**Infrastructure Optimization**

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:**

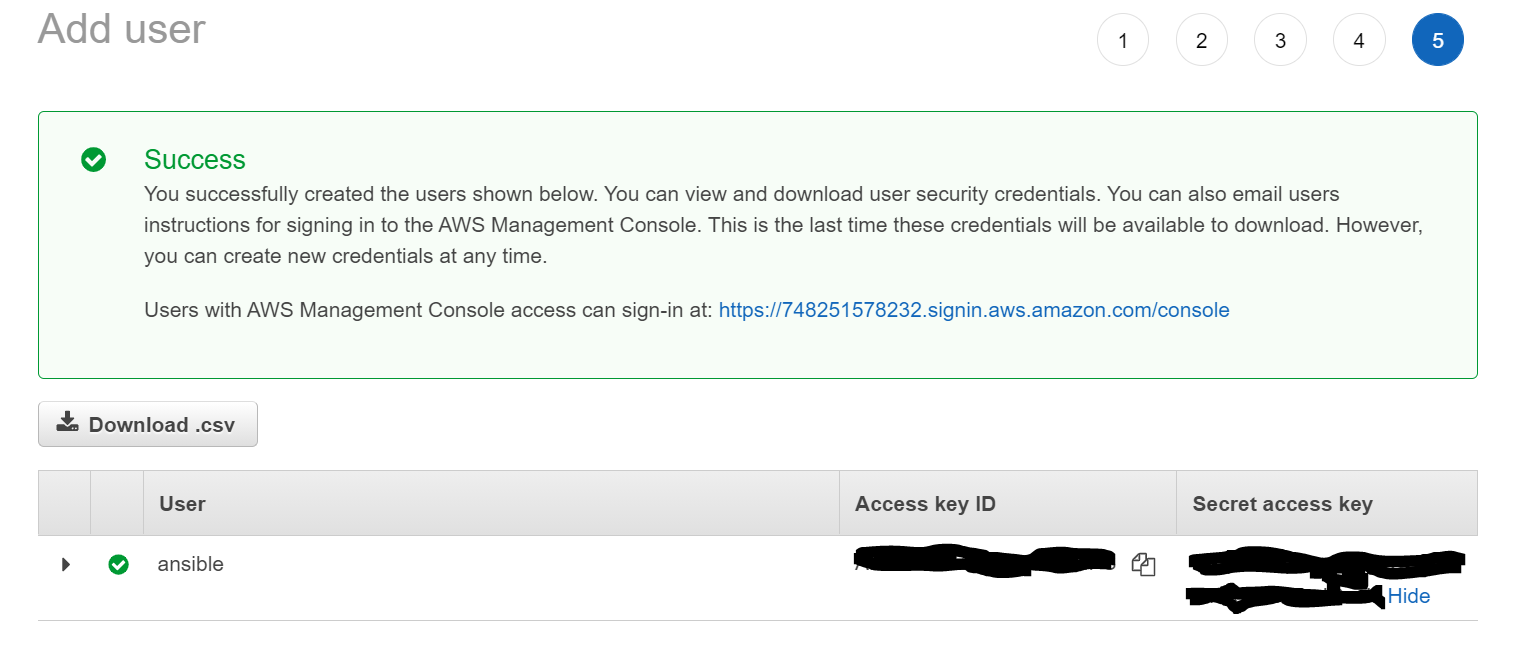
1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and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

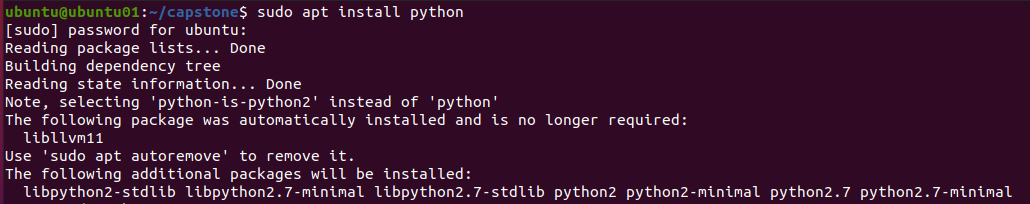
**Prerequisites:** kubeadm and kubectl should be installed

**Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)**

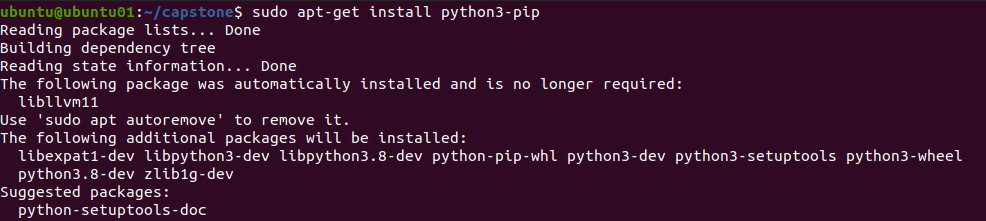
Step 1 - Create AWS IAM User for provisioning through anisible:



Step 2 - Install python



Step 3 - Install python3-pip



Step 4 - Install ansible



**Automate the provisioning of an EC2 instance using Ansible or Chef Puppet**

Step 1 - Configure ansible “localhost”





Step 2 - Create ansible playbook to create ec2 nodes: playbook.html

# AWS playbook

---

- name: Create ec2 instances

hosts: localhost

connection: local

gather\_facts: False

vars:

key\_name: easypay

region: us-east-2

image: ami-0f93b5fd8f220e428

id: "payment-app"

sec\_group: "{{ id }}-sec"

tasks:

- name: Facts

block:

- name: Get instances facts

ec2\_instance\_facts:

aws\_access\_key: "{{ec2\_access\_key}}"

aws\_secret\_key: "{{ec2\_secret\_key}}"

region: "{{ region }}"

register: result

- name: Instances ID

debug:

msg: "ID: {{ item.instance\_id }} - State: {{ item.state.name }} - Public DNS: {{ item.public\_dns\_name }}"

loop: "{{ result.instances }}"

tags: always

- name: Provisioning EC2 instances

block:

- name: Upload public key to AWS

ec2\_key:

name: "{{ key\_name }}"

key\_material: "{{ lookup('file', '/home/ubuntu/.ssh/{{ key\_name }}.pub') }}"

region: "{{ region }}"

aws\_access\_key: "{{ec2\_access\_key}}"

aws\_secret\_key: "{{ec2\_secret\_key}}"

- name: Create security group

ec2\_group:

name: "{{ sec\_group }}"

description: "Sec group for app {{ id }}"

# vpc\_id: 12345

region: "{{ region }}"

aws\_access\_key: "{{ec2\_access\_key}}"

aws\_secret\_key: "{{ec2\_secret\_key}}"

rules:

- proto: tcp

ports:

- 22

cidr\_ip: 0.0.0.0/0

rule\_desc: allow all on ssh port

register: result\_sec\_group

- name: Provision instance(s)

ec2:

aws\_access\_key: "{{ec2\_access\_key}}"

aws\_secret\_key: "{{ec2\_secret\_key}}"

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

group\_id: "{{ result\_sec\_group.group\_id }}"

image: "{{ image }}"

instance\_type: t2.micro

region: "{{ region }}"

wait: true

count: 3

# exact\_count: 2

# count\_tag:

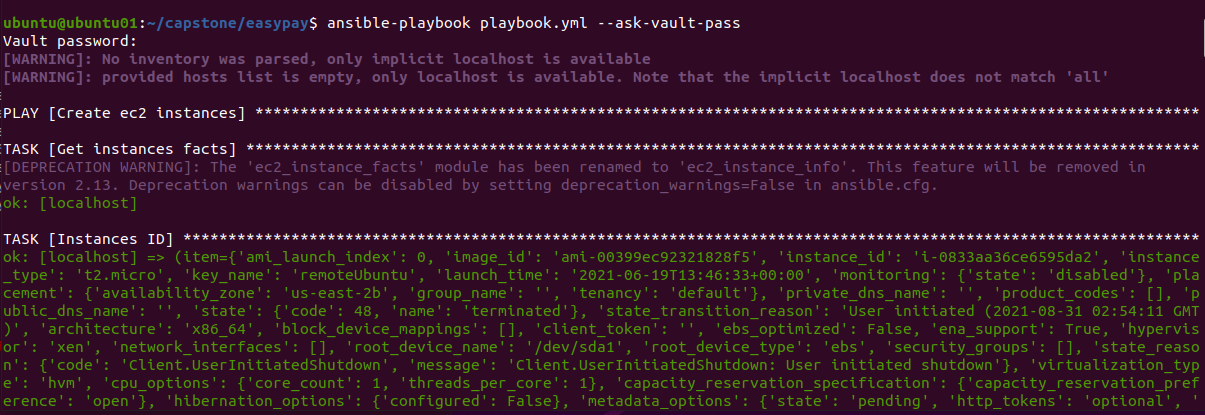
# Name: App

# instance\_tags:

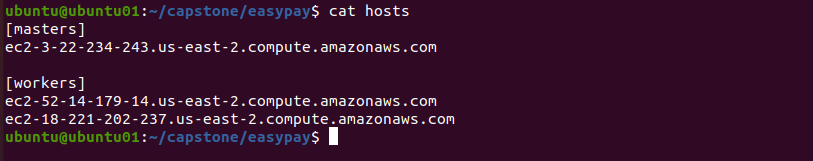
# Name: App

tags: ['never', 'create\_ec2']

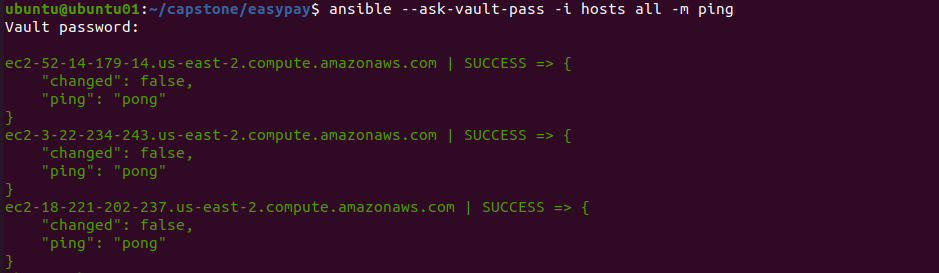
Step 3 - Run the playbook



Step 4 - Create ansible host resource file



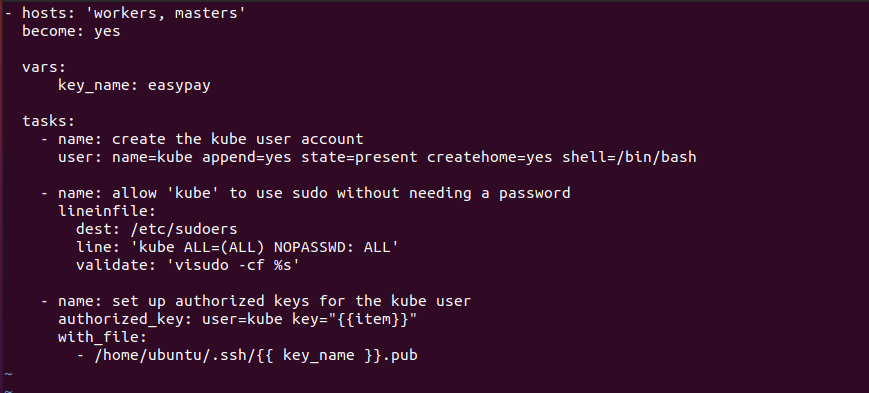
Step – 5 Test the connection to ec2 instances



**Install Docker and Kubernetes on the cluster**

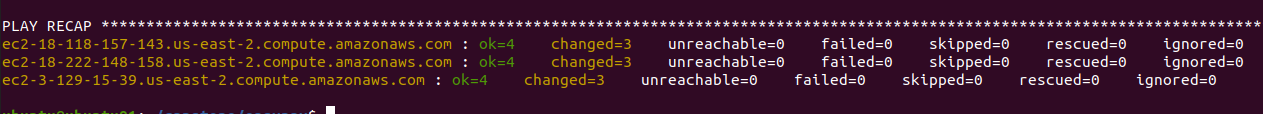
Step 1 – Create Kubernetes user with Ansible







Should get the following results



Step 2 – Install Docker



Credits and reference examples:

Simplilearn Devops Training course: <https://www.simplilearn.com>

Using ansible to create ec2 instances - <https://medium.datadriveninvestor.com/devops-using-ansible-to-provision-aws-ec2-instances-3d70a1cb155f>

Using ansible to create k8s cluster - <https://buildvirtual.net/deploy-a-kubernetes-cluster-using-ansible/>

Using ansible to create load balancer - <https://www.nickhammond.com/load-balancing-with-aws-and-ansible/>