National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Vikrant Anil Ghode
Student ID: 23324180

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vikrant Anil Ghode
Student ID:	23324180
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	11/12/2025
Project Title:	Cost and Carbon Aware Reinforcement Learning Autoscaling for Multi-Cloud Kubernetes Spot Instances
Word Count:	1860
Page Count:	19

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:	Vikrant Anil Ghode
Date:	10th December 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

Vikrant Anil Ghode
23324180

1 Overview

This configuration manual is a record of the way to install, configure and run the Carbon-The development of Aware Reinforcement Learning Autoscaler was a part of the MSc Research Project. It describes the entire configuration stack of local environment variables all the way to multi-cloud. Deployment and monitoring of Kubernetes.

The multi-layered system configuration of the autoscaler is:

- `.env`: environment variables and cloud credentials.
- `configs/cloud_config.yaml`: multi-cloud infrastructure and autoscaling parameters.
- `configs/training_config.yaml`: reinforcement learning (RL) training hyperparameters and reward weights.
- `k8s/*/deployment.yaml`: K8s manifests of AWS, Azure and GCP deployments.
- `Dockerfile`: container image definition used across all clouds.

This document is systematised in a way that an individual can begin with a clean machine and run the thesis experiments on the following configuration files only.

2 Environment Configuration (`.env`)

All sensitive credentials and high-level environment variables necessary to the autoscaler are stored in the centralised file called the `.env` file. It is dynamically loaded by the application and it should not be placed under version control.

2.1 Example structure (sanitised)

```
# ============================================================================
# AWS CREDENTIALS
# ============================================================================
AWS_ACCESS_KEY_ID=YOUR_AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY=YOUR_AWS_SECRET_ACCESS_KEY
AWS_DEFAULT_REGION=eu-west-1

# ============================================================================
```

```
# AZURE CREDENTIALS
# ==============================================================================
AZURE_SUBSCRIPTION_ID=YOUR_AZURE_SUBSCRIPTION_ID
AZURE_TENANT_ID=YOUR_AZURE_TENANT_ID
AZURE_CLIENT_ID=YOUR_AZURE_CLIENT_ID
AZURE_CLIENT_SECRET=YOUR_AZURE_CLIENT_SECRET

# ==============================================================================
# GCP CREDENTIALS
# ==============================================================================
GCP_PROJECT_ID=your-gcp-project-id
GOOGLE_APPLICATION_CREDENTIALS=/absolute/path/to/service-account-key.json

# ==============================================================================
# THIRD-PARTY APIS
# ==============================================================================
ELECTRICITYMAP_API_KEY=YOUR_ELECTRICITYMAP_API_KEY

# ==============================================================================
# KUBERNETES CONFIGURATION
# ==============================================================================
KUBECONFIG=~/.kube/config

# ==============================================================================
# EXPERIMENT SETTINGS
# ==============================================================================
EXPERIMENT_DURATION_HOURS=24
LOG_LEVEL=INFO
DATA_DIR=/path/to/Carbon-aware-autoscaler/data
RESULTS_DIR=/path/to/Carbon-aware-autoscaler/experiments/results

# ==============================================================================
# BUDGET ALERTS
# ==============================================================================
CLOUD_BUDGET_LIMIT_USD=170
ALERT_THRESHOLD_PERCENT=80
```

2.2 Key parameters

Table 1: Selected `.env` parameters and their roles.

Parameter	Purpose	Example	Notes
`AWS_ACCESS_KEY_ID`	AWS API authentication	–	Required for AWS deployments
`AZURE_SUBSCRIPTION_ID`	Azure subscription identifier	–	Used by Azure SDK
`GCP_PROJECT_ID`	GCP project ID	`named-chariot-01`	Must match project in GCP console
`ELECTRICITYMAP_API_KEY`	Carbon-intensity data source	–	Optional but recommended for carbon-aware runs
`EXPERIMENT_DURATION_HOURS`	Real-world evaluation length	`24`	Can be shortened for test runs
`CLOUD_BUDGET_LIMIT_USD`	Safety budget cap	`170`	Triggers alerts when cumulative cost exceeds this limit

2.3 Security notes

- Do *not* commit the `.env` file; instead commit a `.env.example` template.
- Rotate cloud secrets regularly and prefer IAM roles when necessary.
- Move secrets into a secret manager (AWS Secrets Manager, Azure Key Vault, GCP Secret Manager).

Figure 1: Screenshot of the final `.env` file opened in an editor (with secrets blurred or replaced by placeholders).

3 Cloud Configuration (cloud_config.yaml)

The available clouds, regions and type of instances accessible to the autoscaler and shared autoscaling, SLO, budget and carbon-intensity settings are set in the file `configs/cloud_config.yaml`.

3.1 Core structure

`configs/cloud_config.yaml`

The configuration is divided into the following blocks:

- `clouds`: per-provider regions, instance types and Spot/preemptible settings.
- `kubernetes`: logical cluster definitions for EKS, AKS and GKE.
- `autoscaling`: minimum/maximum pod replicas and CPU thresholds.
- `slo`, `budget`, `carbon`: high-level policy constraints.
- `monitoring`, `logging`: Prometheus/Grafana and logging configuration.

3.2 Multi-cloud example (excerpt)

```
clouds:
  aws:
    enabled: true
    regions:
      - eu-west-1       # Ireland (low carbon)
      - us-east-1
    instance_types:
      - t3.medium
      - t3.large
    spot_settings:
      max_price_multiplier: 1.5
      interruption_handling: migrate

  azure:
    enabled: true
    regions:
      - swedencentral   # Very low carbon
      - westeurope
    instance_types:
      - Standard_B2s
      - Standard_D2s_v3

  gcp:
    enabled: true
    regions:
      - europe-north1   # Finland (low carbon)
      - europe-west1
```

3.3 Autoscaling, SLO and carbon settings

```
autoscaling:
  min_instances: 2
  max_instances: 20
  scale_up_threshold: 0.7
  scale_down_threshold: 0.3
  cooldown_period: 300

slo:
  latency_p95_ms: 200
  availability_target: 0.99

carbon:
  daily_limit_kg_co2: 5
  prefer_green_regions: true
```

These parameters were adjusted in favor of low-carbon regions at a low cost but remaining within budget and keeping the SLOs employed in the thesis experiments.

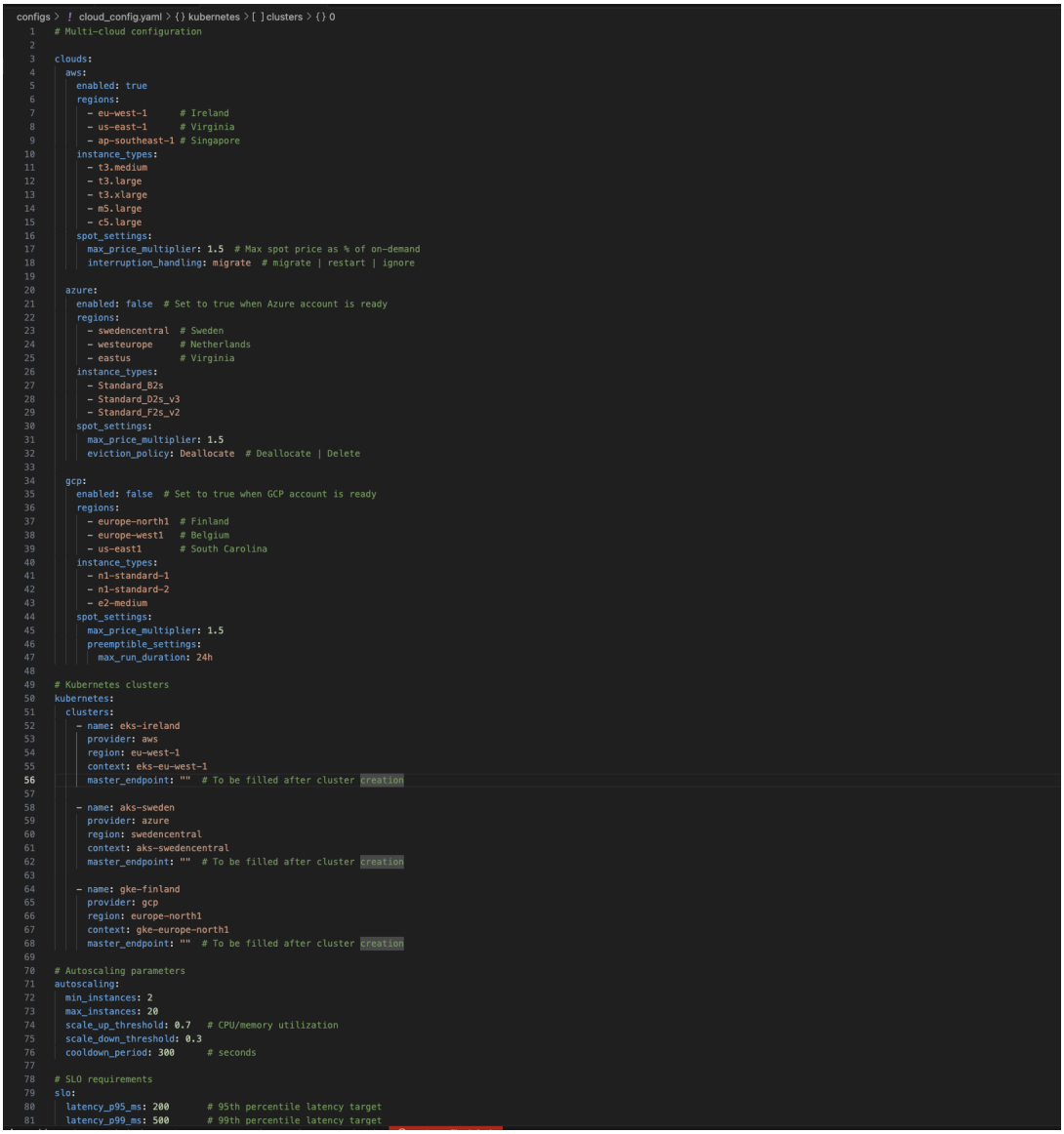

Figure 2: Screenshot of `configs/cloud_config.yaml` showing multi-cloud regions and autoscaling, SLO and carbon settings.

4 Training Configuration (`training_config.yaml`)

The training hyperparameters of the reinforcement learning (RL) training are set in the file in the `configs/training_config.yaml` directory that contains the environment configuration and reward weights to be used to train the carbon-aware autoscaler.

4.1 PPO hyperparameters

The autoscaler uses Proximal Policy Optimisation (PPO). A representative configuration is:

```
ppo:
  learning_rate: 0.0003
  n_steps: 2048
  batch_size: 64
  n_epochs: 10
  gamma: 0.99
  gae_lambda: 0.95
  clip_range: 0.2
  ent_coef: 0.01
  vf_coef: 0.5
  max_grad_norm: 0.5
```

4.2 Environment and reward configuration

```
environment:
  max_steps_per_episode: 1000
  n_parallel_envs: 1
  normalize_observations: true
  normalize_rewards: true

reward:
  cost_weight: 0.3
  carbon_weight: 0.3
  slo_weight: 10.0
  cost_normalization: 10.0
  carbon_normalization: 5.0
```

In the course of experimentation, when the baseline was broken by placing the `slo_weight = 0.2`, 4922.6 SLO violations were obtained and the tuned settings provided above delivered very high SLO compliance. This setting has been the reference set up in the thesis.

4.3 Execution and logging

```
training:
  total_timesteps: 1000000
  eval_freq: 10000
  save_freq: 50000
```

```
  log_interval: 100

tensorboard:
  enabled: true
  log_dir: experiments/logs/tensorboard
```

Figure 3: Screenshot of `configs/training_config.yaml` highlighting the final reward weights and PPO hyperparameters.

5 Kubernetes Deployment Configuration

The auto scaler is implemented in each cloud-specific cluster as a Kubernetes Deployment. The AWS (EKS), Azure (AKS) and GCP (GKE) separate manifests are given under the directory `k8s/` .

5.1 AWS (EKS) deployment

`k8s/aws/deployment.yaml`

Key elements:

- `metadata.labels.cloud = aws` to distinguish cloud provider.
- Container image pulled from Amazon ECR.
- Environment variables `CLOUD_PROVIDER=aws`, `MODEL_PATH`, `DECISION_INTERVAL`.
- CPU/memory requests: 250m / 400Mi; limits: 500m / 800Mi.

5.2 Azure (AKS) deployment

`k8s/azure/deployment.yaml`

Key elements:

- Image hosted in Azure Container Registry (ACR) with `imagePullSecrets`.
- Higher resource requests (500m CPU, 1Gi RAM) to account for VM overhead.
- Azure service principal credentials injected from a Kubernetes secret.

5.3 GCP (GKE) deployment

`k8s/gcp/deployment.yaml`

Key elements:

- Image hosted on Google Container Registry (GCR) with AMD64 architecture.
- Lowest resource requests (100m CPU, 256Mi RAM) and no explicit limits.
- Built-in Prometheus integration via `PROMETHEUS_URL`.

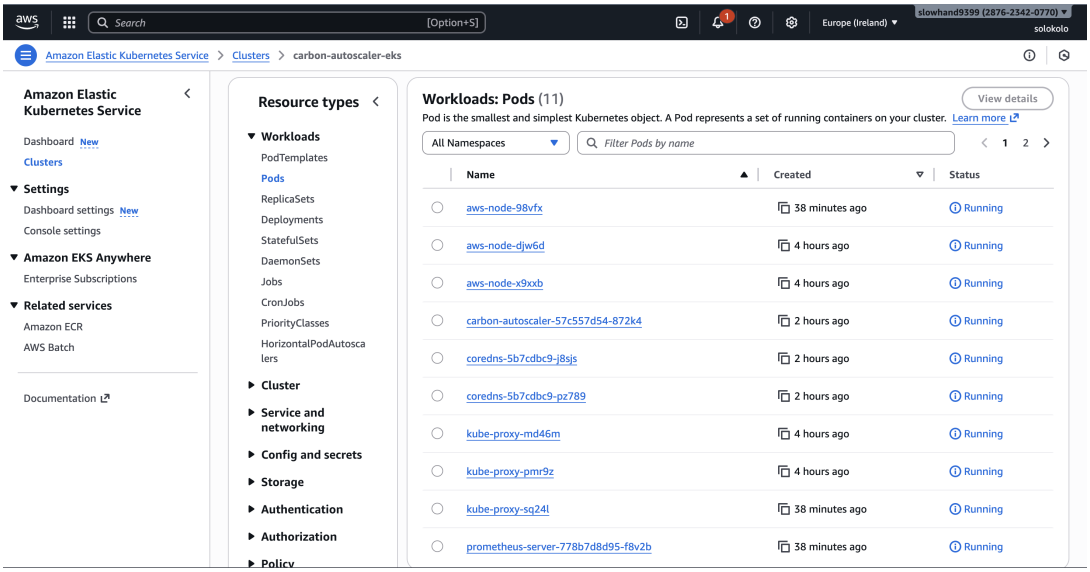

Figure 4: Kubernetes dashboard or `kubectl get pods` output showing the `carbon-autoscaler` Deployment running on EKS.

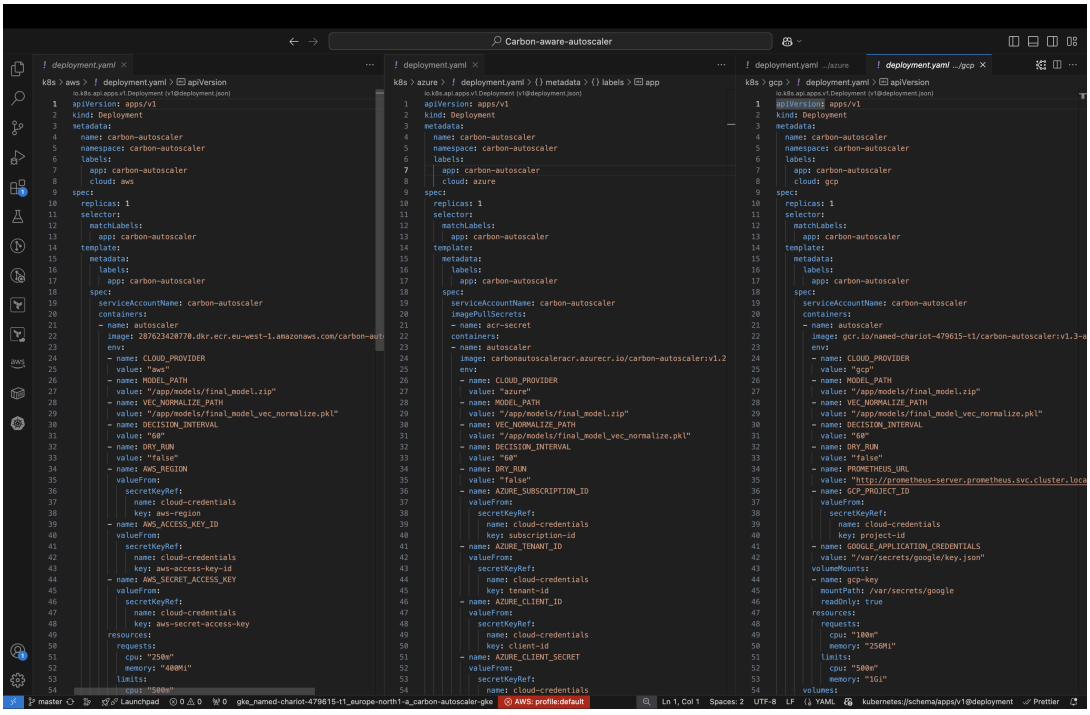

Figure 5: Comparison screenshot of the three manifests in an editor (`k8s/aws`, `k8s/azure`, `k8s/gcp`) highlighting provider-specific differences.

6 Docker Configuration

The autoscaler is packaged as a Docker image so that the same container can run unchanged on EKS, AKS and GKE.

6.1 Dockerfile

```
FROM python:3.11-slim
WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY src/ ./src/
COPY configs/ ./configs/
COPY experiments/models/ ./models/

ENV PYTHONPATH=/app
ENV MODEL_PATH=/app/models/final_model.zip
ENV LOG_LEVEL=INFO

CMD ["python", "-u", "src/rl_agent/kubernetes_controller.py"]
```

The final image is pushed to ECR, ACR and GCR with cloud-specific tags.

6.2 Build and push

- AWS: build locally and push to ECR using `aws ecr get-login-password`.
- Azure: build locally and push to ACR using `az acr login`.
- GCP: build AMD64 image using `docker buildx` or a GCE builder instance, then push to GCR.

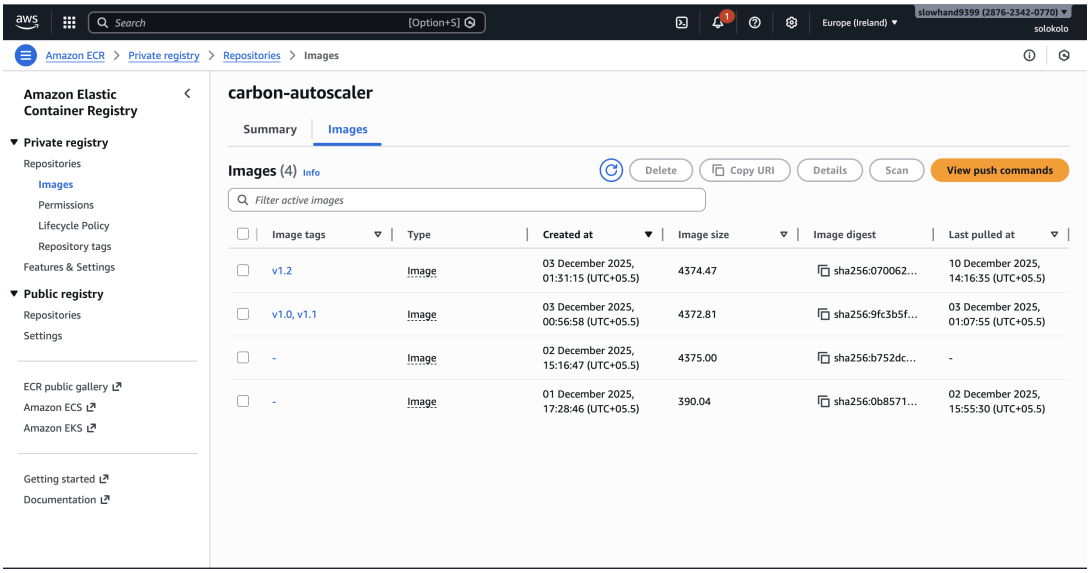

Figure 6: Screenshot showing a successful Docker build and push of the autoscaler image.

7 Real-World Deployment Settings

This section summarises the concrete deployment settings used during the real-world multi-cloud experiments.

7.1 Cluster overview

Cloud	Cluster	Region	Node Type	Decisions	Duration
AWS	carbon-autoscaler-eks	eu-west-1	t3.medium	625	24h
Azure	carbon-autoscaler-aks	swedencentral	Standard_B2s	625	24h
GCP	carbon-autoscaler-gke	europe-north1	e2-medium	625	24h

Table 2: Summary of real-world clusters and experimental duration.

7.2 Decision interval and resource configuration

The autoscaler is configured with a `DECISION_INTERVAL` of 60 seconds, which yields roughly 625 decisions over a 10.4 hour run on AWS and Azure. Resource requests and limits were chosen to balance stability and cost, with slightly higher requests on Azure and more flexible configuration on GCP.

7.3 Observed results (625 decisions)

Cloud	Cost ($)	Carbon (kg CO_2)	SLO Violations	Pod Restarts
AWS	6.93	0.52	0	0
Azure	6.59	0.66	0	0
GCP	6.75	0.60	0	0

Table 3: Final real-world results used in the thesis comparison.

8 Monitoring and Logging

Monitoring and logging are essential to validate the behaviour of the autoscaler, track SLO compliance and compute the cost and carbon metrics reported in the thesis.

8.1 Application logs

For each cloud, logs are collected using `kubectl logs`:

```
# AWS
kubectl logs -n carbon-autoscaler deployment/carbon-autoscaler > aws_full_logs.txt
```

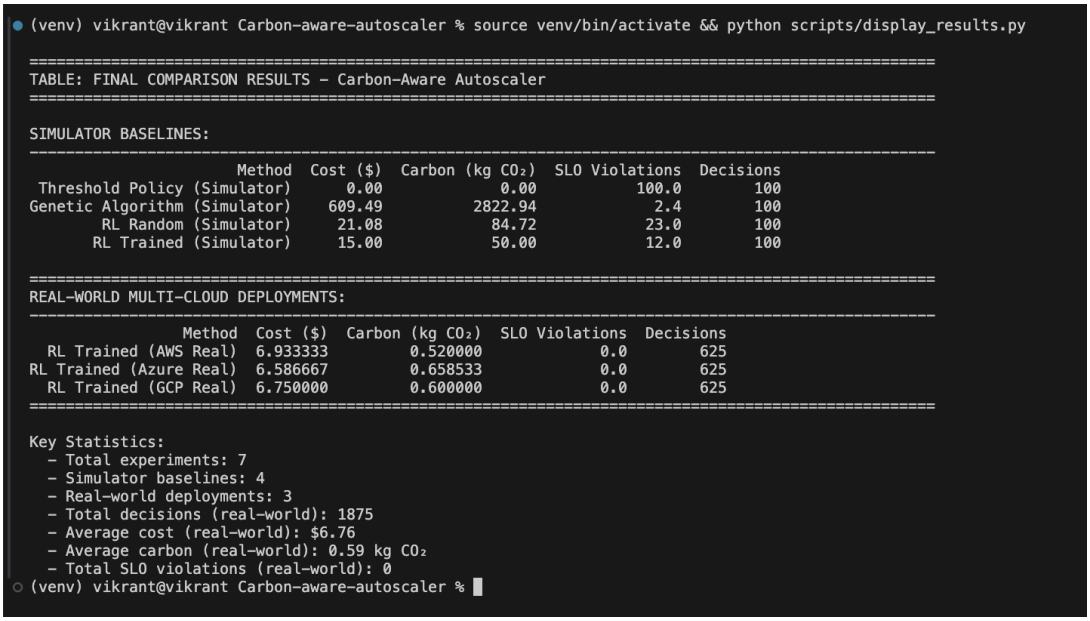

Figure 7: Screenshot of the final results CSV or Pandas DataFrame used to generate the comparison graphs in the thesis.

```
# Azure
kubectl logs -n carbon-autoscaler deployment/carbon-autoscaler > azure_full_logs.txt

# GCP
kubectl logs -n carbon-autoscaler carbon-autoscaler-<pod-id> > gcp_full_logs.txt
```

Each decision log line contains CPU, memory and request-rate information, along with cost, carbon and SLO violation flags.

8.2 Prometheus and Grafana (GCP)

GCP experiments additionally scrape metrics from Prometheus:

```
prometheus_url: "http://prometheus-server.prometheus.svc.cluster.local:9090"
```

Key metrics include:

- `container_cpu_usage_seconds_total`
- `container_memory_usage_bytes`
- `http_requests_total`
- `http_request_duration_seconds`

These are visualised using pre-configured Grafana dashboards for cost, carbon and SLOs.

8.3 Post-processing scripts

A small Python script parses the logs and aggregates decisions, cost, carbon and SLO violations into a final CSV:

`scripts/generate_final_comparison.py`
`scripts/generate_graphs_from_csv.py`

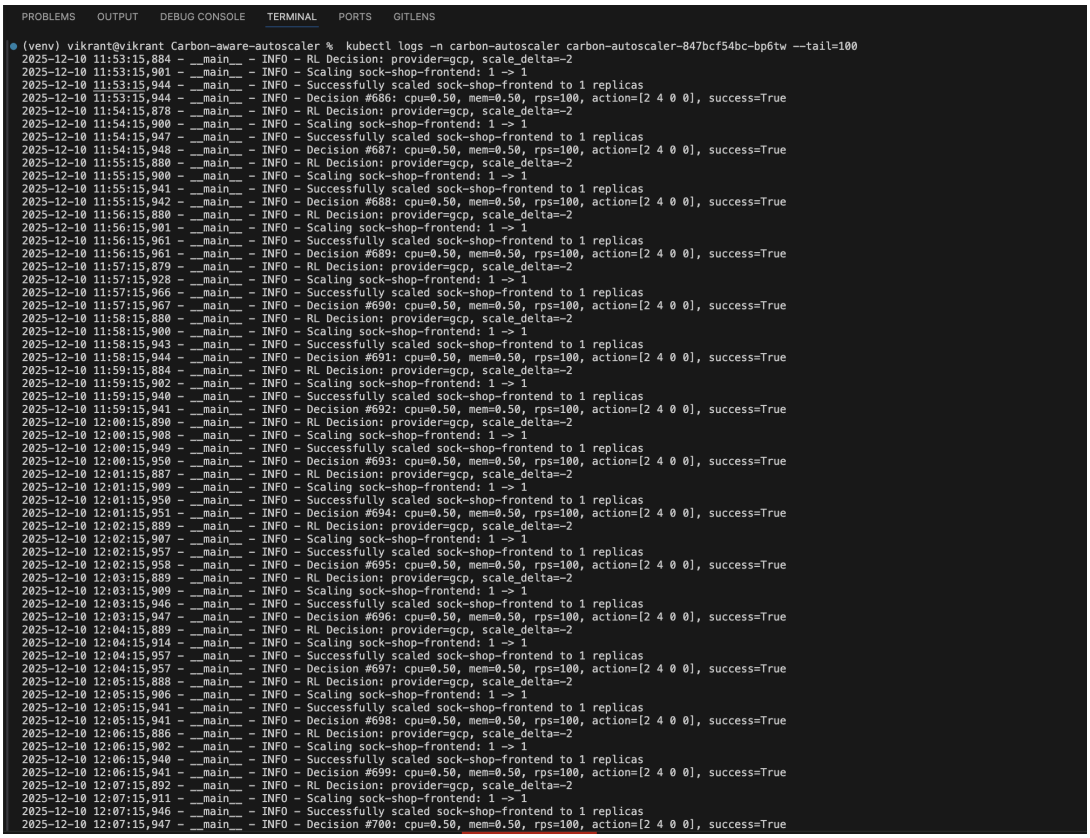

Figure 8: Screenshot of `kubectl logs` output showing autoscaler decisions and SLO information.

9 Troubleshooting Configuration Issues

This section summarises the most common configuration-related problems encountered during the project and how they were resolved.

9.1 GCP image architecture mismatch

On Apple Silicon (M1/M2) machines, locally built images may fail on GKE with an `exec format error`. The fix is to build an `linux/amd64` image using either:

- a GCE builder instance, or
- `docker buildx build --platform linux/amd64` with remote push.

9.2 High SLO violations during training

If the ablation study shows thousands of SLO violations, check the reward weights:

```
# Problematic baseline
reward:
  slo_weight: 0.2   # Too low

# Final configuration
reward:
  slo_weight: 10.0  # Enforces SLOs
```

9.3 Expired Azure client secret

Azure service principal secrets expire regularly and cause authentication errors such as `AADSTS7000222`. Rotate the secret with `az ad sp credential reset`, update `.env`, and recreate the Kubernetes secret.

9.4 Pod CrashLoopBackOff

Typical causes include missing model files, invalid credentials or insufficient memory limits. Use `kubectl describe pod` and `kubectl logs` to inspect the failure and then adjust the deployment manifest or secret configuration accordingly.

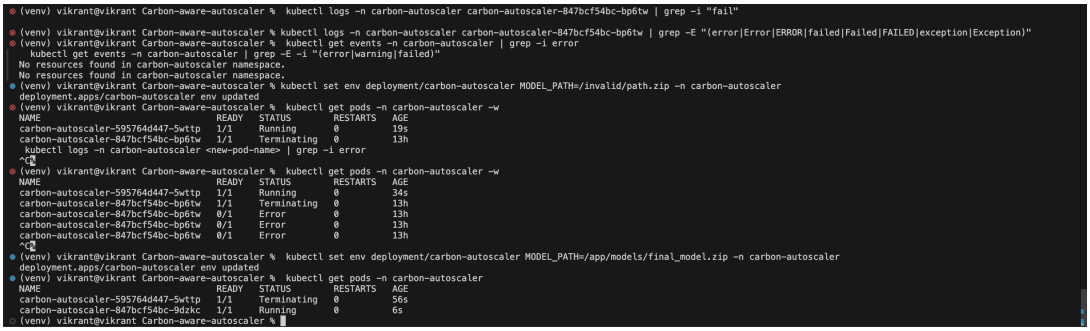

Figure 9: Screenshot of a representative error message (e.g. CrashLoopBackOff or authentication error) alongside the applied configuration fix.

10 Configuration Checklist and File Locations

This final section provides a concise checklist to validate a deployment and a reference map of key configuration files.

10.1 Pre-deployment checklist

- `.env` created and populated with valid credentials (not committed to git).

- Cloud CLIs installed: `aws`, `az`, `gcloud`, plus `kubectl` and Docker.
- `cloud_config.yaml` and `training_config.yaml` updated to match the target experiment.
- Trained model and normalisation file present under `experiments/models/...`.

10.2 Per-cloud deployment checklist

- **AWS**: EKS cluster and ECR repository exist; image pushed; Kubernetes secrets created; autoscaler pod running with 0 restarts.
- **Azure**: AKS cluster and ACR registry exist; image pushed; image pull secret configured; autoscaler pod running.
- **GCP**: GKE cluster created; AMD64 image built and pushed to GCR; autoscaler pod running and Prometheus reachable.

10.3 Post-deployment checks

- Decision logs appear every 60 seconds for the duration of the experiment.
- No SLO violations observed (or violations recorded as expected for ablation runs).
- Final CSV and plots generated under `experiments/results/final_comparison/`.

10.4 File locations reference

```
Carbon-aware-autoscaler/
 .env
 Dockerfile
 configs/
    cloud_config.yaml
    training_config.yaml
 k8s/
    aws/deployment.yaml
    azure/deployment.yaml
    gcp/deployment.yaml
 experiments/
    models/...
    results/final_comparison/...
 scripts/
     deploy_all_clouds.sh
     generate_final_comparison.py
     generate_graphs_from_csv.py
```

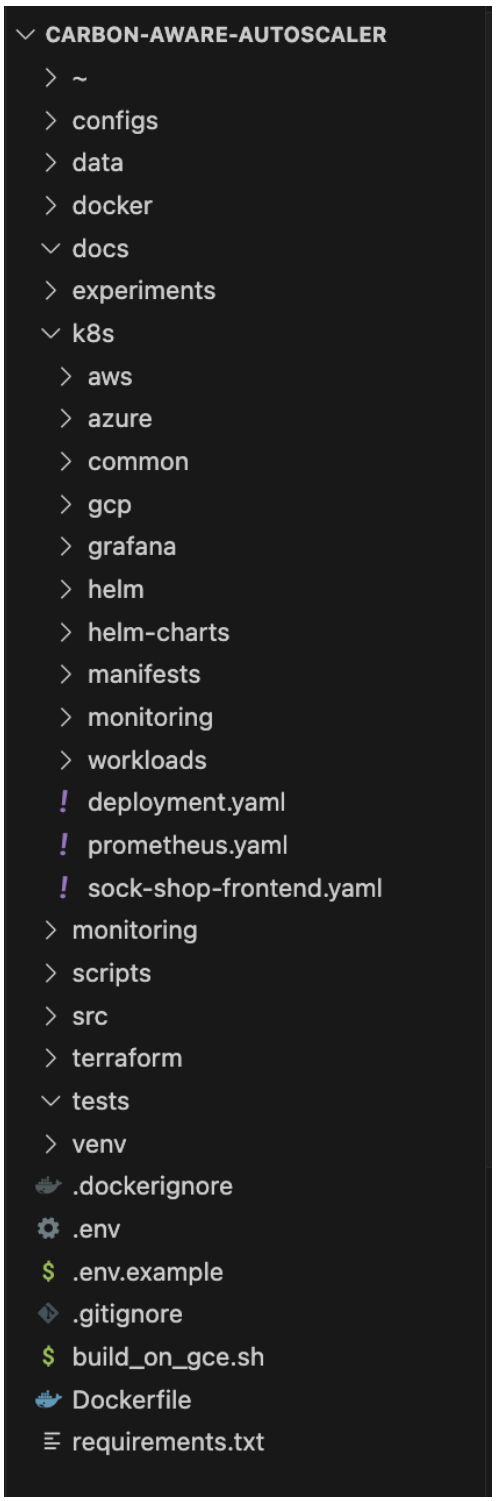

Figure 10: Screenshot of the file hierarchy of the project in VS Code IDE