

Configuration Manual

MSc Research Project
Cloud Computing

Sahil Shaikh
Student ID: x23220228

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Sahil Shaikh
Student ID:	x23220228
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	9

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:	23rd April 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Sahil Shaikh
x23220228

1 Introduction

This configuration guide explains the specification of the entire configuration and deployment of the intelligent multi-cloud CI/CD system built up in this research. You then build it across GCP, Azure and AWS using GitHub Actions, Kubernetes, Docker and Azure Machine Learning to deploy a Djangoweb app. The arrangement allows for AI based cloud selection as well as Kubernetes rollback on performance failures. This guide serves practitioners and researchers so they can reproduce, replicate, and extend the system.

2 System Configuration Setup

2.1 Software Requirements

Tool/Platform	Version / Notes
OS	Ubuntu 22.04 / Windows 11 / macOS Ventura
Python	3.10
Docker	24.x
GitHub CLI	Latest version
Terraform	1.4.x
Google Cloud SDK	With <code>gke-gcloud-auth-plugin</code>
Azure CLI	Latest version
AWS CLI	v2
VS Code (Recommended)	With Docker, Python & Kubernetes extensions
kubectrl	v1.26+
Azure Machine Learning SDK	<code>azureml-core</code> , <code>azure-identity</code> , <code>joblib</code> , etc.

Table 1: Tools and Platforms Used

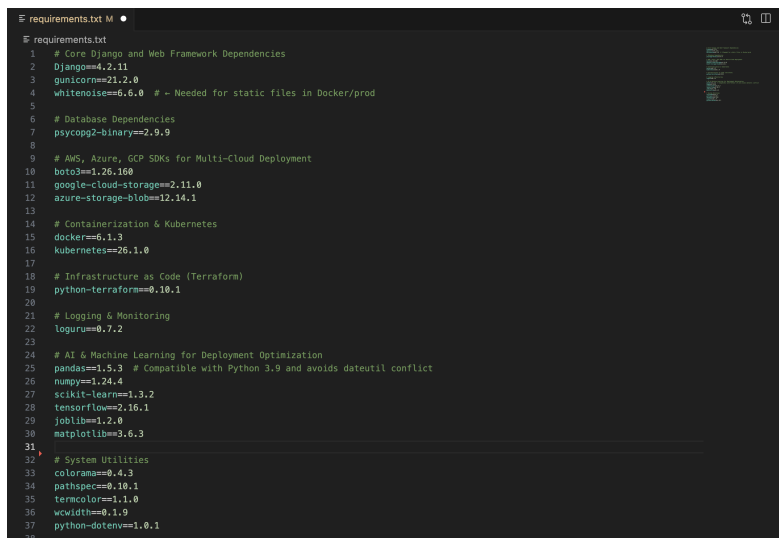
2.2 Hardware Requirements

- **Processor:** Intel i5/i7 or Apple M1
- **RAM:** Minimum 8 GB (*16 GB recommended*)
- **Disk Space:** Approximately 20 GB (for images, logs, data)
- **Network:** Stable broadband internet connection

3 Environmental Setup

3.1 GitHub Repository Setup

- **Github Repository Setup:** -
git clone https://github.com/sahil0480/NCI_Research_Project.git
- **Setup a Python environment:** -
python -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
- **Install Docker and enable Buildx:** docker buildx install



```
requirements.txt
# Core Django and Web Framework Dependencies
Django==4.2.11
gunicorn==21.2.0
whitenoise==6.6.0 # - Needed for static files in Docker/prod

# Database Dependencies
psycopg2-binary==2.9.9

# AWS, Azure, GCP SDKs for Multi-Cloud Deployment
boto3==1.26.160
google-cloud-storage==2.11.0
azure-storage-blob==12.14.1

# Containerization & Kubernetes
docker==6.1.3
kubernetes==26.1.0

# Infrastructure as Code (Terraform)
python-terraform==0.10.1

# Logging & Monitoring
loguru==0.7.2

# AI & Machine Learning for Deployment Optimization
pandas==1.5.3 # Compatible with Python 3.9 and avoids dateutil conflict
numpy==1.24.4
scikit-learn==1.3.2
tensorflow==2.16.1
joblib==1.2.0
matplotlib==3.6.3

# System Utilities
colorama==0.4.3
pathspec==0.10.1
termcolor==1.1.0
wcwidth==0.1.9
python-dotenv==1.0.1
```

Figure 1: Requirement.txt

4 Cloud & Credential Setup

4.1 GitHub Secrets Configuration

All secrets are stored under GitHub repo → Settings → Secrets → Actions.

Secret Name	Purpose
GCP_SA_KEY	GCP Service Account JSON
AZURE_KEY	Azure Client ID, Secret, Tenant ID
AZURE_STORAGE_KEY	Azure Blob Storage Key
AWS_KEY	AWS IAM User Key (in JSON format)
AWS_ACCESS_KEY_ID	AWS Key ID (if separate)
AWS_SECRET_ACCESS_KEY	AWS Secret (if separate)

Table 2: Secrets and Their Purpose in Multi-Cloud Deployment

The screenshot shows the Github interface for repository secrets. At the top, there are tabs for 'Secrets' and 'Variables'. Below the tabs, the title 'Repository secrets' is displayed next to a green button labeled 'New repository secret'. A table lists the existing secrets with columns for 'Name' and 'Last updated'. The table contains four entries: 'AWS_KEY' (updated last week), 'AZURE_KEY' (updated 2 weeks ago), 'AZURE_STORAGE_KEY' (updated 2 weeks ago), and 'GCP_SA_KEY' (updated 2 weeks ago). Each entry has a lock icon on the left and edit/delete icons on the right.

Name	Last updated
AWS_KEY	last week
AZURE_KEY	2 weeks ago
AZURE_STORAGE_KEY	2 weeks ago
GCP_SA_KEY	2 weeks ago

Figure 2: Github Secret Keys

5 Infrastructure Setup

5.1 Terraform Scripts

The Terraform configuration for this project has been structured into three cloud environments AWS, Azure, and GCP each in separate folder and deployment scripts. This ensures modular and cloud-specific infrastructure provisioning.

Terraform AWS Infrastructure

`terraform/aws/` is the directory used to provision an Amazon EKS (Elastic Kubernetes Service) cluster on AWS. It contains the following key files and directories:

- **provider.tf** – Specifies the AWS provider configuration including access credentials and region.
- **main.tf** – Contains the core infrastructure logic to create a VPC, subnets, IAM roles, and the EKS cluster itself.
- **aws-auth.yaml** – A configuration file for granting access to EKS worker nodes via `kubectl`.
- **outputs.tf** – Defines output variables such as the EKS cluster name and kubeconfig for access.
- **variables.tf** – Declares input variables like region and cluster name to enable parameterized deployments.
- **modules/** – An optional directory for reusable and modular Terraform code components.
- **.tfstate files** – Automatically generated files that track the current state of infrastructure resources for consistent and idempotent deployment.

Terraform Azure Infrastructure

`terraform/azure/` provisions an AKS (Azure Kubernetes Service) cluster. Key components include:

- **providers.tf** – Configures the Azure provider with subscription ID and tenant ID.
- **main.tf** – Provisions the AKS cluster and its associated node pools.
- **terraform.tfvars** – Contains values for variables such as location, node count, and resource group.
- **outputs.tf** – Defines output values such as kubeconfig credentials for accessing the AKS cluster.
- **variables.tf** – Declares input variables like region, resource group, and node count for customization.

Terraform GCP Infrastructure

`terraform/gcp/` provisions a GKE (Google Kubernetes Engine) cluster. The folder includes the following:

- **provider.tf** – Configures the Google Cloud provider using credentials and project information.
- **main.tf** – Creates the GKE cluster, including node pools and necessary networking resources.
- **terraform.tfvars** – Stores values for variables such as GCP project ID, zone, and cluster name.
- **outputs.tf** – Outputs important details such as the cluster endpoint and kubeconfig for access.
- **variables.tf** – Accepts input parameters like region, project name, and desired cluster name.

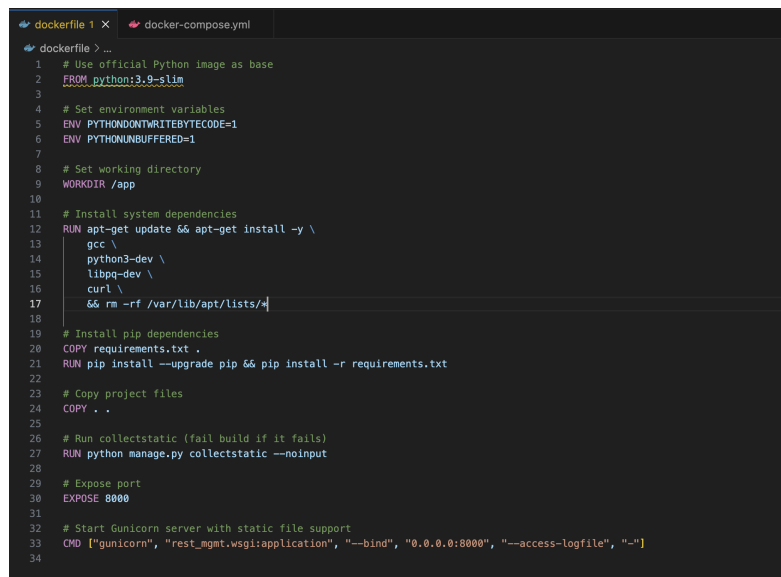


Figure 3: Terraform Folder Structure

6 Application Setup - Containerization with Docker

6.1 Dockerfile

Docker for containerization (for consistency across dev, testing, production) for the Django RMSApplication. If we look at the Dockerfile located at the root of the project, we see that it starts from an official Python 3.9 slim image. It also sets some environment variables to boost Python runtime and sets /app as the working dir. It installs some essential system dependencies (gcc, libpq-dev, curl) that are required to make the database work and also it will generate the run environment. Python dependencies are installed from requirements.txt and copy over the entire project files into the container. Prior to launching the application, Django's collectstatic command is run, collecting all static files and preparing for serving in a production-like environment. The container listens on port 8000 and runs the app through Gunicorn, a high-performance WSGI server, bound to all interfaces.

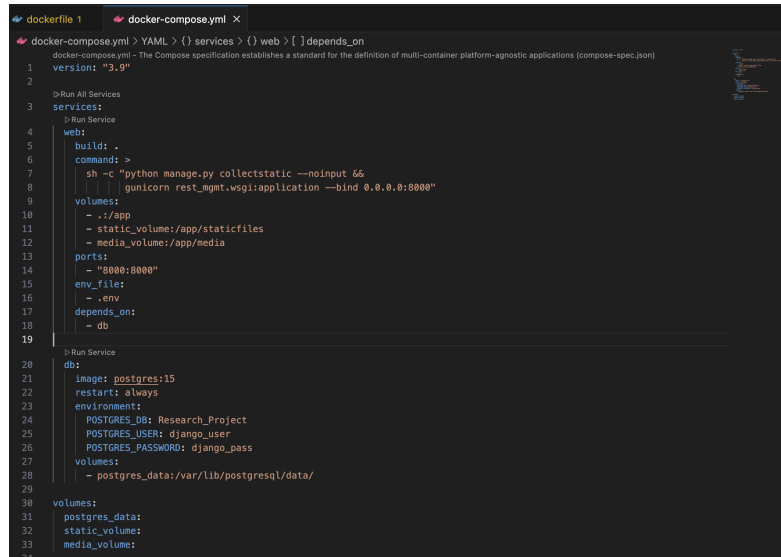
A screenshot of a code editor showing a Dockerfile. The file is named 'dockerfile' and is located in a directory named 'docker-compose.yml'. The code is as follows:

```
1 # Use official Python image as base
2 FROM python:3.9-slim
3
4 # Set environment variables
5 ENV PYTHONUNBUFFERED=1
6 ENV PYTHONUNBUFFERED=1
7
8 # Set working directory
9 WORKDIR /app
10
11 # Install system dependencies
12 RUN apt-get update && apt-get install -y \
13     gcc \
14     python3-dev \
15     libpq-dev \
16     curl \
17     && rm -rf /var/lib/apt/lists/*
18
19 # Install pip dependencies
20 COPY requirements.txt .
21 RUN pip install --upgrade pip && pip install -r requirements.txt
22
23 # Copy project files
24 COPY . .
25
26 # Run collectstatic (fail build if it fails)
27 RUN python manage.py collectstatic --noinput
28
29 # Expose port
30 EXPOSE 8000
31
32 # Start Gunicorn server with static file support
33 CMD ["gunicorn", "rest_mgmt.wsgi:application", "--bind", "0.0.0.0:8000", "--access-logfile", "-"]
34
```

Figure 4: Docker File Code Snippet

6.2 Dockerfile-Compose

Docker Compose (docker-compose.yml) serves as the orchestrator of this multi-container environment, bringing together the Django web application and PostgreSQL database. The web service constructs the docker image based on the defined Dockerfile, it mounts the local project directory into the docker container and uses named volumes to persist static and media files. It is reading configuration variables from the .env file, runs collectstatic and starts the app with Gunicorn. The db service pulls an official PostgreSQL 15 image and sets up a database from some environment variables including the database name, user, and password. We attach volumes to persist data, static, and media. It allows for smooth local development, and can easily be moved to run on the cloud as part of Kubernetes and CI/CD.



```

1  version: "3.9"
2
3  services:
4    > Run Service
5    web:
6      build: .
7      command: >
8        sh -c "python manage.py collectstatic --noinput 66
9          | | | gunicorn rest_mgmt.wsgi:application --bind 0.0.0.0:8000"
10     volumes:
11       - ./app
12       - static_volume:/app/staticfiles
13       - media_volume:/app/media
14     ports:
15       - "8000:8000"
16     env_file:
17       - .env
18     depends_on:
19       - db
20
21   > Run Service
22   db:
23     image: postgres:15
24     restart: always
25     environment:
26       POSTGRES_DB: Research_Project
27       POSTGRES_USER: django_user
28       POSTGRES_PASSWORD: django_pass
29     volumes:
30       - postgres_data:/var/lib/postgresql/data/
31
32 volumes:
33   postgres_data:
34   static_volume:
35   media_volume:

```

Figure 5: Docker-Compose File Code Snippet

7 CI/CD Pipeline Setup

On each commit to the master branch, it automatically triggers the CI/CD pipeline. It starts with building and pushing Docker images to all three prominent cloud providers' container registries, namely Google Cloud Artifact Registry, Azure Container Registry (ACR), and Amazon Elastic Container Registry (ECR). After that, it will deploy the app on Kubernetes clusters that are hosted on GKE, AKS, and EKS. Once deployed, the pipeline collects the deployment metrics and logs from each cloud environment, ultimately storing them in their appropriate storage services: Google Cloud Storage (GCS), Azure Blob Storage, and AWS S3. These logs are then aggregated into a single dataset, which spurs an AI model training job to initiate inside Azure Machine Learning. The intelligent deployment to the best cloud provider is done on the basis of the model prediction. In case the deployment fails it triggers an automatic Kubernetes rollback mechanism to bring back the last stable version.

(a) Step 1: Initialize and Define Variables

(b) Step 2: Azure & AWS Login

(c) Step 3: Multi-Cloud Log Check

(d) Step 4: Initial Deployment (GCP & Azure)

(e) Step 5: Log Aggregation

(f) Step 6: AI-Based Cloud Selection & Final Deploy

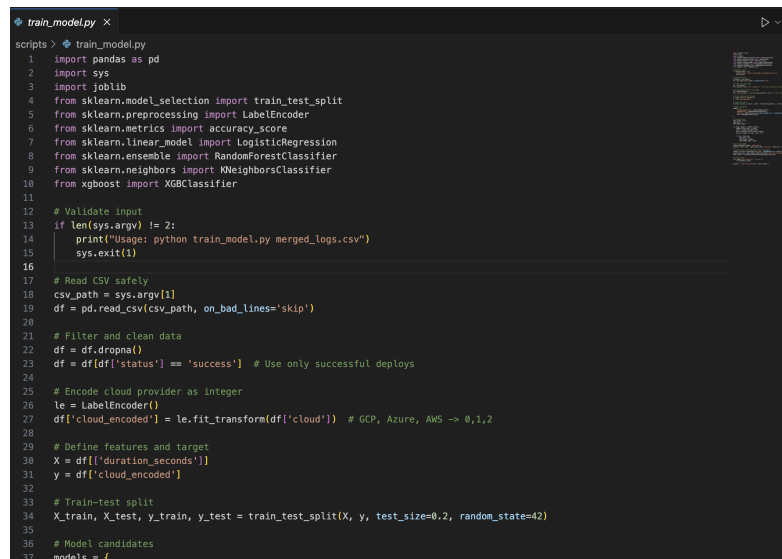
Figure 6: GitHub Actions Workflow for Multi-Cloud Deployment

8 AI Model Training via Azure ML

Training and deployment decisions are executed automatically within the pipeline.

- **ML models trained:** Logistic Regression, Random Forest, XGBoost, KNN

- **Dataset:** Merged deployment logs from all cloud providers
- **Platform:** Azure ML Studio with Python SDK



```

1  import pandas as pd
2  import sys
3  import joblib
4  from sklearn.model_selection import train_test_split
5  from sklearn.preprocessing import LabelEncoder
6  from sklearn.metrics import accuracy_score
7  from sklearn.linear_model import LogisticRegression
8  from sklearn.ensemble import RandomForestClassifier
9  from sklearn.neighbors import KNeighborsClassifier
10 from xgboost import XGBClassifier
11
12 # Validate input
13 if len(sys.argv) != 2:
14     print("Usage: python train_model.py merged_logs.csv")
15     sys.exit(1)
16
17 # Read CSV safely
18 csv_path = sys.argv[1]
19 df = pd.read_csv(csv_path, on_bad_lines='skip')
20
21 # Filter and clean data
22 df = df.dropna()
23 df = df[df['status'] == 'success'] # Use only successful deploys
24
25 # Encode cloud provider as integer
26 le = LabelEncoder()
27 df['cloud_encoded'] = le.fit_transform(df['cloud']) # GCP, Azure, AWS -> 0,1,2
28
29 # Define features and target
30 X = df[['duration_seconds']]
31 y = df['cloud_encoded']
32
33 # Train-test split
34 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
35
36 # Model candidates
37 models = {

```

Figure 7: AI Model Training Code Snippet

9 Rollback Strategy

- **Triggered using:** `kubectl rollout undo deployment/django-app`
- **Activated if:**
 - Logs show sub-threshold metrics
 - AI model predicts poor performance post-deployment

10 Dataset Summary

- **Total Records:** 9000
- **Columns:** cloud, start_time, end_time, duration_seconds, cpu_utilization, memory_usage, deployment_size, cost_estimate, status_code, retry_count, infra_load_score, region, status
- **Cloud Platforms:** GCP, Azure, AWS
- **Duration Range (seconds):** 30 to 600
- **Status Types:** success

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	cloud	start_time	end_time	duration_seconds	cpu_utilization	memory_usage	deployment_size	cost_estimate	status_code	retry_count	infra_load_score	region	status	
1	GCP	2025-04-11T11:01:30Z	2025-04-11T11:03:53Z	134	36.84	116.37	280.57	0.0809	200	1	72.63	eu-west-1	success	
2	GCP	2025-04-11T11:03:53Z	2025-04-11T11:05:32Z	273	80.52	1311.71	1443.21	0.0801	202	2	73.61	eu-west-1	success	
3	GCP	2025-04-11T11:05:32Z	2025-04-11T11:06:00Z	404	44.62	2838.41	223.38	0.0808	200	1	77.38	eu-west-1	success	
4	GCP	2025-04-11T11:06:00Z	2025-04-11T11:07:33Z	527	62.06	7333.98	1218.58	0.106	200	1	51.98	eu-west-1	success	
5	GCP	2025-04-11T11:07:33Z	2025-04-11T11:08:33Z	439	77.13	7596.86	1487.78	0.044	201	2	63.05	eu-west-1	success	
6	GCP	2025-04-11T11:08:33Z	2025-04-11T11:09:00Z	362	28.89	786.73	374.5	0.0726	202	0	64.59	eu-west-1	success	
7	GCP	2025-04-11T11:09:00Z	2025-04-11T11:10:33Z	437	81.56	2440.21	257.72	0.1901	202	2	64.7	eu-west-1	success	
8	GCP	2025-04-11T11:10:33Z	2025-04-11T11:11:33Z	32	38.06	7299.22	1497.43	0.0603	201	1	81.13	eu-west-1	success	
9	GCP	2025-04-11T11:11:33Z	2025-04-11T11:12:44Z	528	53.91	1289.22	863.21	0.0907	201	1	96.01	eu-west-1	success	
10	GCP	2025-04-11T11:12:44Z	2025-04-11T11:13:30Z	561	25.05	2439.57	850.84	0.1403	201	0	63.5	eu-west-1	success	
11	GCP	2025-04-11T11:13:30Z	2025-04-11T11:15:04Z	209	22.41	1964.28	1371.1	0.0476	201	1	97.41	eu-west-1	success	
12	GCP	2025-04-11T11:15:04Z	2025-04-11T11:17:00Z	599	88.63	2634.21	498.43	0.0367	202	1	74.49	eu-west-1	success	
13	GCP	2025-04-11T11:17:00Z	2025-04-11T11:18:00Z	362	50.56	5892.44	1363.54	0.1345	202	0	87.61	eu-west-1	success	
14	GCP	2025-04-11T11:18:00Z	2025-04-11T11:19:00Z	208	84.49	2810.87	542.04	0.1902	201	0	85.78	eu-west-1	success	
15	GCP	2025-04-11T11:19:00Z	2025-04-11T11:20:00Z	587	45.2	6011.17	1142.79	0.0404	202	0	59.84	eu-west-1	success	
16	GCP	2025-04-11T11:20:00Z	2025-04-11T11:21:00Z	428	51.01	2017.48	1079.49	0.1289	200	1	68.82	eu-west-1	success	
17	GCP	2025-04-11T11:21:00Z	2025-04-11T11:22:00Z	481	54.85	182.01	1406.52	0.0667	200	2	74.14	eu-west-1	success	
18	GCP	2025-04-11T11:22:00Z	2025-04-11T11:23:00Z	387	64.26	7855.4	964.87	0.0128	200	0	64.85	eu-west-1	success	
19	GCP	2025-04-11T11:23:00Z	2025-04-11T11:24:00Z	262	46.94	2453.61	700.15	0.0754	201	1	68.21	eu-west-1	success	
20	GCP	2025-04-11T11:24:00Z	2025-04-11T11:25:00Z	34	22.09	3635.57	1260.03	0.189	201	1	56.84	eu-west-1	success	
21	GCP	2025-04-11T11:25:00Z	2025-04-11T11:26:00Z	62	36.35	4662.85	1153.24	0.1991	200	0	65.73	eu-west-1	success	
22	GCP	2025-04-11T11:26:00Z	2025-04-11T11:27:00Z	194	40.48	497.52	1022.64	0.0382	202	2	68.75	eu-west-1	success	
23	GCP	2025-04-11T11:27:00Z	2025-04-11T11:28:00Z	513	40.08	5374.49	222.95	0.1413	202	1	71.37	eu-west-1	success	
24	GCP	2025-04-11T11:28:00Z	2025-04-11T11:29:00Z	591	54.73	4252.59	1215.86	0.1322	202	1	71.59	eu-west-1	success	
25	GCP	2025-04-11T11:29:00Z	2025-04-11T11:30:00Z	359	59.53	6596.62	391.39	0.1148	201	0	59.32	eu-west-1	success	
26	GCP	2025-04-11T11:30:00Z	2025-04-11T11:31:00Z	253	81.08	5038.33	963.4	0.0403	202	2	75.87	eu-west-1	success	
27	GCP	2025-04-11T11:31:00Z	2025-04-11T11:32:00Z	551	73.55	2272.06	728.83	0.1287	201	1	87.44	eu-west-1	success	
28	GCP	2025-04-11T11:32:00Z	2025-04-11T11:33:00Z	482	55.36	1931.97	948.67	0.0846	201	1	77.78	eu-west-1	success	
29	GCP	2025-04-11T11:33:00Z	2025-04-11T11:34:00Z	254	77.78	3885.37	567.31	0.0629	201	0	91.85	eu-west-1	success	
30	GCP	2025-04-11T11:34:00Z	2025-04-11T11:35:00Z	153	79.91	3977.43	390.15	0.0384	202	0	50.98	eu-west-1	success	
31	GCP	2025-04-11T11:35:00Z	2025-04-11T11:36:00Z	555	50.57	6106.32	388.11	0.1005	201	0	74.3	eu-west-1	success	
32	GCP	2025-04-11T11:36:00Z	2025-04-11T11:37:00Z											

Figure 8: Dataset Snippet

11 Output & Prediction

- Final cloud decision stored in best_cloud.txt.
- Automated deployment to predicted cloud provider (GCP/AWS/Azure).

12 Troubleshooting

Issue	Solution
Deployment failed	Check log output in CI job, validate credentials
Rollback not triggered	Verify thresholds & log parsing script
AI not training	Check Azure ML authentication or dataset format

Table 3: Common Issues and Their Solutions