

Infra Optimization.

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)

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)**Select EC2 instance once you login to AWS Web**
[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)
Step 1: Choose an Amazon Machine Image (AMI)

Step 1: Choose an Amazon Machine Image (AMI)

Amazon RDS

[Launch a database using RDS](#)

SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0fde50fcbcd46f2f7 (64-bit x86) / ami-05f2f5f76d89313bb (64-bit Arm)

SUSE Linux **Free tier eligible** Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-09e67e426f25ce0d7 (64-bit x86) / ami-00d1ab6b335f217cf (64-bit Arm)

Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Free tier eligible Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0747bdcabd34c712a (64-bit x86) / ami-08353a25e80bbea3e (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Free tier eligible Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

Microsoft Windows Server 2019 Base - ami-0fa60543f60171fe3

Windows **Free tier eligible** Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

Initially selected instance type as t2.micro but later changed to t2.medium post deployment for all the three nodes to be compatible with Kubernetes

[aws](#) [Services ▾](#) [\[Alt+S\]](#)
[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)
Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: [All instance families](#) [Current generation](#) [Show/Hide Columns](#)

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input checked="" type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input checked="" type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input checked="" type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes

Selected Default VPC and Auto Assign Public IP

Screenshot of Step 3: Configure Instance Details. The page shows configuration options for launching an instance. Key settings include:

- Number of instances:** 1
- Purchasing option:** Request Spot Instances (unchecked)
- Network:** vpc-052806ff73a59b30d (default) - Create new VPC (button)
- Subnet:** No preference (default subnet in any Availability Zone) - Create new subnet (button)
- Auto-assign Public IP:** Use subnet setting (Enable)
- Placement group:** Add instance to placement group (unchecked)
- Capacity Reservation:** Open
- Domain join directory:** No directory - Create new directory (button)
- IAM role:** None - Create new IAM role (button)
- Shutdown behavior:** Stop
- Stop - Hibernate behavior:** Enable hibernation as an additional stop behavior (unchecked)
- Enable termination protection:** Protect against accidental termination (unchecked)
- Monitoring:** Enable CloudWatch detailed monitoring (unchecked)
- Tenancy:** Shared - Run a shared hardware instance

Screenshot of Step 4: Add Storage. The page shows storage device settings for the instance. A single root volume is listed:

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-04753b6396410a049	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Configure Security Group for all traffic for LAB purpose only

Screenshot of Step 6: Configure Security Group. The page shows rules for the security group. A warning message states:

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Step 7: Review Instance Launch

To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about [free usage tier](#) eligibility and usage restrictions.

[Don't show me this again](#)

AMI Details

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0747bdcabd34c712a
Free tier eligible Ubuntu Server 18.04 LTS (HVM).EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root Device Type: ebs Virtualization type: hvm

[Edit AMI](#)

Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.large	-	2	8	EBS only	-	Low to Moderate

Security Groups

[Edit security groups](#)

Security group name: launch-wizard-2
Description: launch-wizard-2 created 2021-05-30T18:13:36.917+05:30

Type (i)	Protocol (i)	Port Range (i)	Source (i)	Description (i)
SSH	TCP	22	0.0.0.0/0	
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

Instance Details

[Edit instance details](#)

Storage

[Edit storage](#)

Tags

[Edit tags](#)

[Cancel](#) [Previous](#) [Launch](#)

2. Create the Access key and the secret key for the root user or IAM user for the AWS CLI configuration

The screenshot shows the AWS IAM User Summary page for a user named 'capstone'. The 'Security credentials' tab is selected, showing the 'Console sign-in link' (https://962386878476.signin.aws.amazon.com/console) and other credential details. Below this, the 'Access keys' section is shown, with a table listing an active access key (AKIA6AEV2MAGKQZT5OGK) and its details. A 'Create access key' button is visible.

Access key ID	Created	Last used	Status
AKIA6AEV2MAGKQZT5OGK	2021-05-30 16:44 UTC+0530	2021-05-30 16:58 UTC+0530 with ec2 in us-east-1	Active Make Inactive

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

Created a PEM for SSH and got the Private Key to be used for SSH from local workstation

Used the below mentioned commands to install Ansible on all the EC2 instances

```
sudo apt-get install -f
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

```
ubuntu@ip-172-31-54-80:~$ ansible --version
ansible 2.9.22
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Feb 27 2021, 15:10:58) [GCC 7.5.0]
```

Post the installation of Ansible on the master EC2 node created a playbook for ansible to automate the EC2 instance deployment using the below mentioned playbook yaml

```
- hosts: localhost
  tasks:
    - ec2:
        aws_access_key: AKIA6AEV2MAGKQZT5OGK
        aws_secret_key: 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
        key_name: capstone
```

```

group: sg-0ebc95f6e95b48772
instance_type: t2.micro
image: ami-0747bdcabd34c712a
wait: yes
wait_timeout: 500
count: 1
instance_tags:
  Name: ClusterInstance-1
monitoring: yes
region: us-east-1
vpc_subnet_id: subnet-0cd8d2a7867fc82ae
assign_public_ip: yes

```

```

PLAY [localhost] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [ec2] *****
changed: [localhost]

PLAY RECAP *****
localhost : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-54-80:~ $ 

```

Playbook execution completed successfully for 2 more EC2 instances with Name ClusterInstance-1 and ClusterInstance-2 respectively

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
-	i-0364bf2da40c5ab53	Running	t2.micro	2/2 checks passed	No alarms	us-east-1e	ec2-52-91-184-36.com...	52.91.184.36	-
ClusterInstance-2	i-01e5e8f1e127a1040	Running	t2.micro	Initializing	No alarms	us-east-1e	ec2-54-209-31-37.com...	54.209.31.37	-
ClusterInstance-1	i-0a37e4daed8cb119c	Running	t2.micro	2/2 checks passed	No alarms	us-east-1e	ec2-54-157-132-95.co...	54.157.132.95	-

Steps to create the Load Balancer and Taget Group and Attach the instance in the Target Group

Select HTTP/HTTPS LB

Application Load Balancer	Network Load Balancer	Gateway Load Balancer	Classic Load Balancer
HTTP HTTPS	TCP TLS UDP	IP	PREVIOUS GENERATION for HTTP, HTTPS, and TCP
Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.	Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.	Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.	Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.
Learn more >	Learn more >	Learn more >	Learn more >
Create	Create	Create	Create

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

Name	<input type="text" value="EasyPay-loadbalancer"/>
Scheme	<input checked="" type="radio"/> internet-facing <input type="radio"/> internal
IP address type	<input type="text" value="ipv4"/>

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
<input type="text" value="HTTP"/>	<input type="text" value="80"/>
Add listener	

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 1: Configure Load Balancer

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

VPC	<input type="text" value="vpc-052806ff73a59b30d (172.31.0.0/16) (default)"/>
Availability Zones	<input checked="" type="checkbox"/> us-east-1a Subnet-0e27f9f6920948537 IPv4 address Assigned by AWS
	<input checked="" type="checkbox"/> us-east-1b Subnet-01e7b46551b2a3c82 IPv4 address Assigned by AWS
	<input checked="" type="checkbox"/> us-east-1c Subnet-0bddf811e1d44bcbb IPv4 address Assigned by AWS
	<input checked="" type="checkbox"/> us-east-1d Subnet-086296f74fa1957e8 IPv4 address Assigned by AWS
	<input checked="" type="checkbox"/> us-east-1e Subnet-0cdbd2a7867fcbb2ae IPv4 address Assigned by AWS
	<input checked="" type="checkbox"/> us-east-1f Subnet-08de355c82590e677 IPv4 address Assigned by AWS

Add-on services

Additional AWS services can be integrated with this load balancer at launch when you enable them below. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 2: Configure Security Settings

⚠ Improve your load balancer's security. Your load balancer is not using any secure listener.

If your traffic to the load balancer needs to be secure, use the HTTPS protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under [Basic Configuration](#) section. You can also continue with current settings.

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group Create a new security group
 Select an existing security group

Filter

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-0efd7084e042fed52	default	default VPC security group	Copy to new
<input checked="" type="checkbox"/> sg-073e3702aae36c792	launch-wizard-1	launch-wizard-1 created 2021-05-30T16:05:20.679+05:30	Copy to new

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify here. It also performs health checks on the targets using these settings. The target group you specify in this step will apply to all of the listeners configured on this load balancer. You add listeners after the load balancer is created.

Target group

Target group

Name

Target type Instance IP Lambda function

Protocol

Port

Protocol version HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
 HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
 gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Health checks

Protocol

Path

[1. Configure Load Balancer](#) [2. Configure Security Settings](#) [3. Configure Security Groups](#) [4. Configure Routing](#) [5. Register Targets](#) [6. Review](#)

Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

<input type="checkbox"/>	Instance	Name	Port	State	Security groups	Zone
No instances available.						

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

on port 80

<input type="checkbox"/>	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input type="checkbox"/>	i-0364bf2da40c5ab53		running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fc2ae	172.31.48.0/20
<input type="checkbox"/>	i-01e5e81e1e127a1040	ClusterInstance-2	running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fc2ae	172.31.48.0/20
<input type="checkbox"/>	i-0a37e4daed8cb119c	ClusterInstance-1	running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fc2ae	172.31.48.0/20

Step 6: Review

Please review the load balancer details before continuing

Load balancer

Name EasyPay-loadbalancer
Scheme internet-facing
Listeners Port:80 - Protocol:HTTP ▲
IP address type ipv4
VPC vpc-052806ff73a59b30d
Subnets subnet-0e27f9f6920948537, subnet-0bddf811e1d44bc8, subnet-086296f74fa1957e8, subnet-0cdbd2a7867fc2ae, subnet-08de35c6b2590e77
Tags

Security groups

Security groups sg-073e3702aee36c792

Routing

Target group New target group
Target group name EasyPay-Target
Port 80
Target type instance
Protocol HTTP
Protocol version HTTP1
Health check protocol HTTP
Path /
Health check port traffic port
Healthy threshold 5
Unhealthy threshold 2
Timeout 5
Interval 30
Success codes 200

Targets

[Edit](#) [Cancel](#) [Previous](#) [Create](#)

AWS Services

Search for services, features, marketplace products, and docs

[Alt+S]



Corestack_Role/pnaiju_vmware @ 9623-8687-8476 ▾ N. Virginia ▾ Support ▾

Load Balancer Creation Status

Successfully created load balancer

Load balancer **EasyPay-loadbalancer** was successfully created.
 Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic, and for the targets to complete the registration process and pass the initial health checks.

Suggested next steps

- Discover other services that you can integrate with your load balancer. Visit the [Integrated services](#) tab within **EasyPay-loadbalancer**
- Consider using AWS Global Accelerator to further improve the availability and performance of your applications. [AWS Global Accelerator console](#) ▾

[Close](#)

Elastic IP configuration for all the three instances

[EC2](#) > [Elastic IP addresses](#) > Associate Elastic IP address

Associate Elastic IP address

Choose the instance or network interface to associate to this Elastic IP address (18.213.139.151)

Elastic IP address: 18.213.139.151

Resource type
 Choose the type of resource with which to associate the Elastic IP address.
 Instance
 Network interface

Instance

Private IP address
 The private IP address with which to associate the Elastic IP address.

Reassociation
 Specify whether the Elastic IP address can be reassigned with a different resource if it already associated with a resource.
 Allow this Elastic IP address to be reassigned

[Cancel](#) [Associate](#)

Associate Elastic IP address

Choose the instance or network interface to associate to this Elastic IP address (52.45.16.42)

Elastic IP address: 52.45.16.42

Resource type
Choose the type of resource with which to associate the Elastic IP address.

Instance
 Network interface

⚠️ If you associate an Elastic IP address to an instance that already has an Elastic IP address associated, this previously associated Elastic IP address will be disassociated but still allocated to your account. [Learn more](#)

Instance
Q i-00b664c4b5bd70468

Private IP address
The private IP address with which to associate the Elastic IP address.
Q 172.31.57.225

Reassociation
Specify whether the Elastic IP address can be reassigned with a different resource if it already associated with a resource.
 Allow this Elastic IP address to be reassigned

⌚ Elastic IP address associated successfully.
Elastic IP address 52.45.16.42 has been associated with instance i-00b664c4b5bd70468

Elastic IP addresses (1/3)							Actions ▲	Allocate Elastic IP address
Name	Allocated IPv4 add...	Type	Allocation ID	Associated instance ID	Creation ID	Subnet ID	View details	Release Elastic IP addresses
–	18.213.139.151	Public IP	eipalloc-0f6e0af0466d0f12e	i-0195c69318a332e84	oci-0895cf3cd40ce4	subnet-0cdbd2a7867fc2ae	<input type="button" value="Associate Elastic IP address"/>	<input type="button" value="Disassociate Elastic IP address"/>
–	52.45.16.42	Public IP	eipalloc-04b96c90ebe2368b1	i-00b664c4b5bd70468	172.31.57.225	ipassoc-0370c68ebce4cd	<input type="button" value="Associate Elastic IP address"/>	<input type="button" value="Disassociate Elastic IP address"/>
<input checked="" type="checkbox"/>	52.5.98.24	Public IP	eipalloc-0ad9f1a1384dd0537	–	–	–	<input type="button" value="Associate Elastic IP address"/>	<input type="button" value="Disassociate Elastic IP address"/>

Available instances (3)

Instance ID	Name	State	Security groups	Zone	Subnet ID
i-0195c69318a332e84	ClusterInstance-1	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae
i-00b664c4b5bd70468	ClusterInstance-2	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae
i-0bf423b2e63b3c904	ClusterInstance-2	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae

0 selected

Ports for the selected instances
Ports for routing traffic to the selected instances (separate multiple ports with commas):
80

3 selections are now pending below. Include more or register targets when ready.

Targets (3)

All	Status	Instance ID	Name	Port	State	Security groups	Zone	Subnet ID
X	Pending	i-0195c69318a332e84	ClusterInstance-1	80	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae
X	Pending	i-00b664c4b5bd70468	ClusterInstance-2	80	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae
X	Pending	i-0bf423b2e63b3c904	ClusterInstance-2	80	running	launch-wizard-2	us-east-1e	subnet-0cdbd2a7867fc2ae

3 pending

The screenshot shows the AWS EC2 service interface. On the left, there's a sidebar with navigation links like 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances' (selected), 'Instances New', 'Instance Types', and 'Launch Templates'. The main content area is titled 'Elastic IP addresses (1/1)' and displays a single row of data:

Name	Allocated IPv4 add...	Type	Allocation ID	Associated instance ID	Private IP address	Association ID
-	52.45.16.42	Public IP	eipalloc-04b96c90ebe2368b1	-	-	-

At the top right, there are buttons for 'Actions' and 'Allocate Elastic IP address'.

Install Docker and Kubernetes on the cluster

completed the installation of Docker on all the three instances first.

```
ubuntu@ip-172-31-54-80:~$ history
1 sudo apt-get update
2 sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
4 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu/$lsb_release -cs" stable
5 sudo apt-get install docker-ce
6 docker --version
7 sudo docker --version
```

Configured Master node with Kubeadm init and joined the rest of the nodes to the master

```
sudo kubeadm init
mkdir -p $HOME/.kube
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl get nodes
```

Applied the overlay network for K8

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
kubectl get pods -n kube-system
```

```
Preparing to unpack .../libcontainerd.io_1.4.6-1_amd64.deb ...
Unpacking libcontainerd.io_1.4.6-1 ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../docker-ce-cli_5%3a20.10.6-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.6-3-0-ubuntu-bionic) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../3-docker-ce_5%3a20.10.6-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce (5:20.10.6-3-0-ubuntu-bionic) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../4-docker-ce-rootless-extras_5%3a20.10.6-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.6-3-0-ubuntu-bionic) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../5-docker-scan-plugin_0.7.0-ubuntu-bionic_amd64.deb ...
Unpacking docker-scan-plugin (0.7.0-ubuntu-bionic) ...
Selecting previously unselected package liblibleveld17:amd64.
Preparing to unpack .../6-liblibleveld17_2.4.6-2_amd64.deb ...
Unpacking liblibleveld17:amd64 (2.4.6-2) ...
Setting up containerd.io (1.4.6-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up docker-ce-rootless-extras (5:20.10.6-3-0-ubuntu-bionic) ...
Setting up docker-scan-plugin (0.7.0-ubuntu-bionic) ...
Setting up liblibleveld17:amd64 (2.4.6-2) ...
Setting up docker-ce-cli (5:20.10.6-3-0-ubuntu-bionic) ...
Setting up pign (2.4-1) ...
Setting up docker-ce (5:20.10.6-3-0-ubuntu-bionic) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/docker.service → /lib/systemd/system/docker.socket.
Processing triggers for liblibleveld17:amd64 (2.4.6-2) ...
Processing triggers for systemd (2.37-3ubuntu10.46) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-54-80:~$ docker -version
docker: '-version' is not a docker command.
See 'docker --help'.
ubuntu@ip-172-31-54-80:~$ sudo docker --version
Docker version 20.10.6, build 370c289
ubuntu@ip-172-31-54-80:~$
```

```

ubuntu@ip-172-31-54-80:~$ sudo kubeadm token create --print-join-command
kubeadm join 172.31.54.80:6443 --token ljr0n4.pdx18yy9xtt7v1 --discovery-token-ca-cert-hash sha256:b4ee6f520fb56bd1e7e59df30f009458578148c33ad1176d2ea0e68f93059ffe
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME          STATUS   ROLES      AGE     VERSION
p-172-31-49-129  NotReady   control-plane,master   9m40s   v1.21.1
p-172-31-57-225  NotReady   <none>        14s    v1.21.1
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME          STATUS   ROLES      AGE     VERSION
p-172-31-49-129  NotReady   <none>        17s    v1.21.1
p-172-31-54-80  NotReady   control-plane,master   11m    v1.21.1
p-172-31-57-225  NotReady   <none>        2m7s   v1.21.1
ubuntu@ip-172-31-54-80:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset/applications created
ubuntu@ip-172-31-54-80:~$ kubectl get pods -n kube-system
NAME          READY   STATUS    RESTARTS   AGE
coredns-558bd4d45db-5p7nh   0/1   Pending   0          17m
coredns-558bd4d45db-t425w   0/1   Pending   0          17m
etcd-ip-172-31-54-80       1/1   Running  0          18m
kube-apiserver-ip-172-31-54-80  1/1   Running  0          18m
kube-controller-manager-ip-172-31-54-80  1/1   Running  0          18m
kube-proxy-cwv7q            1/1   Running  0          8m40s
kube-proxy-vsjht            1/1   Running  0          6m50s
kube-proxy-wpxvp            1/1   Running  0          17m
kube-scheduler-ip-172-31-54-80  1/1   Running  0          18m
weave-net-52x9j             2/2   Running  1          12s
weave-net-551l               2/2   Running  1          12s
weave-net-gkkcq              2/2   Running  1          12s
ubuntu@ip-172-31-54-80:~$ kubectl get pods -n kube-system
NAME          READY   STATUS    RESTARTS   AGE
coredns-558bd4d45db-5p7nh   0/1   Pending   0          18m
coredns-558bd4d45db-t425w   0/1   Pending   0          18m
etcd-ip-172-31-54-80       1/1   Running  0          18m
kube-apiserver-ip-172-31-54-80  1/1   Running  0          18m
kube-controller-manager-ip-172-31-54-80  1/1   Running  0          18m
kube-proxy-cwv7q            1/1   Running  0          6m45s
kube-proxy-vsjht            1/1   Running  0          6m55s
kube-proxy-wpxvp            1/1   Running  0          18m
kube-scheduler-ip-172-31-54-80  1/1   Running  0          18m
weave-net-52x9j             2/2   Running  1          17s
weave-net-551l               2/2   Running  1          17s
weave-net-gkkcq              2/2   Running  1          17s
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME          STATUS   ROLES      AGE     VERSION
p-172-31-49-129  Ready    <none>        7m2s   v1.21.1
p-172-31-54-80  Ready    control-plane,master   18m    v1.21.1
p-172-31-57-225  Ready    <none>        8m52s  v1.21.1

```

Implement the network policies at the database pod to allow ingress traffic from the front-end application pod

Created below mentioned yaml to create the network policy at the database pod to allow ingress traffic from front-end application POD

```

ubuntu@ip-172-31-54-80:~$ cat networkpolicy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
      - namespaceSelector:
          matchLabels:
            project: myproject
        - podSelector:
            matchLabels:
              role: frontend
        ports:
          - protocol: TCP
            port: 6379
    egress:
      - to:
          - ipBlock:
              cidr: 10.0.0.0/24
            ports:
              - protocol: TCP
ubuntu@ip-172-31-54-80:~$ kubectl create -f networkpolicy.yaml
networkpolicy.networking.k8s.io/test-network-policy created

```

Implement the network policies at the database pod to allow ingress traffic from the front-end application pod

```

ubuntu@ip-172-31-54-80:~$ cat > userrole.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list", "create", "delete", "update"]
^C
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f userrole.yaml
role.rbac.authorization.k8s.io/pod-reader created
ubuntu@ip-172-31-54-80:~$ 

```

Configure application on the pod

```

ubuntu@ip-172-31-54-80:~$ cat > pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    role: db
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f pod.yaml
pod/static-web created
ubuntu@ip-172-31-54-80:~$ 

```

```

pi@odroidc1: ~
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f pod.yaml
pod/static-web created
ubuntu@ip-172-31-54-80:~$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
static-web  1/1     Running   0          38s
ubuntu@ip-172-31-54-80:~$ 

```

Take snapshot of ETCD database

Download and configured ETCD from GitHub

```

ubuntu@ip-172-31-54-80:~$ etcd --version
etcd Version: 3.3.13
git SHA: 99d3084
go Version: go1.10.8
go OS/Arch: linux/amd64
ubuntu@ip-172-31-54-80:~$ 

```

Executed below mentioned command to create a backup WAL for ETCD database

```

2021-05-30 15:00:45.992291 I | ignoring v3 raft entry
2021-05-30 15:00:45.992243 I | ignoring v3 raft entry
2021-05-30 15:00:45.992255 I | ignoring v3 raft entry
2021-05-30 15:00:45.992270 I | ignoring v3 raft entry
2021-05-30 15:00:45.992285 I | ignoring v3 raft entry
2021-05-30 15:00:45.992299 I | ignoring v3 raft entry
2021-05-30 15:00:45.992311 I | ignoring v3 raft entry
ubuntu@ip-172-31-54-80:~$ ls
cluster.yaml  etcd-backup  etcd-v3.3.13-linux-amd64  etcd-v3.3.13-linux-amd64.tar.gz  member  networkpolicy.yaml  pod.yaml  userrole.yaml
ubuntu@ip-172-31-54-80:~$ cd /backupdir/
ubuntu@ip-172-31-54-80:~/backupdir/ Permission denied
ubuntu@ip-172-31-54-80:~$ ls
cluster.yaml  etcd-backup  etcd-v3.3.13-linux-amd64  etcd-v3.3.13-linux-amd64.tar.gz  member  networkpolicy.yaml  pod.yaml  userrole.yaml
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/member/wal/
0000000000000000-0000000000000000.wal
ubuntu@ip-172-31-54-80:~$ sudo etcdctl backup --data-dir /var/lib/etcd/ --backup-dir /backupdir/
member
ubuntu@ip-172-31-54-80:~$ sudo cd /backupdir/
sudo: cd: command not found
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/member
snap  wal
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/member/wal/
0000000000000000-0000000000000000.wal
ubuntu@ip-172-31-54-80:~$ sudo etcdctl backup --data-dir /var/lib/etcd/ --backup-dir /backupdir/

```

Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

Below mentioned autoscaling yaml was used for the same

```

ubuntu@ip-172-31-16:~$ cat podautoscaler.yml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: web
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Pod
    name: static-web
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
    target:
      type: Utilization
      averageUtilization: 50

ubuntu@ip-172-31-16:~$ kubectl apply -f podautoscaler.yml
horizontalpodautoscaler.autoscaling/web created
ubuntu@ip-172-31-16:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
web      Pod/static-web  <unknown>/50%  1         10        0          18s

```

Complete code has been pushed to GitHub for review

complete code is pushed to GitHub for review