National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Abhishek Medhane
Student ID: X21182787

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Abhishek Medhane
Student ID:	X21182787
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	14/08/2023
Project Title:	Configuration Manual
Word Count:	987
Page Count:	4

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:	15th September 2023

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

Abhishek Medhane
X21182787

Table of Contents

1 Introduction

This configuration manual describes the purpose of this research and the evaluation matrices which we have considered during performing the experiments. The flow of this manual describes the environment setup required along with the hardware and software requirements. For setting up the test beds we have used various frameworks and the installation for the same has been briefly described in this manual. Also, I have added screenshots and a troubleshooting section, so that we can ensure any user can seamlessly setup the environment.

1.1 Purpose

The purpose of this manual is to provide an in depth guide for effectively setting up and tailoring stateful serverless computing using stream processing to diverse environments and user needs. Recognizing the pivotal role of precise configuration in achieving optimal performance and technicality, we describe a systematic walkthrough of prerequisites, detailed configuration steps, and advanced best practices.

1.2 Evaluation Methodology

2 Environment Setup

2.1 System Requirements

- Operating System: Linux based distribution like Ubuntu, CentOS, RedHat, etc, we are setting up our environment locally on MacOS.
- Memory: At least 8GB of RAM (16GB recommended).
- Disk: 30 GB of SSD. The actual storage requirement will depend on the job's state size.

2.2 Software Requirements

- JDK: Java 8 or 11 (Preferably Java 11) `https://openjdk.org/install/`

  ```
  java version "11.0.20" 2023-07-18 LTS
  Java(TM) SE Runtime Environment 18.9 (build 11.0.20+9-LTS-256)
  Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.20+9-LTS-256, mixed mode)
  ```

- Python: Python Version less than equal to 3.8, Apache Flink does not have support for versions greater than Python 3.8.

  ```
  python3 --version
   Python 3.8.10
  ```

- Docker and Docker Compose: Installation can be found on official documentation.

  ```
  docker --version
  Docker version 24.0.2, build cb74dfc
  docker-compose --version
  Docker Compose version v2.19.1
  ```

- Editor: We are creating our project in VS Code.

3 Installation of Required Frameworks

Firstly, we create a blank project inside VS Code were we will maintain all the installation and requirement specification. The source code for installation of each frameworks can be found in the code repository submitted inside zip on NCI Moodle.

3.1 Getting Started with Apache Kafka and Zookeeper

Kafka is used in real time streaming data architectures for data analytics and the purpose of zookeeper is to Manage and coordinate the Kafka broker. We will the set up both of them inside docker container using docker-compose. Create a new directory named Kafka where we will create our configuration file.

```
mkdir Kafka
vim docker-compose.yml
docker-compose up -d
```

3.2 Getting Started with Apache Flink

Apache Flink is a powerful distributed stream processing frameworks that offer capabilities for processing and analyzing data in real time. We will create a new directory for the installation of Flink.

```
mkdir Flink
vim docker-compose.yml
docker-compose up -d
```

3.3 Getting Started with PyFlink

Apache Flink provides with a default Python API called PyFlink which allows us to create ETL pipelines which can be used for processing large real time streaming workloads.

```
python3 -m venv venv
source venv/bin/activate
pip3 install apache-flink
```

3.4 Getting Started with Elasticsearch and Kibana

Elasticsearch and Kibana are powerful data analysis and visualization tools which are very efficient in providing data storage, searching capability and interactive graphs. We will also setup these inside docker container. Elasticsearch can be found on port :9200 whereas, Kibana Dashboards can be found on port :5601.

```
mkdir Kafka
vim docker-compose.yml
docker-compose up -d
```

4 End-to-end testing the pipeline for Stream Processing

4.1 Launch Confluent Kafka Docker Environment

In same kakfa directory as docker-compose.yaml

```
cd Kafka
vim docker-compose.yml
docker-compose up -d
```

4.2 Create Kafka source topic named salesitems

```
docker exec -it broker kafka-topics --create \
    --bootstrap-server localhost:9092 \
    --topic salesitems
```

4.3 Create Kafka sink topics named processedsales and processedsales2

```
docker exec -it broker kafka-topics --create \
    --bootstrap-server localhost:9092 --topic processedsales
```

4.4 Download the Kafka Connector Jar using HTTPie HTTP Shell Client

```
http --download https://repo.maven.apache.org/maven2/org/apache/flink/flink-sql-conne
```

4.5 Run sales_producer py program

```
python sales_producer.py --bootstrap-server localhost:9092 --topic salesitems
```

4.6 In another terminal / cmd prompt (with the Python Virtual Environment activated) run the Flink program

```
python eventtime_windows.py
```

4.7 Verify that the processedsales and processedsales2 topic has data using the kafka-console-consumer

```
docker exec -it broker kafka-console-consumer --from-beginning \
    --bootstrap-server localhost:9092 \
    --topic processedsales
```

References

Download and install prebuilt OpenJDK packages `https://openjdk.org/install/`

Install Docker Engine `https://docs.docker.com/engine/install/ubuntu/`

Download and Install Kafka `https://kafka.apache.org/downloads`

Install Apache Flink Python `https://pypi.org/project/apache-flink/`