National College of Ireland

# Configuration Manual

MSc Research Project
Cloud Computing


Laura Mendez
Student ID: x23172061

School of Computing
National College of Ireland


Supervisor: Punit Gupta

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Laura Mendez |
| **Student ID:** | x23172061 |
| **Programme:** | Cloud Computing |
| **Year:** | 2018 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Punit Gupta |
| **Submission Due Date:** | 12/08/2024 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 1456 |
| **Page Count:** | 8 |

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:** | 14th September 2024 |

**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

Laura Mendez
x23172061

## Contents

# 1 Introduction

The Drift Detection Solution is designed to identify and risk-evaluate state drifts in Infrastructure as Code (IaC) projects.

- **User Interface (UI) Module**: The User Interface (UI) Module is designed to provide a user-friendly platform for interacting with the API. This module must allow users to easily check the health of their IaC projects. In the Web Interface, the user could set up IaC projects, initiate drift detection, view the results, and get the details needed to prioritize remediation activities.
- **API Module**: The API Module is designed to provide a programmable interface for storing the input data, accessing and managing the state drift analysis results, and facilitating integration with external systems or tools.
- **Database**: The database stores the configuration and results data, which is encrypted to ensure security.

This document provides step-by-step setup, configuration, and running of the Drift Detection Solution.

# 2 Required Tools

- **Node.js**: Version 14.x or higher
- **Python**: Version 3.8 or higher
- **Docker**: Version 25 or higher
  - **MySQL Docker Image**: 8.0
  - **Keycloak Docker Image**: latest

# 3 Accessing the Code

To set up and run the Drift Detection Solution, you will need access to the source code. Either through the git private repositories (you need to request access before proceeding) or a zip file with the two projects inside (delivered through Moodle).

To get the code through the git private repositories follow the following steps, otherwise, you can go directly to the Environment Variables Section.

## 3.1 Requesting Access

Requesting Access to Code Repositories:

1. Mail to `x23172061@student.ncirl.ie` with the subject line "Access Request for Drift Detection Solution Repositories."
2. Kindly include in your mail, your full name, organization, and a brief description of why you need access to these repositories.

Following your request, you will have the invitation upon acceptance to access the repositories.

## 3.2 Cloning the Repositories

Clone the repositories using the following steps:

1. **Clone the User Interface (UI) Module**:

   ```
   git clone https://github.com/laurusik/drift_detector_user_interface.git
   ```

2. **Clone the API Module**:

   ```
   git clone https://github.com/laurusik/drift_detector_api.git
   ```

3. **Navigate to the main branch in each repository** (if not already on it):

   ```
   git checkout main
   ```

# 4 Environment Variables

With the repositories cloned or the shared file unzipped, you can now proceed with the setup instructions provided in this document. First, you should configure the necessary environment variables to set up the components of the application. These variables should be exported into your environment, or a `.env` file, before running the different components.

The following describes what key environment variables need to be set up:

## 4.1 UI Frontend

- `VITE_API_BASE_URL`: The base URL for the API that the frontend will communicate with.
- `VITE_API_PROXY_TARGET`: The URL of the Keycloak server for handling authentication.

## 4.2 API Backend

- `ENCRYPTION_KEY`: Used to encrypt and decrypt sensitive information stored in the database. This key must be securely stored.
- `CORS_ORIGIN`: Specifies which domains are allowed to access the API.
- `DB_USER`: The root user for the MySQL database.
- `DB_PASSWORD`: The root password for the MySQL database.
- `DB_HOST`: The host where the MySQL database is allocated.
- `DB_PORT`: The port where the MySQL database is accessed.
- `DB_NAME`: The database name for the project.

### 4.2.1 Sample UI Frontend .env File

The following example `.env` file sets out environment variables:

```
VITE_API_BASE_URL=http://127.0.0.1:5000
VITE_API_PROXY_TARGET=http://localhost:8080
```

# 5 Component Support Setup

## 5.1 Setting Up Keycloak

Keycloak is an open-source Identity and Access Management solution used to provide authentication mechanisms for the application.

**Note:** *The users made in Keycloak are those who will log in to the application via the frontend.*

1. **Pull the Keycloak Docker image**:

   ```
   docker pull quay.io/keycloak/keycloak:latest
   ```

2. **Run the Keycloak container**:

   ```
   docker run -d --name keycloak -p 8080:8080 \
   -e KEYCLOAK_USER=<username> \
   -e KEYCLOAK_PASSWORD=<password> \
   quay.io/keycloak/keycloak:latest start-dev
   ```

3. **Configure Keycloak**:

   - Access the Keycloak admin console at `http://localhost:8080`.
   - Create a new realm named `drift`.
   - Under the `drift` realm, create a client named `drift-detector`.
   - Set the `Valid redirect URIs` and `Web origins` to the URL of your frontend application (e.g., `http://localhost:5173`).
   - Create a user in the `drift` realm with appropriate roles for accessing the application.

## 5.2 Setting Up the Database

1. **Pull the MySQL Docker image**:

   ```
   docker pull mysql:8.0
   ```

2. **Run the MySQL container**:

```
docker run --name secure-mysql -e MYSQL_ROOT_PASSWORD=<password> \
-v ./datadir:/var/lib/mysql -p 3306:3306 -d mysql:8.0
```

3. **Initialize the database**:

```
CREATE DATABASE drift_detector;
USE drift_detector;
```

4. **Perform initial inserts**:

   - Create the tables and insert the necessary data using the SQL scripts provided in the `data/mysql` folder in the API project.

# 6 Web Application Setup (Frontend)

The frontend is a Vue.js application interacting with the API to present a user-friendly interface for drift detection.

**Tip:** You can also go to the `README.md` file inside the `drift_detector_user_interface` project for setup guide.

1. **Install the project dependencies**:

```
npm install
```

2. **Run the frontend in development mode**:

```
npm run dev
```

**Tip:** The frontend will run on port `5173` by default.

## 6.1 Frontend Routes Overview

The frontend application is structured around key routes, each mapped to a specific component. Below is an overview of these routes:

- **Home** (`/`):
  - **Component**: `HomeView.vue`
  - **Description**: The main landing page of the application.
- **Login** (`/login`):
  - **Component**: `LoginView.vue`
  - **Description**: The login page where users authenticate to access the application.

- **Settings** (`/settings`):
  - **Component**: `SettingsView.vue`
  - **Description**: A protected route for setting up the IaC projects.
  - **Authentication Required**: Yes
- **Drift Detection** (`/healthcheck`):
  - **Component**: `HealthCheckView.vue`
  - **Description**: A protected route where users can initiate drift detection and view detailed reports on their infrastructure.
  - **Authentication Required**: Yes

**Note:** Settings and Drift Detection routes require authentication. When a user requests access to a certain protected route and is not authenticated, they will be redirected to the login page.

# 7 API Backend Setup

The API backend is built with Flask, and it forms the core logic of the drift detection service.

**Tip:** You can also go to the `README.md` file inside the `drift_detector_api` project for setup guide.

1. **Install dependencies**:

   ```
   pip install -r requirements.txt
   ```

2. **Generate the encryption key**:

   ```
   python util/generate_key.py
   export ENCRYPTION_KEY=<generated_key>
   ```

   **Warning:** Save this key securely, as it will be required to encrypt/decrypt data in the database.

3. **Run the API**:

   ```
   flask run --host=0.0.0.0 --port=5000
   ```

   **Tip:** The API will run on port `5000` by default.

## 7.1 API Endpoints Overview

The API backend exposes several endpoints handling the core functionality of the drift detection service. Below is a description of available endpoints:

- **Health Check** (`GET /service/healthcheck`):
  - **Description**: Checks if the service is up and running.
  - **Response**: `200 OK` with a message indicating the service status.
- **Drift Report** (`POST /drift/report`):
  - **Description**: Generates a report based on the provided infrastructure ID.
  - **Parameters**: `infrastructureId` (required)
  - **Response**: `200 OK` with the drift report or `500 Internal Server Error` if an error occurs.
- **Drift Details** (`POST /drift/details`):
  - **Description**: Retrieves details of detected drifts for a specific infrastructure ID.
  - **Parameters**: `infrastructureId` (required)
  - **Response**: `200 OK` with drift details or `500 Internal Server Error` if an error occurs.
- **Create Project** (`POST /project/create`):
  - **Description**: Stores a new project and its configuration.
  - **Parameters**: `name`, `cloud`, `tool`, `state_file`, `configuration_file`, `cloud_config` (all required)
  - **Response**: `200 OK` with a confirmation message or `500 Internal Server Error` if an error occurs.
- **Project Details** (`GET /project/details`):
  - **Description**: Retrieves details for a specific project by name.
  - **Parameters**: `project_name` (required)
  - **Response**: `200 OK` with project details or `404 Not Found` if the project does not exist.
- **All Projects** (`GET /project/all`):
  - **Description**: Returns a list of all stored projects.
  - **Response**: `200 OK` with the list of projects.

# 8 Communication Ports Guide

Following is a list of the different application components with the ports used by each component:

- **Keycloak**: `8080` - for authentication services.
- **MySQL Database**: `3306` - for database operations.
- **API Backend**: `5000` - for API requests.
- **Frontend**: `5173` - for the web interface.

## 8.1 Communication Flow

Here's a brief overview of how the components communicate with each other:

- **Frontend to API**: The frontend communicates with the API backend by sending requests to `VITE_API_BASE_URL`. This communication needs to be enabled over your network.
- **API to Database**: For storing and fetching data, the API connects with a MySQL database using port `3306`. This port needs to be opened between the API server and the database server.
- **Frontend to Keycloak**: The frontend communicates with Keycloak to authenticate users through the `VITE_API_PROXY_TARGET`. Ensure this communication is allowed.
- **External Users to Frontend**: Users access the frontend at port `5173`. Ensure this port is accessible from the user's network.

# 9 Troubleshooting

- **Keycloak Issues**: Check if the Keycloak container is running, and the address for it is available at `http://localhost:8080`.
- **Database Connection Errors**: Verify that the MySQL container is running, and `MYSQL_ROOT_PASSWORD` is set correctly.
- **API Not Responding**: Make sure that the `ENCRYPTION_KEY` is correctly exported and that the API is running on port `5000`.
- **Frontend Not Loading**: Confirm that the frontend is running on port `5173` and that the `VITE_API_BASE_URL` is correctly set.

# 10 Appendices

- Keycloak Documentation
- Flask Documentation
- Vue.js Documentation