National College of Ireland

Configuration Manual

MSc Research Project
MSc Cloud Computing

Saiteja Konda
Student ID: 23411520

School of Computing
National College of Ireland

Supervisor: Prof Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Saiteja Konda
Student ID:	23411520
Programme:	MSc Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Prof Aqeel Kazmi
Submission Due Date:	28/01/2026
Project Title:	Configuration Manual
Word Count:	899
Page Count:	16

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:	Saiteja Konda
Date:	28th January 2026

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

Saiteja Konda
23411520

1 Github Link

https://github.com/saitejanci/kubernetes-custom-scheduler.git

2 Introduction

This project improves Kubernetes autoscaling for machine learning workloads by adding a smarter, resource-aware custom scheduler that better understands the unpredictable GPU and memory needs of ML jobs, instead of relying only on the usual static requests and limits. The manual you're reading is a complete, straightforward guide that walks you through setting up the entire environment on AWS with Cloud9 and EC2 instances, installing Prometheus and Grafana for monitoring, and lists every prerequisite—hardware, software versions, AWS services, and tools—so anyone can replicate the setup, run their own training jobs, test the new autoscaling behavior, and verify that it actually works better in practice.

3 Hardware Requirements

Table 1 summarises the minimum hardware requirements for running the experimental setup used in this project.

Component	**Minimum Requirement**
vCPUs	4
RAM	8–16 GB
Storage	30 GB
Network	Stable internet connection

Table 1: Hardware requirements for the experimental environment.

4 Software Requirements

The following software stack is required to configure, deploy, and execute the Kubernetes-based experimental environment used in this project. These tools support cluster creation, workload execution, custom scheduler deployment, monitoring, and performance visualisation.

Software	Version / Description
Python	3.8+
Docker	Docker CE
Minikube	For single-node Kubernetes cluster
Kubectl	Kubernetes command-line tool
Prometheus Stack	Monitoring and metrics collection
Grafana	Metrics visualisation
Git	Version control
IDE	VS Code or AWS Cloud9

5 Tools and Technologies Used

5.1 Backend Language

Python: Used to develop the custom scheduler logic, metric collection scripts, and automation utilities that interact with Kubernetes APIs and support workload orchestration.

5.2 Cluster Tools

Minikube: Provides a local single-node Kubernetes cluster used to test custom scheduler behaviour and autoscaling logic.

Kubectl: Command-line interface for managing Kubernetes resources, deploying workloads, and inspecting pod and node status.

Docker: Container runtime used to package ML workloads, custom scheduler components, and auxiliary services into portable container images.

5.3 Monitoring Tools

Prometheus: Captures CPU utilisation, memory usage, pod scaling events, and node-level metrics for both the default and custom schedulers.

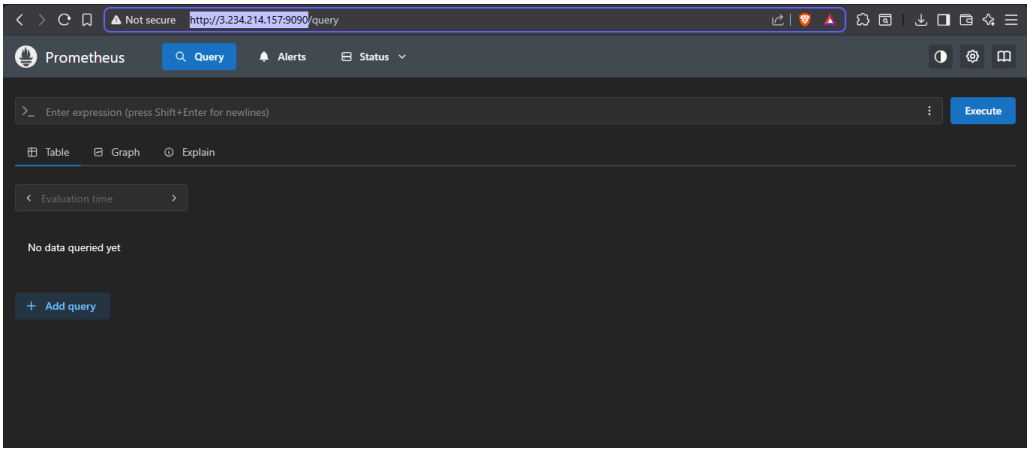


Figure: Prometheus dashboard showing real-time metrics used for autoscaling decisions.

Grafana: Visualises Prometheus metrics and generates dashboards used to compare performance and resource utilisation across scheduler types.

Figure: Grafana dashboard visualising CPU, memory, and pod replica metrics for both schedulers.

5.4 Data & Utility Libraries

- **numpy**: Efficient numerical operations for workload generation and metric analysis.
- **pandas**: Tabular data processing for experiment results and summarisation.
- **psutil**: System-level resource usage capture for ML workloads, including CPU and memory utilisation.

6 Project Structure

7 Environment Setup

7.1 Install Dependencies

```
sudo apt update
sudo apt install -y docker.io

# Minikube
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube

# kubectl
sudo install kubectl /usr/local/bin/kubectl
```

7.2 Start Minikube

```
minikube start --cpus=2 --memory=4096
```

8 Prometheus and Grafana Setup

8.1 Deploy Prometheus

```
kubectl apply -f prometheus-deployment.yaml    { for deploying Prometheus in default
kubectl apply -f prometheus-service.yaml       { Exposing Prometheus via a cluster IP
```

```
voclabs:~/environment $ ls *prometheus*
prometheus-deployment.yaml  prometheus-service.yaml
```

Verify Prometheus is Running state:

```
kubectl get pods | grep Prometheus
```

Access the Prometheus Web UI:

```
kubectl port-forward --address 0.0.0.0 pod/prometheus-server-{pod name} 9090:9090
kubectl port-forward svc/prometheus-server 9090:9090 --address 0.0.0.0
```

We are using Prometheus to scrape the metrics like real time CPU, memory metrics of the pod.

8.2 Deploy Grafana

Apply Grafana:

```
kubectl apply -f grafana-config.yaml
kubectl apply -f grafana-pvc.yaml
kubectl apply -f grafana-deployment.yaml
```

```
voclabs:~/environment $ ls *grafana*
grafana-config.yaml  grafana-deployment.yaml  grafana-pvc.yaml
```

Verify Grafana is running:

```
kubectl get pods | grep Grafana
```

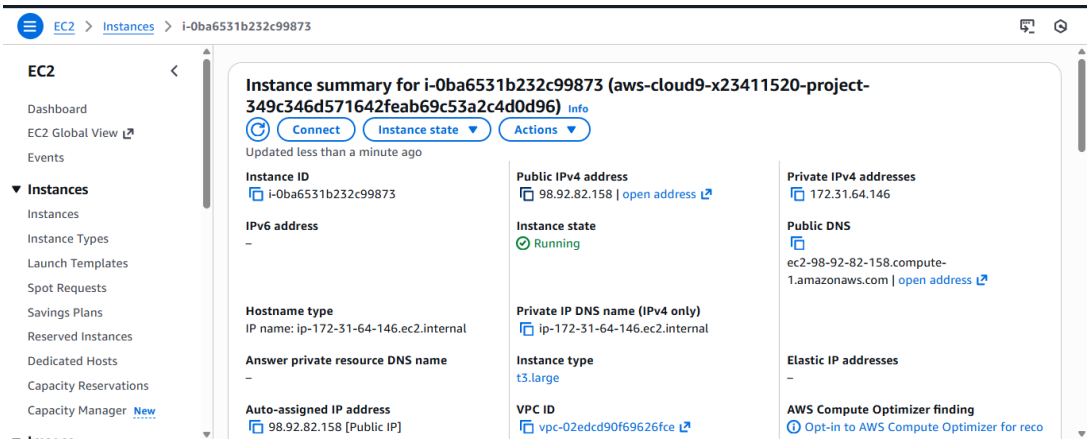
Access the Grafana web UI:

```
kubectl port-forward --address 0.0.0.0 pod/Grafana-{Grafana-pod-name} 3030:3000
```

9 AWS Services Used

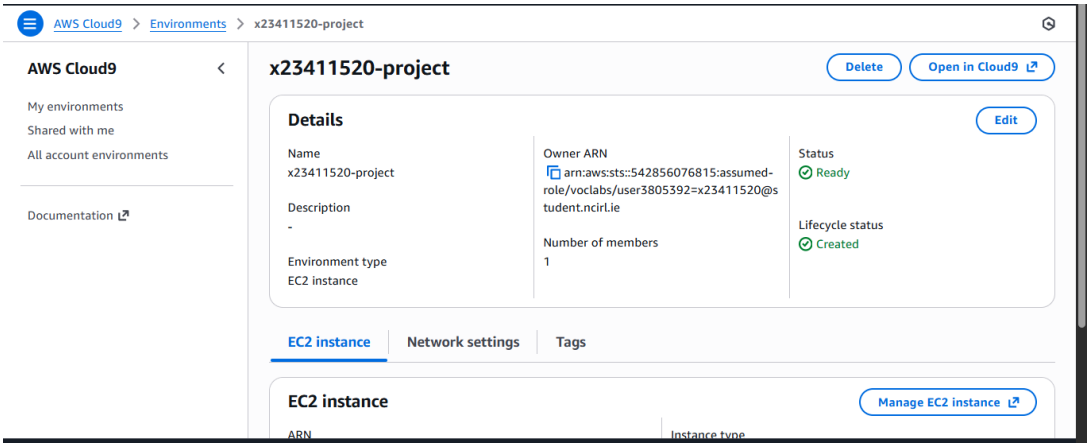
9.1 Amazon EC2

Used to host the environment where Kubernetes, Minikube, Docker, Prometheus, and Grafana are installed. Acts as compute infrastructure for ML workload execution and autoscaling tests.



9.2 AWS Cloud9

Provides a cloud-based IDE to manage development, cluster configuration, custom scheduler coding, testing scripts, and monitoring setup.



10 Project Configuration Flow

The overall configuration and execution flow for the project is:

1. Install and set up Minikube, Prometheus, Grafana, `kubectl`, and Docker on the AWS Cloud9 instance.
2. Create a Kubernetes cluster using Minikube.
3. Create a pod inside the cluster using the default Kubernetes scheduler.
4. Implement and configure the custom scheduler.

5. Create a pod inside the cluster that is scheduled using the custom scheduler.
6. Apply the ML workload on both pods (default scheduler pod and custom scheduler pod).
7. Collect metrics using Prometheus.
8. Display and compare metrics on Grafana.

11 Important Commands

This section lists the key commands used to set up the environment and execute the ML workloads.

1. Start Minikube Cluster

```
minikube start -p minikube-scheduler
```

2. Start and Port-Forward Prometheus and Grafana

```
kubectl port-forward --address 0.0.0.0 pod/grafana-b96c94d44-2fddd 3030:3000

kubectl port-forward --address 0.0.0.0 pod/prometheus-server-d599cf8dc-kmvn6 9090:909
```

3. Execute ML Workload in Pod Using Default Scheduler

```
kubectl exec -it ml-default -- /bin/bash
python3 train.py
```

4. Execute ML Workload in Pod Using Custom Scheduler

```
kubectl exec -it ml-custom-workload-64df4d5c66-vhkdr -- /bin/bash
python3 train.py
```

5. Collect Metrics from Grafana

Once Prometheus and Grafana are running and port-forwarded, use the Grafana web interface to:

- Select the appropriate Prometheus data source.
- Open the dashboards configured for CPU, memory, pod replicas, and node metrics.
- Export or capture graphs for comparison between the default and custom schedulers.

12 Experiment results when executed for 7 min, 15 min and 30 min range run

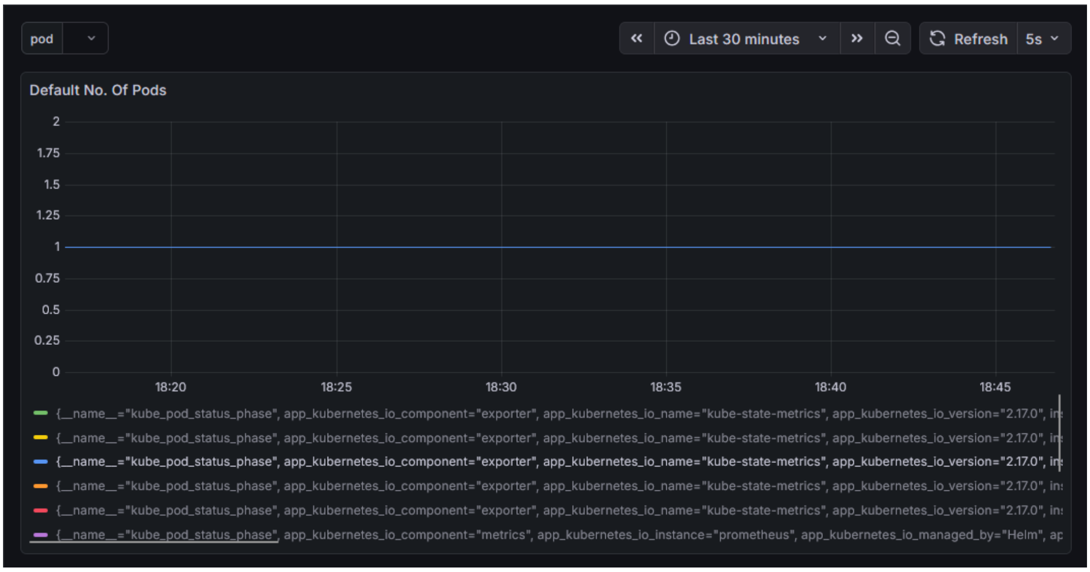
12.1 Experiment 1: Short Run (7 Min)

Default Scheduler:

CPU: CPU Core Utilization for Default Scheduler during 7-Minute Short Run

Memory: Memory Usage Pattern for Default Scheduler during 7-Minute Short Run

Pod: Pod Count for Default Scheduler during 7-Minute Short Run



Custom Scheduler:

CPU: CPU Core Utilization for Custom Scheduler during 7-Minute Short Run

Memory: Memory Usage Pattern for Custom Scheduler during 7-Minute Short Run

Pod: Pod Count for Custom Scheduler during 7-Minute Short Run

HPA Autoscaling Metrics for Custom Scheduler during 7-Minute Short Run

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
ml-custom-workload	Deployment/ml-custom-workload	cpu: 2%/60%	1	10	2	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 23%/60%	1	10	2	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 232%/60%	1	10	1	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 206%/60%	1	10	2	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 110%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 58%/60%	1	10	4	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 67%/60%	1	10	4	6d3h

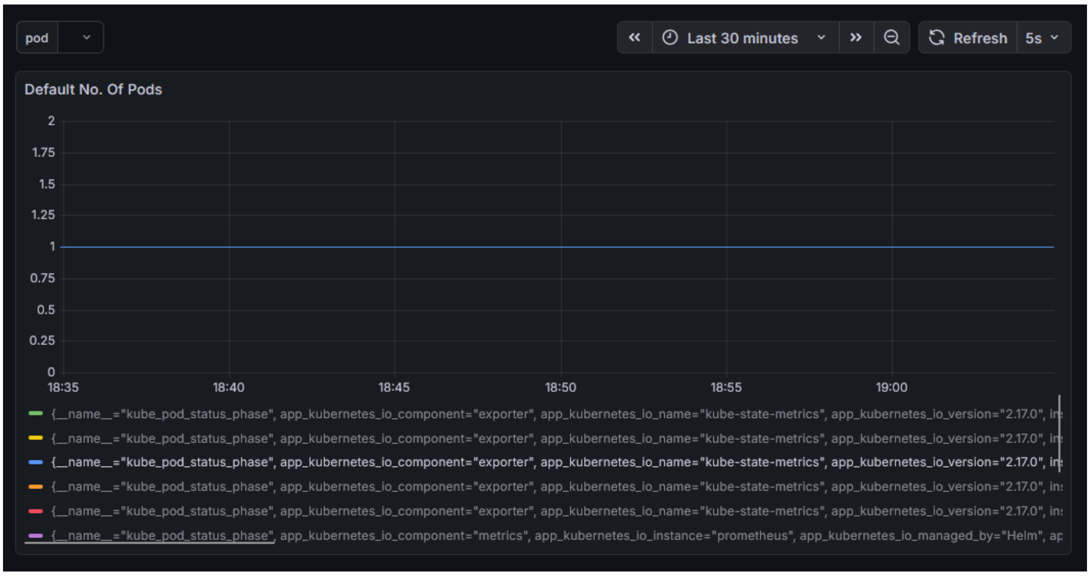
12.2 Experiment 2: Medium Run (15 Min)

Default Scheduler:

CPU: CPU Core Utilization for Default Scheduler during 15-Minute Medium Run

Memory: Memory Usage Pattern for Default Scheduler during 15-Minute Medium Run

Pod: Pod Count for Default Scheduler during 15-Minute Medium Run



Custom Scheduler:
CPU: CPU Core Utilization for Custom Scheduler during 15-Minute Medium Run

Memory: Memory Usage Pattern for Default Scheduler during 15-Minute Medium Run

Pod: Pod Count for Custom Scheduler during 15-Minute Medium Run

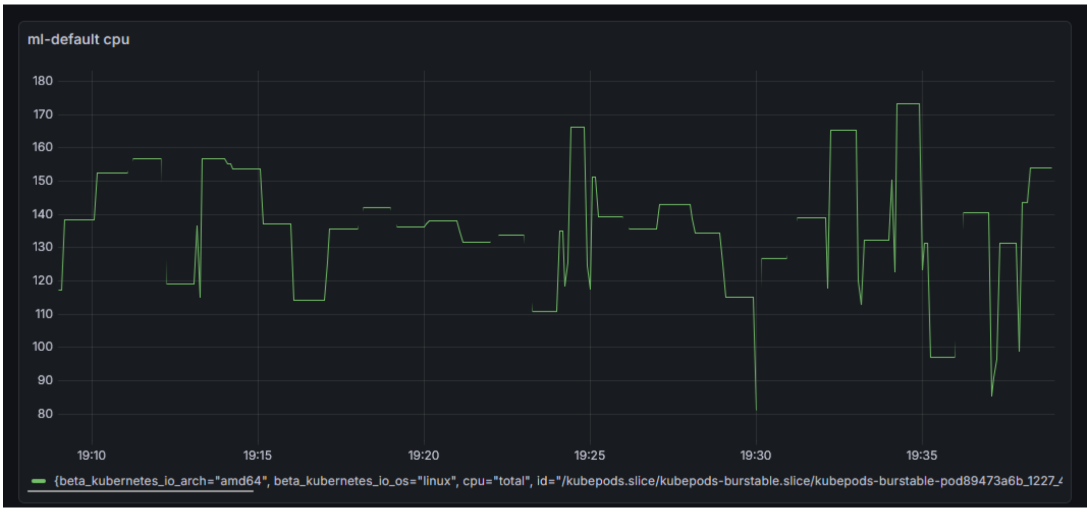
HPA: HPA Autoscaling Metrics for Custom Scheduler during 15-Minute Medium Run

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
ml-custom-workload	Deployment/ml-custom-workload	cpu: 196%/60%	1	10	1	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 292%/60%	1	10	2	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 148%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 42%/60%	1	10	5	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 74%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 50%/60%	1	10	4	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 46%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 56%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 94%/60%	1	10	3	6d3h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 39%/60%	1	10	5	6d3h

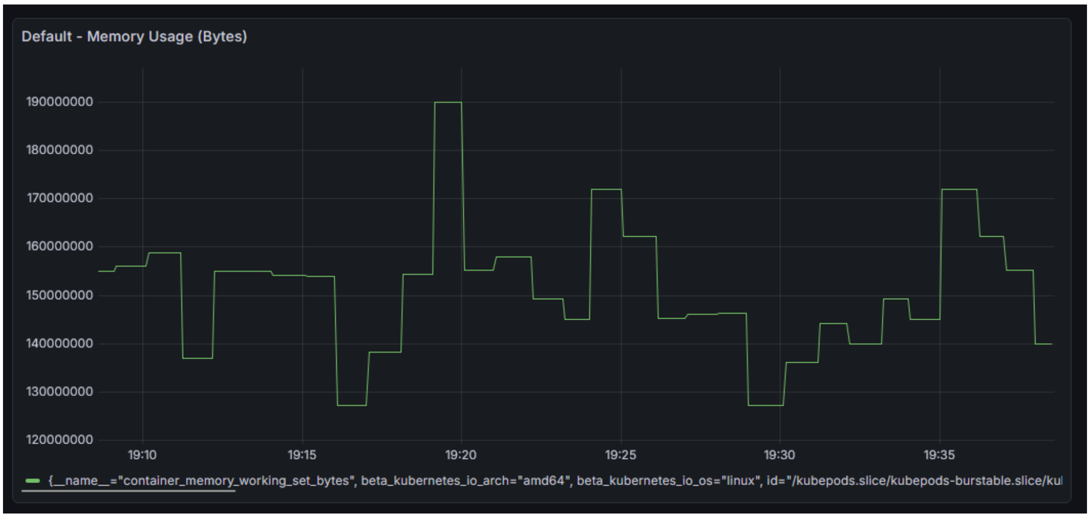
12.3 Experiment 3: Long Run (30 Min)

Default Scheduler:

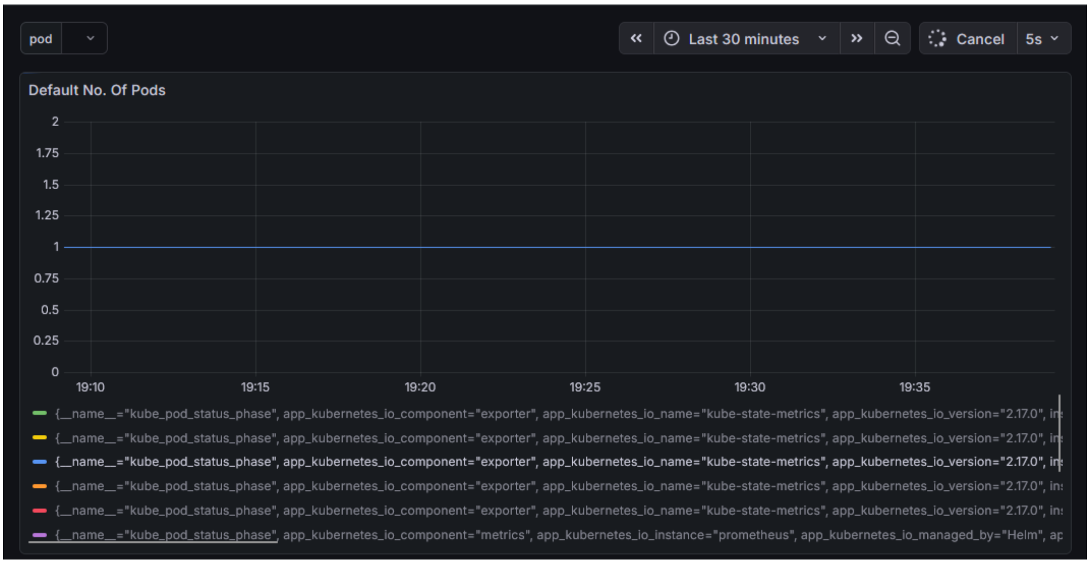
CPU: Core CPU Utilization for Default Scheduler during 30-Minute-Long Run



Memory: Memory Usage Pattern for Default Scheduler during 30-Minute-Long Run



Pod: Pod Count for Default Scheduler during 30-Minute-Long Run



Custom Scheduler:
CPU: CPU Core Utilization for Custom Scheduler during 30-Minute-Long Run

Memory: Memory Usage Pattern for Custom Scheduler during 30-Minute-Long Run

Pod: Pod Count for Custom Scheduler during 30-Minute-Long Run

HPA Autoscaling Metrics for Custom Scheduler during 30-Minute Long Run

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
ml-custom-workload	Deployment/ml-custom-workload	cpu: 76%/60%	1	10	2	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 142%/60%	1	10	3	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 43%/60%	1	10	5	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 44%/60%	1	10	5	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 53%/60%	1	10	4	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 20%/60%	1	10	4	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 109%/60%	1	10	2	6d5h
ml-custom-workload	Deployment/ml-custom-workload	cpu: 88%/60%	1	10	3	6d5h