

Configuration Manual

MSc Research Project Cloud Computing

Shubham Singh Student ID: 22170341

School of Computing National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Shubham Singh
Student ID:	22170341
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	1699
Page Count:	11

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.

<u>ALL</u> 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:	Shubham Singh
Date:	13th December 2023

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

Shubham Singh 22170341

1 Introduction

This document provides the steps to create and install the required softwares, tools and files to perform the research mentionted in research report with title "Enhancing Microservices Resilience: Chaos Engineering with Istio Service Mesh on Kubernetes".

2 Deploying Kubernetes Cluster on Google Kubernetes Engine

To deploy a Kubernetes cluster on GKE, first we create a kubernetes cluster using the UI of Google Cloud as in figure 1 and 2. We will select the standard creation and provide the necessary details like cluster name, location, zone, type and memory Google Cloud (2023).

		Cluster basics
reate cluster		The new cluster will be created with the name, version, and in the location you spe here. After the cluster is created, name and location can't be changed.
elect the cluster mode that you want to use.		• To experiment with an affordable cluster, try My first cluster in the Clu set-up guides
Autopilot: Google manages your cluster (Recommended) A pay-per-Pod Kubernetes cluster where GKE manages your nodes with minimal configuration required. Learn more [2]	CONFIGURE	Name cluster-1
Standard: You manage your cluster A pay per node Kubernetes cluster where you configure and manage your nodes. Learn more C	CONFIGURE	Cluster names must start with a lowercase letter followed by up to 39 lowercase letter numbers, or hyphens. They can't end with a hyphen. You cannot change the cluster's no once it's created. Location type Resource prices may vary between certain regions. Learn more [2]
Compare cluster modes to learn more about their differences.	COMPARE	Ozonal
	CANCEL	O Regional Zone us-central1-c ▼

Figure 1: Cluster Modes

Figure 2: Cluster Configurations

3 Microservices application creation and deployment

3.1 Building the Microservices

Two microservices named test-app-hello and test-app-employee are created using python FastAPI as in figure 3.

<pre>flaktb: nopa f flaktb: nopa import is import is import is import fastapi.import FastAPI, HTTPException, Request, Response from fastapi.import fastAPI, HTTPException, Request, Response from fastapi.import RedirectResponse import ingitient import ingitingitient import ingitient import ingitient import ingiti</pre>	/ KORK KOPP // KORK KORK KORK KORK KORK KORK KORK K	
<pre>import 6 import 6 import</pre>	¢ flake8: noqa	# flake8: noga
Jugort 63 (rom fastapi import FastAPI, HTPException, Request, Response (import fastAPI, HTPException, Request, Response (import gastAPI, attrException, Request, Response (import fastAPI, attrExcep		import random
frm takei uper fastd, mittigin, keest, kepong import uuid fom takei uper fastd, mittigin, keest, kepong import uuid fom takei uper fastd, mittigin fom fastgi, responses import RedirectResponse import avait import uuid fom fastgi, might gi, responses import RedirectResponse import uuid fom fastgi, might gi, responses import RedirectResponse import uuid fom fastgi, might gi, responses import RedirectResponse import uuid fom fastgi, might gi, responses import RedirectResponse import uuid fom fastgi, might gi, responses import RedirectResponse import uuid fom fastgi, might gi, responses import RedirectResponse import systematics fom fastgi, might gi, responses import RedirectResponse app, name = 'hello' app, name = 'hello' app = FastAPi(titleasp_name, version='0.8.1', description=app_name, swagger_ui_parame fom fastgi, execting in the tioper Presentement	import os	from fastapi import FastAPI, HTTPException, Request, Response
jest usid fram falsi, isosana isoori RedirectResponse import logging import logging import logging import logging import logging import logging import logging import syst fram starlette, exporter import PrometheusHiddleware, handle_metrics Reprodi. Repro	from fastapi import FastAPI, MTTPException, Request, Response	import uuid
frm table.respons jourt Regists inport requests import loging inport requests import loging inport requests import loging inport togging import loging inport loging	import uid	from fastapi, responses import RedirectResponse
<pre>igent_deging igent_deging igent_deging igent_deging import togging import togging import togging import togging import or togging imp</pre>	from fastapi.responses import RedirectResponse	import requests
<pre>import Sys import Sys import</pre>	inport logging	import Logging
imm datig, an sport from starigtte_exporter import PrometheusMiddleware, handle_metrics implant from starigtte_exporter import PrometheusMiddleware, handle_metrics implant app_name = 'hello' imm datig_identient app = FastAFT(titlesapp_name, version='0.0.1', description=app_name, swagger_ui_parame imm datig_identient app = FastAFT(titlesapp_name, version='0.0.1', description=app_name, swagger_ui_parame imm datig_identient prometheusMiddleware, imm tarigit.estiont app = FastAFT(titlesapp_name, version='0.0.1', description=app_name, swagger_ui_parame imm tarigit.estiont app = FastAFT(titlesapp_name, version='0.0.1', description=app_name, version='0.0.1', description=app_name, version='0.0.1', description=app_name, version='0.0.1', description=app_name, version='0.0.1', description=app_name, version='0.0.1', tartient	import datatime	import sys
Feld, app_name Regression app_name Ten medic gineri giner	from redis_on import (from starlette exporter import PrometheusMiddleware, handle metrics
minitedit, app_name = 'hello' app_name = 'hello' app_name, version='0.0.1', description=app_name, swagger_ui_parame inn tariet_extint_port_ficition app_name, version='0.0.1', description=app_name, swagger_ui_parame inn tariet_extint_port_ficition promethoushidleware, opp_name, version='0.0.1', description=app_name, swagger_ui_parame inn tariet_extint_port_ficition promethoushidleware, opp_name, restrict inn tariet_extint_port_ficition promethoushidleware, opp_name, restrict inn tariet_extint_port_ficition promethoushidleware, opp_name, restrict	Field.	
Regreter app_name = 'hello' frm redix meror set_redix innerts in meror set_redix innerts innert	HashModel,	
i mordis minori per resion="0.0.1", description=app_name, version="0.0.1", description=app_name, swagger_ui_parame jeger:0.00 from tatai_textient inport fortilizer from tatai_textient inport foretheaddeleare, hadig_metrics from tatai_textuer inport fromthaddeleare, hadig_metrics from tatai_textient inport foretheaddeleare, hadig_metrics from tatai_textuer inport	Migrator	ann name - 'hello'
Imm diag, major at case time opp. add. stidth case time Spp. add. stidth case time opp. add. stidth case time Imm target spit. Sp		ann - EastAPT(fitte-ann name, version-10.0.1) description-ann name, swanner ui paramet
import 201 Opprovidence in the soft of	from redis_om import get_redis_connection	app = rd middleware(
im strukticapiteritetti por institutti im strukticapiteri hori institutti im strukticapiteri hori herikantikeleari, kaolu attrics profixe app ; tabeta=("sorvice": "hello",	Indext. CSV	PromotheusMidd) avare
In advertig day for upper memory memory memory memory and the set of the set	Trom Tastabl testclient import lestclient	
tabels=("service": "hello",	From startette exporter inport Prometheuskubateware, manute metrics	prefixe spot
service": "hello",		lable-
Selvice . Hetto,		Courteent Thellet
And a second sec	and a star - United lines	service. Herco,
section and the sector of the	approximate - supervise and - Supervised approximate participation of a supervised of a supervised and a supervised and a supervised and the super	aroup patherTrue
an of followed	and add add add and a construction of the second se	buckets=100;
uprogrammer up and a second se	Prostbautid (di)ware	skin pathe [//dest. //oppand icon! //i]
aro newskowy	are lanead to start.	skip_paths=(/docs, /openapi.json, /),
profile and	arefixe and	Skip_methods={ or rioks };
180155	label set	and another (II (another and II) another another (II)
"service": "molone".	"service": "employee".	approve (meetics) manute_meetics)
grup paths/rup,	group paths=True,	
butkets=(8.1, 8.5, 1, 2.5, 18),	buckets=(0.1, 0.5, 1, 2.5, 10),	logger = (ogging.getCoggr(app_name)
skip_paths=(/docs', '/opmapi_jstat', '/'), O Do you went to instal the recommended Bain	skip_paths=['/docs', '/optnapi.json', '/'],	togget sectered (togging. Debug)
dia subdes(/0010021)	ckie settede=(100T0001)	CI = COUNTICEST CANADATE (1)

(a) test-app-employee main.py

(b) test-app-hello main.py



3.2 Application UI

The application looks like as in figure 4.

hello (CA: 8.3) Agenta Jun helo	
default	^
GET / Root	6 🗸
GET /api/vl/hello Helo	~
GET /api/vl/fallible/{pass_weight}/{fail_weight} Fallble Holo	~
GET /api/vl/employee_id} GetEmployee	~
GET /api/vl/httpbin/headers GetEmpkyee	~
Schemas	^
HTTPValidationError > Expand all object	
ValidationError > Depart all edject	



3.3 **REDIS** Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: redis
    namespace: test-app
    labels:
        app: redis
spec:
        selector:
        matchLabels:
        app: redis
```

```
replicas: 1
strategy:
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
  type: RollingUpdate
template:
  metadata:
    labels:
            redis
      app:
  spec:
    containers:
    – name: redis
      image: docker.io/redis/redis-stack-server:latest
      resources:
        limits:
          memory: 128Mi
        requests:
          memory: 128Mi
      livenessProbe:
        tcpSocket:
          port: 6379
        initialDelaySeconds: 3
        timeoutSeconds: 2
        successThreshold: 1
        failureThreshold: 3
        periodSeconds: 10
      readinessProbe:
        tcpSocket:
          port: 6379
        initialDelaySeconds: 3
        timeoutSeconds: 2
        successThreshold: 1
        failureThreshold: 3
        periodSeconds: 10
      ports:
      - containerPort: 6379
        name:
               redis
    restartPolicy: Always
```

3.4 Deploying the application

Docker image is created for the microservices and pushed to docker as in figure 6. The same image name is used in YAML file for the deployment. The microservice will be deployed by running below command.



Figure 5: Exposing test-app-hello

```
#kubectl apply -f test-app-hello.yaml
```

shubham4294singh@cloudshell:~\$ kubectl get pods --all-namespaces -o=jsonpath='{range .items[*]}{.spec.containers[*].image}{"\n"}{end}' | sort | uniq locker.io/hihellobolke/test-app-employee:2 docker.io/istio/proxyv2:1.20.0



For exposing the application to outside world a service of type load balancer named test-app-hello-ext is created and exposed to outer world on IP-34.134.225.91 as in figure 5.

#kubectl apply -f test-app-hello-ext.yaml

Once this is done our both microservices are up and running having 1 pod each. But as we will be injecting failures in later stage we need more number of pods. For this edit the deployment yaml file and changed the replica count to 5 for both the microservices.

4 Installation and Configuration of Locust on AWS

Locust is installed on AWS EC2 instance, so first create an EC2 instance by logging into AWS account and the configurations of instance is visible in figure 7 Amazon Web Services (2023).

Now connect to created EC2 instance from the local command prompt using SSH.

Instance summary for i-012b47d27ff01b4ca (x2 Updated less than a minute ago	2170341_RIC) Info	Connect Instance state V Actions V
Instance ID D I-012b47d27ff01b4ca (x22170341_RIC) IPv6 address -	Public IPv4 address Instance state Running	Private IPv4 addresses 17 172.31.35.78 Public IPv4 DNS 17 ec2-15-229-83-80.sa-east-1.compute.amazonaws.com Jopen address [7]
Hostname type IP name: ip-172-31-35-78.sa-east-1.compute.internal Answer private resource DNS name IPv4 (A)	Private IP DNS name (IPv4 only) ip-172-31-35-78.sa-east-1.compute.internal Instance type t2.large	Elastic IP addresses -

Figure 7: AWS Instance

```
#ssh -i "ric -2.pem" ubuntu@ec2-15-229-83-80.sa-east-1.
    compute.
    amazonaws.com
```

Now first install the python on EC2 instance and after that set up a Python virtual environment using Python 3.10.

#sudo apt install python3 - installing python

#sudo apt update //Update the package lists to get the latest available versions #sudo apt install python3.10-venv //Install Python 3.10 venv package # python3.10 -m venv .venv //Create a virtual environment named .venv using Python 3.10

Now activate the virtual environment using the below command.

#source .venv/bin/activate //Activate the virtual
 environment

After running these commands in order, a virtual environment called.venv will be created, activated for usage, and the Python 3.10 venv package will be installed. Any installation or execution of Python-related software will be contained within the virtual environment when it has been activated.

Now install locust using command

```
#pip3 install -r requirements.txt
```

pip reads the requirements.txt file, locates each package mentioned along with its version, and installs them into your Python environment.

Once the installation is complete verify the installation by running the below command as in figure 8. ubuntu@ip-172-31-35-78:~/ws/locust\$ locust --version
locust 1.4.3

Figure 8: Locust

5 Installation & Configuration of Chaos Engineering

In the GKE cluster create a directory called chaos and inside it write a bash script to inject the failure in the system as in figure 9.

In a Kubernetes cluster, the script restarts a selected number of pods from the testapp-employee deployment. After retrieving a list of ready pods, it chooses a certain number at random and restarts them at a set period of time. The absence of pods is logged for a restart if none of the pods satisfy the requirements.

#!/bin/	/bash
#set -e	euo pipefail
max_pod kill_in #set -x	is_to_kill="\${1:-3}" iterval="\${2:-5}" <
while t	rue; do
pod	<pre>i_list="\$(kubectl -n test-app get pods \ selector "app in (test-app-employee)" \ -o custom-columns=POD:metadata.name,READY-true:status.containerStatuses[*].ready \ -n-no-headers \ grep true \ awk '[print 5]}' \ shuf -n "\${max_pods_to_kill}" \ xargs echo)"</pre>
if els fi	<pre>[["x\${pod_list}" == "x"]]; then echo "\$(date +%y%m%d-%H%M%S) - no pods to kill" ee echo "\$(date +%y%m%d-%H%M%S) - deleting pods: \${pod_list}" kubectl -n test-app delete pod \${pod_list} 2>\$1 sed 's/^/ /'</pre>
sle	ep "\${kill_interval}"

Figure 9: Chaos Engineering Bash Script

6 Installation & Configuration of Service Mesh Istio

To install Istio first create a separate namespace called istio-system. Now install Istio and create an ingress gateway and a service called VirtualService as in figure 10 and 11. Istio Documentation (2023).

```
#istioctl install — set values.pilot.env.
PILOT_ENABLE_STATUS=true — set values.pilot.env.
PILOT_ENABLE_CONFIG_DISTRIBUTION_TRACKING=true — set
values.global.istiod.enableAnalysis=true
```

Now istio-ingressgateway will be exposed on an external IP at 35.202.105.8. Create Istiod service as type of ClusterIP as in figure 12.



Figure 10: Virtual Services



Figure 11: Ingress Gateway

shubham4294singh@cloudshell:~/to	est-app-employee/k8s\$	kubectl get ser	vices -n istio-system	
NAME TYPE	CLUSTER-IP	EXTERNAL-IP	PORT (S)	AGE
istio-ingressgateway LoadBala	ncer 10.100.8.232	35.202.105.82	15021:32478/TCP,80:31924/TCP,443:32653/TCP	11d
istiod ClusterI	10.100.9.173	<none></none>	15010/TCP,15012/TCP,443/TCP,15014/TCP	11d

Figure 12: istio-system Services

7 Simulating Load on Microservice

To start load testing and hitting the microservice with large user requests a bash script is written as in figure 13.

Now to run this bash script first make sure the virtual environemnt is activated. After that navigate to path having the bash script. We can run it using ./test-employee.sh

<pre>ubuntu@ip-172-31-35-78:~/ws/locust\$ cat test-employee.sh #!/bin/bash</pre>
export _now="\$(date +%y%m%d-%H%M%S)" export _ip="\${1-34.134.225.91}" export _locustfile="employee.py"
<pre>locustheadless \ users 400\ spawn-rate 100 \ run-time 3600s \ reset-stats \ locustfile "locustfiles/\${_locustfile}" \ host http://\${_ip} \ host http://\${_ip} \ </pre>
locustfile "locustfiles/\${_locustfile}" \ host http://\${_ip} \ html "reports/\${_locustfile}-\$(hostname)-\${_now}.html"

Figure 13: Bash Script-Load Testing

2	GET /api/v1/employee/709: BadStatusCode('http://34.134.225.91/api/v1/employee/709', code=50	(00
1	GET /api/v1/employee/938: BadStatusCode('http://34.134.225.91/api/v1/employee/938', code=56	(00
1	GET /api/v1/employee/711: BadStatusCode('http://34.134.225.91/api/v1/employee/711', code=50	(00
1	GET /api/v1/employee/559: BadStatusCode('http://34.134.225.91/api/v1/employee/559', code=56	(00
2	GET /api/v1/employee/513: BadStatusCode('http://34.134.225.91/api/v1/employee/513', code=50	(00
2	GET /api/v1/employee/764: BadStatusCode('http://34.134.225.91/api/v1/employee/764', code=50	(00
3	GET /api/v1/employee/366: BadStatusCode('http://34.134.225.91/api/v1/employee/366', code=56	(00
1	GET /api/v1/employee/919: BadStatusCode('http://34.134.225.91/api/v1/employee/919', code=50	(00
3	GET /api/v1/employee/406: BadStatusCode('http://34.134.225.91/api/v1/employee/406', code=50	(00
1	GET /api/v1/employee/163: BadStatusCode('http://34.134.225.91/api/v1/employee/163', code=50	(00
2	GET /api/v1/employee/457: BadStatusCode('http://34.134.225.91/api/v1/employee/457', code=50	(00
1	GET /api/v1/employee/863: BadStatusCode('http://34.134.225.91/api/v1/employee/863', code=50	(00
1	GET /api/v1/employee/124: BadStatusCode('http://34.134.225.91/api/v1/employee/124', code=50	00)

Figure 14: Locust Execution

L	_ocus	t Test Repor	t							
D	During: 26/11/2023, 19:28:38 - 26/11/2023, 19:38:36									
Ta	arget Host: h	nttp://34.134.225.91								
s	cript: employ	yee.py								
F	Request	Statistics								
	Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
	GET	/api/v1/employee/1	65		225	146	4392	341	0.1	0.0
	GET	/api/v1/employee/10	50		179	146	442	345	0.1	0.0
	GET	/api/v1/employee/100	61		169	146	610	353	0.1	0.0
	GET	/api/v1/employee/101	54		164	145	421	359	0.1	0.0
	GET	/api/v1/employee/102	52		164	145	386	353	0.1	0.0
	GET	/api/v1/employee/103	67		260	147	3737	345	0.1	0.0
	GET	/api/v1/employee/104	67		205	145	2984	347	0.1	0.0

Figure 15: Locust Test Execution Result

The system searches the current directory for a file called test-employee.sh when you type ./test-employee.sh, then tries to launch it using the default shell interpreter Locust

Documentation (2023).

Once the test run completes will get the result in the form of a .html file as in figure 15 and 16.



Figure 16: Locust Test Execution Result

8 Injecting the Failures

To inject the failure run the restart-pods-randomly.sh bash script.

Keep changing the max_pods_to_kill and kill_interval. Observe the behaviour of microservice everytime the numbers are changed as in figure 17.

```
#max pods to_kill="${1:-3)"
#kill interval="$(2:-5)"
```

shubham4294singh@cloudshell:~/chaos\$ cat restart-pods-randomly.sh #!/bin/bash
#set -euo pipefail
<pre>max_pods_to_kill="\${1:-3}" kill_interval="\${2:-5}" #set -x</pre>
while true; do
<pre>pod_list="\$(kubectl -n test-app get pods \ selector "app in (test-app-employee)" \ -o custom-columnsFOD:metadata.name,READY-true:status.containerStatuses[*].ready \ no-headers \</pre>
if [["x\${pod_list}" == "x"]]; then echo "\${date +%y%m%d-%H%M%S} - no pods to kill" else
echo "\$(date +%y%m%d-%H%M%S) - deleting pods: \$(pod_list)" kubectl -n test-app delete pod \$(pod_list) 2>61 sed 's/^/ /' fi
<pre>sleep "\${kill_interval}" done</pre>

Figure 17: Bash Script to Inject Pod Failure

shubham4294singh@cloudshell:~/chaos\$./restart-pods-randomly.sh	
231130-212628 - deleting pods: test-app-employee-5458b5f756-2vc8k test-app-employee-5458b5f756-fxwh	5 test-app-employee-5458b5f756-gn87x
pod "test-app-employee-5458b5f756-2vc8k" deleted	
pod "test-app-employee-5458b5f756-fxwh6" deleted	
pod "test-app-employee-5458b5f756-gn87x" deleted	
231130-212641 - deleting pods: test-app-employee-5458b5f756-sw749 test-app-employee-5458b5f756-1kbg	test-app-employee-5458b5f756-zh2hr
pod "test-app-employee-5458b5f756-sw749" deleted	
pod "test-app-employee-5458b5f756-1kbg4" deleted	
pod "test-app-employee-5458b5f756-zh2hr" deleted	
231130-212653 - deleting pods: test-app-employee-5458b5f756-8hlkx test-app-employee-5458b5f756-55hb	f test-app-employee-5458b5f756-gz8g8
pod "test-app-employee-5458b5f756-8hlkx" deleted	
pod "test-app-employee-5458b5f756-55nbf" deleted	
pod "test-app-employee-5458b5f756-gz8g8" deleted	

Figure 18: Failure Creation

9 Running the Istio

To run the locust first we need to enable the istio injection. Open the all yaml file and comment out the line mentioning istio-injection disabled as in figure 19.

After that delete the deployment for test-app-employee, test-app-hello, and Redis.

```
kubectl delete deployment test-app-employee test-app-
hello redis
```

#kubectl apply -k.

Execute the above command. The command **kubectl apply -k** will read the instructions included in the kustomization.yaml file and apply the appropriate resources to the Kubernetes cluster in accordance with the configurations specified in it.

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
namespace: test-app
resources:
```

```
deployments/all.yaml
ns/all.yaml
servicemonitor/all.yaml
svc/all.yaml
istio/all.yaml
```

After the deployment are available the test can be run with Istio enabled.



Figure 19: Istio-Injection

References

Amazon Web Services (2023). Amazon ec2 user guide for linux instances. Retrieved November 29, 2023.

URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Google Cloud (2023). Deploy an application to a cluster — kubernetes engine documentation. Retrieved November 29, 2023.

URL: https://cloud.google.com/kubernetes-engine/docs/deploy-app-cluster

Istio Documentation (2023). Installing istio — istio documentation. Retrieved November 29, 2023. URL: https://istio.io/latest/docs/setup/install/istioctl/

Locust Documentation (2023). Writing a locustfile — locust documentation. Retrieved November 29, 2023. URL: https://docs.locust.io/en/stable/writing-a-locustfile.html