National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Keshav Singh
Student ID: X22187201

School of Computing
National College of Ireland

Supervisor: Sai Emani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …Keshav Nandkishor Singh….
……

Student ID: ……X22187201……
……..……

Programme: … ……MSC in Cloud Computing **Year:** ……2024……..
……

Module: MSc Research Project……..……

Lecturer: …Sai Emani……..……

Submission Due Date: ……16-09-2024……..……

Project Title: … Enhancing Virtual Machine Migration through the utilization of Reinforcement Learning and Apache Kafka Messaging Services……..…
……

Word Count: 562 …… **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: ……Keshav Nandkishor Singh
……

Date: ……16-09-2024
……

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

Keshav Singh
Student ID: X22187201

1 System Requirements

Before proceeding to implementation make sure the systems meet the following requirements.

Operating System	Windows, macOS, Linux
Processor	Multi-core
RAM	16 GB
Storage	At least 10GB of free disk space
Python version	3.7 or higher
Java version	8 or higher
Apache Kafka	Latest compatible version

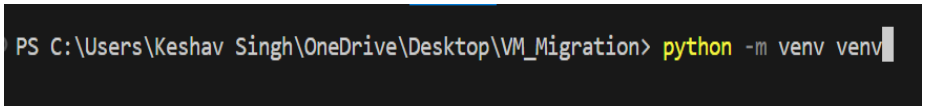
Table 1. System Requirements

2 Software Requirements

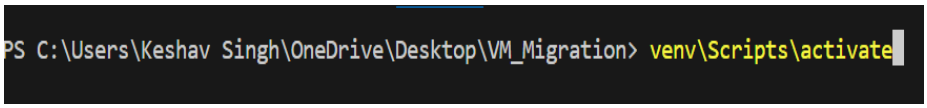
2.1 Python and Virtual Environment

1. Download and install Python
2. Create virtual environment
    - Navigate to project directory

    - Create a virtual environment



    - Activate



3. Install the required libraries
    - pip install tensorflow gym kafka-python

2.2 Java (Prerequisite for Apache Kafka)

2.3 Apache Kafka

1. Download and extract Kafka

2. **Setup Kafka**
    - Navigate to the directory
      cd C:\kafka_2.13-3.7.1\bin\windows
    - Start Zookeeper
      > zookeeper-server-start.bat ..\..\config\zookeeper.properties
    - In new terminal start Kafka broker
      kafka-server-start.bat ..\..\config\server.properties

3 Kafka configuration

3.1 Create Kafka Topic

1. **Create Topic for the VM Metrics**

kafka-topics.bat --create --topic vm-metrics --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1

3.2 Start Kafka Producer and Consumer

1. **Kafka Producer**
    - Kafka Producer will generate and send VM metrics to the kafka topic i.e. vm metrics topic
    - Create python script for producer named kafka_producer.py

```python
import time
import json
import random
from kafka import KafkaProducer

producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda v: json.dumps(v).encode('utf-8'))

iterations = 5000  # Number of iterations before stopping

for _ in range(iterations):
    vm_metrics = {
        'vm_id': random.randint(1, 10),
        'cpu_usage': random.uniform(0, 100),
        'memory_usage': random.uniform(0, 100),
        'network_usage': random.uniform(0, 100)
    }
    producer.send('vm-metrics', vm_metrics)
    print(f"Sent: {vm_metrics}")
    #time.sleep(1)
```

Fig 1. Kafka Producer script

    - Run the script

      python kafka_producer.py

2. Kafka consumer and RL Agent

- Create python script named rl_agent.py

```python
class RLAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.model = self.build_model()

    def build_model(self):
        model = tf.keras.models.Sequential([
            tf.keras.layers.Input(shape=(self.state_size,)),
            tf.keras.layers.Dense(24, activation='relu'),
            tf.keras.layers.Dense(24, activation='relu'),
            tf.keras.layers.Dense(self.action_size, activation='linear')
        ])
        model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(learning_rate=0.001))
        return model

    def train(self, env, episodes, batch_size=32):
        for e in range(episodes):
            state = env.reset()
            state = np.reshape(state, [1, self.state_size])
            for time in range(500):
                action = np.argmax(self.model.predict(state)[0])
                next_state, reward, done, _ = env.step(action)
                next_state = np.reshape(next_state, [1, self.state_size])
                self.model.fit(state, np.array([action]), epochs=1, verbose=0)
                state = next_state
                if done:
                    print(f"episode: {e}/{episodes}, score: {time}")
                    break
```

```python
# Initialize Kafka consumer
consumer = KafkaConsumer('vm-metrics', bootstrap_servers='localhost:9092',
                         value_deserializer=lambda m: json.loads(m.decode('utf-8')),
                         auto_offset_reset='earliest',  # Ensure we consume from the beginning if no off
                         enable_auto_commit=True,        # Commit offsets automatically
                         group_id='vm-migration-group')  # Group ID for consumer group
print("Consumer initialized. Waiting for messages...")
env = VMEnvironment()
agent = RLAgent(state_size=env.observation_space.shape[0], action_size=env.action_space.n)
max_messages = 10
messages_processed = 0

for message in consumer:
    if messages_processed >= max_messages:
        break

    print(f"Received message: {message.value}")  # Debugging statement to print the received message
    vm_metrics = message.value
    state = np.array([vm_metrics['cpu_usage'], vm_metrics['memory_usage'], vm_metrics['network_usage']])
    state = np.reshape(state, [1, env.observation_space.shape[0]])
    env.state = state[0]  # Ensure the environment state is properly set
    action = np.argmax(agent.model.predict(state)[0])
    next_state, reward, done, _ = env.step(action)
    next_state = np.reshape(next_state, [1, env.observation_space.shape[0]])
    agent.model.fit(state, np.array([action]), epochs=1, verbose=0)
    print(f"Processed message. Action: {action}, Reward: {reward}")  # Debugging statement to confirm pr
    messages_processed += 1
    if done:
        state = env.reset()
    time.sleep(1)

print("Finished processing messages.")
```

Run the RL agent script

- python rl_agent.py

4 Run complete implementation

1. Start Zookeeper and Kafka Broker
2. Run Kafka Producer – this script will start sending vm metrics to the kafka topic
3. Run RL Agent – this script will start processing vm metrics and make migration decisions
4. Monitor Outputs – kafka producer script will print the vm metrics and rl_agent script will print received metrics, action and rewards.

5 Stopping services

```bash
deactivate
```