-4: Create the inventory file and ansible.cfg file and test the connection.

# vim myinventory

[webserver]

<public ip of worker ec2 instance>

Save the file



**# pwd**

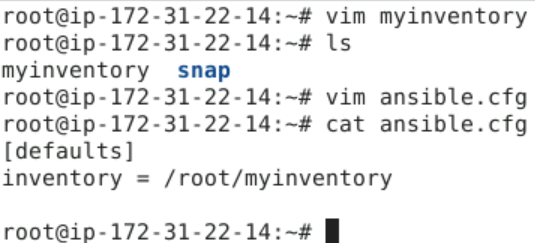
**Copy the path of the directory**

**# vim ansible.cfg**

[defaults]

inventory = /root/myinventory

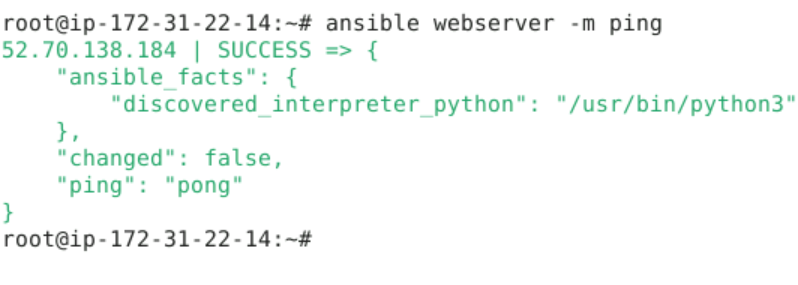
**Save the file**



Run ping command to check the connection

Validate the setup :

# ansible webserver -m ping



Step 5: Write a YAML playbook with tasks for installing Nginx

# vim playbook.yml

- name: Deploying project 1

  hosts: webserver

  become: true

  vars:

   pkg\_name: nginx

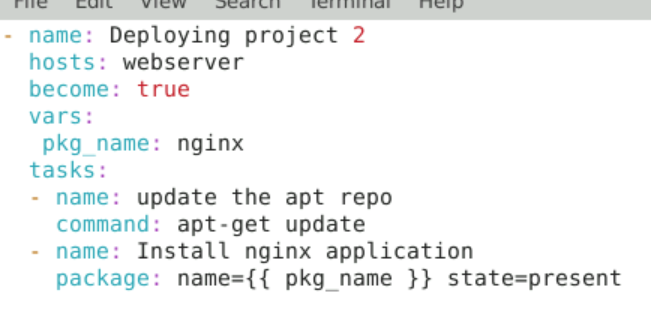
  tasks:

  - name: update the apt repo

    command: apt-get update

  - name: Install nginx application

    package: name={{ pkg\_name }} state=present



Save and run the playbook



Step 6: Update the nginx configuration using Jinja2 template

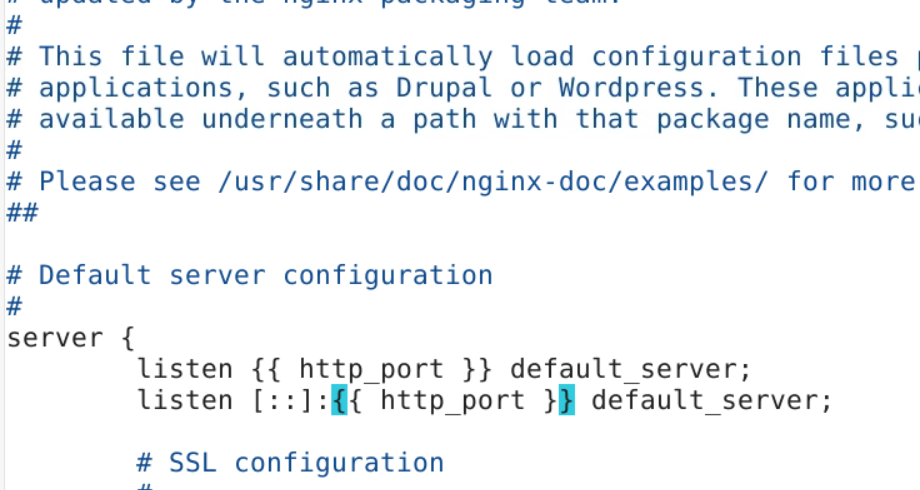
# cd

# cp /etc/nginx/sites-available/default .

#  mv default default.j2

# vim default.j2

Update port 80 with a variable {{ http\_port }} , as shown below



Step-7: Update the playbook with jina2 template tasks

# vim playbook.yml

- name: Deploying project 1

  hosts: webserver

  become: true

  vars:

   pkg\_name: nginx

   http\_port: 8282

  tasks:

  - name: update the apt repo

    command: apt-get update

  - name: Install nginx application

    package: name={{ pkg\_name }} state=present

  - name: Update the ports in nginx config file

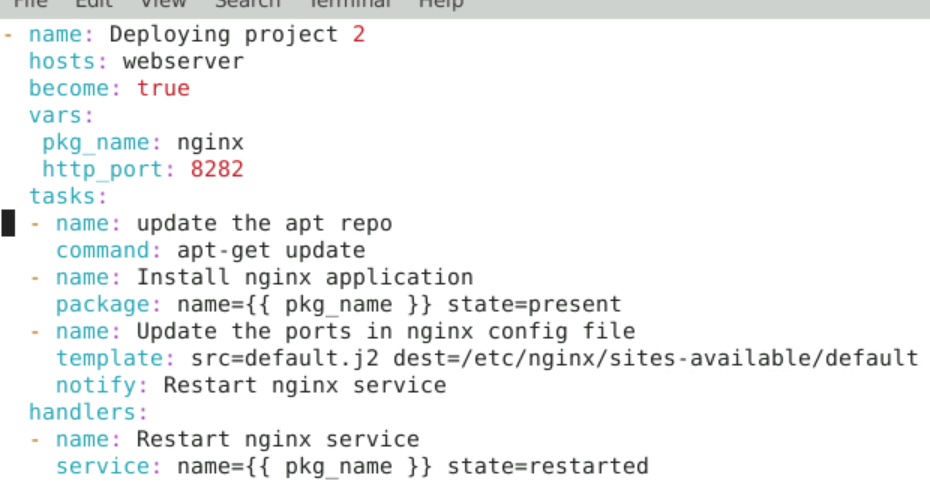
    template: src=default.j2 dest=/etc/nginx/sites-available/default

    notify: Restart nginx service

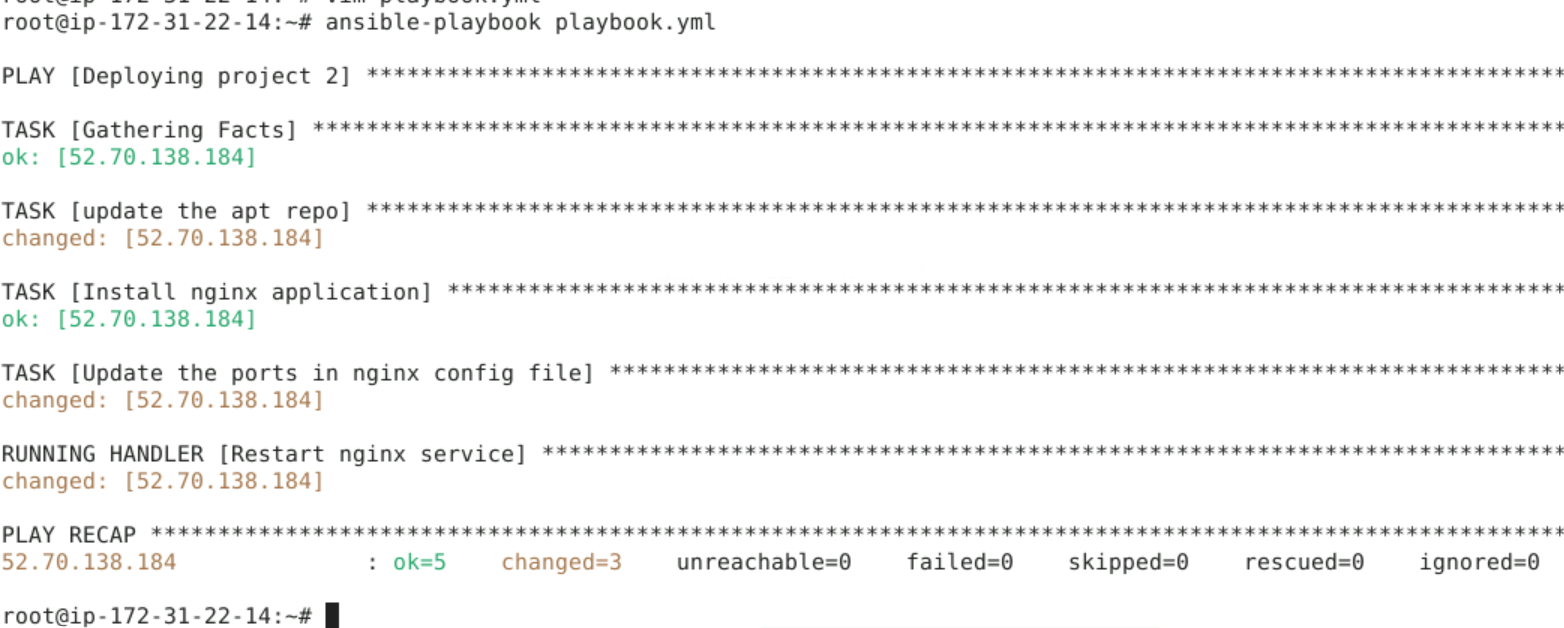
  handlers:

  - name: Restart nginx service

    service: name={{ pkg\_name }} state=restarted



Save the playbook and run it



Step-8: Create an HTML code and deploy it on nginx sever

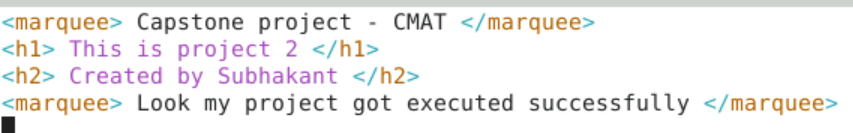
# vim index.html

<marquee> Capstone project - CMAT </marquee>

<h1> This is project 2 </h1>

<h2> Created by Subhakant </h2>

<marquee> Look my project got executed successfully </marquee>



Save the file

Update the playbook with the new tasks

# vim playbook.yml

- name: Deploying project 1

  hosts: webserver

  become: true

  vars:

   pkg\_name: nginx

   http\_port: 8282

  tasks:

  - name: update the apt repo

    command: apt-get update

  - name: Install nginx application

    package: name={{ pkg\_name }} state=present

  - name: Update the ports in nginx config file

    template: src=default.j2 dest=/etc/nginx/sites-available/default

    notify: Restart nginx service

  - name: Deploy HTML code on nginx server

    copy: src=index.html dest=/var/www/html

    notify: Restart nginx service

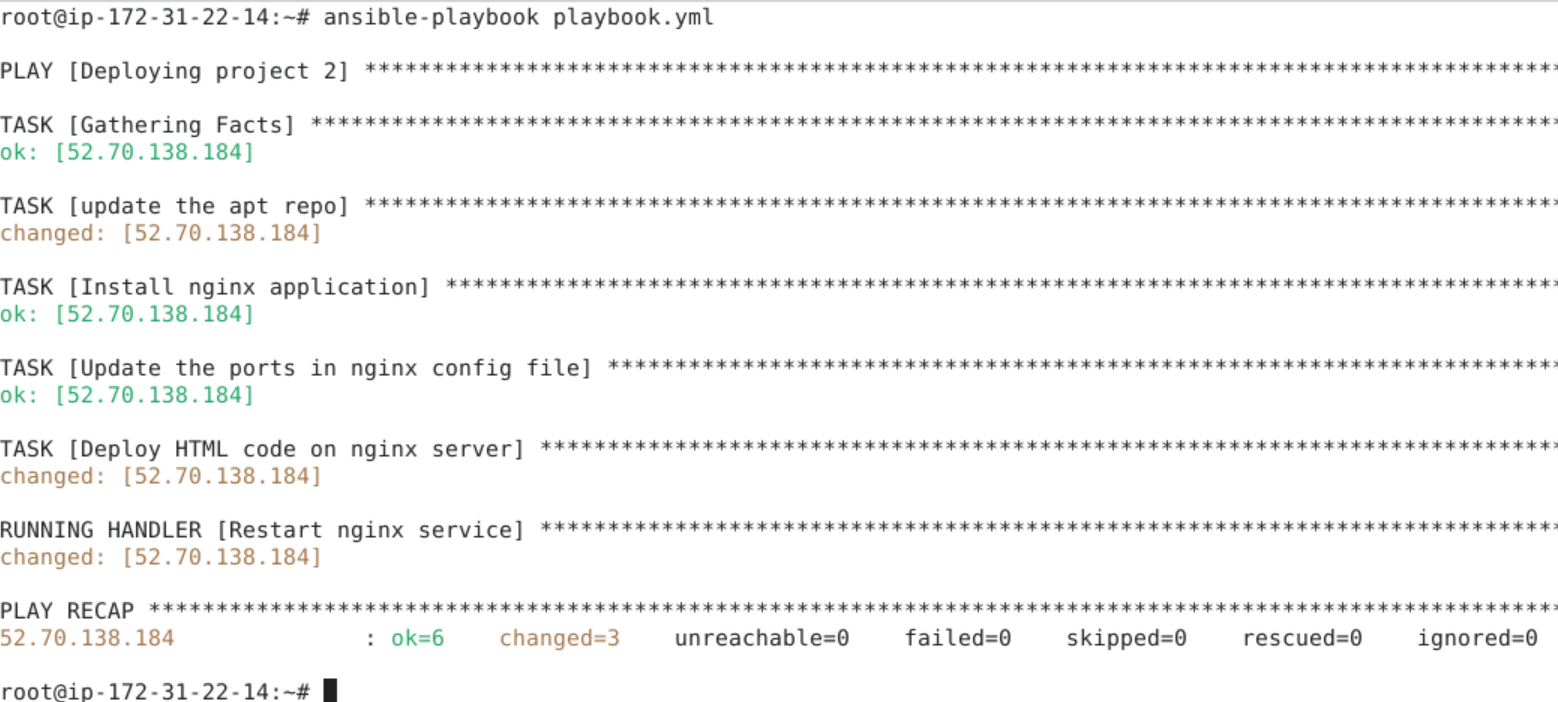
  handlers:

  - name: Restart nginx service

    service: name={{ pkg\_name }} state=restarted



Save the file and execute the playbook.



Go to the browser, take the public ip of worker node

Publicip:8282

