National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Saksham Shailesh Gurbhele
Student ID: 23266724

School of Computing
National College of Ireland

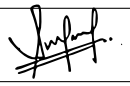
Supervisor: Prof. Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Saksham Shailesh Gurbhele
Student ID:	23266724
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Prof. Diego Lugones
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	1348
Page Count:	6

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:	
Date:	10th August 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 for KubeGreen

Saksham Gurbhele
23266724

1 Overview

This manual guides you through the complete configuration and execution of the smart scheduler described in the provided run guide. It covers prerequisites, project setup on Google Cloud Platform (GCP), cluster provisioning, monitoring setup with Prometheus, preparing configuration files, running the scheduler and recording results, as well as sanity checks, troubleshooting and clean-up.

2 Local prerequisites

The following general tools and software installed on your local machine before getting started:

- **Operating system:** macOS, Linux, or Windows Subsystem for Linux (WSL).
- **Python:** version 3.9 or newer.
- **Command line tools:** `kubectl` and `gcloud` command line interfaces. These are used to interact with Kubernetes clusters and Google Cloud, respectively.
- **Helm:** version v3. Helm is used to install Prometheus on each cluster.

3 Authenticate and set your GCP project

Use the `gcloud` CLI to Authenticate and set up your project with gcloud CLI. Replace `PROJECT_ID` with your own project identifier.

```
# Login to Google Cloud interactively
gcloud auth login


# Set the active project
PROJECT_ID="your-gcp-project-id"
gcloud config set project "$PROJECT_ID"

# Enable the GKE API
gcloud services enable container.googleapis.com
```

4 Create GKE clusters

Provision a small number of single-zone clusters for testing. Each cluster should have at least one node (the run guide uses machine type `e2-standard-2`). Choose zones appropriate for your use case.

```
# Choose zones for each cluster
ZONE_LONDON="europe-west2-a"
ZONE_FRANKFURT="europe-west3-a"
ZONE_MADRID="europe-southwest1-a"

# Create clusters (one node each)
gcloud container clusters create gke-london \
  --zone "$ZONE_LONDON" --project "$PROJECT_ID" \
  --machine-type e2-standard-2 --num-nodes 1
gcloud container clusters create gke-frankfurt \
  --zone "$ZONE_FRANKFURT" --project "$PROJECT_ID" \
  --machine-type e2-standard-2 --num-nodes 1
gcloud container clusters create gke-madrid \
  --zone "$ZONE_MADRID" --project "$PROJECT_ID" \
  --machine-type e2-standard-2 --num-nodes 1
```

5 Fetch kubeconfig contexts

To interact with the new clusters using `kubectl`, pull the cluster credentials. This creates contexts in your `/.kube/config` file. Afterwards you can list contexts and optionally rename them.

```
# Pull kube credentials (creates contexts)
gcloud container clusters get-credentials gke-london \
  --zone "$ZONE_LONDON" --project "$PROJECT_ID"
gcloud container clusters get-credentials gke-frankfurt \
  --zone "$ZONE_FRANKFURT" --project "$PROJECT_ID"
gcloud container clusters get-credentials gke-madrid \
  --zone "$ZONE_MADRID" --project "$PROJECT_ID"

# List contexts
kubectl config get-contexts

# Optional: rename long context names
# kubectl config rename-context <current-name> gke-london
# kubectl config rename-context <current-name> gke-frankfurt
# kubectl config rename-context <current-name> gke-madrid
```

Verify each cluster is reachable:

```
kubectl --context=gke-london get nodes
kubectl --context=gke-frankfurt get nodes
kubectl --context=gke-madrid get nodes
```

6 Install Prometheus monitoring

Monitoring is performed using the `kube-prometheus-stack` Helm chart. Install this chart into a `monitoring` namespace on each cluster, disabling Grafana and Alertmanager by default. First add the Helm repository and update its index:

```
# Add and update the Prometheus community Helm chart repo
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update

# Install Prometheus stack into each cluster
```

```
for CTX in gke-london gke-frankfurt gke-madrid; do
  # Create monitoring namespace (ignore if it already exists)
  kubectl --context=$CTX create namespace monitoring || true

  # Install Prometheus with Grafana and Alertmanager disabled
  helm --kube-context $CTX install prometheus \
    prometheus-community/kube-prometheus-stack \
    -n monitoring \
    --set grafana.enabled=false \
    --set alertmanager.enabled=false
done
```

Wait for the Prometheus pods to reach the `Running` state on each cluster:

```
kubectl --context=gke-london -n monitoring get pods
kubectl --context=gke-frankfurt -n monitoring get pods
kubectl --context=gke-madrid -n monitoring get pods
```

7 Port-forward Prometheus to local ports

Expose each Prometheus service to a particular local port of `kubectl port-forward`. Open another terminal for each cluster so you can leave it to keep the port-forward going during testing. Change port numbers as needed.

```
# London  localhost:9090
kubectl --context=gke-london -n monitoring \
  port-forward svc/prometheus-kube-prometheus-prometheus 9090:9090

# Frankfurt  localhost:9091
kubectl --context=gke-frankfurt -n monitoring \
  port-forward svc/prometheus-kube-prometheus-prometheus 9091:9090

# Madrid  localhost:9092
kubectl --context=gke-madrid -n monitoring \
  port-forward svc/prometheus-kube-prometheus-prometheus 9092:9090
```

8 Prepare project files

The repository root should contain several configuration files that are required in order to execute the scheduler.

8.1 `clusters.json`

The file is a mapping between cluster context names and metadata including a region, carbon intensity zone and URL of the locally forwarded Prometheus instance.

```
{
  "gke-london": {
    "context": "gke-london",
    "region": "London",
    "carbon_zone": "UK",
    "prometheus_url": "http://localhost:9090"
  },
  "gke-frankfurt": {
    "context": "gke-frankfurt",
    "region": "Frankfurt",
```

```
    "carbon_zone": "DE",
    "prometheus_url": "http://localhost:9091"
  },
  "gke-madrid": {
    "context": "gke-madrid",
    "region": "Spain",
    "carbon_zone": "ES",
    "prometheus_url": "http://localhost:9092"
  }
}
```

8.2 carbon.json

In this file, the recent carbon intensity (gCO_2/kWh) in each zone is given.

```
{
  "UK": 175,
  "DE": 224,
  "ES": 70
}
```

8.3 weights.json

The scheduler reads weighting factors α, β and γ from this file each run. Adjust these coefficients to prioritise latency, CPU availability and carbon footprint respectively.

```
{ "alpha": 0.3, "beta": 0.2, "gamma": 0.5 }
```

8.4 job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: test-job
spec:
  template:
    spec:
      containers:
      - name: test
        image: busybox
        command: ["sh", "-c", "echo Hello from smart scheduler && sleep 10"]
      restartPolicy: Never
  backoffLimit: 0
```

9 Run the scheduler

Execute the scheduler in single decision mode. It will read your weights file, query Prometheus on each cluster to gather metrics, compute scores and then deploy the job to the cluster with the highest score.

```
python3 scheduler.py
```

When successful, the script prints per-cluster latency, CPU utilisation, carbon intensity and scores, indicates the selected cluster, submits the job, waits for completion, calculates energy and CO_2 emissions, and appends a line to `results.csv`.

10 Initialise a fresh `results.csv`

If you need to reset the results log, create a new CSV file with the appropriate header:

```
echo "timestamp,cluster,region,latency,cpu,carbon,score,duration,emissions,selected,
    alpha,beta,gamma" > results.csv
```

11 Quick sanity checks

After setup, verify that Prometheus is reachable and that metrics are collected correctly.

- **Prometheus readiness:** query the readiness endpoint on each forwarded port. For example:

  ```
  curl -s http://localhost:9090/-/ready
  # Repeat for 9091 and 9092
  ```

- **CPU metrics:** ensure that the `container_cpu_usage_seconds_total` time series returns data:

  ```
  curl -s "http://localhost:9090/api/v1/query?query=sum(rate(
      container_cpu_usage_seconds_total[1m]))"
  ```

  Repeat the query for each port to confirm metrics are scraped.

- **Kubernetes job status:** inspect the status and logs of submitted jobs on the selected cluster.

  ```
  kubectl --context=<CTX> get jobs -A
  kubectl --context=<CTX> logs job/test-job
  ```

  If a job name already exists, you can delete it before rerunning:

  ```
  kubectl --context=<CTX> delete job test-job --ignore-not-found
  ```

12 Troubleshooting

This part outlines some of the frequent problems and their solutions.

Port-forward fails If Port-forward does not work, Check the name of the service made by the Helm chart twice. List services in the `monitoring` namespace and look for the Prometheus service:

```
kubectl --context=<CTX> -n monitoring get svc | grep prometheus
```

Update the service name in your port-forward command if it differs.

CPU query returns empty Prometheus may not have scraped the container metrics. Wait Count a minute or two, after installing the chart, and the query again. Ensure your clusters have at least one node.

Clusters scaled to zero The scheduler requires at least one node in each cluster. If autoscaling scales a cluster to zero, resize it back to one node:

```
gcloud container clusters resize gke-london --zone "$ZONE_LONDON" --num-nodes 1
```

Auth/account mismatch Confirm which account is active in your `gcloud` configuration and switch if necessary:

```
gcloud auth list
gcloud config list
```

13 Clean up

To avoid ongoing costs, Don t forget to delete the GKE clusters once you are finished. The following loop deletes multiple clusters by name. It queries the cluster location to avoid hard-coding zones. Reorder the cluster list as per the clusters you formed.

Don t forget to delete the GKE clusters once you are done to erase ongoing expenses. The next rotation removes several clusters named. It interrogates the cluster location to prevent hard coded zones. Reorder the cluster list as per the clusters you formed.

```
for CL in gke-london gke-frankfurt gke-madrid; do
  gcloud container clusters delete $CL \
    --zone $(gcloud container clusters list --format="value(LOCATION)" --filter="NAME=
        $CL") \
    --quiet
done
```

References

- Google Cloud CLI (`gcloud`) Documentation. Available at: `https://cloud.google.com/sdk/docs`. Accessed August 2025.
- Google Kubernetes Engine (GKE) Documentation. Available at: `https://cloud.google.com/kubernetes-engine/docs`. Accessed August 2025.
- Kubernetes Command-Line Tool (`kubectl`) Documentation. Available at: `https://kubernetes.io/docs/reference/kubectl/`. Accessed August 2025.
- Helm v3 Documentation. Available at: `https://helm.sh/docs/`. Accessed August 2025.
- Prometheus Community Helm Charts. Available at: `https://prometheus-community.github.io/helm-charts`. Accessed August 2025.
- Prometheus Documentation. Available at: `https://prometheus.io/docs/`. Accessed August 2025.