National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cloud Computing

Suyog Rajesh Redkar
Student ID: X23419504

School of Computing
National College of Ireland

Supervisor: Prof. Shreyas Setlur Arun

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Suyog Rajesh Redkar……

Student ID: X23419504……

Programme: Msc in Cloud Computing…… **Year:** 2025……..

Module: Msc research Project……

Lecturer: Prof Shreyas Setlur Arun……

Submission Due Date: 12/12/2025……

Project Title: Enhancing Resilience of Terraform Provisioned Kubernetes Cluster using Chaos Engineering……

Word Count: …… **Page Count:** ……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Suyog Rajesh Redkar……

Date: 12/12/2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

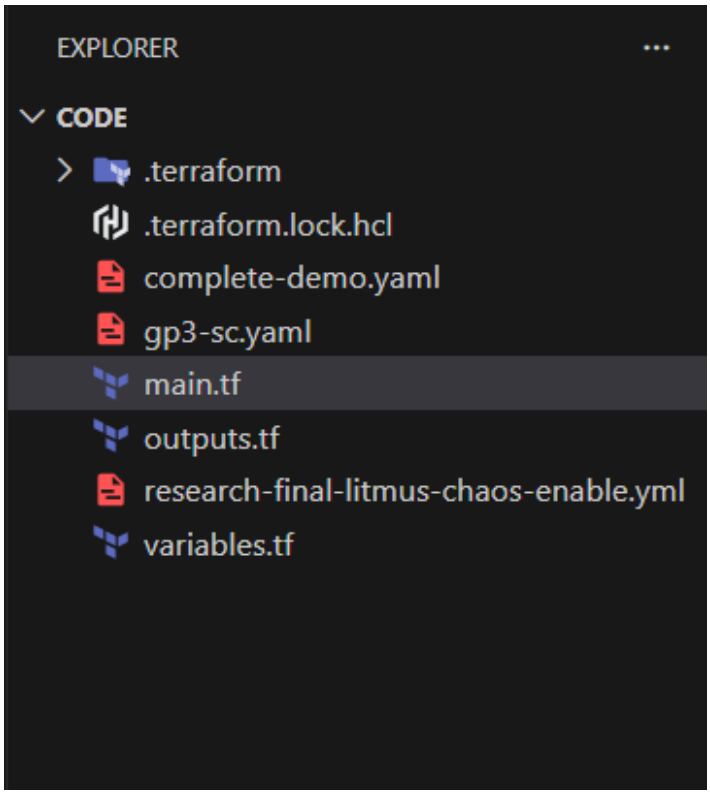
Suyog Rajesh Redkar
Student ID: x23419504

1 System Requirements

I. AWS EKS Cluster
- 3 x Worker nodes (t3.medium)
- OS: Amazon Linux
- Cluster Version: Kubernetes >= 1.30

II. Terraform >= 1.11
III. Helm >= 3.18
IV. Litmus Chaos >= 3.20
V. Prometheus >= 2.51
VI. AWS Cli v2
VII. Kubectl >= 1.29
VIII. Git
IX. VS Code

2 Project Directory and Structure



3 Setup Guide

3.1 Terraform Script

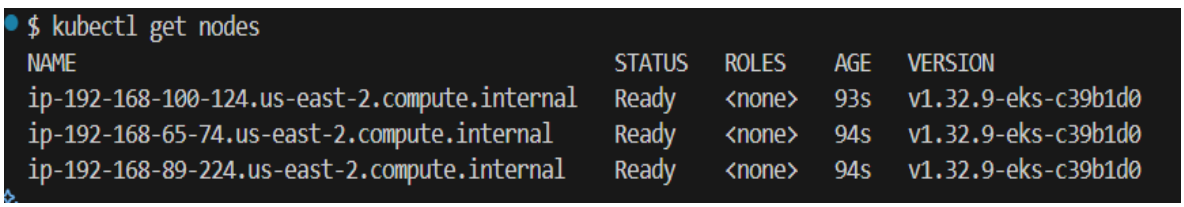
This terraform script is used to provision the infrastructure. It includes the main.tf, output.tf and variables.tf files which provisions the EKS cluster, node group, Vpc and Subnets.

We initialize terraform using

- Terraform init
- Terraform plan
- Terraform apply

We verify the AWS EKS cluster creation using command

- kubectl get nodes



3.2 Sock-Shop application Deployment

Install the Sock-Shop application onto the EKS Cluster using the complete-demo.yaml file using command:-

- kubectl apply -f complete-demo.yaml

Verify the installation using:-

- kubectl get pods -n default.

```
$ kubectl get pods -n default
NAME                            READY   STATUS    RESTARTS   AGE
carts-db-594b68dc85-bs7tc       1/1     Running   0          3m52s
carts-f4f6c98b9-8v5gv           1/1     Running   0          3m54s
catalogue-6d8b8988d-rwggr       1/1     Running   0          3m51s
catalogue-db-5c796cfb68-2tcp8   1/1     Running   0          3m50s
front-end-55c969566c-qs28z      1/1     Running   0          3m49s
orders-685d756fb5-mdwz9         1/1     Running   0          3m48s
orders-db-67c8d5f459-dm62q      1/1     Running   0          3m47s
payment-56bcfc857f-dnn24        1/1     Running   0          3m46s
queue-master-5f4477db4c-4brkw   1/1     Running   0          3m45s
rabbitmq-97cbcbdbc-69hhd        2/2     Running   0          3m44s
session-db-67c8bfcb9c-hpftd     1/1     Running   0          3m43s
shipping-5c7bc7f48f-jn99j       1/1     Running   0          3m41s
user-6776ddd957-rswst           1/1     Running   0          3m40s
user-db-7dbb5fcd6f-vdchq        1/1     Running   0          3m39s
```

All pods must be ready and in running state.

We can access the application frontend using

- kubectl port-forward svc/front-end 8080:80

The application will be live on

- http://localhost:8080

3.3 LitmusChaos Setup

Install LitmusChaos via helm using the below command

- helm repo add litmuschaos https://litmuschaos.github.io/litmus-helm/
- helm install chaos litmuschaos/litmus --namespace=litmus --set mongodb.image.registry=docker.io --set mongodb.image.repository=bitnamilegacy/mongodb --set mongodb.image.tag=5.0.8
- kubectl apply -f gp3-sc.yaml

Verify litmus chaos installation using the below command

- kubectl get pods -n litmus

```
$ kubectl get pods -n litmus
NAME                                        READY   STATUS    RESTARTS   AGE
chaos-litmus-auth-server-665585fcf7-hhpvp   1/1     Running   0          4m24s
chaos-litmus-frontend-fb9dff8-brq6s         1/1     Running   0          4m24s
chaos-litmus-server-755bc4d7b5-shvqk        1/1     Running   0          4m24s
chaos-mongodb-0                             1/1     Running   0          58s
chaos-mongodb-1                             1/1     Running   0          39s
```

In order to access the Litmus ChaosCenter UI we use the below command

- kubectl port-forward svc/litmus-frontend-service -n litmus 9091:9091

The ChaosCenter UI will be live on

- http://localhost: 9091

3.4 Prometheus monitoring setup

Install Prometheus via helm using the command

- helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
- helm install monitoring prometheus-community/kube-prometheus-stack --namespace monitoring --create-namespace --set alertmanager.persistentVolume.storageClass="gp2",server.PersistentVolume.storageClass="gp2"

Verify installation using

- kubectl get pods -n monitoring

```
$ kubectl get pods -n monitoring
NAME                                                     READY   STATUS    RESTARTS   AGE
alertmanager-monitoring-kube-prometheus-alertmanager-0   2/2     Running   0          57s
monitoring-grafana-6587d4bd48-n2hvb                      3/3     Running   0          61s
monitoring-kube-prometheus-operator-9888744d8-zps8m      1/1     Running   0          61s
monitoring-kube-state-metrics-689d998768-7ghx8           1/1     Running   0          61s
monitoring-prometheus-node-exporter-lcdcl                1/1     Running   0          61s
monitoring-prometheus-node-exporter-m6nzt                1/1     Running   0          61s
monitoring-prometheus-node-exporter-vgkmp                1/1     Running   0          61s
prometheus-monitoring-kube-prometheus-prometheus-0       2/2     Running   0          57s
```

Access the prometheus dashboard using

- kubectl port-forward -n monitoring svc/monitoring-kube-prometheus-prometheus 9090:9090

The prometheus dashboard will be live on

- http://localhost:9090

Use the below Prom Query in-order to scrape the data during the failure injection.

i. Network-Loss
   - rate(container_network_receive_bytes_total{pod=~"carts.*", pod!~"carts-db.*"}[1m])

ii. Pod Delete

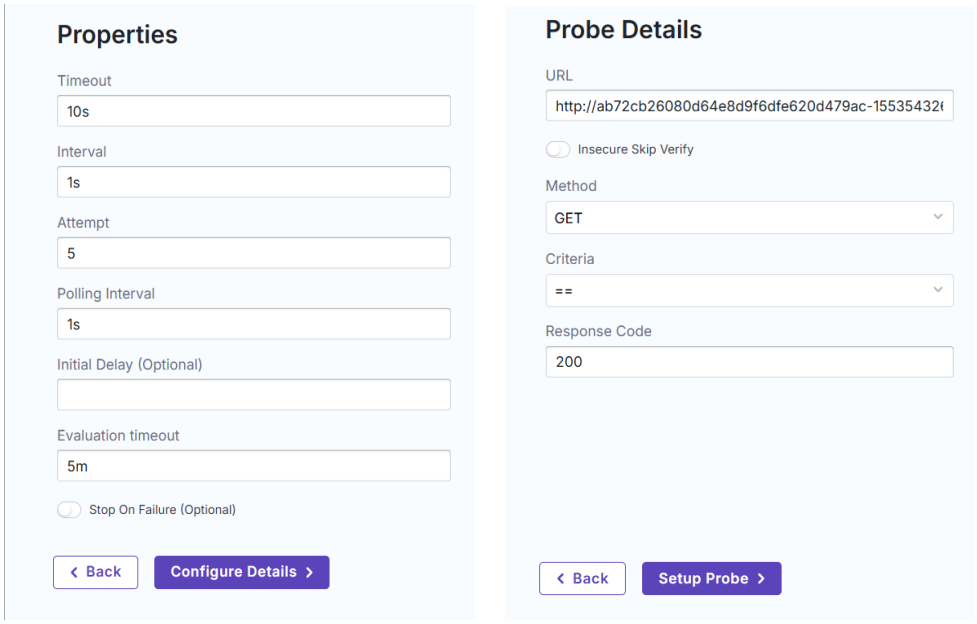
- sum(kube_pod_status_ready{namespace="default",pod=~"carts.*",pod!~"carts-db.*", condition="true"})

iii. CPU Utilization

- avg(sum by (pod) (rate(container_cpu_usage_seconds_total{pod=~"carts.*",pod!~"carts-db.*",container!="POD"}[1m])))

4 Failure Injection

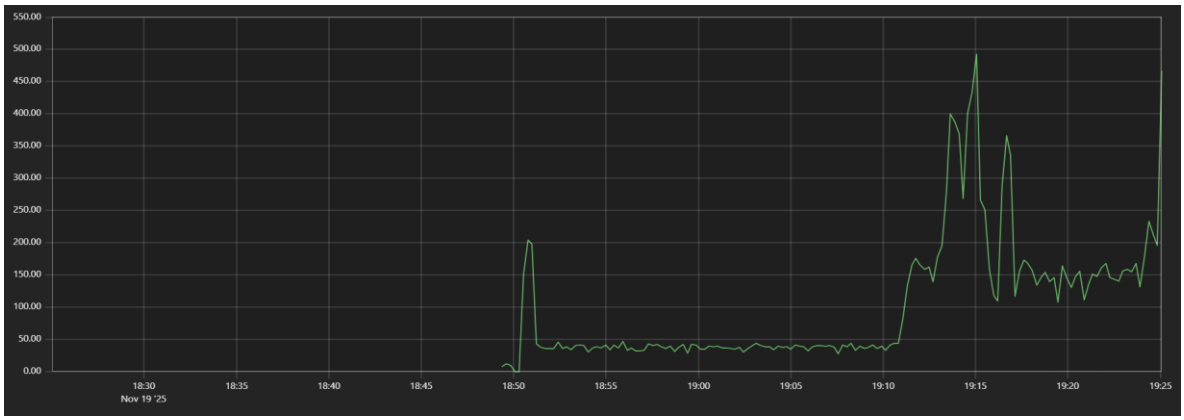
4.1 LitmusChaos Probe Configuration



5 Outputs

5.1 Network Loss

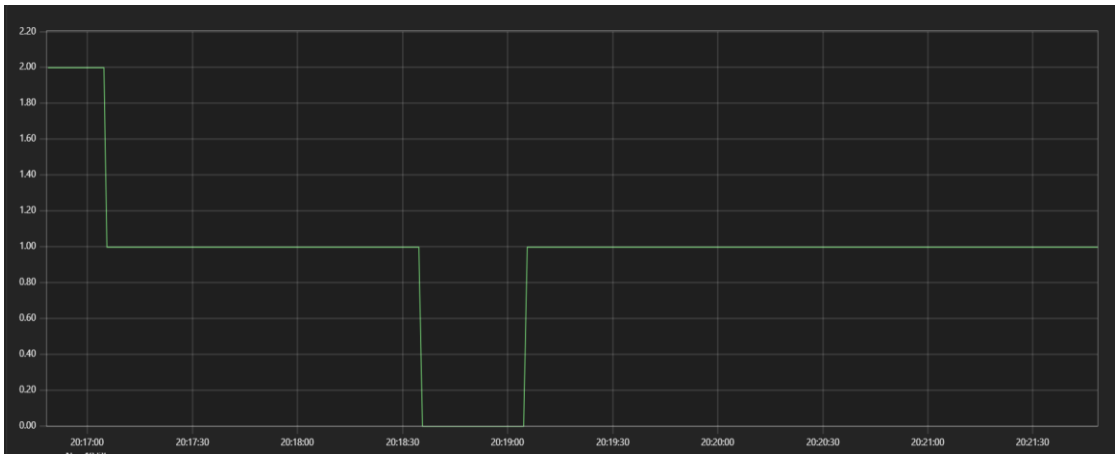
The network loss experiment showed sudden spike in the network usage by the carts pod making microservice unavailable.



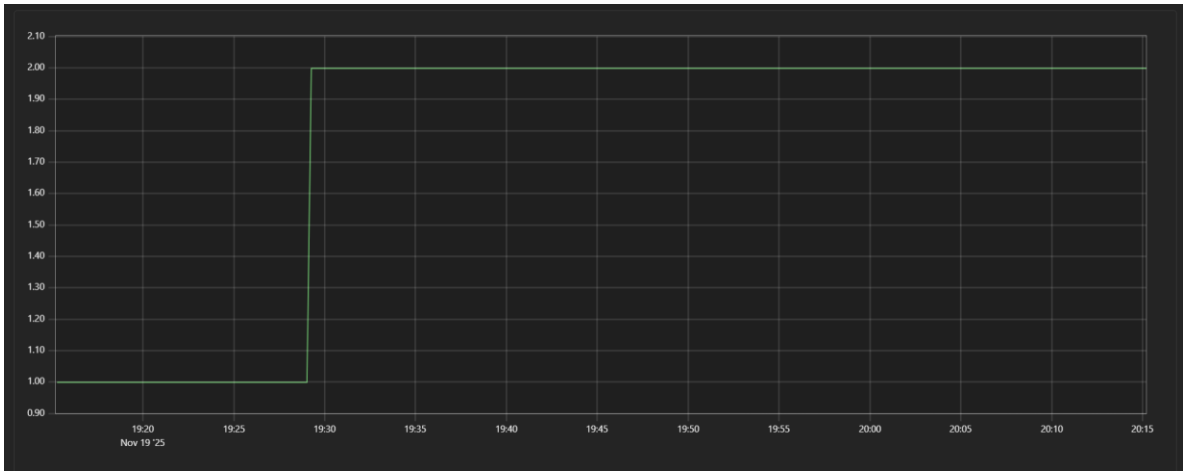
5.2 Pod Delete

The pod delete experiment was run in two phases i.e. with replica set and without replica set. While running the experiment without replica set the carts microservice was completely unavailable while the pod was recovering for the failure, whereas in the second phase the service was still available as the configuration was made to maintain replicas (up to 2) such that even if one pod fails the other was ready to take requests.

5.2.1 Without replica set configuration



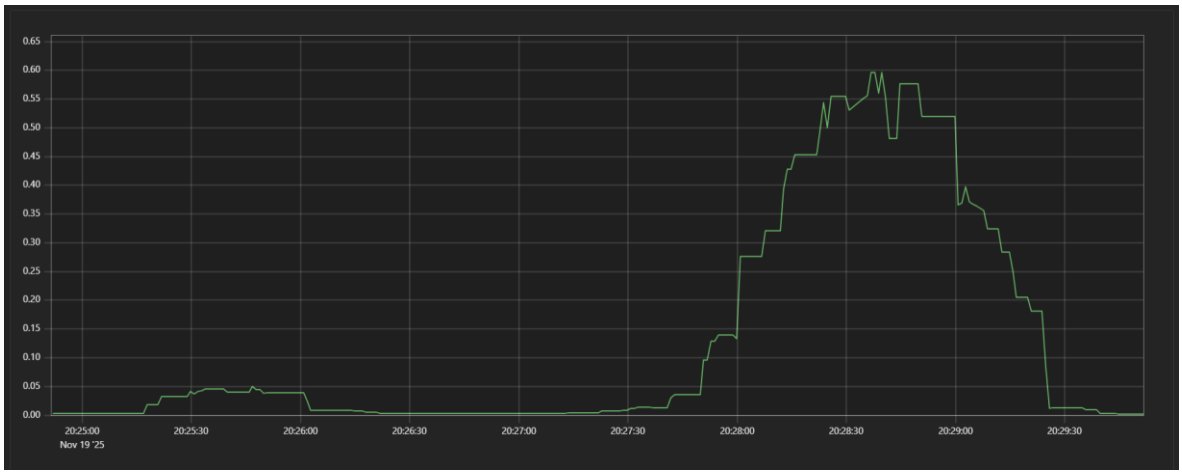
5.2.2 With replica set configuration



5.3 CPU Utilization

The CPU utilization experiment was run in two phases i.e. with autoscaling and without autoscaling. While running the experiment without autoscaling the carts microservice was utilized ~60% of the available CPU which caused other microservices to fail due to CPU deallocate, whereas in the second phase the CPU utilization averaged ~35% where the HPA would have a minimum of 1 and maximum of 6 pods if the pod CPU utilization cross 50%.

5.3.1 Without autoscaling configuration



5.3.2 With autoscaling configuration