National College of Ireland

Configuration Manual

MSc Research Project
MSCCLOUD

Ammar Naeemul Haque Ansari
Student ID: x22197567

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ansari Ammar Naeemul Haque		
Student ID:	X22197567		
Programme:	MSCCLOUD	**Year:**	2023-24
Module:	Research in Computing		
Lecturer:	Aqeel Kazmi		
Submission Due Date:	03/01/2025		
Project Title:	Feedback Control loops for performance SLAs in Cloud Computing		
Word Count:	3 **Page Count:** 852		

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:	Ammar
Date:	03/01/2025

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

Ammar Naeemul Haque Ansari
X22197567

1 System Requirements and Environment Setup

Our cloud resource management framework is implemented mainly in Google Colab. Since it provides a constant reproducible development environment across users, on the available platform, supported by 12GB of RAM and GPU, that is usually NVIDIA T4 or P100, can well meet our computational needs. Even though local hardware implementation is achievable, we consider encouraging Colab due to many reproducibility-related arguments for avoiding hardware inconsistencies among others.The only local requirements are a stable internet connection and a modern web browser; Chrome is preferred for maximum compatibility with Colab.
Implementation Language Used: Here, the advantages of Python 3.10 are taken for machine learning practices. Almost all the essential libraries come inbuilt with Google Colab to ease setup. The one thing the user needs is a Google account to be able to access Colab. That is a Kaggle account too for dataset access as the extra requirement. Besides that, Google drive space is needed, about 2GB for your use case in storing the data, model checkpoints, or different output files.
If possible, we recommend the use of Colab's GPU runtime for the best performance. This will greatly speed things up for model training and other computationally expensive operations that are especially important during the initial setup or validation phases. The overall system will work without GPU acceleration, but processing large workload patterns will take several times longer.

2 Data and Model Configuration

It uses the Google 2019 cluster trace dataset, which provides real-world workload patterns for testing and validation. Initial setup involves correctly configuring the data pipeline: accessing the dataset through the Kaggle API and setting up appropriate storage paths in the Colab environment. Here is the essential configuration code:

```
import kagglehub
import pandas as pd
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor, IsolationForest

 Core system configuration
system_config = {
        'data_source': "derrickmwiti/google-2019-cluster-sample",
        'model_params': {
        'prediction_window': 100,
        'update_interval': 5,
```

```
        'pid_control': {'kp': 0.1, 'ki': 0.05, 'kd': 0.01}
        }
}
```

Advanced cleaning and feature extraction methodologies are applied to the data processing pipeline. Our framework automatically processes missing values and outliers, though these parameters can be tuned if needed. The system processes data in batches for efficient memory management, and checkpoint saving is automated in case of long training sessions where data could get lost.

3 Implementation and Monitoring

In the design of our resource management framework, there are three major integrations: workload prediction, anomaly detection, and feedback control. Workload prediction is implemented through a Gradient Boosting Regressor, which has been carefully tuned for time series and resource utilization patterns. For anomaly detection, the implementation relies on an Isolation Forest algorithm with adaptive contamination factors, while the controller is a PID with parameters optimized through extensive testing.

Performance monitoring forms a core basis of our implementation. In the system, it is implemented to continuously monitor prediction accuracy, anomaly detection rate, resource utilization pattern, and response time. The logging of all the metrics in both the Colab environment and optionally saves to Google Drive for persistence. At initial setup or optimization phases, the above-listed metrics should be constantly observed by any user for assured optimum performance.

These are common implementation challenges: dataset access errors, memory limitations, and availability of runtime GPU. In case of dataset access errors, it may be necessary to standardize Kaggle API credentials configuration. Memory limitations may be overcome by adjusting batch sizes or data sampling in the case of big workloads. If there is no GPU runtime, the system will harvest processing on the CPU, but execution may slow down.

This includes periodic saving of model checkpoints in Google Drive, monitoring resource utilization via the Colab interface, and cleaning up old outputs. The system uses automatic logging for debug purposes, which can be controlled according to the requirement with the help of logging level adjustment. It is worth pointing out that keeping at least one backup of every critical model checkpoint and configuration file saves you from data loss.

The framework is designed to be flexible, easily allowing users to adjust parameters as necessary for their use while preserving performance characteristics. Regular review of system logs and performance metrics will provide insight into adjusting configuration parameters as needed. For production deployment, we recommend implementing additional monitoring alerts and establishing regular backups.

4 References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing
You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* 27(5): 1310–1333.

Feng, G. and Buyya, R. (2016). Maximum revenue-oriented resource allocation in cloud, *IJGUC* 7(1): 12–21.

Gomes, D. G., Calheiros, R. N. and Tolosana-Calasanz, R. (2015). Introduction to the special issue on cloud computing: Recent developments and challenging issues, *Computers & Electrical Engineering* 42: 31–32.

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.