National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Nikhil Mondhe
Student ID: x21174105

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nikhil Mondhe
Student ID:	x21174105
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Rejwanul Haque
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	356
Page Count:	8

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:	Nikhil Mondhe
Date:	13th August 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

Nikhil Mondhe
x21174105

1 Introduction

This configuration manual provides in-depth, step-by-step instructions for installing, configuring, and deploying all of the software, tools, and files that are necessary for the implementation of the proposed system.

2 Prerequisites

AWS Kubernetes Cluster		
EC2 Instance	AWS	t3.medium
OS	Ubuntu	Version (20.04)
Orchestration tool	Kubernetes	Version = 1.19
Container tool	Docker	Version = 18.01

Table 1: Cluster Configurations

Virtual Machine (VPC)	
vCPU	8
Memory	32GiB
Network Performance	5 gbps upto
Cost of services	$0.3741/ hr

Table 2: Virtual machine Configurations

3 Implementation

3.0.1 Install required tools

- Install the AWS Command Line Interface (CLI) and kubectl.
- Configure AWS CLI with your credentials using aws configure.

3.0.2 Create an Amazon EKS Cluster

Usign following commands:-

"aws eks create-cluster –name clustername –role-arn your-eks-role-arn –resources-vpc-config subnetIds=subnet-1,subnet-2,securityGroupIds=sg-1"

3.0.3 Configure Kubectl to Use the Cluster

aws eks update-kubeconfig –name eks

3.0.4 Verify Cluster

```
(base) ➜ eks eksctl get cluster

NAME    REGION          EKSCTL CREATED
eks     us-east-1       True
(base) ➜ eks kubectl get nodes -o wide
NAME                           STATUS   ROLES    AGE   VERSION              INTERNAL-IP      EXTERNAL-IP      OS-IMAGE         KERNEL-VERSION                CONTAINER-RUNTIME
ip-192-168-36-130.ec2.internal   Ready    <none>   21h   v1.26.6-eks-a5565ad   192.168.36.130   18.206.222.214   Amazon Linux 2   5.10.184-175.731.amzn2.x86_64   containerd://1.6.19
```

Figure 1: Verifying EKS Cluster

3.0.5 Create Kubernetes Deployment YAML

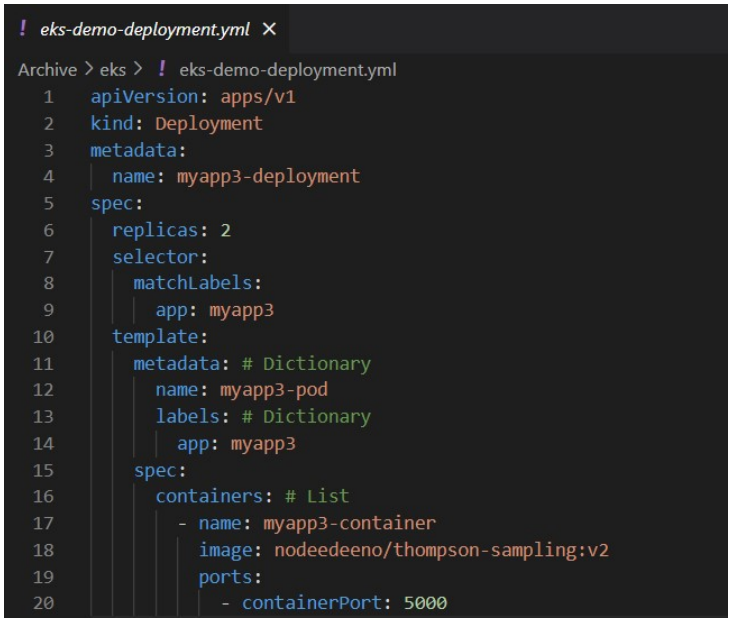

Figure 2: Deployment YAML File

Run the following command to apply the deployment configuration:

kubectl apply -f eks.demo.deployment.yaml

To Verify the deployment commands :

```
(base) ➜ eks kubectl get Deployment -A
NAMESPACE     NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
default       grafana                       1/1     1            1           46h
default       myapp3-deployment             2/2     2            2           47h
default       prometheus-operator           1/1     1            1           46h
default       thompson-nikhil-deployment    1/1     1            1           17h
default       thompson-scheduler            1/1     1            1           19h
kube-system   coredns                       2/2     2            2           2d18h
monitoring    prometheus-deployment         1/1     1            1           17h
(base) ➜ eks
```

Figure 3: EKS Deployment

3.0.6 Expose the Service to the cluster

First, a Service YAML file must be created.

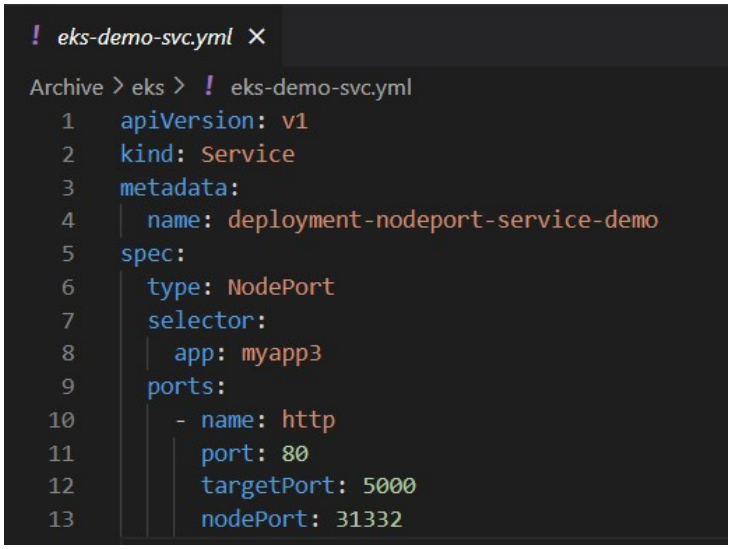

Figure 4: Service YAML file

Apply the service configuration using the kubectl apply command:

kubectl apply -f my-app-service.yaml

To Verify the service commands:

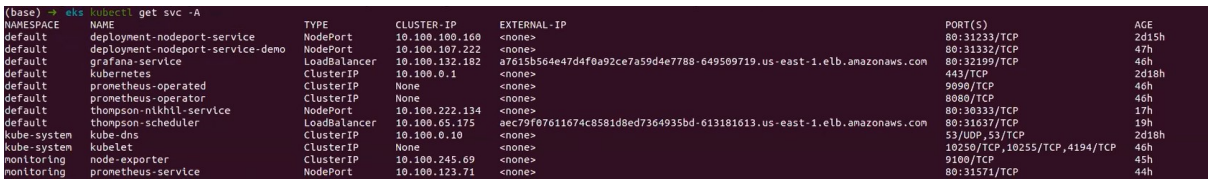

Figure 5: Expose the Service

To get the pods following commands:

```
(base) ➜ eks kubectl get pods -A -o wide
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE    IP               NODE                                NOMINATED NODE   READINESS GATES
default       grafana-567959f665-z6vzt                  1/1     Running   0          46h    192.168.39.3     ip-192-168-36-130.ec2.internal   <none>           <none>
default       myapp3-deployment-6d87df5694-cg4mn        1/1     Running   0          47h    192.168.55.133   ip-192-168-36-130.ec2.internal   <none>           <none>
default       myapp3-deployment-6d87df5694-rmmdm        1/1     Running   0          47h    192.168.33.76    ip-192-168-36-130.ec2.internal   <none>           <none>
default       prometheus-operator-98cb56dc9-dzhvw       1/1     Running   0          46h    192.168.43.17    ip-192-168-36-130.ec2.internal   <none>           <none>
default       thompson-nikhil-deployment-57fcfc79-lvckk 1/1     Running   0          17h    192.168.58.223   ip-192-168-36-130.ec2.internal   <none>           <none>
default       thompson-scheduler-6bbc7b6df-k4j6s        1/1     Running   0          19h    192.168.54.36    ip-192-168-36-130.ec2.internal   <none>           <none>
kube-system   aws-node-6kggb                            1/1     Running   0          2d15h  192.168.36.130   ip-192-168-36-130.ec2.internal   <none>           <none>
kube-system   coredns-55fb5d545d-khksz                  1/1     Running   0          2d16h  192.168.55.10    ip-192-168-36-130.ec2.internal   <none>           <none>
kube-system   coredns-55fb5d545d-trr9s                  1/1     Running   0          2d16h  192.168.56.237   ip-192-168-36-130.ec2.internal   <none>           <none>
kube-system   kube-proxy-lgv74                          1/1     Running   0          2d15h  192.168.36.130   ip-192-168-36-130.ec2.internal   <none>           <none>
monitoring    node-exporter-2tw7s                       1/1     Running   0          45h    192.168.57.192   ip-192-168-36-130.ec2.internal   <none>           <none>
monitoring    prometheus-deployment-5c5fff48b7-5snmf    1/1     Running   0          17h    192.168.32.227   ip-192-168-36-130.ec2.internal   <none>           <none>
```

Figure 6: Worker Pods

4 Algorithm Implementation

thompson_nikhil.py

Archive > OG thompson algo > thompson_nikhil.py > ThompsonSampling > update

```python
from flask import Flask, jsonify
import numpy as np

app = Flask(__name__)

class ThompsonSampling:
    def __init__(self, counts, values):
        self.counts = counts
        self.values = values

    def initialize(self, n_arms):
        self.counts = np.zeros(n_arms)
        self.values = np.zeros(n_arms)

    def select_arm(self):
        n_arms = len(self.counts)
        theta_samples = np.random.beta(1 + self.values, 1 + self.counts - self.values)
        return np.argmax(theta_samples)

    def update(self, chosen_arm, reward):
        self.counts[chosen_arm] += 1
        n = self.counts[chosen_arm]
        value = self.values[chosen_arm]
        self.values[chosen_arm] = ((n - 1) / float(n)) * value + (1 / float(n)) * reward

@app.route("/")
def home():
    ts = ThompsonSampling(None, None)
    ts.initialize(10)  # initialize with 10 arms

    # Simulate the selection and update process for a number of steps
    for _ in range(100):
        chosen_arm = ts.select_arm()
        reward = np.random.binomial(1, p=0.1*chosen_arm)  # simulate reward
        ts.update(chosen_arm, reward)

    return jsonify(ts.values.tolist())

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Figure 7: Thompson sampling Algorithm in Python

5 Deployment of Algorithm

- Install Docker using commands

5.0.1 Create a docker file to deployed in Container

```
Dockerfile
Archive > OG thompson algo > Dockerfile
1  # Use an official Python runtime as a parent image
2  FROM python:3.7-slim
3
4  # Set the working directory in the container
5  WORKDIR /app
6
7  # Add the current directory contents into the container at /app
8  ADD . /app
9
10 # Install any needed packages specified in requirements.txt
11 RUN pip install --trusted-host pypi.python.org -r requirements.txt
12
13 # Run ThompsonSampling.py when the container launches
14 CMD ["python", "thompson_nikhil.py"]
15
16 EXPOSE 5000
17
```

Figure 8: Docker file

5.0.2 Deployment on the Cluster using YAML file

```
thompson-nikhil.yml
Archive > eks > thompson-nikhil.yml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: thompson-nikhil-deployment
5    labels:
6      app: thompson-nikhil
7  spec:
8    replicas: 1
9    selector:
10     matchLabels:
11       app: thompson-nikhil
12   template:
13     metadata:
14       labels:
15         app: thompson-nikhil
16     spec:
17       containers:
18       - name: thompson-nikhil
19         image: nodeedeeno/thompson_nikhil:v2
20         ports:
21         - containerPort: 5005
```

Figure 9: Deployment Algorithm Yaml file

6 Installation of Monitoring and Visualisation Tools

Created a kubernetes service of Grafana and Prometheus using YAML file.

Figure 10: Garfana Service Yaml File

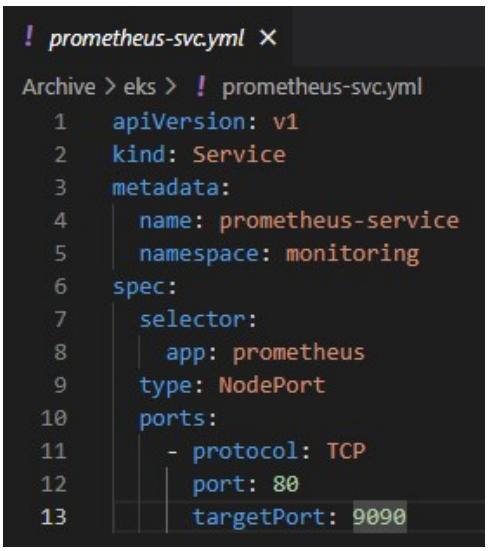

Figure 11: Prometheus Service Yaml File

Deploy this service using create a deployment yaml file and deployed using kubectl commands as shown below:

kubectl apply -f filename.yaml

Below are the yaml files for deployment this tools :

Figure 12: Garfana Deployment Yaml File

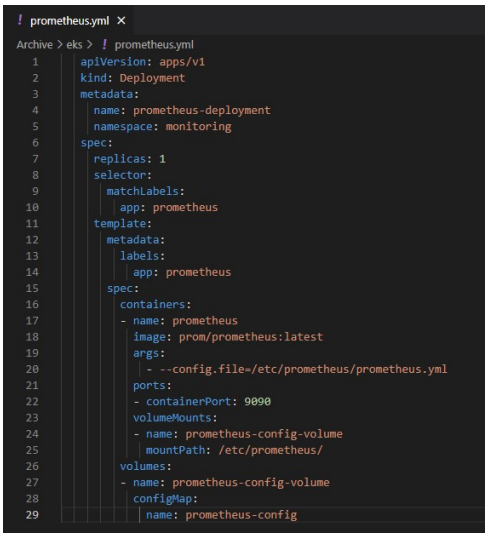

Figure 13: Prometheus Deployment Yaml File

7 Generate Load on Application

We need to install the Locust tool to generate the application load for testing the system. Refer to the official Locust documentation for installation and configuration.

After we setup the locust tool we need to mention the users for generating the load and requests per second for 5 minutes.

8 Observations

Grafana Dashboard is used to display comprehensive insights regarding memory performance and CPU utilisation as shown in Figure 14

Figure 14: CPU Memory Utilization