National College of Ireland

Configuration Manual

MSc Research Project

MSc in Cloud Computing

Sai Kumar Reddy Palnati

Student ID: 24104191

School of Computing

National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sai Kumar Reddy Palnati
Student ID:	24104191
Programme:	MSC in Cloud Computing **Year:** 2025-2026
Module:	Research Project
Lecturer:	Yasantha Samarawickrama
Submission Due Date:	28th January 2026
Project Title:	Reinforcement Learning-Based Cost and Latency Optimization in Kubernetes-Enabled Multi-Cloud Environments
Word Count:	816 **Page Count:** 7

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: Sai Kumar Reddy Palnati

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

Sai Kumar Reddy Palnati

Student ID: 24104191

This configuration manual will provide a complete guide for anyone to replicate my RL-based Kubernetes scheduler. The setup uses local Kind clusters to simulate a multi-cloud environment (AWS and Azure) and integrates real 2025 pricing/latency data. All commands were tested on macOS with the following hardware/software baseline:

1 Tools, Libraries, Frameworks, and Versions Used

Category	Tool/Library/Framework	version	Purpose
Package Manager	Homebrew	4.4+	Installing CLI tools
Container Runtine	Docker Desktop	4.28+	Running Kind clusters
Conda Environment	Miniconda	Latest	Isolated Python env
Python	Python	3.12	Core language
RL Framework	Ray RLib	2.40.0	PPO Training
GYM Environment	Gymnasium	0.29+	Custom env base
Kubernetes Local	Kind (Kubernetes IN Docker)	0.23+	Local clusters
Kubernetes CLI	Kubectl	1.31+	Cluster management
Helm	helm	3.15+	Installing Prometheus
Monitoring	Kube-prometheus-stack(Helm chart)	Latest	Grafana+Prometheus
Load Testing	Locust	2.29+	Workload generation
Data Processing	Pandas,Numpy,Scikit-learn	Latest	Data Normalization
Visualization	TensorBoard,Matplotlib	Latest	Training Graphs

Table 1 : Required Tools, Libraries, Frameworks, and Versions

2 Step-by-Step Setup Instructions

1. **Install Homebrew (if not already installed)**

   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

2. **Install Docker Desktop**

   Download Docker from https://www.docker.com/products/docker-desktop/

   Install and start Docker (grant permissions, allocate 4–6 GB RAM in Settings > Resources)

3. **Install Miniconda**

   brew install --cask miniconda

   conda init zsh # or bash if using bash

   - Restart terminal

4. **Create and Activate Conda Environment**

   conda create -n rl-k8s python=3.12

   conda activate rl-k8s

Figure 1: Activating Conda

5. **Install Python Dependencies**

   pip install ray[rllib]==2.40.0 gymnasium flask pandas numpy scikit-learn locust kubernetes prometheus-api-client

   conda install pytorch torchvision torchaudio -c pytorch

6. **Install Kubernetes Tools**

   brew install kind kubectl helm

7. **Create Kind Clusters (Simulated AWS and Azure)**
   - Create config files in k8s_manifests/:
   - aws-cluster-config.yaml (with node label cloud=aws, podSubnet 10.244.0.0/16)
   - azure-cluster-config.yaml (cloud=azure, podSubnet 10.245.0.0/16)

   kind create cluster --name kind-aws --config k8s_manifests/aws-cluster-config.yaml

   kind create cluster --name kind-azure --config k8s_manifests/azure-cluster-config.yaml

   kubectl config rename-context kind-kind-aws kind-aws

   kubectl config rename-context kind-kind-azure kind-azure

8. **Install Prometheus + Grafana on Both Clusters**

   helm repo add prometheus-community
   https://prometheus-community.github.io/helm-charts

   helm repo update

   # AWS

   kubectl config use-context kind-aws

   helm install prom-aws prometheus-community/kube-prometheus-stack -n monitoring --create-namespace

   # Azure

   kubectl config use-context kind-azure

   helm install prom-azure prometheus-community/kube-prometheus-stack -n monitoring --create-namespace

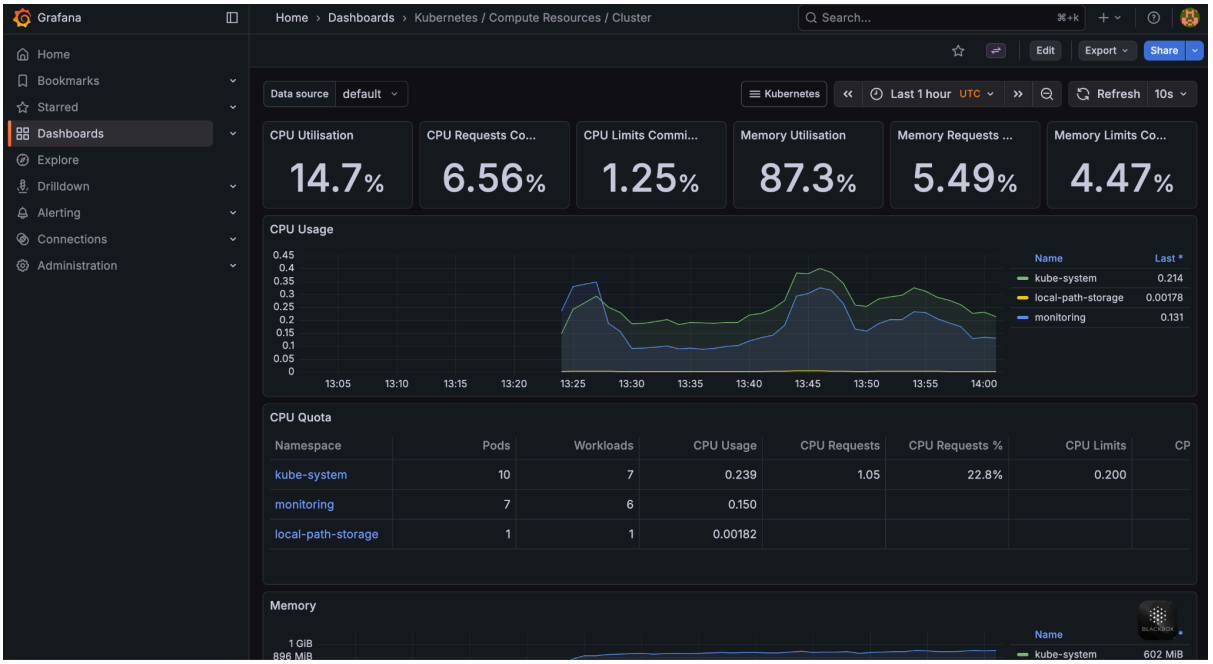

Figure 2: Metrics from Grafana dashboard

9. Port-Forward Grafana (for viewing)

AWS: kubectl port-forward -n monitoring svc/prom-aws-grafana 3000:80

Azure: kubectl port-forward -n monitoring svc/prom-azure-grafana 3001:80

Access at http://localhost:3000 and 3001 (username: admin, password from secret decode)

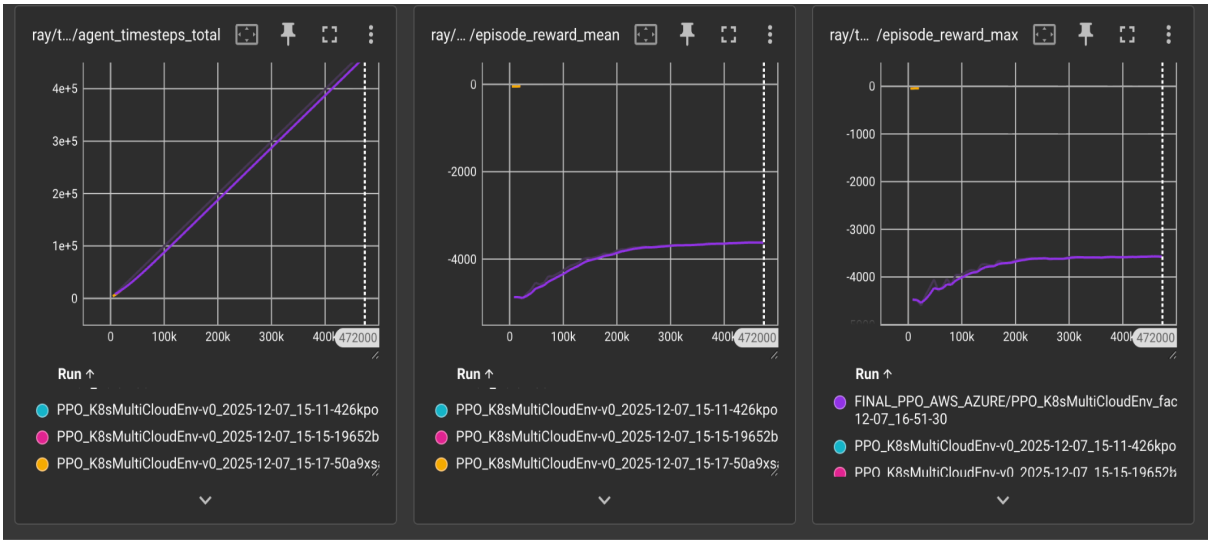

Figure 3: Graphs from Tensorboard

10. Generate Real Pricing/Latency Data

Run generate_real_pricing.py to create data/real_prices.csv and data/real_latencies.csv based on Dec 2025 rates:

11. **Normalize Data**

    python normalize_data.py

3 Training Parameters

Algorithm: PPO (Proximal Policy Optimization)
Framework: Torch with Apple Metal acceleration

Key Hyperparameters:

- train_batch_size: 8000
- sgd_minibatch_size: 512
- rollouts: 6 workers, 4 envs per worker

Hardware Utilization: ~70% CPU, 10–12 GB RAM during training

4 Evaluation

- Baseline: Deploy simple-service.yaml, run Locust without extender.
- RL Scheduler: deploy with custom policy ConfigMap.
- Evaluation: Run final_evaluation.py (100 episodes) or Locust with varying users (20, 100, 300+).

```
2025-12-09 13:39:00,968 WARNING util.py:68 -- Install gputil for GPU system monitoring.
Episode  20 → cost = $-3599.120
Episode  40 → cost = $-3590.862
Episode  60 → cost = $-3602.812
Episode  80 → cost = $-3620.842
Episode 100 → cost = $-3595.316

============================================================
FINAL EVALUATION RESULTS (100 episodes)
============================================================
Average cost per episode       : $-3603.3045
Total decisions total           : 9900
Agent chose AWS                : 4236 times (42.8%)
Agent chose Azure              : 5664 times (57.2%)

Improvement vs greedy baseline ($4.765): 75720.2% better
```

Figure 4: Final Improvement Logs

References

1. Jayanetti, A., Halgamuge, S. & Buyya, R., 2024. A Deep Reinforcement Learning Approach for Cost Optimized Workflow Scheduling in Cloud Computing Environments. In: Proceedings of the 2024 Asia Pacific Conference on Computing Technologies, Communications and Networking (CTCNet 2024). ACM, New York, NY, USA.
Available at: https://doi.org/10.1145/3685767.3685780

2. Jeyasri, S., 2023. Multi-Cloud Strategies for Distributed AI Workflows and Application. International Research Journal of Engineering and Technology (IRJET), 11(11), pp.1–6. Available
at: https://www.irjet.net/archives/V11/i11/IRJET-V1111195.pdf

3. Katakam, H. V. N., 2025. Dynamic Workload Placement across Multi-Clouds: AI-Driven Cost Optimization without Downtime. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 6(4), pp.96–100.
Available at: https://doi.org/10.63282/3050-9262.IJAIDSML-V6I4P114

4.Dwivedi, P. N., Pandey, A. & Tripathi, R., 2025. AI-Powered Cloud Optimization: A Smart Framework for Auto-Healing, Cost Control, and Load Balancing. International Journal of Scientific Research and Engineering Development, 8(4), pp.647–653.
Available at: https://www.ijsred.com/volume8/issue4/IJSRED-V8I4P57.pdf

5.Nunavath, V., 2025. AI-enhanced self-healing Kubernetes for scalable cloud operations. World Journal of Advanced Engineering Technology and Sciences, 16(2), pp.21–29.
Available at: https://doi.org/10.30574/wjaets.2025.16.2.1255

6. Katakam, H. V. N., 2025. Intelligent Spot Instance Management: AI-Driven Decision Framework for Cost-Optimized Cloud Computing. International Journal of Innovative Research in Multidisciplinary and Practical Studies, 13(5), pp.1–12.
Available at: https://www.ijirmps.org/papers/2025/5/232827.pdf

7. Bokra, D. & Khairnar, S., 2025. Machine learning-based cloud resource allocation algorithms: a comprehensive comparative review. Frontiers in Computer Science, 7, Article 1678976.
Available at: https://doi.org/10.3389/fcomp.2025.1678976

8. Diwan, P. D., 2025. GenAI-driven cloud management for AWS and Kubernetes environments. World Journal of Advanced Research and Reviews, 26(1), pp.1475–1484.
Available at: https://doi.org/10.30574/wjarr.2025.26.1.1165

9. Barua, B. & Kaiser, M. S., 2024. AI-Driven Resource Allocation Framework for Microservices in Hybrid Cloud Platforms. arXiv preprint arXiv:2412.02610.
Available at: https://arxiv.org/abs/2412.02610
10. Morabito, R., Kjällman, J. and Komu, M., 2018. Hypervisors vs. lightweight virtualization: A performance comparison. Journal of Cloud Computing: Advances, Systems and Applications, 7(1), pp.1–14.
Available at: https://doi.org/10.1186/s13677-018-0120-2