**CAPESTONE PROJECT-1**

**POST GRADUATE PROGRAM IN DEVOPS**

**NAME: TATA SAIKRISHNA**

**MAIL:dcgm.crmr@gmail.com**

**Mobile:7382801892**

**Registration ID:731272**

**https://github.com/tatasaikrishna/devops-capstone-project.git**

**Infra Optimization.**

Course-end Project 1

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**  
A popular payment application, **EasyPay**where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with  
the connectivity of the database has been the reason for the issue.  
While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.  
Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**The following tools must be used:**

1. EC2
2. Kubernetes
3. Docker
4. Ansible or Chef or Puppet

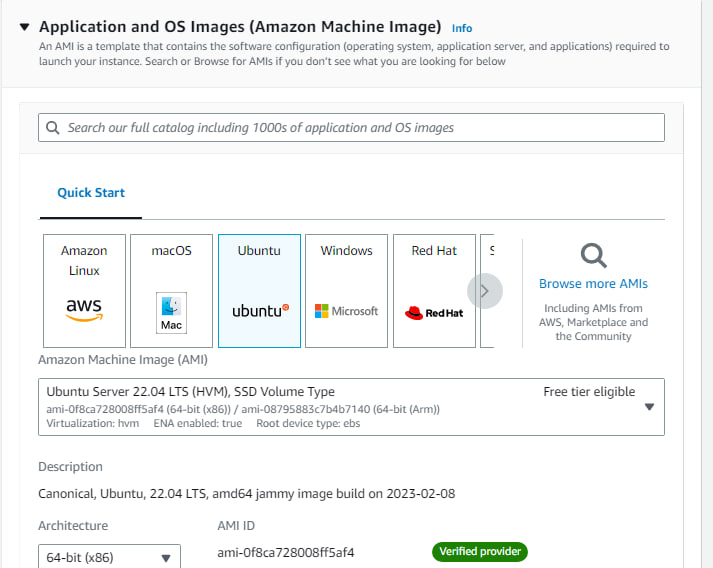
**The following things to be kept in check:**

1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and recording the results.
4. You need to submit the final specification document, which includes:

* Project and tester details
* Concepts used in the project
* Links to the GitHub repository to verify the project completion
* Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

**2. To automate the provisioning of an EC2 instance using Ansible**

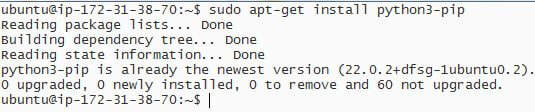
**Step:1**

****

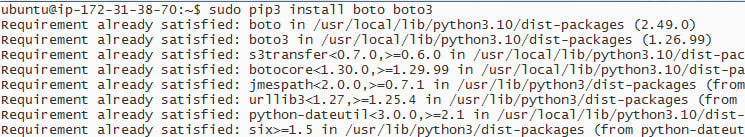
**$ sudo apt-get update**

****

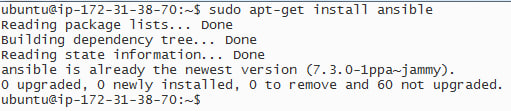
**$sudo apt-get install python3-pip**

****

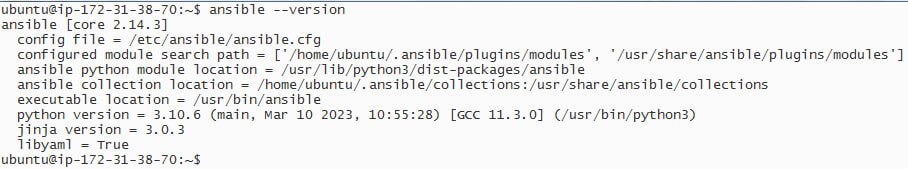
**$sudo pip3 install boto boto3**

****

**$sudo apt-get install ansible**

****

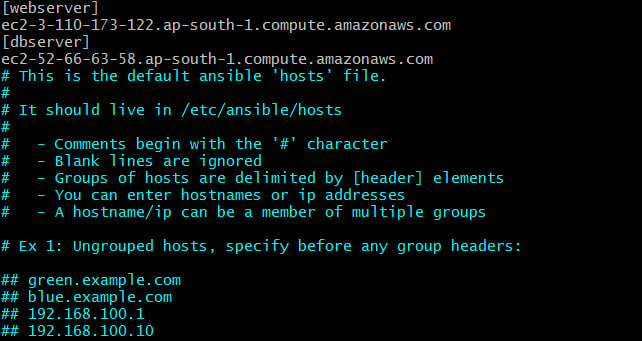
**$ansible –version**

****

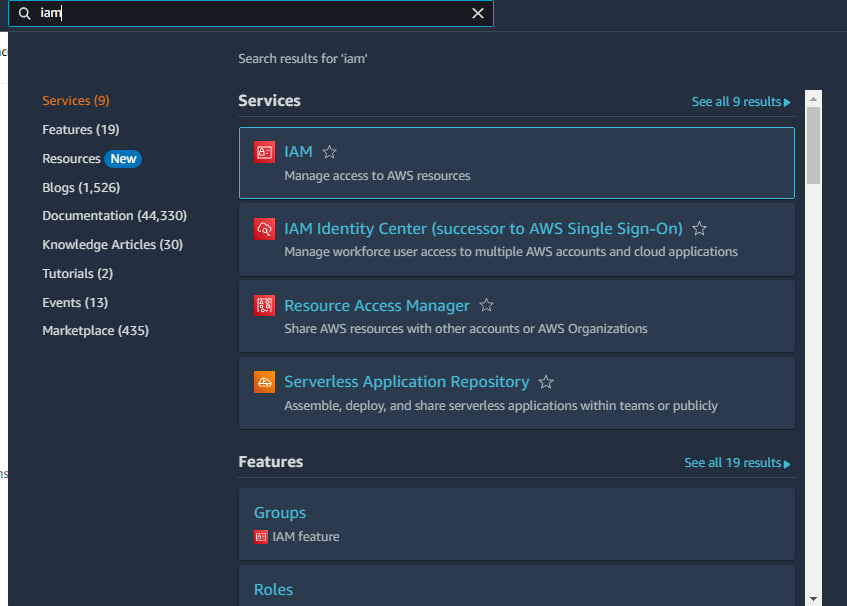
**Step:2**

**example inventory file in YAML format**

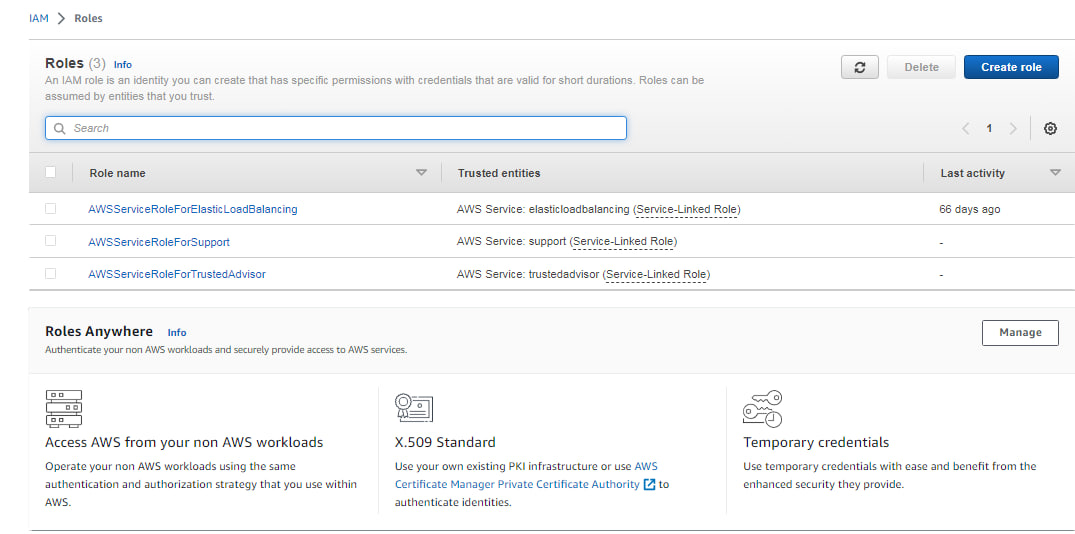
**$vim /etc/ansible/hosts**

****

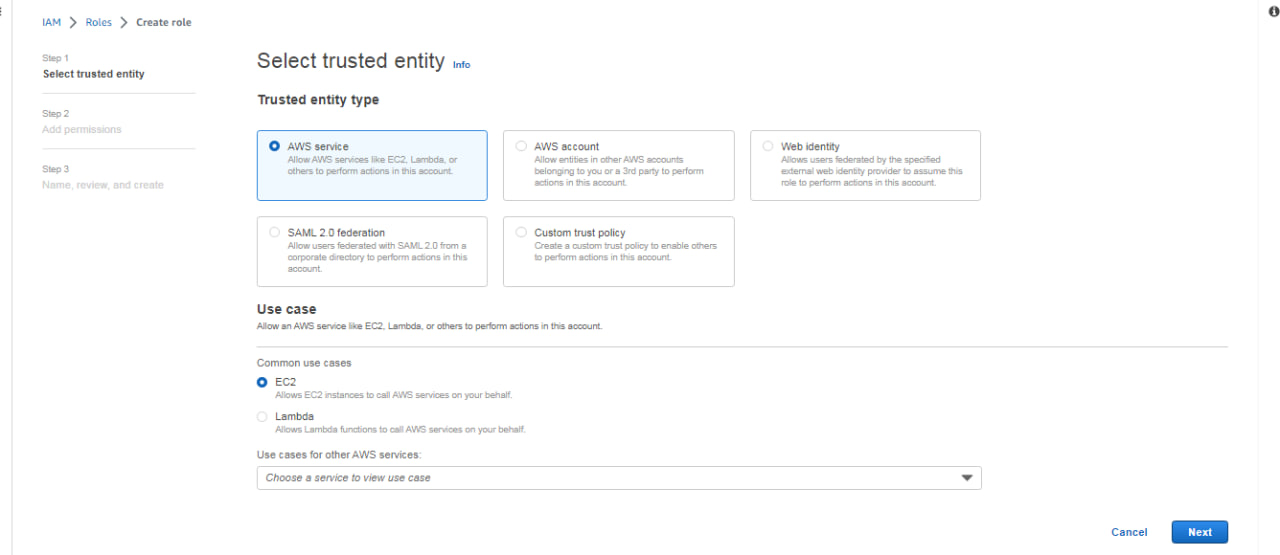
**Create IAM role and attach to instance**

****

**Click on Creae role**

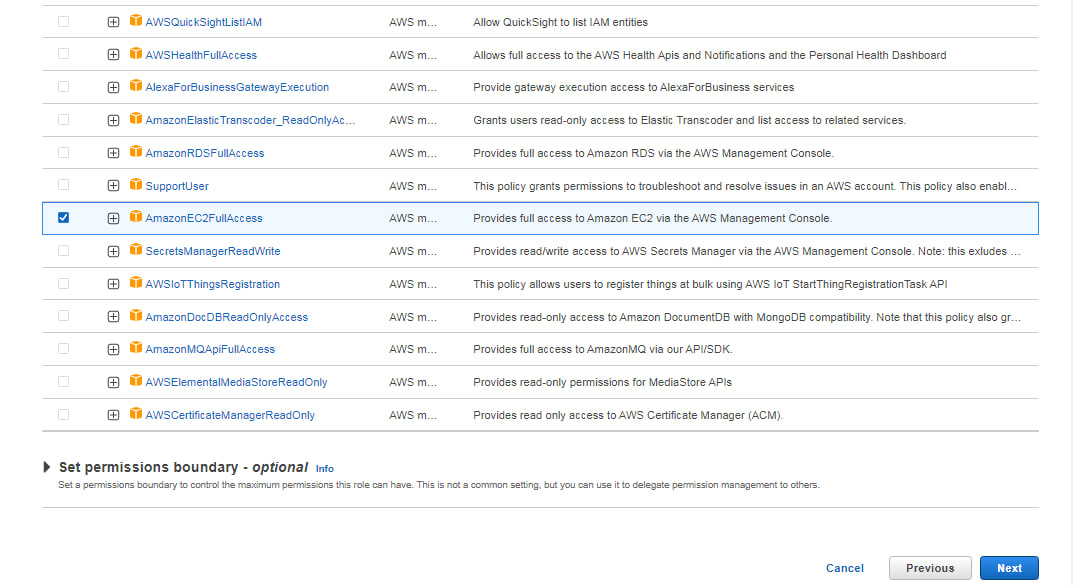
****

**Select EC2 and press Next**

****

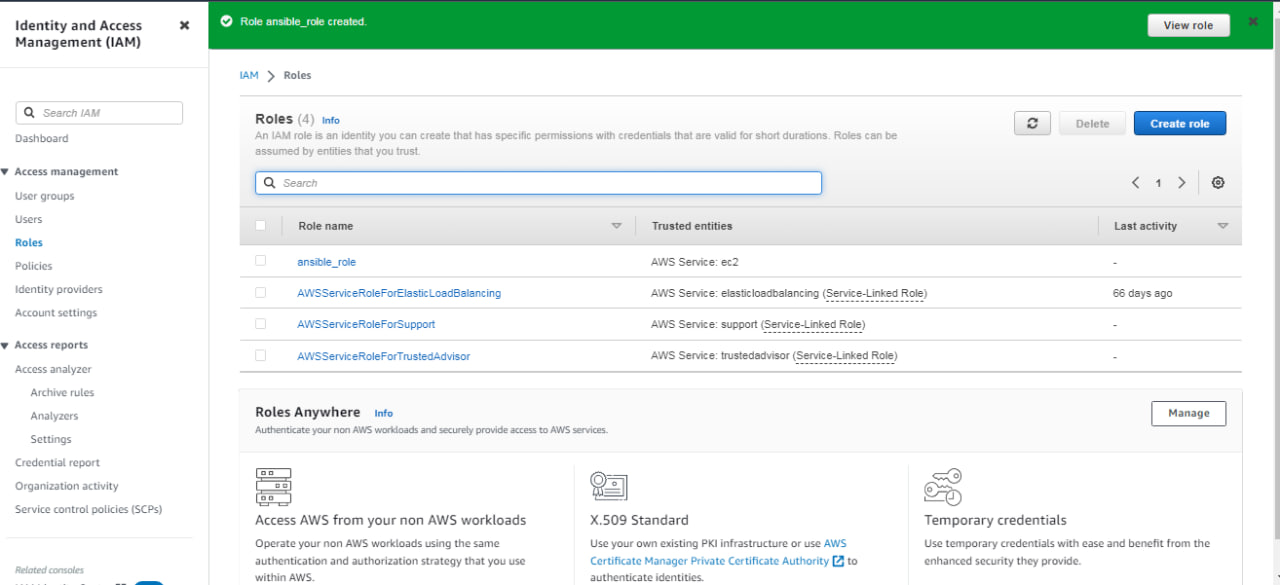
**Select AmazonEc2FullAccess**

**Click Next**

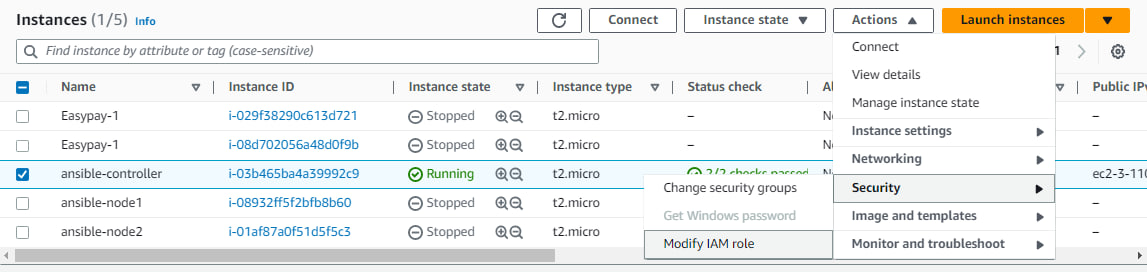
****

**Give Name: ansible\_role**

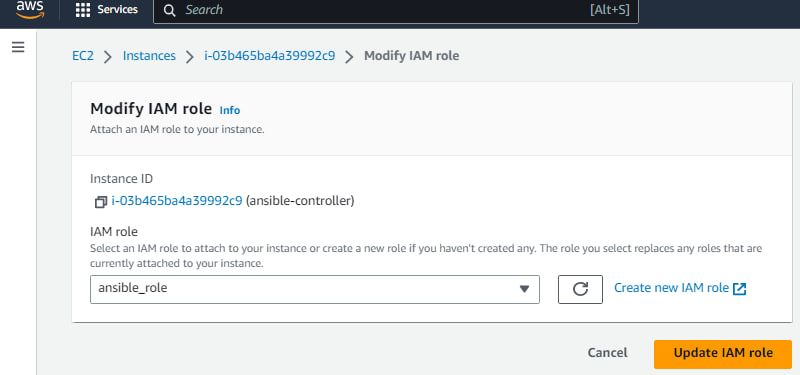
**Click on Create**

****

**Go to Instances and Assign IAM role**

****

**Click on update IAM role**

****

1. **Ansible playbook that creates an EC2 instance on AWS and performs some basic configuration tasks on it:**

$vim ansible-ec2.yaml

---

- name: Create and Configure EC2 Instance

hosts: localhost

connection: local

gather\_facts: False

vars:

key\_name: Mumbai

region: ap-south-1a

instance\_type: t2.micro

ami: ami-0f8ca728008ff5af4

security\_group: easypay-1

tasks:

- name: Launching EC2 Instance

ec2:

key\_name: "{{ key\_name }}"

region: "{{ region }}"

instance\_type: "{{ instance\_type }}"

image: "{{ ami }}"

wait: yes

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

count: 1

instance\_tags:

Name: my\_instance

register: ec2

- name: Add the new instance to the host group

add\_host:

name: "{{ ec2.instances[0].public\_dns\_name }}"

groups: launched

- name: Wait for SSH to come up

wait\_for:

host: "{{ ec2.instances[0].public\_dns\_name }}"

port: 22

delay: 60

timeout: 320

- name: Update the apt package index

become: yes

become\_user: ubuntu

apt:

update\_cache: yes

- name: Install nginx web server

become: yes

become\_user: ubuntu

apt:

name: nginx

state: latest

- name: Start nginx web server

become: yes

become\_user: ubuntu

service:

name: nginx

state: started



