**Real-time Project**

**Automating AWS Infrastructure and Application Deployment**

***with***

**Ansible**

**Project Roadmap**

1. **Create the Ansible Server**
2. **Install Ansible**
3. **Install and Configure AWS CLI**
4. **Create Key Pair Using Ansible**
5. **Save the Key Pair Generated by Ansible**
6. **Create VPC Using Ansible**
7. **Create Frontend (FE) and Backend (BE) Servers Using Ansible**
8. **Create VPC Peering Connection**
9. **Deploy the Frontend and Backend Applications Using Ansible**

**Step-by-Step Instructions**

**1. Create Ansible Server (Bastion Host)**

* Launch an EC2 instance:
  + Name: Ansible
  + OS: Ubuntu
  + Security Group: Allow All Traffic for testing purposes (adjust later for production).
* Connect to the instance via MobaXterm or any SSH client.

**2. Install Ansible**

Run the following commands on the Ansible server:

sudo apt update && sudo apt -y full-upgrade

sudo apt-get install software-properties-common -y

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible -y

sudo apt install python3-pip -y

sudo pip3 install boto boto3

ansible --version

**3. Install and Configure AWS CLI**

Follow the official guide to install AWS CLI:

sudo apt install unzip -y

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

rm -rf aws awscliv2.zip

aws --version

Configure AWS CLI:

**aws configure**

**4. Create Key Pair Using Ansible**

* Create a playbook

vim create\_keypair.yaml

- name: Create EC2 Key Pair

hosts: localhost

connection: local

gather\_facts: false

tasks:

- name: Create EC2 Key Pair

ec2\_key:

name: my-ansible-keypair

region: us-east-1

state: present

register: keypair

- name: Save private key to file

copy:

content: "{{ keypair.key.private\_key }}"

dest: ~/my-ansible-keypair.pem

mode: 0400

Run the playbook:

ansible-playbook create\_keypair.yaml

**5. Save the Key Pair**

Ensure the private key is saved securely with proper file permissions:

chmod 400 ~/my-ansible-keypair.pem

**6. Create VPC Using Ansible**

Create the playbook vpc.yaml:

vim vpc.yaml

- hosts: localhost

connection: local

gather\_facts: false

tasks:

- name: Create VPC

ec2\_vpc\_net:

name: ansible-vpc

cidr\_block: 10.0.0.0/16

region: us-east-1

register: vpc\_result

- name: Create Internet Gateway

ec2\_vpc\_igw:

vpc\_id: "{{ vpc\_result.vpc.id }}"

state: present

register: igw

- name: Create Public Subnet for FE

ec2\_vpc\_subnet:

vpc\_id: "{{ vpc\_result.vpc.id }}"

cidr: 10.0.0.0/24

az: us-east-1a

map\_public: true

state: present

register: public\_subnet

- name: Create Private Subnet for BE

ec2\_vpc\_subnet:

vpc\_id: "{{ vpc\_result.vpc.id }}"

cidr: 10.0.1.0/24

az: us-east-1a

state: present

register: private\_subnet

- name: Create NAT Gateway

ec2\_vpc\_nat\_gateway:

subnet\_id: "{{ public\_subnet.subnet.id }}"

state: present

wait: true

register: nat

- name: Configure Public Route Table

ec2\_vpc\_route\_table:

vpc\_id: "{{ vpc\_result.vpc.id }}"

routes:

- dest: 0.0.0.0/0

gateway\_id: "{{ igw.gateway\_id }}"

subnets:

- "{{ public\_subnet.subnet.id }}"

register: public\_route\_table

- name: Configure Private Route Table

ec2\_vpc\_route\_table:

vpc\_id: "{{ vpc\_result.vpc.id }}"

routes:

- dest: 0.0.0.0/0

gateway\_id: "{{ nat.nat\_gateway\_id }}"

subnets:

- "{{ private\_subnet.subnet.id }}"

register: private\_route\_table

Run the playbook:

**ansible-playbook vpc.yaml**

**mkdir roles**

**cd roles**

**mkdir frontend**

**mkdir backend**

**mkdir vpc**

**cd vpc**

**mkdir tasks**

**cd tasks**

In the vpc directory create a tasks directory and move our **vpc.yaml**file there with the name **main.yaml**

**mv ../../../vpc.yaml main.yaml**

Edit the main.yaml file

**cat main.yaml**

cut the following part of the config out of the file then **delete** it.

**- hosts: localhost**

**connection: local**

**gather\_facts: False**

**tasks:**

**vim main.yaml**

- name: Create a VPC with dedicated tenancy and a couple of tags

ec2\_vpc\_net:

name: ansible-vpc

cidr\_block: 10.0.0.0/16

region: us-east-1

tenancy: dedicated

tags:

Name: ansible-vpc

register: vpc\_result

- name: Create Internet Gateway

ec2\_vpc\_igw:

vpc\_id: "{{ vpc\_result.vpc.id }}"

state: present

register: igw

- name: Create Public Subnet for FE servers

ec2\_vpc\_subnet:

state: present

vpc\_id: "{{ vpc\_result.vpc.id }}"

az: "us-east-1a"

cidr: 10.0.0.0/24

map\_public: true

tags:

Name: Public Subnet

register: public\_subnet

- name: Set up Public Subnet Route Table

ec2\_vpc\_route\_table:

vpc\_id: "{{ vpc\_result.vpc.id }}"

tags:

Name: Public RTB

subnets:

- "{{ public\_subnet.subnet.id }}"

routes:

- dest: 0.0.0.0/0

gateway\_id: "{{ igw.gateway\_id }}"

register: public\_route\_table

- name: Create Private Subnet for BE and DB servers

ec2\_vpc\_subnet:

state: present

vpc\_id: "{{ vpc\_result.vpc.id }}"

az: "us-east-1a"

cidr: 10.0.1.0/24

tags:

Name: Private Subnet

register: private\_subnet

- name: Create new NAT Gateway and allocate new EIP if a NAT Gateway does not yet exist in subnet

ec2\_vpc\_nat\_gateway:

state: present

subnet\_id: "{{ public\_subnet.subnet.id }}"

wait: true

if\_exist\_do\_not\_create: true

register: nat

- name: Set up Private Subnet Route Table

ec2\_vpc\_route\_table:

vpc\_id: "{{ vpc\_result.vpc.id }}"

tags:

Name: Private RTB

subnets:

- "{{ private\_subnet.subnet.id }}"

routes:

- dest: 0.0.0.0/0

gateway\_id: "{{ nat.nat\_gateway\_id }}"

register: private\_route\_table

- name: Create Security Group with Rule Descriptions

ec2\_group:

name: Ansible VPC SG

description: SG with rule descriptions

vpc\_id: "{{ vpc\_result.vpc.id }}"

rules:

- proto: tcp

from\_port: 0

to\_port: 65535

cidr\_ip: 0.0.0.0/0

rule\_desc: allow all

register: sg

**---**

**cd ~**

**vim playbook.yaml**

- hosts: localhost

connection: local

gather\_facts: False

roles:

- vpc

**ansible-playbook playbook.yaml**

Create FE and BE servers with Ansible

In the **roles/frontend** directory create **tasks** directory and create a **main.yaml**file in there

**cd roles/frontend**

**mkdir tasks**

**cd tasks**

*mv ../../../aws-ec2.yaml main.yaml*

**vim main.yaml**

- name: Create EC2 KeyPair

amazon.aws.ec2\_key:

name: ansible-ec2-keypair

no\_log: true

register: key\_out

- name: Show the Keypair contents

debug:

var: key\_out.key.private\_key

when: key\_out.changed

- name: Save the Private Key

copy:

content: "{{ key\_out.key.private\_key }}"

dest: ansible-key.pem

mode: '0600'

when: key\_out.changed

- name: Create Security Group

amazon.aws.ec2\_group:

name: ansible-sg

description: "Security group for Ansible-created EC2 instance"

vpc\_id: "{{ vpc\_result.vpc.id }}"

rules:

- proto: tcp

ports:

- 22

cidr\_ip: 0.0.0.0/0

- proto: tcp

ports:

- 80

cidr\_ip: 0.0.0.0/0

register: sg

- name: Start an instance with a public IP address

amazon.aws.ec2\_instance:

name: "frontend"

key\_name: "{{ key\_out.key.name }}"

vpc\_subnet\_id: "{{ public\_subnet.subnet.id }}"

instance\_type: t2.micro

security\_group: "{{ sg.group\_id }}"

network:

assign\_public\_ip: true

image\_id: ami-0c7217cdde317cfec

**cd**

**vim playbook.yaml**

- hosts: localhost

connection: local

gather\_facts: False

roles:

- vpc

- frontend

**ansible-playbook playbook.yaml**

A frontend instance should be created with all the specified config

Go to the **roles/frontend** directory and copy the tasks directory into the backend. Then edit the main.yaml file in the backend/tasks directory

**cd roles/frontend**

**cp -r tasks/ ../backend**

**cd ../backend**

**cd tasks**

**vim main.yaml**

- name: Create EC2 KeyPair

amazon.aws.ec2\_key:

name: ansible-ec2-keypair

no\_log: true

register: key\_out

- name: Show the Keypair contents

debug:

var: key\_out.key.private\_key

when: key\_out.changed

- name: Save the Private Key

copy:

content: "{{ key\_out.key.private\_key }}"

dest: ansible-key.pem

mode: '0600'

when: key\_out.changed

- name: Start a BE instance with a public IP address

amazon.aws.ec2\_instance:

name: "Backend"

key\_name: "{{ key\_out.key.name }}"

vpc\_subnet\_id: "{{ private\_subnet.subnet.id }}"

instance\_type: t2.micro

security\_group: "{{ sg.group\_id }}"

network:

assign\_public\_ip: true

image\_id: ami-0c7217cdde317cfec

**cd**

**vim playbook.yaml**

Insert the following config with the backend as a newly added role

**- hosts: localhost  
  connection: local  
  gather\_facts: False  
  roles:**

**- vpc**

**- frontend**

**- backend**

**---**

**ansible-playbook playbook.yaml**

Create VPC Peering connection

**mkdir infra deployment**

Move the **playbook.yaml** file and **roles** directory into the **infra**directory. Then move the **ansible-key.pem**into the deployment directory

**mv playbook.yaml roles/ infra**

**mv ansible-key.pem deployment/**

Go to Peering connections in AWS console, and **Create peering connection** in order to establish connection between two different VPCs, give name = **aws-ansible-peer,** select default vpc

Select the **ansible-vpc**as Accepter. Create the peering connection

Accept the request from the **Action**drop-down

Select the route table of **default** VPC and edit the routes

Add a **route** with the **ansible-vpc**CIDR as the destination and newly created peering connection as the Target.

Save the changes.

Select the route table of ansible-vpc and edit the routes

Add a route with the **default vpc**CIDR as the destination and the newly created peering connection as the Target.

Save the changes.

Select the **Private** RTB of the **ansible**-vpc and edit the routes

Add a route with the **default-vpc**CIDR as the destination and newly created peering connection as the Target.

Save the changes.

Select the **Public** RTB of the **ansible**-vpc and edit the routes

Add a route with the **default vpc**CIDR as the destination and the newly created peering connection as the Target.

Save the changes.

Try to telnet to the Backend server private IP address. If telnet is successful, the VPC peering works properly

**telnet Backend\_private\_IP 22**

Deploy the FE and BE app with Ansible

**cd deployment**

**vim hosts** …. Inventory file

[frontend]

frontend\_IP

[backend]

backend\_IP

[frontend:vars]

ansible\_ssh\_private\_key\_file=ansible-key.pem

ansible\_ssh\_user=ubuntu

[backend:vars]

ansible\_ssh\_private\_key\_file=ansible-key.pem

ansible\_ssh\_user=ubuntu

**vim ansible.cfg**

[defaults]

host\_key\_checking = False

inventory = hosts

**vim playbook.yaml**

- hosts: frontend

tasks:

- name: Update apt

become: yes

apt:

update\_cache: yes

- name: Install node and npm

become: yes

apt:

pkg:

- nodejs

- npm

**ansible-playbook playbook.yaml** see facts gathering from frontend server is successful or not!

Create a roles directory and in it, create backend/tasks and frontend/tasks directory

**mkdir roles**

**cd roles**

**mkdir -p backend/tasks**

**mkdir -p frontend/tasks**

Create and edit main.yaml file in the backend/tasks directory

**vim backend/tasks/main.yaml**

- name: Clone git repo for backend

git:

repo: https://github.com/azizulmaqsud/two-tier-app.git

dest: /home/ubuntu/two-tier

clone: yes

update: yes

- name: Update apt cache

become: yes

apt:

update\_cache: yes

- name: Install python and pip

become: yes

apt:

pkg:

- python3-pip

- name: Install requirements

pip:

executable: pip3

requirements: /home/ubuntu/two-tier/backend/requirements.txt

- name: Setup Flask Path

shell: echo $PATH

environment:

PATH: "/home/ubuntu/.local/bin:{{ ansible\_env.PATH }}"

- name: Run the backend

command:

chdir: /home/ubuntu/two-tier/backend

cmd: /home/ubuntu/.local/bin/flask run --host 0.0.0.0

async: 1000

poll: 0

**cd ..**

**vim playbook.yaml**

- hosts: frontend

roles:

- frontend

- hosts: backend

roles:

- backend

In the frontend directory create a **files** directory and create a setupProxy.js file

**cd roles**

**cd frontend**

**mkdir files**

**cd files**

**vim setupProxy.js**

const { createProxyMiddleware } = require('http-proxy-middleware');

module.exports = app => {

app.use(

"/reverser",

createProxyMiddleware({

target: "http://10.0.1.223:5000",

changeOrigin: true

})

);

app.use(

"/summation",

createProxyMiddleware({

target: "http://10.0.1.223:5000",

changeOrigin: true

})

);

};

1. Import the createProxyMiddleware function from the http-proxy-middleware module.
2. Set up proxy middleware for two routes in an Express app (/reverser and /summation).
3. Forward requests to these routes to [http://backend:5000](http://10.0.1.223:5000/).
4. Modify the origin header to match the target's host (changeOrigin: true).

**cd ../tasks**

**vim main.yaml**

- name: Clone git repo for frontend

git:

repo: https://github.com/azizulmaqsud/two-tier-app.git

dest: /home/ubuntu/two-tier

clone: yes

update: yes

ignore\_errors: true

- name: Update apt cache

become: yes

apt:

update\_cache: yes

- name: Install node and npm

become: yes

apt:

pkg:

- nodejs

- npm

- name: Install dependencies

npm:

path: /home/ubuntu/two-tier/frontend

- name: Copy updated setupProxy file

copy:

src: files/setupProxy.js

dest: /home/ubuntu/two-tier/frontend/src/setupProxy.js

- name: Run the application

command:

cmd: npm start

chdir: /home/ubuntu/two-tier/frontend

async: 1000

poll: 0

1. **Clone Frontend Repository**: Clone a specific frontend Git repository, ignoring any errors during the process.
2. **Refresh Package Index**: Update the package index on the target machine using apt.
3. **Install Node.js and npm**: Use apt to install Node.js and npm with elevated privileges.
4. **Install Project Dependencies**: Run npm to install dependencies located in a certain path.
5. **Copy Proxy Configuration**: Transfer a setupProxy.js file to the correct location in the project directory.
6. **Start Frontend Server**: Launch the frontend application using npm start, running asynchronously without continuous checking (polling).

**ansible-playbook playbook.yaml**

**curl <backend pvt ip>:5000**

Visit the **frontend\_server\_IP:3000** address and see two-tier application in action

This project can be an excellent real-world demonstration of automating AWS infrastructure and application deployment with Ansible.

Thank you