National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Abigail Mariam Anil
Student ID: x23433698

School of Computing
National College of Ireland

Supervisor: Prof. Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Abigail Mariam Anil
Student ID:	x23433698
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Prof. Aqeel Kazmi
Submission Due Date:	11/12/2025
Project Title:	Forecasting Knative Workload Traffic with Machine Learning to Minimize Cold Starts
Word Count:	1494
Page Count:	11

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:	Abigail Mariam Anil
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):	

Forecasting Knative Workload Traffic with Machine Learning to Minimize Cold Starts

Abigail Mariam Anil
x23433698

1 Requirements

The predictive autoscaler system implementation and integration into Knative needs both cloud and local applications. The whole arrangement is subdivided into two subsections: Azure cloud setup and configuration, Model Pre-Processing and Training environment.

1.1 Local Workstation

1. On Windows 10/11 have WSL2 Ubuntu enabled.
2. Install Docker Desktop on your local system with WSL2 Integration enabled.
3. In the Docker Desktop app, enable Kubernetes support and also sufficient CPU/RAM for containers - like 4 CPU, 4 GB RAM.
4. Install Visual Studio Code with WSL and Python extensions.
5. Inside WSL Ubuntu terminal on VSC, you can install Python 3.10+, git, azure CLI, kubectl.
6. Create and activate a Python virtual environment in the project directory using the command: python3 -m venv venv

   source venv/bin/activate

1.2 Cloud Accounts and Services

To perform an end-to-end project execution, we will require the following accounts and subscriptions:

1. Student's subscription for Azure.
2. Version control on Github account.
3. Google Account sharing Colab across Google (dataset preprocessing, model training and experimentations).

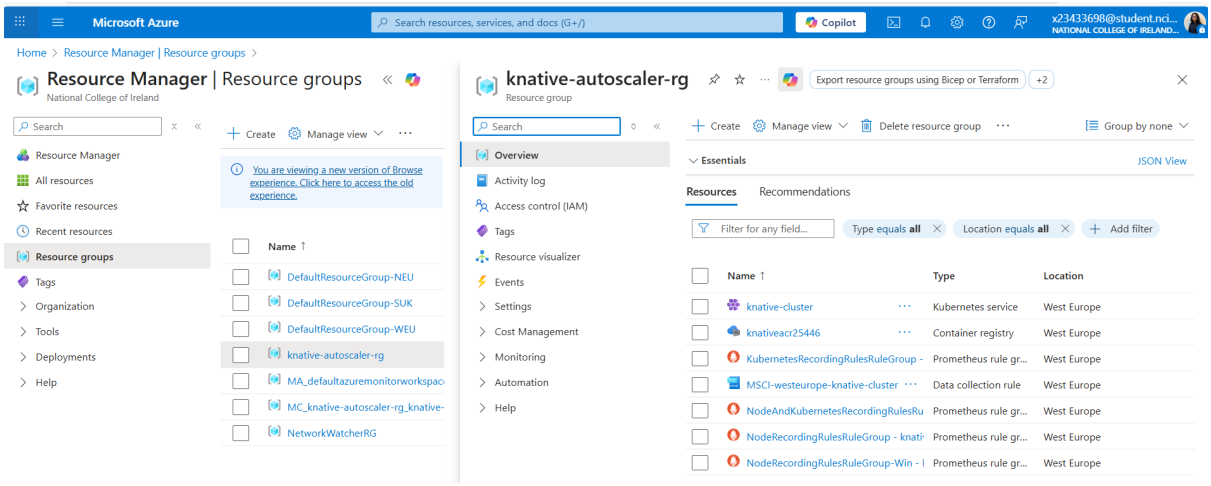

Figure 1: Resource Group Overview on Azure Portal

2 Azure Environment Setup

2.1 Login and Subscription selection

1. Open a WSL terminal in Visual Studio Code IDE.
2. From the terminal, login to Azure using the command : az login
3. List all the available subscriptions using the command : az account list –output table
4. From the available subscriptions, the right subscription can be set using the command: az account set –subscription "$YOUR_SUBSCRIPTION_ID "

2.2 Resource Group

1. Set the following variables in WSL which will be used later at runtime :

   RESOURCE_GROUP="knative-autoscaler-rg"

   LOCATION="westeurope"
2. Create the Resource Group using the command: az group create –name $RESOURCE_GROUP –location $LOCATION
3. The successful creation of resource group can be verified in the overview page of the azure portal as presented in the Figure 1.

2.3 Register Providers

1. Register AKS Providers Microsoft (2024b): az provider register –namespace Microsoft.ContainerService
2. Check registration using the command : az provider show -n Microsoft.ContainerService –query "registrationState"

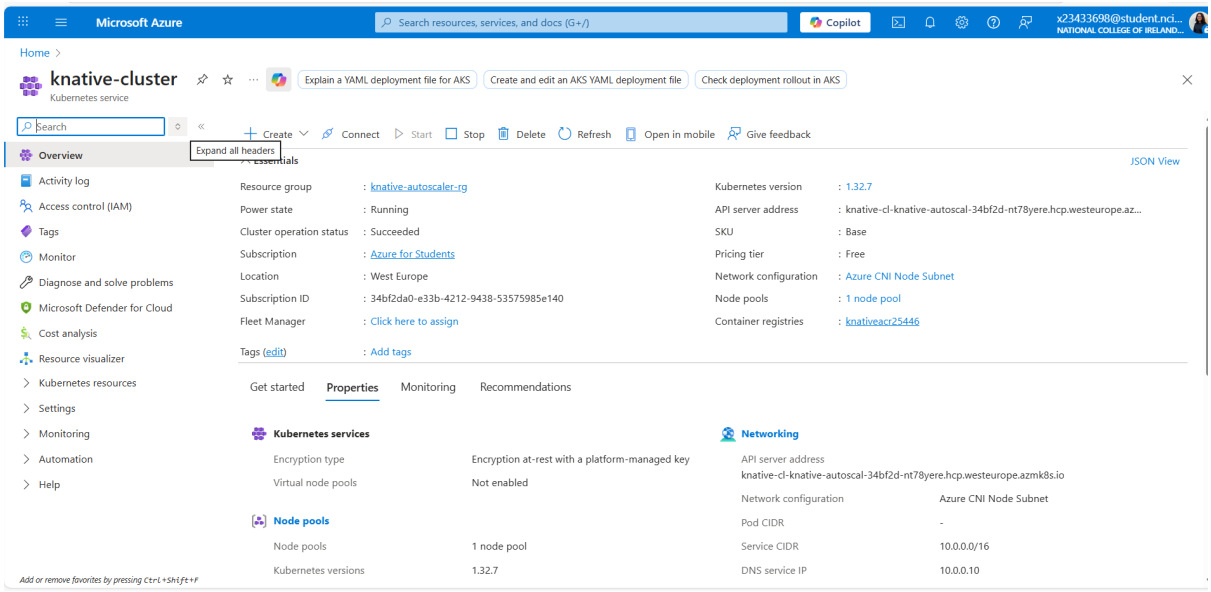

Figure 2: AKS Overview

3. Register the ACR provider Microsoft (2024a): az provider register –namespace Microsoft.ContainerRegistry

4. Check Registration uisng the command :

az provider show -n Microsoft.ContainerRegistry –query ”registrationState”

2.4 AKS Cluster Creation

1. Set the variable for cluster name as : CLUSTER_NAME=”knative-cluster”

2. Create cluster Microsoft (2024b): az aks create –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME –node-count 2 –node-vm-size Standard_B2s –enable-addons monitoring –generate-ssh-keys –enable-managed-identity –network-plugin azure

3. Get AKS Credentials using the command: az aks get-credentials –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME –overwrite-existing

   Figure 2 gives an AKS overview on Azure portal.

4. Check nodes as indicated in Figure 3:

   kubectl get nodes

2.5 Daily Start/Stop

1. To check cluster power state run the command as it is presented in Figure 4: az aks show –name knative-cluster –resource-group knative-autoscaler-rg –query ”powerState” -o table

2. To stop the cluster at the end of the day, run the command: az aks stop –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME

```
root@CHRIS-HP:/mnt/c/Users/Mathe/Downloads/Abigail/Thesis/Project# kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
aks-nodepool1-11238319-vmss000016   Ready    <none>   27m   v1.32.7
aks-nodepool1-11238319-vmss000017   Ready    <none>   26m   v1.32.7
```

Figure 3: Node Pools

```
root@CHRIS-HP:/mnt/c/Users/Mathe/Downloads/Abigail/Thesis/Project# az aks show \
  --name knative-cluster \
  --resource-group knative-autoscaler-rg \
  --query "powerState" -o table
Code
-------
Running
```

Figure 4: Check cluster power state

3. In order to restart the cluster again later use the following command: az aks start –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME

   az aks get-credentials –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME –overwrite-existing

   kubectl get nodes

3 Azure Container Registry and Docker Desktop

1. Assign the registry name variable : ACR_NAME=”knativeacr25446”
2. Registry can be created using the command Microsoft (2024a) : az acr create –resource-group $RESOURCE_GROUP –name $ACR_NAME –sku Basic
3. Login to Registry: az acr login –name $ACR_NAME
4. Attach ACR to AKS: az aks update –resource-group $RESOURCE_GROUP –name $CLUSTER_NAME –attach-acr $ACR_NAME
5. Get login server: export ACR_LOGIN_SERVER=$(az acr show –name $ACR_NAME –query loginServer -o tsv)

3.1 Docker desktop Usage

1. Make sure that it is in running state and that Kubernetes is connected.
2. Confirm that the backend engine is using WSL2 Docker Inc. (2024).
3. Check containers view to verify local registry, kind/local kubernetes containers as shown in Figure 5.

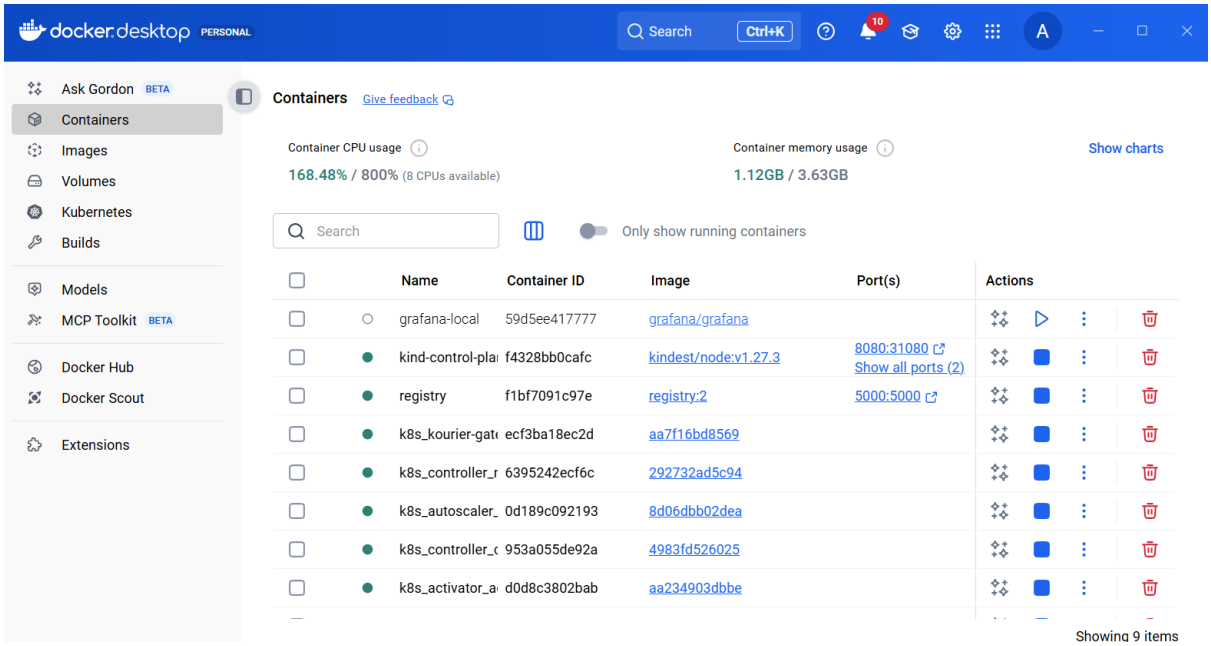

Figure 5: Docker Desktop Containers

```
(venv) root@CHRIS-HP:/mnt/c/Users/Mathe/Downloads/Abigail/Thesis/Project# kubectl get pods -n knative-serving
NAME                                    READY   STATUS    RESTARTS   AGE
activator-58d646674b-j4vmq              1/1     Running   0          12d
autoscaler-5c8fbfdbcb-lv596             1/1     Running   0          12d
controller-57997dbbcf-jh5l5             1/1     Running   0          12d
net-kourier-controller-6db796dd7b-zdgz5 1/1     Running   0          12d
webhook-5b4998477-wtkcc                 1/1     Running   0          12d
```

Figure 6: Watch Knative Pods

4 Knative Serving and Kourier setup on AKS

4.1 Install Knative Serving

1. Apply CRDs Knative Project (2024) using the command:

   kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.14.0/serving-crds.yaml

2. Apply core components using the command:

   kubectl apply -f https://github.com/knative/serving/releases/download/knative-v1.14.0/serving-core.yaml

3. To track Knative pod and wait until activator, autoscaler, controller and Web-hook. are up and running as shown in Figure 6 execute the command : kubectl get pods -n knative-serving -w

4.2 Installation and Configuration of Kourier

1. Install Kourier using the command:

   kubectl apply -f https://github.com/knative/net-kourier/releases/download/knative-v1.14.0/kourier.yaml

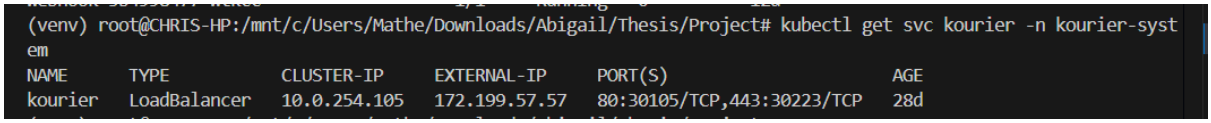

Figure 7: Kourier Service

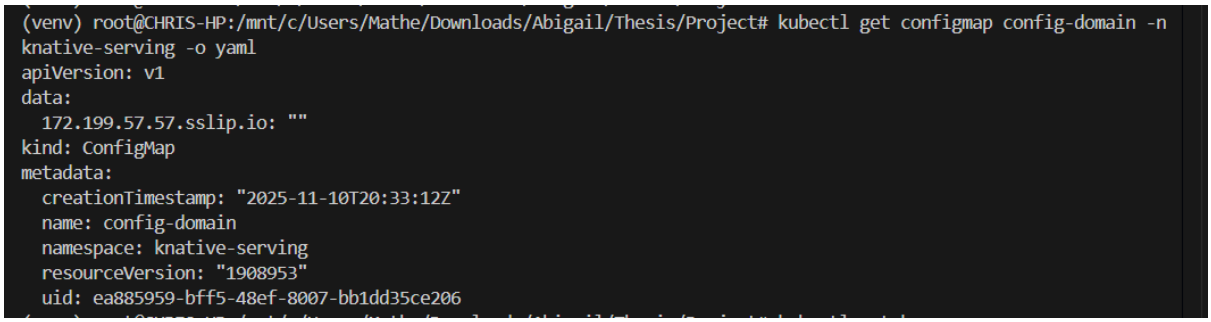

Figure 8: Checking Config Domain

2. Configure Knative to use Kourier:

```
kubectl patch configmap/config-network -n knative-serving \
  --type merge \
  --patch '{"data":{"ingress-class":"kourier.ingress.networking.knative.dev"}}'
```

3. Check Kourier service as illustrated in Figure 7 using the command : kubectl get svc kourier -n kourier-system

4.3 Domain Configuration

1. Get external IP:

   EXTERNAL_IP=$(kubectl get svc kourier -n kourier-system -o jsonpath='{.status.loadBalancer.ingress[0].ip}')

   echo $EXTERNAL_IP

2. Configure config domain using the command:

```
kubectl patch configmap/config-domain -n knative-serving \
  --type merge \
  --patch "{\"data\":{\"$EXTERNAL_IP.sslip.io\":\"\"}}"
```

3. Check config domain, make sure that only a single key exists as illustrated in Figure 8: kubectl get configmap config-domain -n knative-serving -o yaml

5 Repository setup and Python environment

5.1 Clone Github Repository

1. Clone Repo: git clone https://github.com/abigail-anil/knative-predictive-autoscaler.git

   cd knative-predictive-autoscaler/knative-cloud

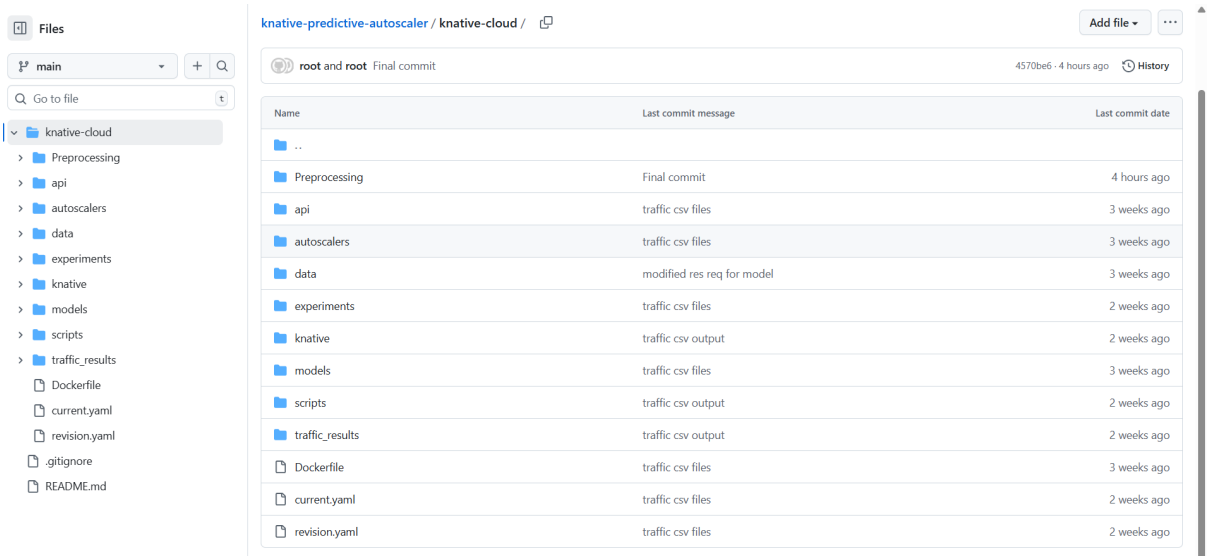

Figure 9: Github Folder Structure

2. Confirm folder structure as shown in Figure 9

5.2 Python Dependencies

Install python dependencies using the requirements file which can be cloned from the github repo: From knative-cloud folder:

source venv/bin/activate pip install -r requirements.txt

6 Data preprocessing and training the models (Colab)

6.1 Preprocessing in Colab

1. Start Google Colab and upload the notebook from the Preprocessing/ directory.
2. Upload the raw Azure Functions Dataset[1] to Google Colab.
3. In the preprocessing notebook, parse timestamps and ensure a uniform 1-minute interval. Other features available include hour, dayofweek, is business hours, is weekend. Sum up the per minute to establish per minute requests. And lastly select the functions to be experimented with and filter(func 235, func 110, etc.).
4. Export a csv of each of the filtered function ID and save them as timeseries such as timeseries func 235.csv.
5. Copy the resulting csv into your local disk and place them under knative-cloud/data/

[1] `https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsInvocationTrace2021.md`

6.2 Model Training and Saving

1. In Google Colab, for each function ID, you may run prophet training Taylor and Letham (2017), LSTM training Guruge and Priyadarshana (2025) and Hybrid trainingShiekhani et al. (2025).

2. Ensure that the training code separates the data into train and test i.e. (80%/20%) and stores training models.

3. Download the models saved on Colab and place them in the directories with the corresponding names below the directory knative-cloud/models/.

7 Building the forecasting API container

1. In knative-cloud directory, build the image Docker Inc. (2024) using the command: docker build -t forecasting-api:v44 .

   Here you may change the image and version name, but maintain consistent image and version for following commands.

2. Tag with ACR Registry using the command:

   docker tag forecasting-api:v44 $ACR_LOGIN_SERVER/forecasting-api:v44

3. Push to ACR using the command: docker push $ACR_LOGIN_SERVER/forecasting-api:v44

4. Verify tags using the command: az acr repository show-tags –name $ACR_NAME –repository forecasting-api –output table

8 Knative Service Deployment

8.1 Generate yaml manifests

1. Make sure that the scripts can be executed by changing its execution permissions.

   chmod +x scripts/generate_aks_yamls.sh

   chmod +x scripts/generate_reactive_yamls.sh

2. These below scripts are run to generate predictive and baseline yamls. This writes predictive yaml files to knative/ and baseline yaml files to knative/reactive/ directory.

   ./scripts/generate_aks_yamls.sh

   ./scripts/generate_reactive_yamls.sh

8.2 Apply Services to AKS

Apply predictive services and wait until every service reports READY=TRUE as depicted in Figure 10: kubectl apply -f knative/
kubectl get ksvc -w

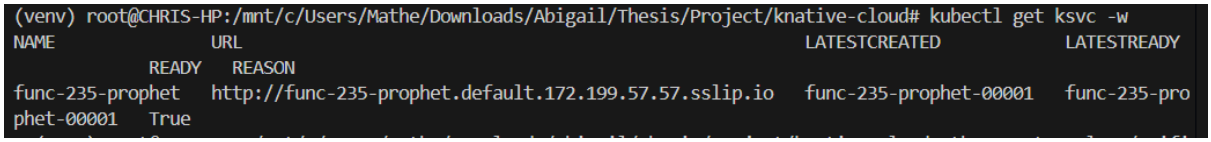

Figure 10: Apply services to AKS

8.3 Verify URL and image version

1. Check the applied service and note down the URL displayed in the output : kubectl get ksvc func-235-prophet

2. Check the image used by the pod and confirm whether it is the latest. Modify the command accordingly for each Knative service:

   kubectl describe pod $(kubectl get pods -l serving.knative.dev/service=func-235-prophet -o jsonpath='{.items[0].metadata.name}') — grep Image

3. Check health of the endpoint and verify HEALTHY status Ramirez (2024):

   curl -v http://func-235-prophet.default.${EXTERNAL_IP}.sslip.io/health

9 Running the predictive autoscaler and experiments

9.1 Predictive Autoscaler

1. In terminal 1, run predictive autoscaler for prophet from knative-cloud directory:

   cd knative-cloud source venv/bin/activate

   python3 autoscalers/unified_predictive_autoscaler.py –function-id func_235 –model-type prophet –interval 20 –use-real-traffic –traffic-csv data/timeseries_func_235.csv

2. Observe the logs showing current requests per pod, predictions, scaling actions, as shown in Figure 11.

3. Launch a new terminal and execute at the same time: watch -n 2 kubectl get pods

9.2 Traffic Generation

1. Open a new terminal and run the traffic generator script in parallel: python3 experiments/aks_traffic_generator_200.py

   func_235 prophet data/timeseries_func_235.csv 200 90

2. Observe the logs where the script prints per-minute stats of the number of requests, success counts, p95latency, etc as shown in Figure 12. Ensure also that the outputs are being generated in. the traffic results/ dictionary.

3. Perform the experiments on all the 3 models for other functions.

```
^C(venv) root@CHRIS-HP:/mnt/c/Users/Mathe/Downloads/Abigail/Thesis/Project/knative-cloudpython3 autoscalers/unifi
ed_predictive_autoscaler.py \\
  --function-id func_235 \
  --model-type prophet \
  --interval 20 \
  --use-real-traffic \
  --traffic-csv data/timeseries_func_235.csv
2025-12-09 13:46:47,398 - Loaded 20053 traffic datapoints
2025-12-09 13:46:47,549 - Traffic stats: min=0, mean=27, max=252
2025-12-09 13:46:49,067 - Kourier IP: 172.199.57.57
2025-12-09 13:46:49,067 - Service URL: http://func-235-prophet.default.172.199.57.57.sslip.io
2025-12-09 13:46:49,067 - Testing service connectivity...
2025-12-09 13:46:49,955 - Service reachable: {'status': 'healthy'}
2025-12-09 13:46:49,956 - Pod capacity set to: 30 req/min
2025-12-09 13:46:49,956 -  PREDICTIVE AUTOSCALER STARTED
2025-12-09 13:46:49,956 -    Service: func-235-prophet
2025-12-09 13:46:49,956 -    Model: PROPHET
2025-12-09 13:46:49,957 -    Traffic: Real CSV Data
2025-12-09 13:46:49,957 -    Capacity: 30 req/min per pod
2025-12-09 13:46:49,958 -    Range: 0-5 pods
2025-12-09 13:46:49,958 -

2025-12-09 13:46:49,959 - [Iteration 1] 13:46:49
2025-12-09 13:46:50,765 -  Current state: 1 running + 1 pending | 69.0 req/min
2025-12-09 13:46:51,553 - PROPHET predictions: ['18.4', '18.0', '17.7']...
2025-12-09 13:46:51,553 -  Peak load: 18.4 req/min | Current capacity: 60 req/min (2 pods)
2025-12-09 13:46:51,553 -  Utilization: 30.7% | Action: SCALE DOWN | Desired: 0 pods
2025-12-09 13:46:51,553 - Scaling: 2 → 0 pods
2025-12-09 13:46:51,977 -  Scaled to 0 pods (minScale updated)
2025-12-09 13:46:51,977 - Sleeping for 20s...
2025-12-09 13:47:12,106 -
```

Figure 11: Predictive Autoscaler Logs

```
(venv) root@CHRIS-HP:/mnt/c/Users/Mathe/Downloads/Abigail/Thesis/Project/knative-cloud# python3 experiments/aks_t
raffic_generator_200.py func_235 prophet data/timeseries_func_235.csv 200 90

 TRAFFIC RUN: func_235-prophet
URL: http://func-235-prophet.default.172.199.57.57.sslip.io
Replay minutes: 200
Speedup: 90x

[2025-12-09T13:47:54+00:00] Minute 1/200: 50 req
→ 50/50 ok | pods=4 | p95=127.3 ms COLD
[2025-12-09T13:47:57+00:00] Minute 2/200: 50 req
→ 50/50 ok | pods=3 | p95=113.7 ms
[2025-12-09T13:48:01+00:00] Minute 3/200: 0 req
[2025-12-09T13:48:01+00:00] Minute 4/200: 50 req
→ 50/50 ok | pods=2 | p95=116.6 ms
[2025-12-09T13:48:03+00:00] Minute 5/200: 50 req
→ 50/50 ok | pods=2 | p95=225.3 ms
[2025-12-09T13:48:07+00:00] Minute 6/200: 50 req
→ 50/50 ok | pods=2 | p95=105.0 ms
[2025-12-09T13:48:11+00:00] Minute 7/200: 2 req
→ 2/2 ok | pods=2 | p95=103.6 ms
[2025-12-09T13:48:11+00:00] Minute 8/200: 22 req
→ 22/22 ok | pods=2 | p95=111.7 ms
[2025-12-09T13:48:12+00:00] Minute 9/200: 50 req
→ 50/50 ok | pods=2 | p95=110.0 ms
[2025-12-09T13:48:15+00:00] Minute 10/200: 13 req
→ 13/13 ok | pods=2 | p95=134.7 ms
[2025-12-09T13:48:16+00:00] Minute 11/200: 0 req
[2025-12-09T13:48:16+00:00] Minute 12/200: 1 req
→ 1/1 ok | pods=2 | p95=95.8 ms
[2025-12-09T13:48:17+00:00] Minute 13/200: 8 req
→ 8/8 ok | pods=2 | p95=104.9 ms
[2025-12-09T13:48:17+00:00] Minute 14/200: 50 req
→ 50/50 ok | pods=2 | p95=357.7 ms
```

Figure 12: Load Generator Logs

10 Grafana Dashboard

The output CSVs made by the load generator were visualized using Cloud Grafana.

1. Push all the csv outputs to Github repository.
2. Configure an infinity datasource in Grafana pointing to these files.
3. Using data, make dashboards of number of pods versus requests per model, comparison of latencies , cost and resource metrics table and cold start counts per model and per function.

11 Github repository

The complete code of the project, configuration, scripts and notebook can be found at: https://github.com/abigail-anil/knative-predictive-autoscaler

All the files, which are referred in this folder, are found in knative-cloud folder.

References

Docker Inc. (2024). Docker documentation, `https://docs.docker.com/`. Accessed Nov 2025.

Guruge, P. and Priyadarshana, Y. (2025). Time series forecasting-based kubernetes autoscaling using facebook prophet and long short-term memory, *Frontiers in Computer Science* **7**. DOI: 10.3389/fcomp.2025.1509165.
URL: *https://doi.org/10.3389/fcomp.2025.1509165*

Knative Project (2024). Knative documentation: Serving, autoscaling, eventing, `https://knative.dev/docs/`. Accessed Nov 2025.

Microsoft (2024a). Azure container registry documentation, `https://learn.microsoft.com/azure/container-registry/`. Accessed Nov 2025.

Microsoft (2024b). Azure kubernetes service (aks) documentation, `https://learn.microsoft.com/azure/aks/`. Accessed Nov 2025.

Ramirez, S. (2024). Fastapi documentation, `https://fastapi.tiangolo.com/`. Accessed Nov 2025.

Shiekhani, L., Wang, H., Shi, W., Liu, J., Qiu, Y., Gu, C. and Ding, W. (2025). The hybrid model: Prediction-based scheduling and efficient resource management in a serverless environment, *Applied Sciences* **15**. DOI: 10.3390/app15147632.
URL: *https://doi.org/10.3390/app15147632*

Taylor, S. and Letham, B. (2017). Forecasting at scale. DOI: 10.7287/peerj.preprints.3190v2.
URL: *https://doi.org/10.1080/00031305.2017.1380080*