

# Guidance

---

## Access to Azure Lab

[Download Remote Desktop App to connect your Azure VM](#)

[Getting it for Mac](#)

[Getting it for Windows](#)

[Start Azure VM](#)

## IntelliJ IDE

[Start an new Python project](#)

[To install a Python dependency package](#)

[Dependencies Needed in Course](#)

## Ubuntu VMs

[To start Ubuntu VMs](#)

[Ubuntu VM's IP](#)

[To get Ubuntu VM's IP address](#)

[SSH to Ubuntu VMs](#)

## Prepare necessary environment for Ubuntu VMs

[Python](#)

[Docker-compose tool](#)

## Transfer files with VMs

[By MobaXterm \(Recommended\)](#)

[By shared folder \(Not Recommended\)](#)

## How to write a docker-compose configuration file

## How to build your code into docker image

[Prepare a Dockerfile](#)

[Prepare a docker-compose configuration file](#)

[Execution flow](#)

## Coursework supplements

[Overview structures](#)

[Task](#)

[Task](#)

[Task](#)

## [Implementation Examples](#)

[How to send HTTP request to get sensor data](#)

[How to define MQTT subscriber in Python](#)

[How to define MQTT publisher in Python](#)

[How to define RabbitMQ consumer in Python](#)

[How to define RabbitMQ producer in Python](#)

[How to use Matplotlib to visualize a line chart](#)

[How to use Machine Learning Engine](#)

## [Issues solving \(Please carefully follow every step and tip in this section\)](#)

[Docker logs cannot print output](#)

[An error occurred while attempting to start VM](#)

[Cannot get sensor data from Urban Observatory](#)

## [Appendix](#)

[About target sensors information](#)

## [For Demonstrators](#)

# Access to Azure Lab

You can access Azure Lab at anywhere (both on campus or out of campus).

Every student has admin privileges for Windows and Ubuntu, which means you can install any software you need.

## Download Remote Desktop App to connect your Azure VM

If you are not using university's desktop, please download this App by yourself ! Otherwise, you could login VM in Azure Lab directly by downloading access file.

### Getting it for Mac

App Store:  Microsoft Remote Desktop

The screenshot shows the Mac App Store preview for Microsoft Remote Desktop. At the top, there's a navigation bar with links for Store, Mac, iPad, iPhone, Watch, AirPods, TV & Home, Only on Apple, Accessories, and Support. Below the navigation is a search bar and a 'View in Mac App Store' button. The main content area features the Microsoft Remote Desktop icon (a red square with a white 'K' shape), the app name, developer information (Microsoft Corporation), a rating of 4.4 stars, and a 'Free' status. A 'View in Mac App Store' button is also present. Below this is a section titled 'Screenshots' with three images showing the app in use on a Mac desktop. A descriptive text block follows, stating: 'Use Microsoft Remote Desktop for Mac to connect to Azure Virtual Desktop, Windows 365, admin-provided virtual apps and desktops, or remote PCs. With Microsoft Remote Desktop, you can be productive no matter where you are.' There are 'GET STARTED' and 'more' buttons, along with a 'Version History' link at the bottom.

## Getting it for Windows

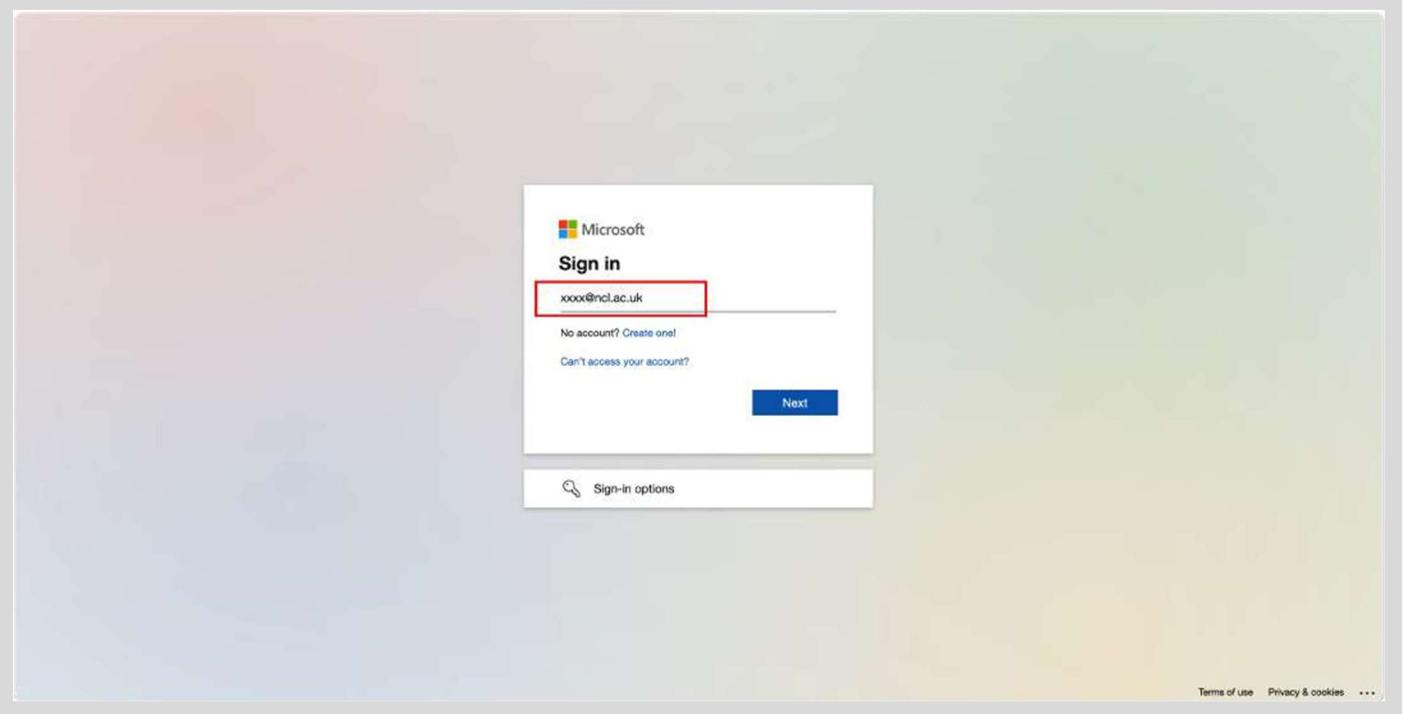
Microsoft Store: Get \$Microsoft Remote Desktop from the Microsoft Store

The screenshot shows the Microsoft Store page for Microsoft Remote Desktop. At the top, there's a navigation bar with links for Microsoft, Windows Apps, Software, Games & Entertainment, All Microsoft, Developers, Sign in, and a user profile icon. The main content area features the Microsoft Remote Desktop icon, the app name, developer information (Microsoft Corporation), a rating of 4.5 stars, and a 'Free' status. A 'Get in Store app' button is also present. Below this is a section titled 'Description' with a text block: 'Use the Microsoft Remote Desktop app to connect to a remote PC or virtual apps and desktops made available by your admin. The app helps you be productive no matter where you are.' There is a 'Getting Started' section with a link to 'Configure your PC for remote access first. Download the Remote Desktop assistant to your PC and let it do the work for you: <https://aka.ms/RDSetu...>'. A 'Read more' link is also present. To the right of the description are sections for 'Details', 'Published by', 'Terms', and 'Developer and IT'. A PEGI 3 rating badge is shown.

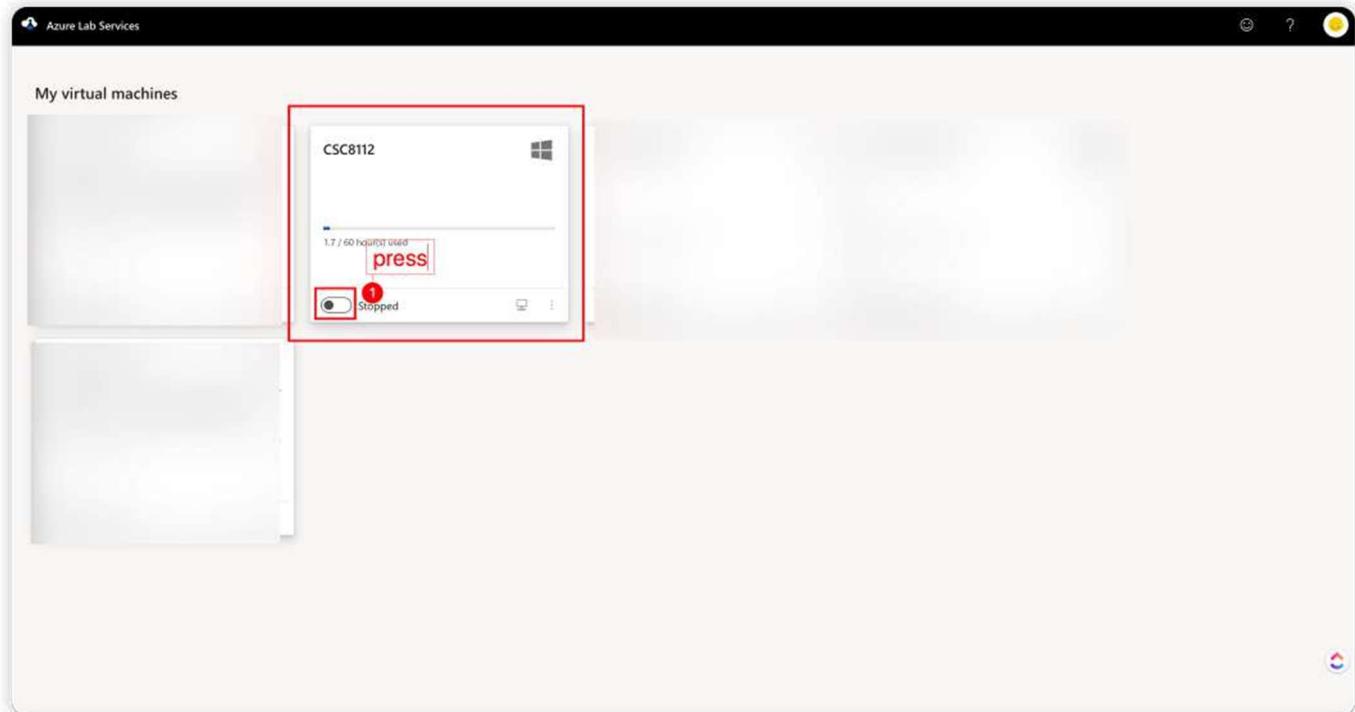
## Start Azure VM

1. Go to: Azure Lab Services

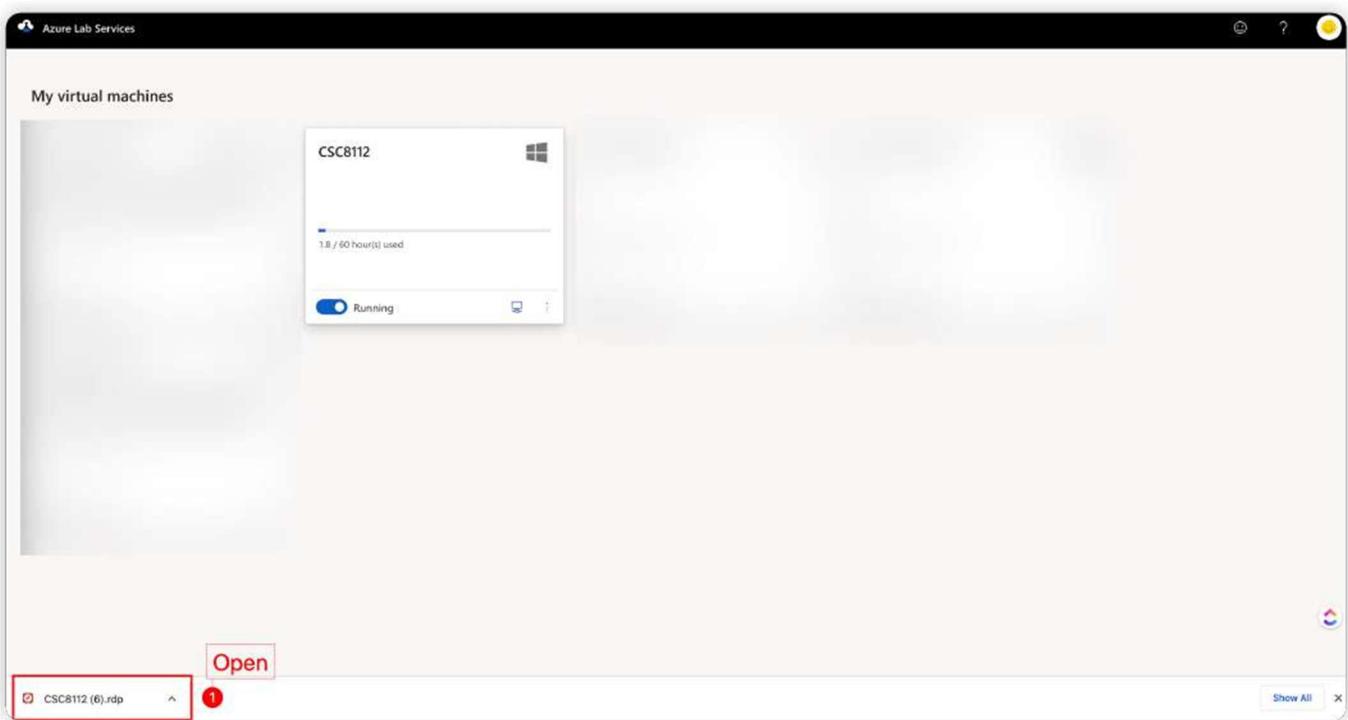
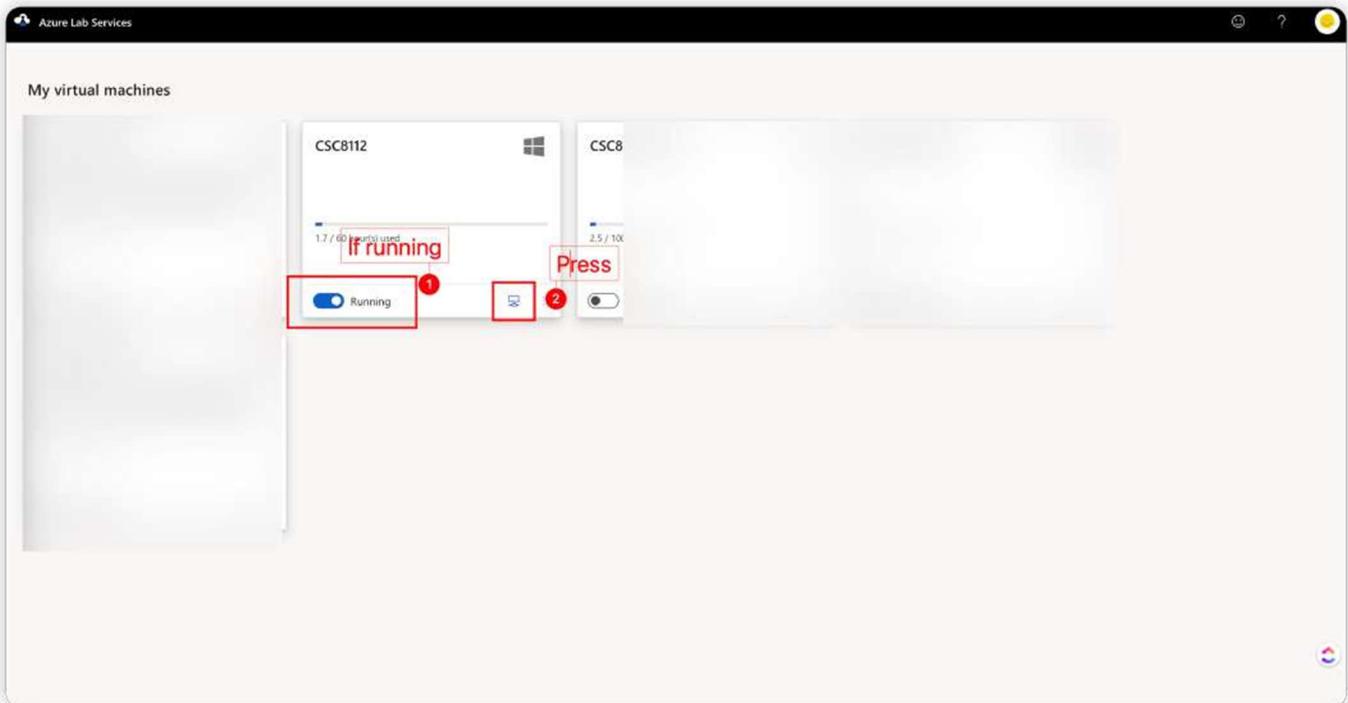
## 2. Login with your University account



## 3. Start your VM

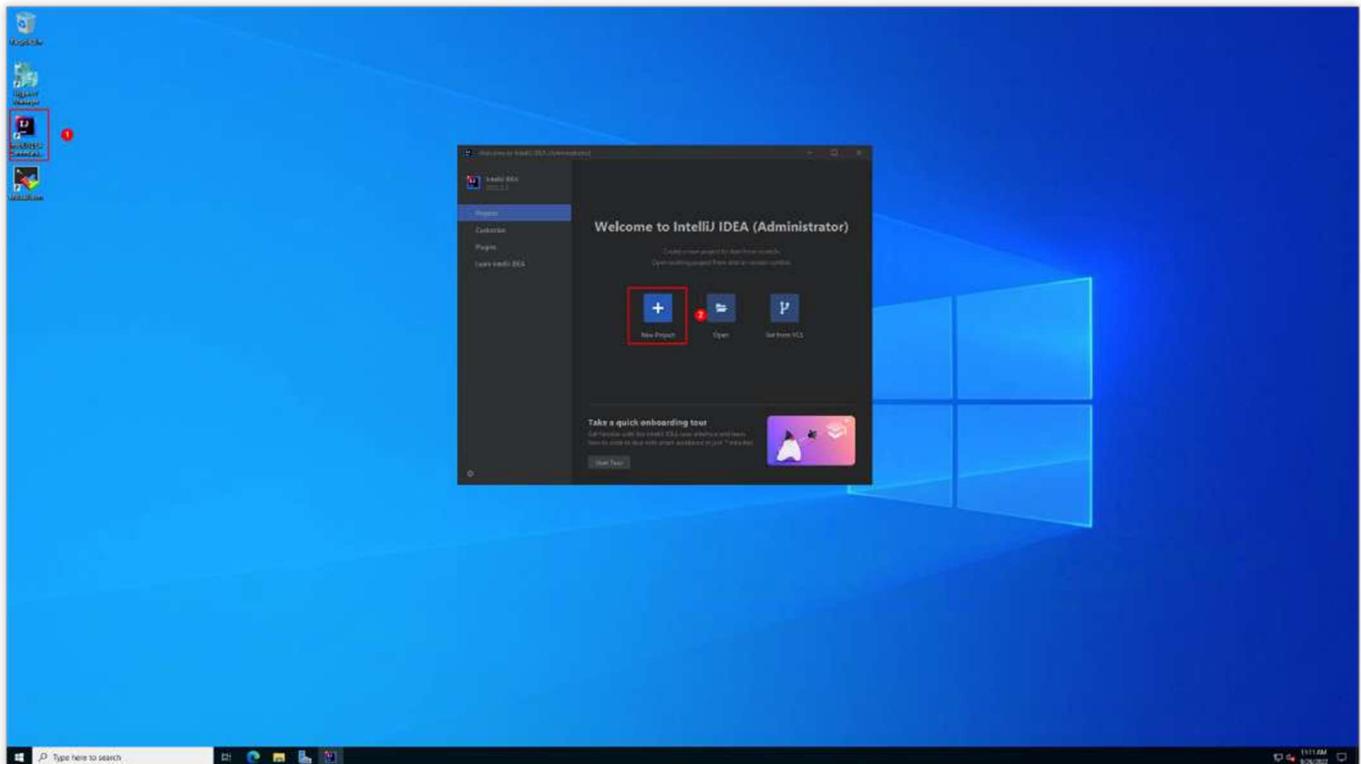


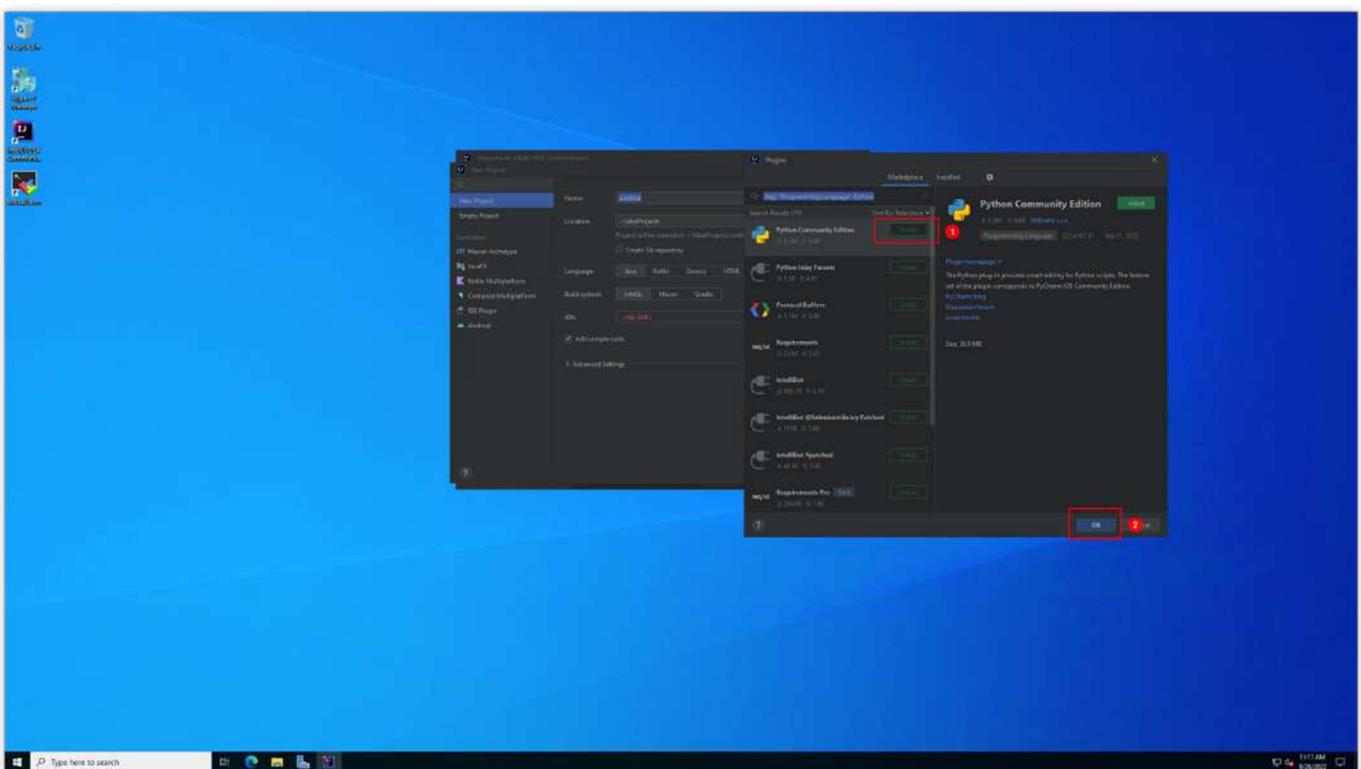
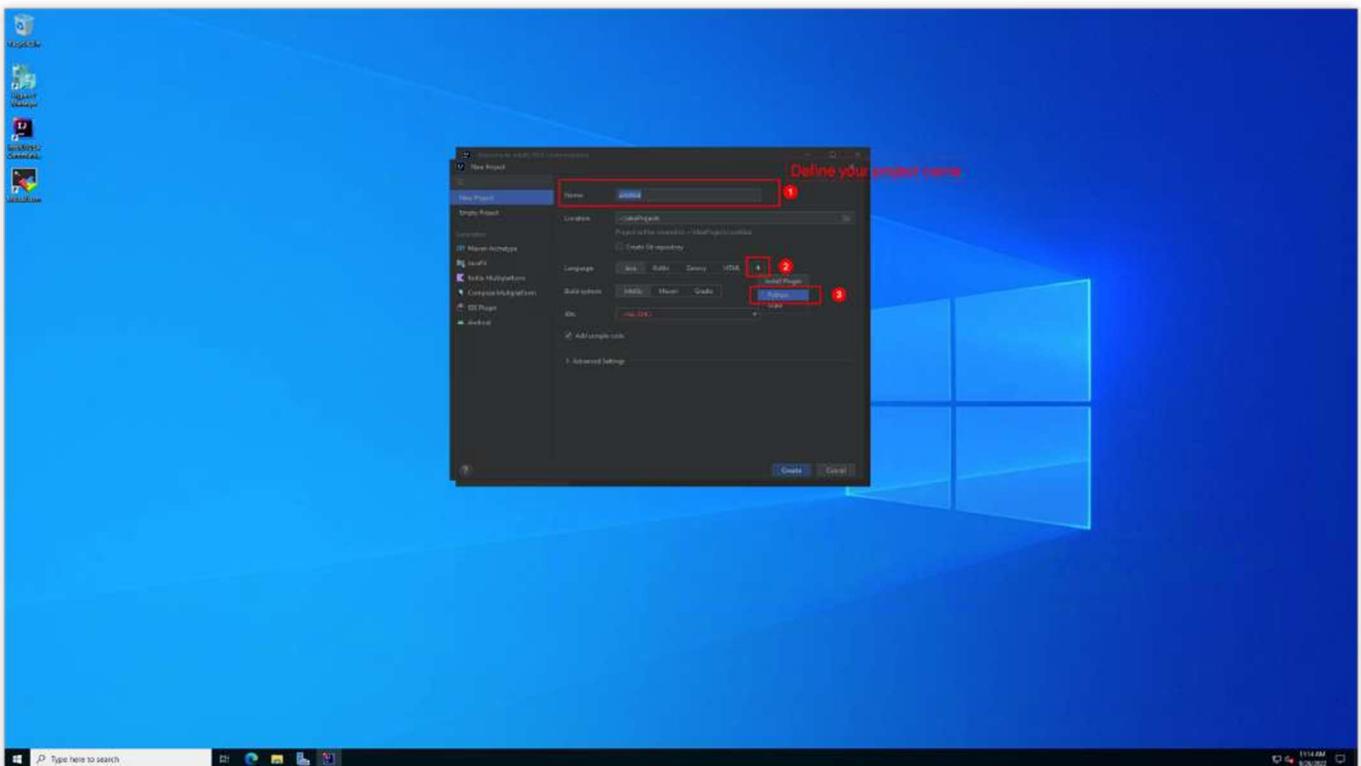
## 4. Connect your VM with password (Please check it in Teams channel)

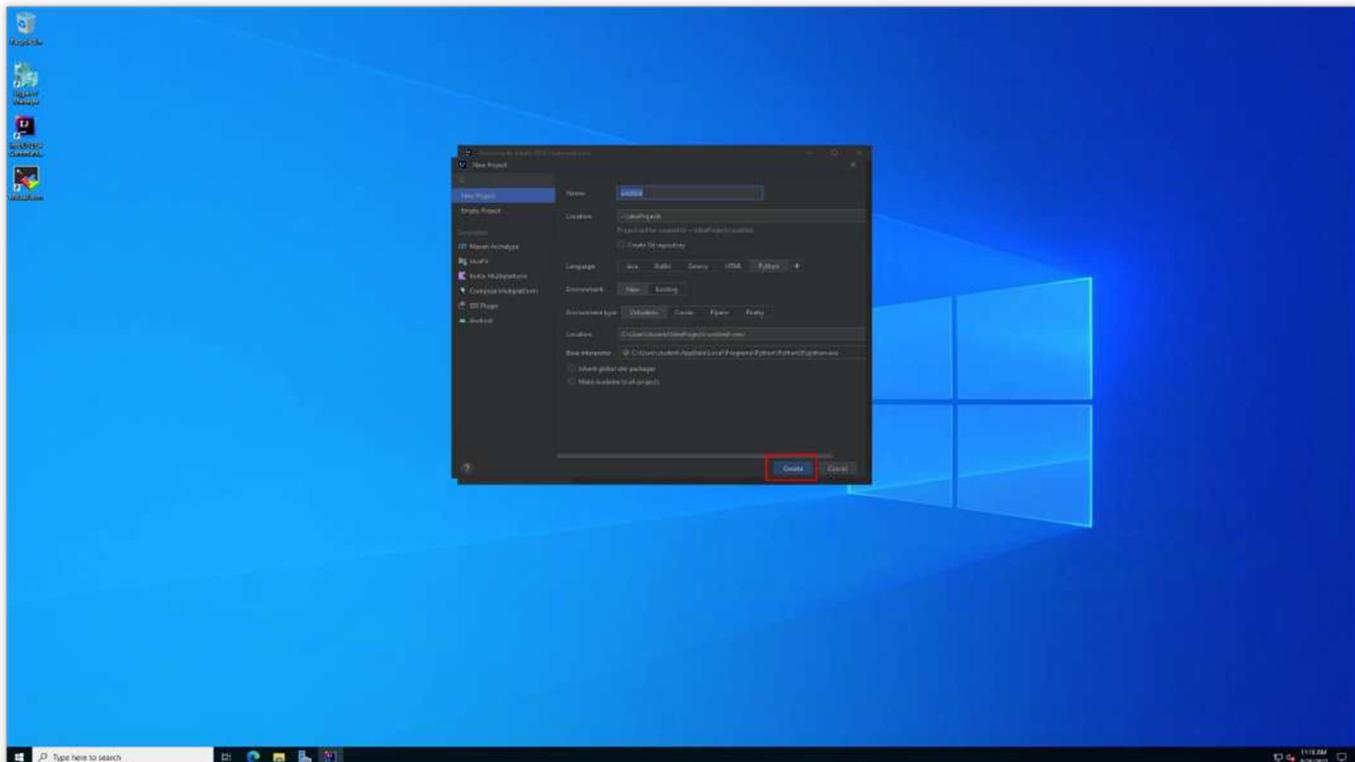


## IntelliJ IDE

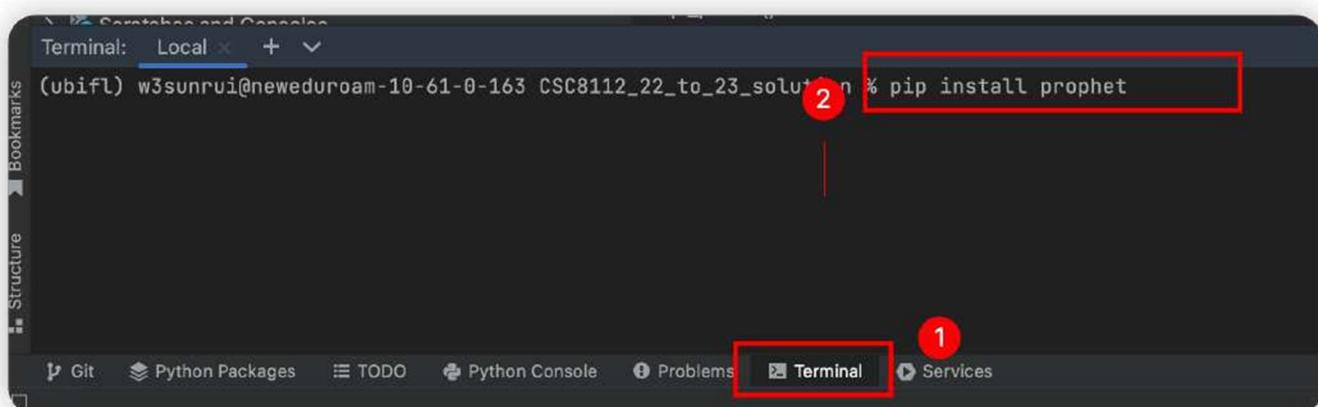
Start an new Python project







To install a Python dependency package



Dependencies Needed in Course

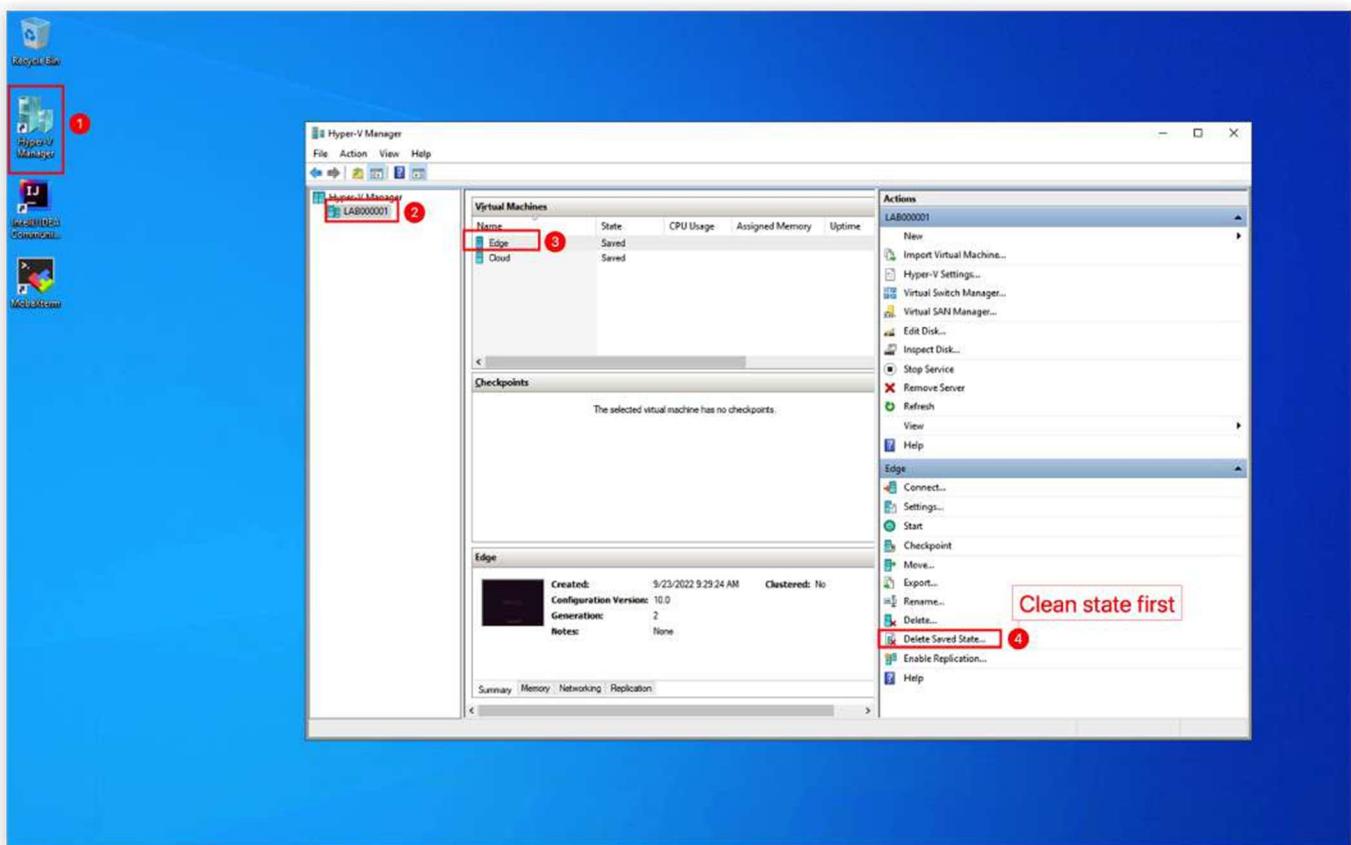
Package Name	Version (== or latest)	Install Command	Task
requests	2.28.1	pip install requests	1
paho-mqtt	1.6.1	pip install paho-mqtt	1 & 2

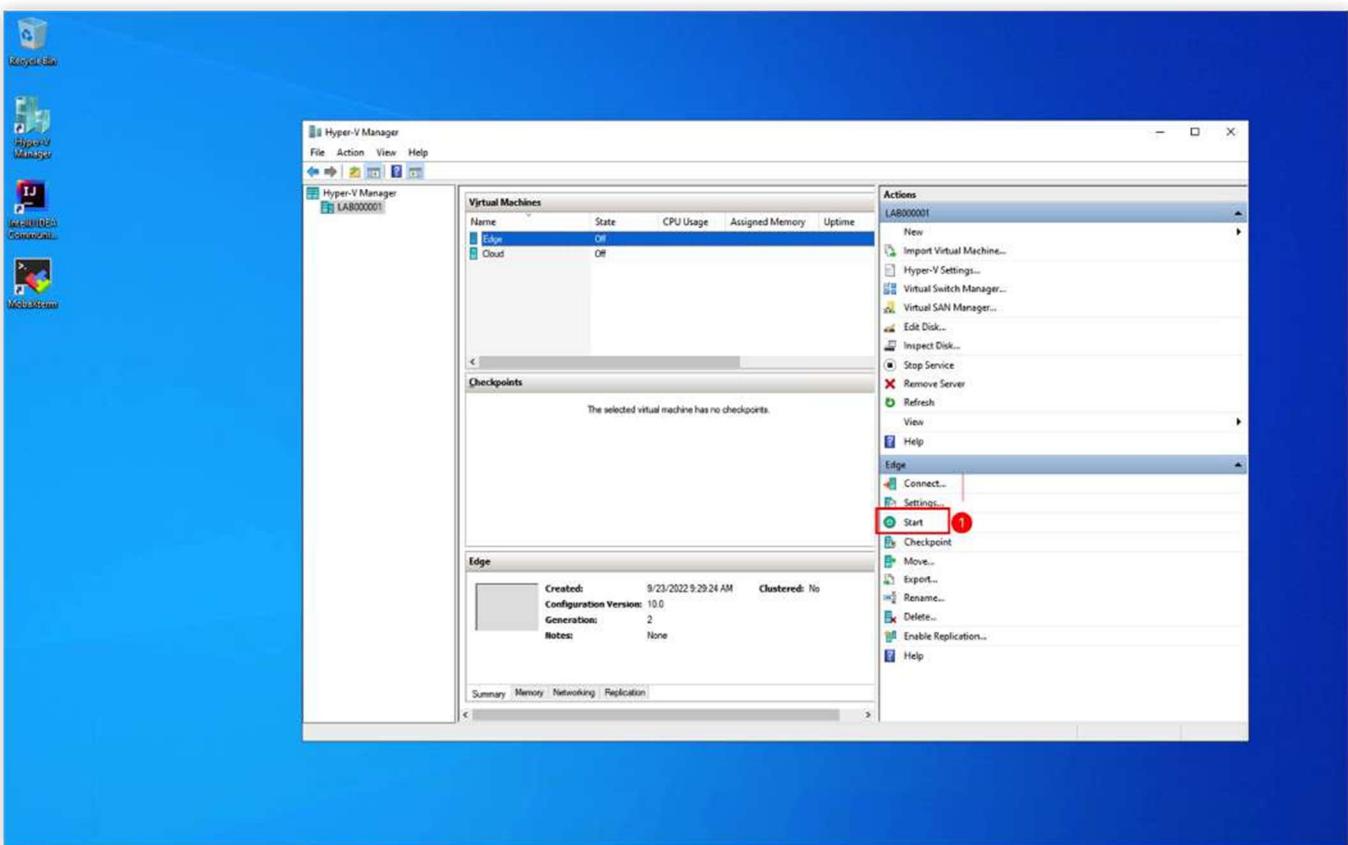
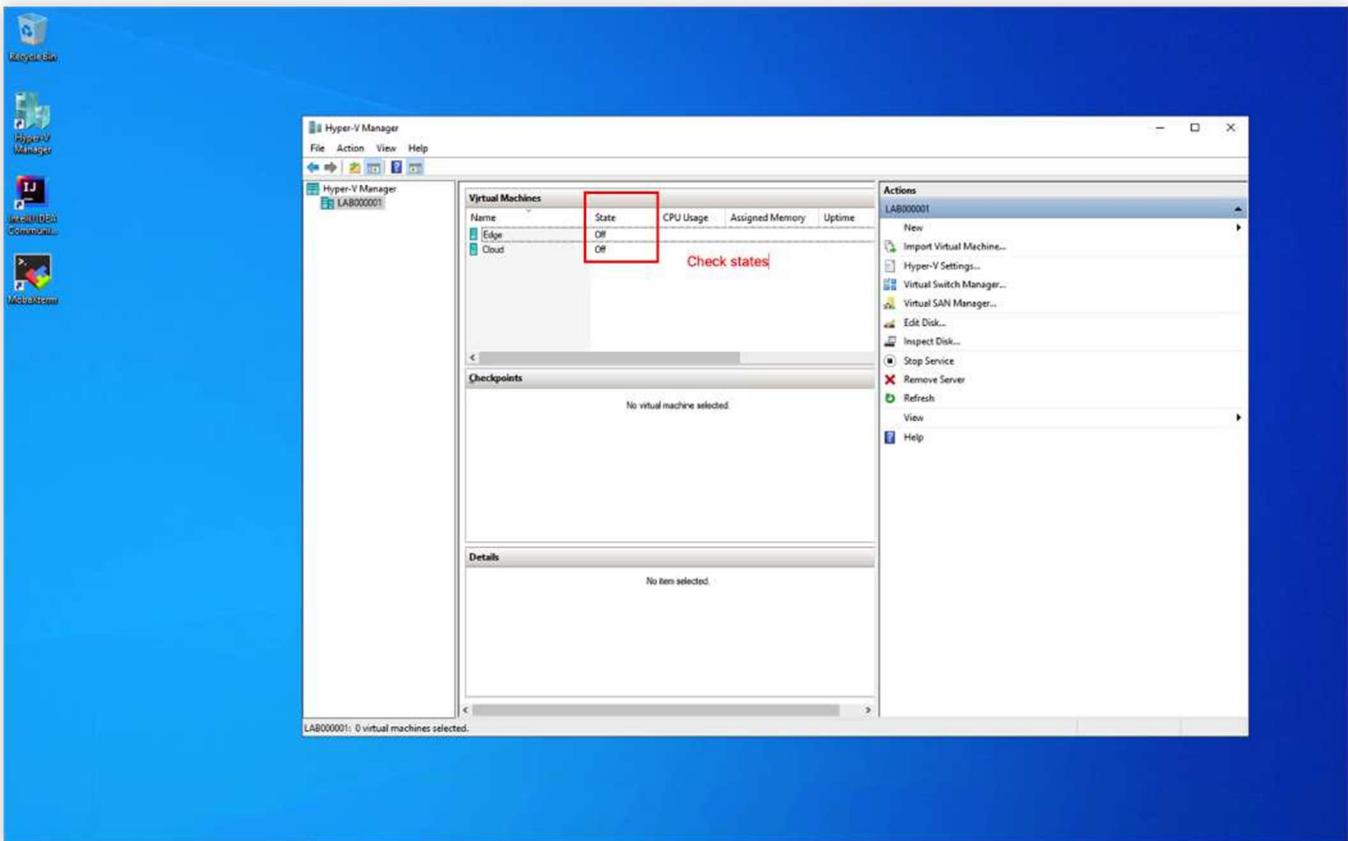
pika	1.3.0	python -m pip install pika --upgrade	2 & 3
prophet	1.1.1	python -m pip install prophet	3
matplotlib	3.6.0	pip install matplotlib	3
tensorflow	2.13.1	pip install tensorflow	4
numpy	1.24.3	pip install numpy	4

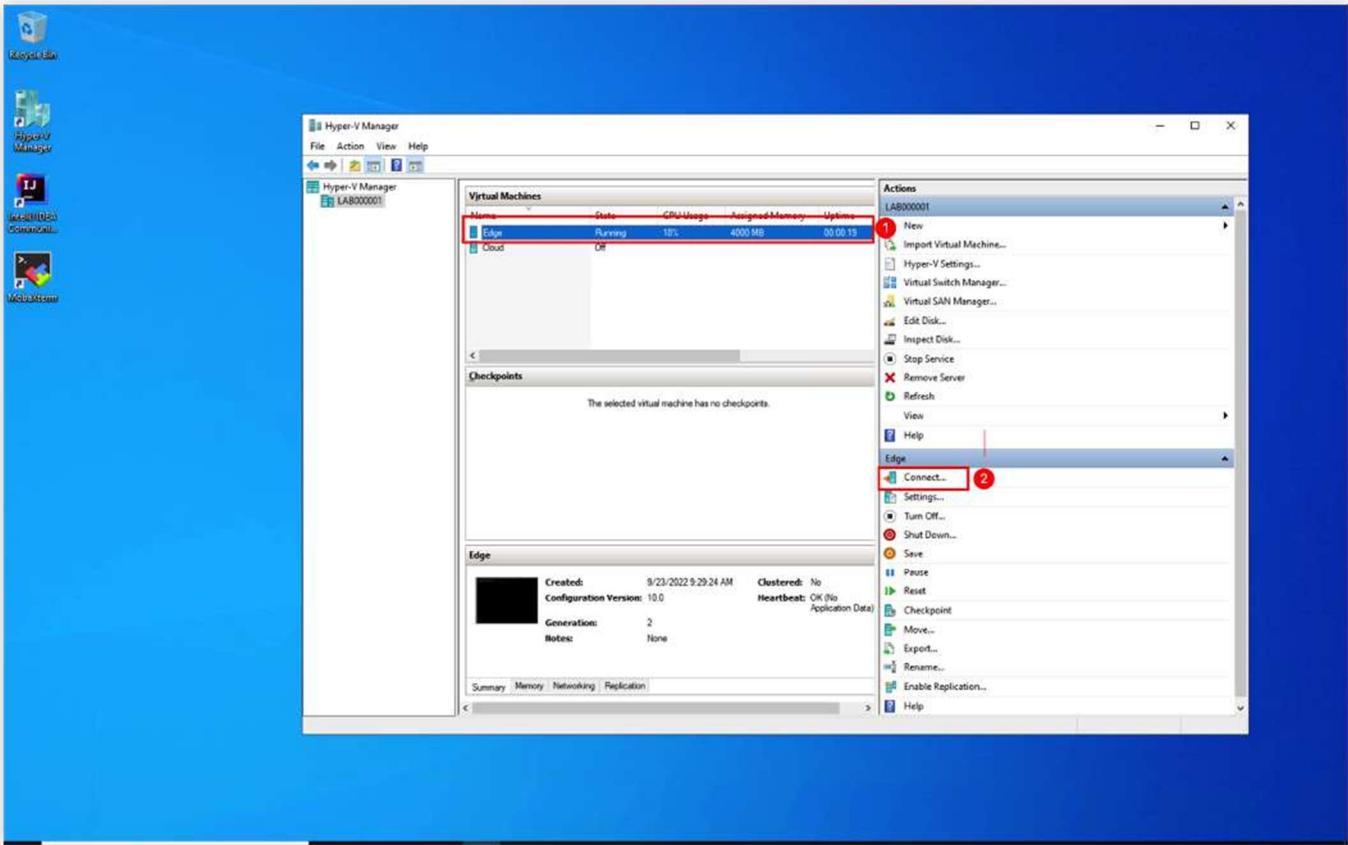
## Ubuntu VMs

### To start Ubuntu VMs

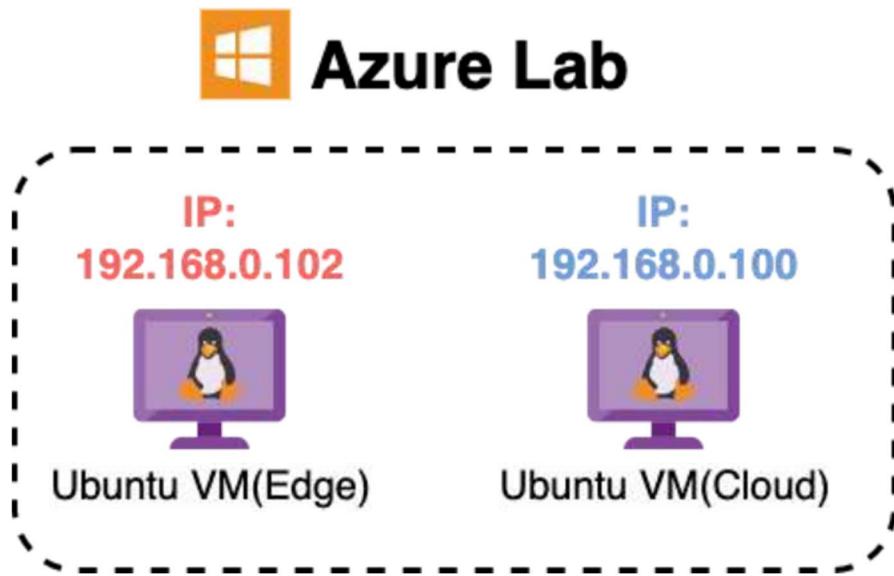
For the first time using, please clean saved state first (both Edge and Cloud), according to the following screenshot !



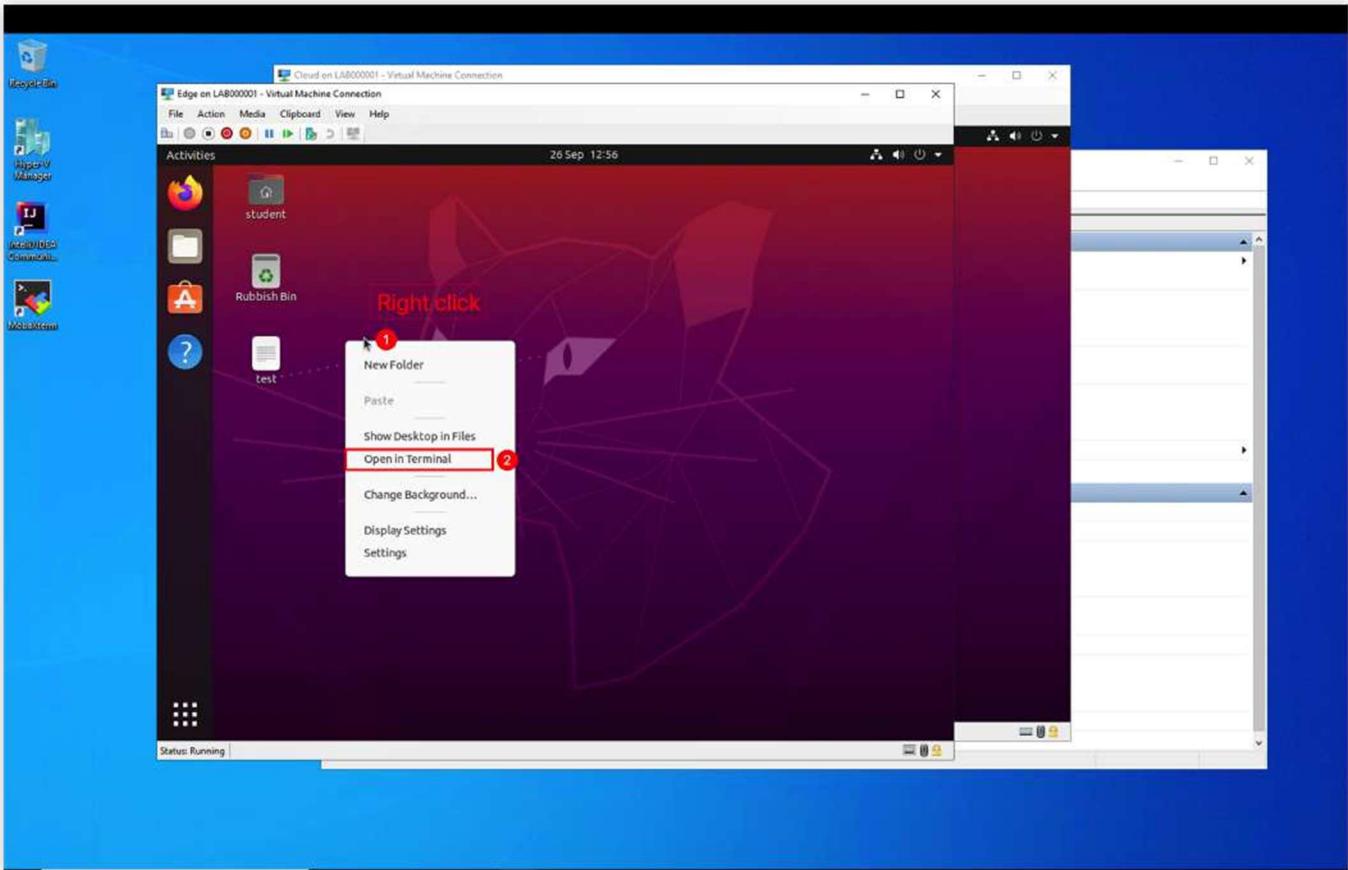




Ubuntu VM's IP



To get Ubuntu VM's IP address

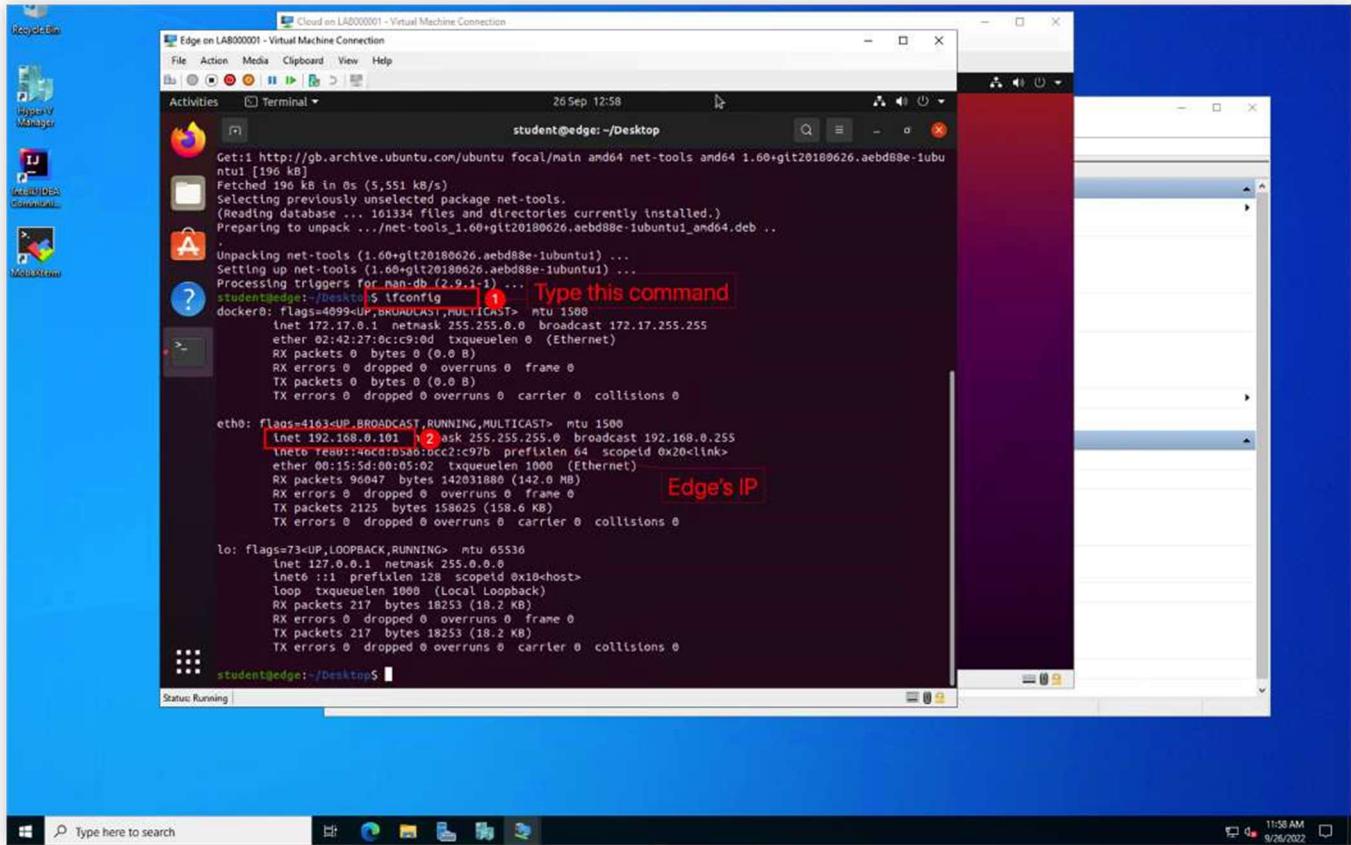


## Install net-tools to system

### ▼ Install net-tools

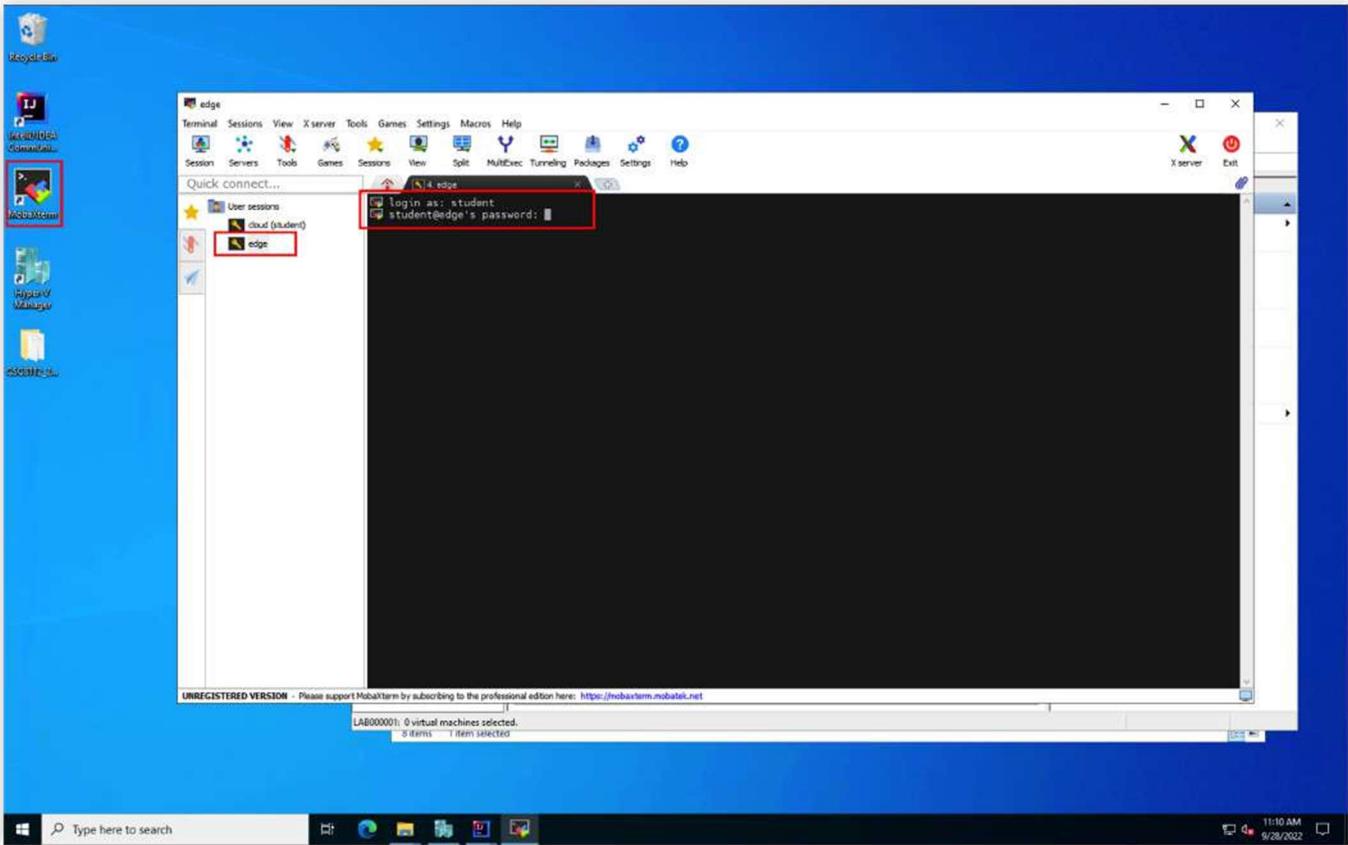
Bash |

```
sudo apt install net-tools
```



## SSH to Ubuntu VMs

The Azure VM provided "MobaXterm" to let you easily connect Ubuntu VMs by terminal way.



## Prepare necessary environment for Ubuntu VMs

### Python 3

Please manually install Python3.8 and pip in both Ubuntu VMs.

#### Install Python 3.8 and Pip

Shell |

```
1 sudo apt install python3  
2  
3 sudo apt install python3-pip
```

### Docker-compose tool

Please manually install docker-compose tool in both Ubuntu VMs.

#### Install docker-compose tool

Shell |

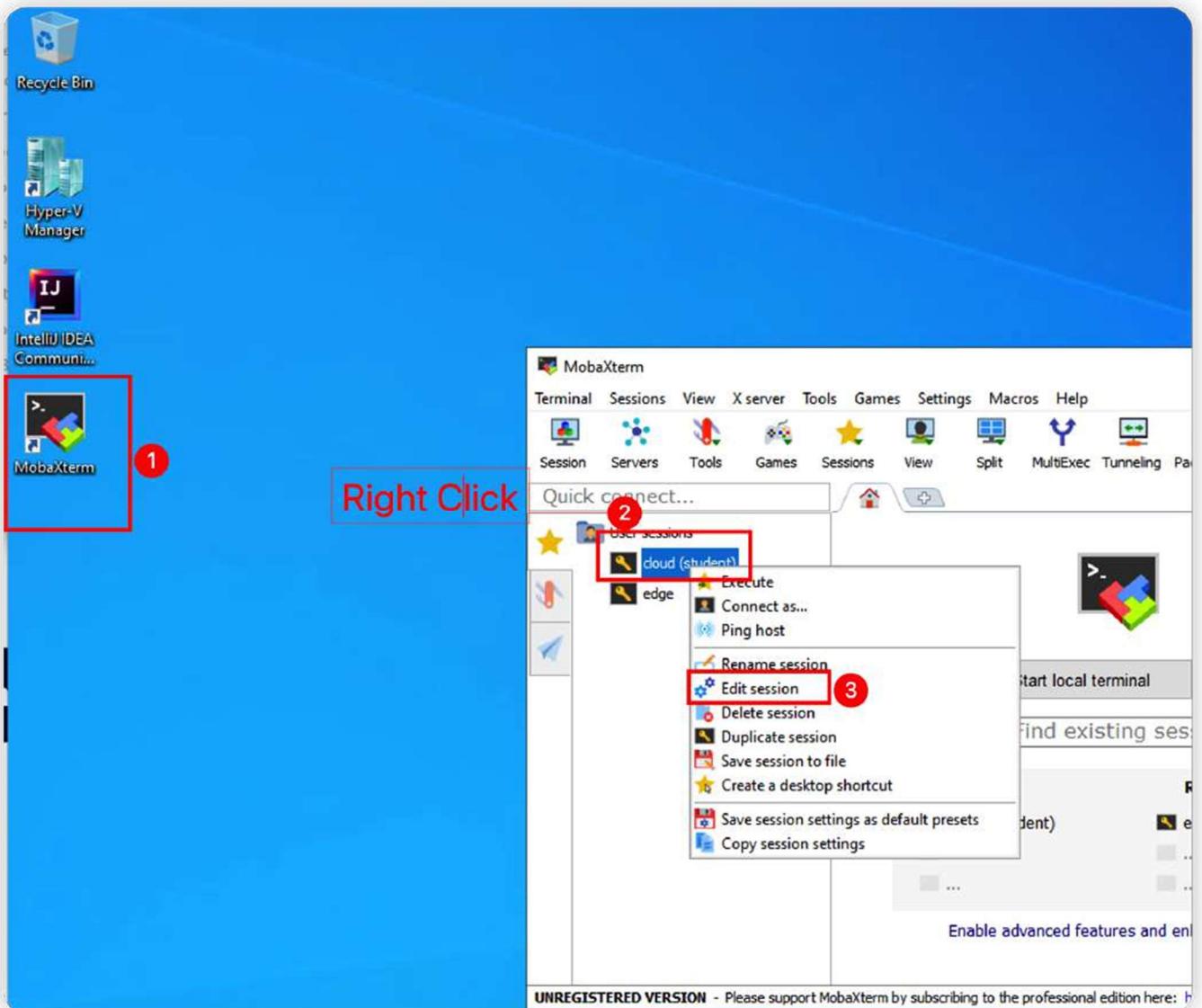
```
1 sudo apt install docker-compose
```

# Transfer files with VMs

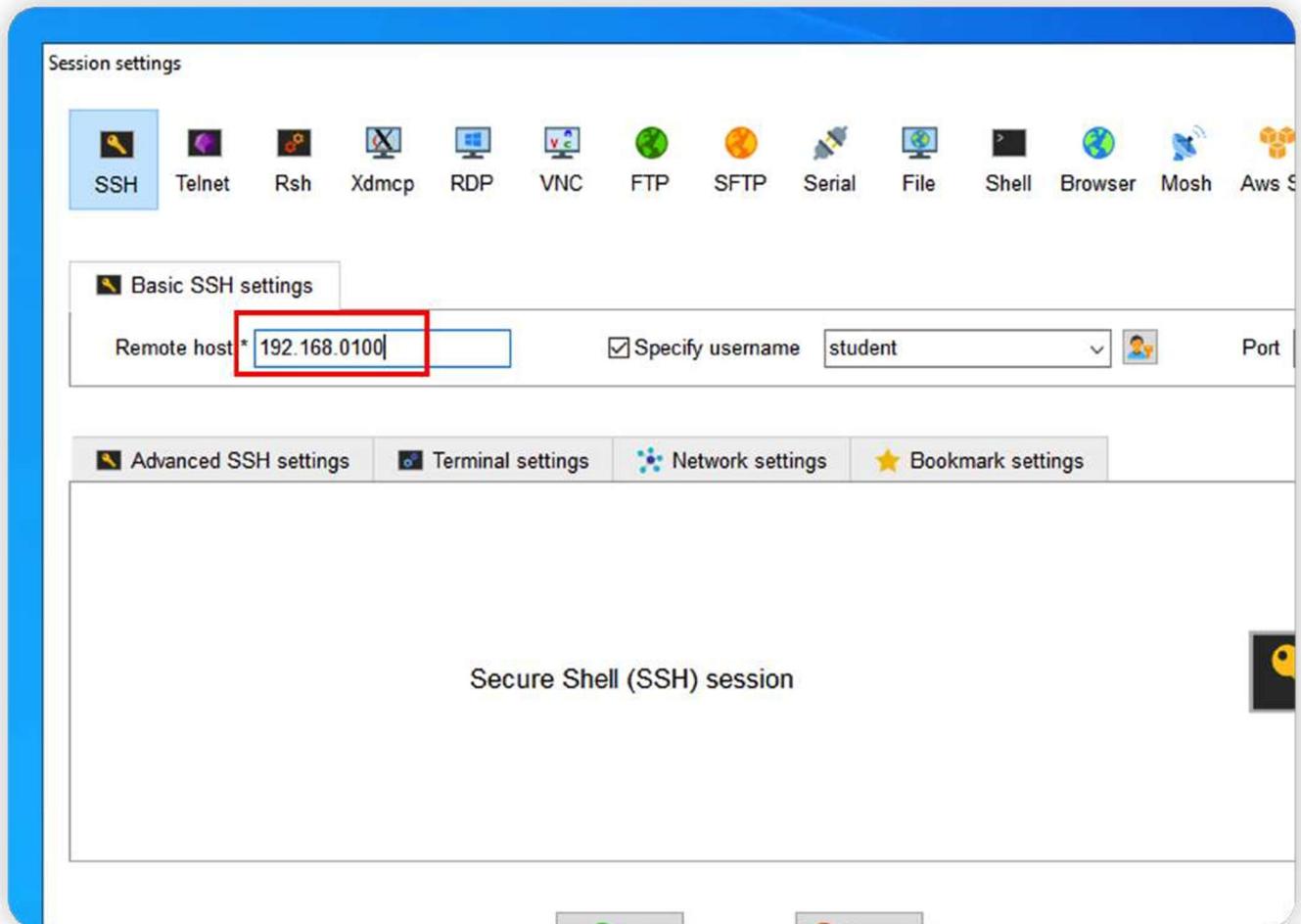
By MobaXterm (Recommended)

If you are using ModeXterm, pls do the following operation first !

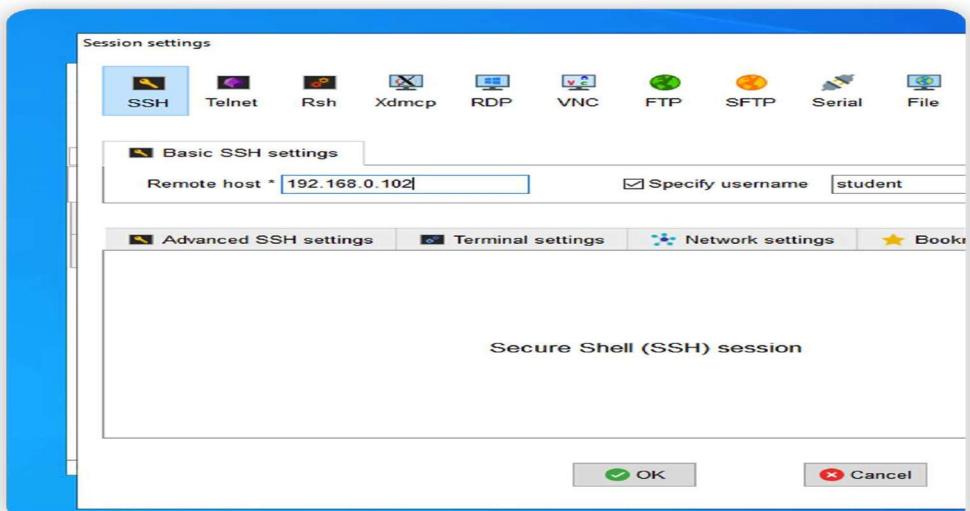
Otherwise the old configuration may refer to the same VM while you are trying to login.

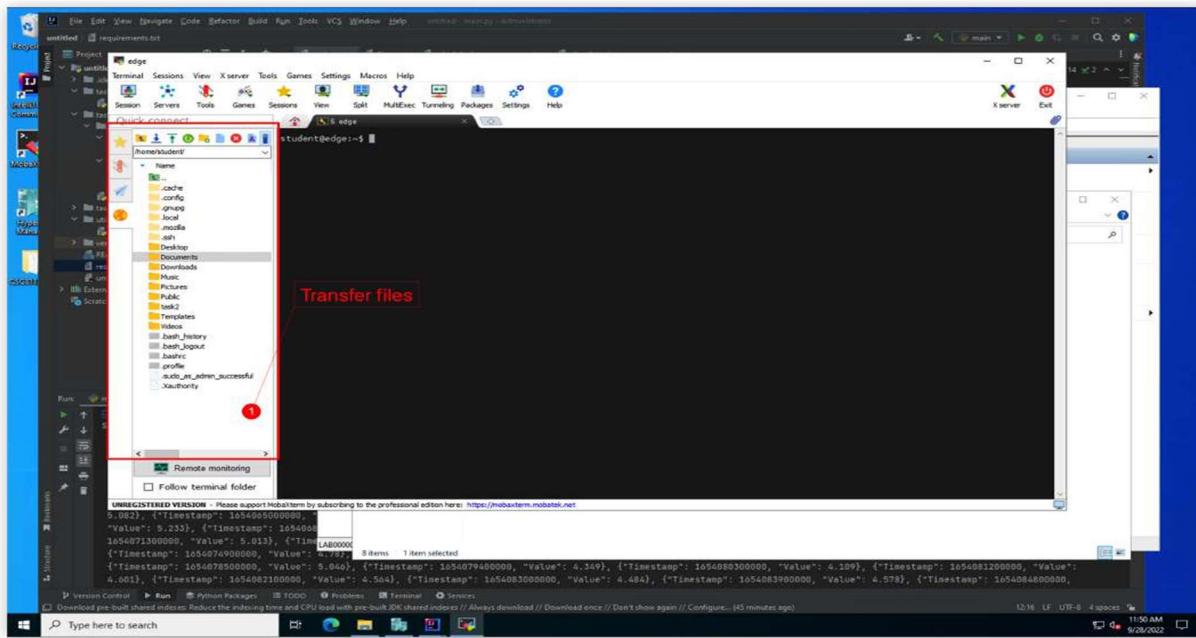


Setting for Cloud



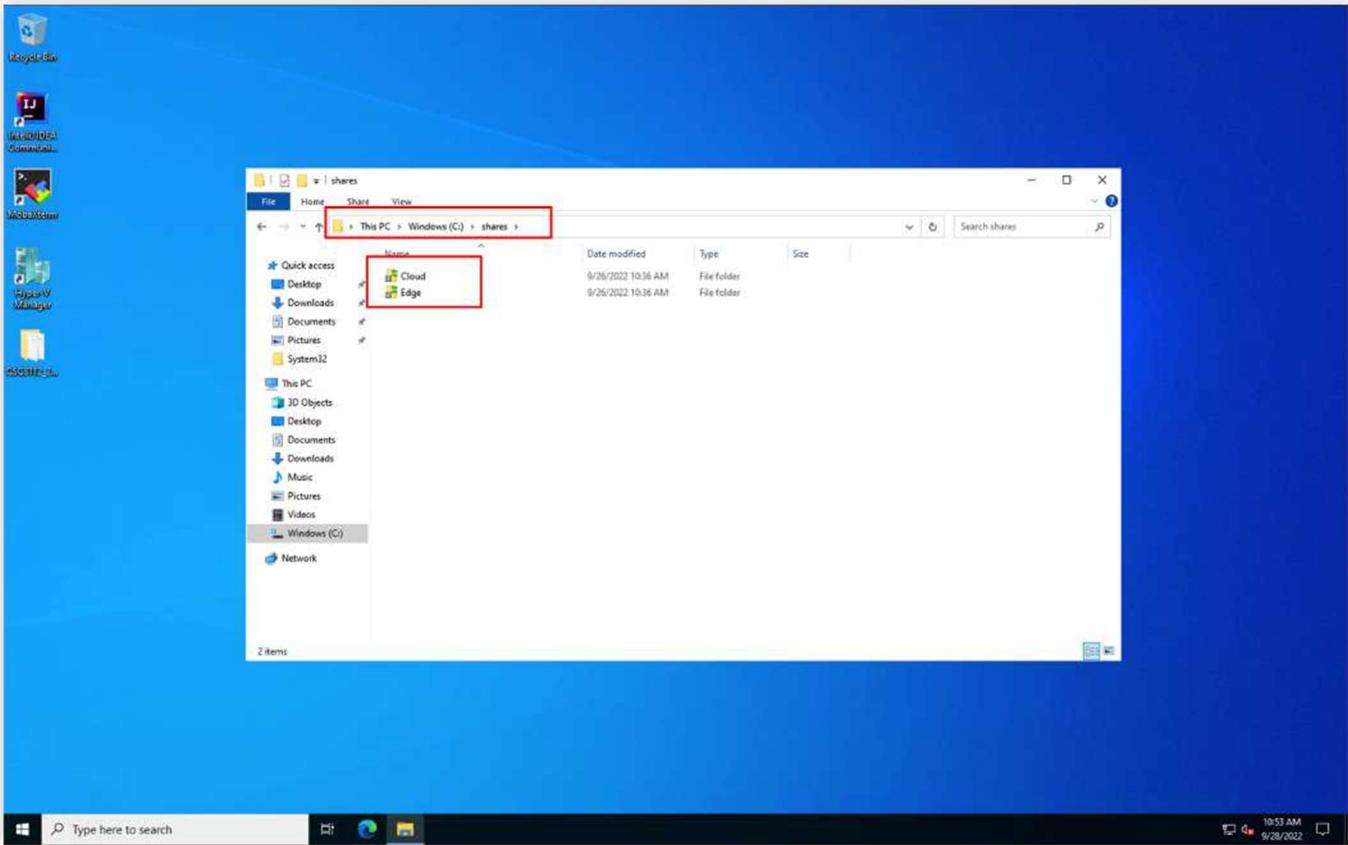
Setting for Edge





## By shared folder (Not Recommended)

We provide two shared directories for Cloud and Edge respectively.



## How to write a docker-compose configuration file

Hint: Refer the docker-compose.yml file in [Use Docker Compose](#)

Example:

```
▼ docker-compose file example           YAML |
```

```
1 version: "3"
2 services:
3   mongo:
4     image: mongo
5     deploy:
6       replicas: 1
7     ports:
8       - '27017:27017'
```

## How to build your code into docker image

## Prepare a Dockerfile

Example:

Dockerfile |

```
1 # Base on image_full_name (e.g., ubuntu:18.04) docker image
2 FROM python:3.8.12
3
4 # Switch to root
5 USER root
6
7 # Copy all sources files to workdir
8 ADD <your project directory> /usr/local/source
9
10 # Change working dir
11 WORKDIR /usr/local/source
12
13 # Prepare project required running system environments
14 # requirements.txt is a document that pre-define any
15 # python dependencies with versions required of your code
16 RUN pip3 install -r requirements.txt
17
18 # Start task
19 CMD python3 <your main .py file>
```

## Prepare a docker-compose configuration file

YAML |

```
version: "3"
services:
  data_injector:
    image: data_injector:latest
```

## Execution flow

1. Run your Dockerfile file first with command:

Build code | Shell |

```
sudo docker build -t <name:tag> <source directory (relative)>
```

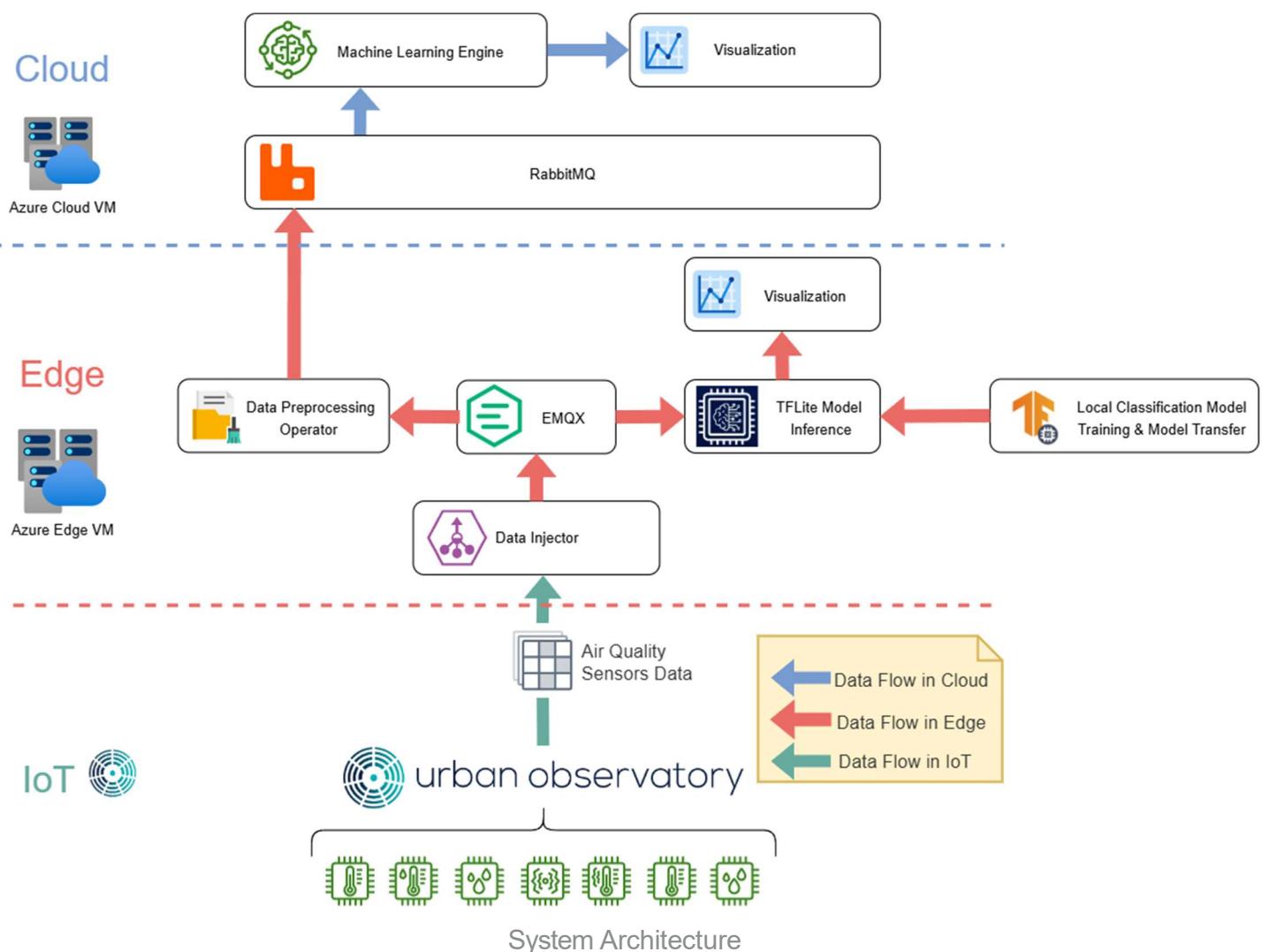
2. Run your docker-compose file with command:

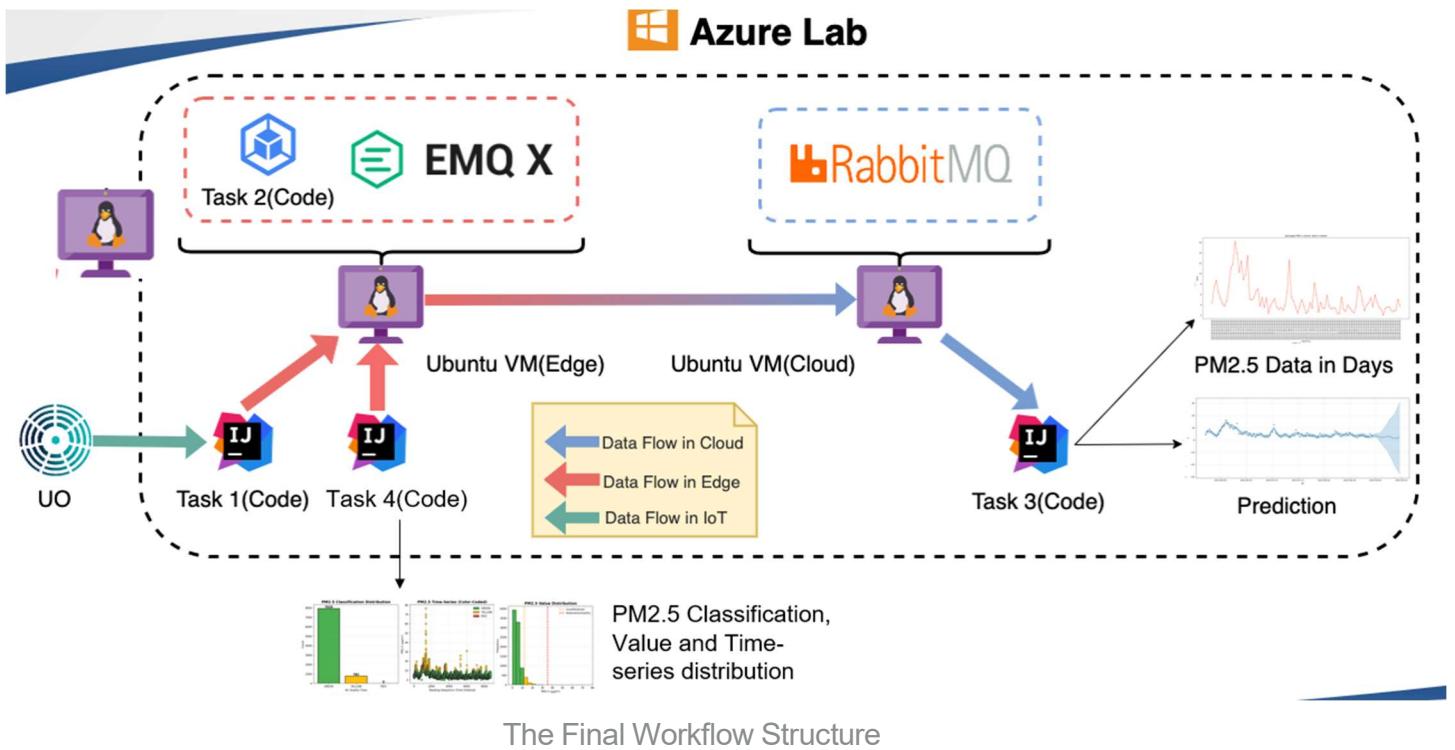
- Please make sure the docker-compose tool already installed, if not, please refer to chapter [Docker-compose tool](#)

```
▼ Start a image | Shell |  
1 sudo docker-compose up
```

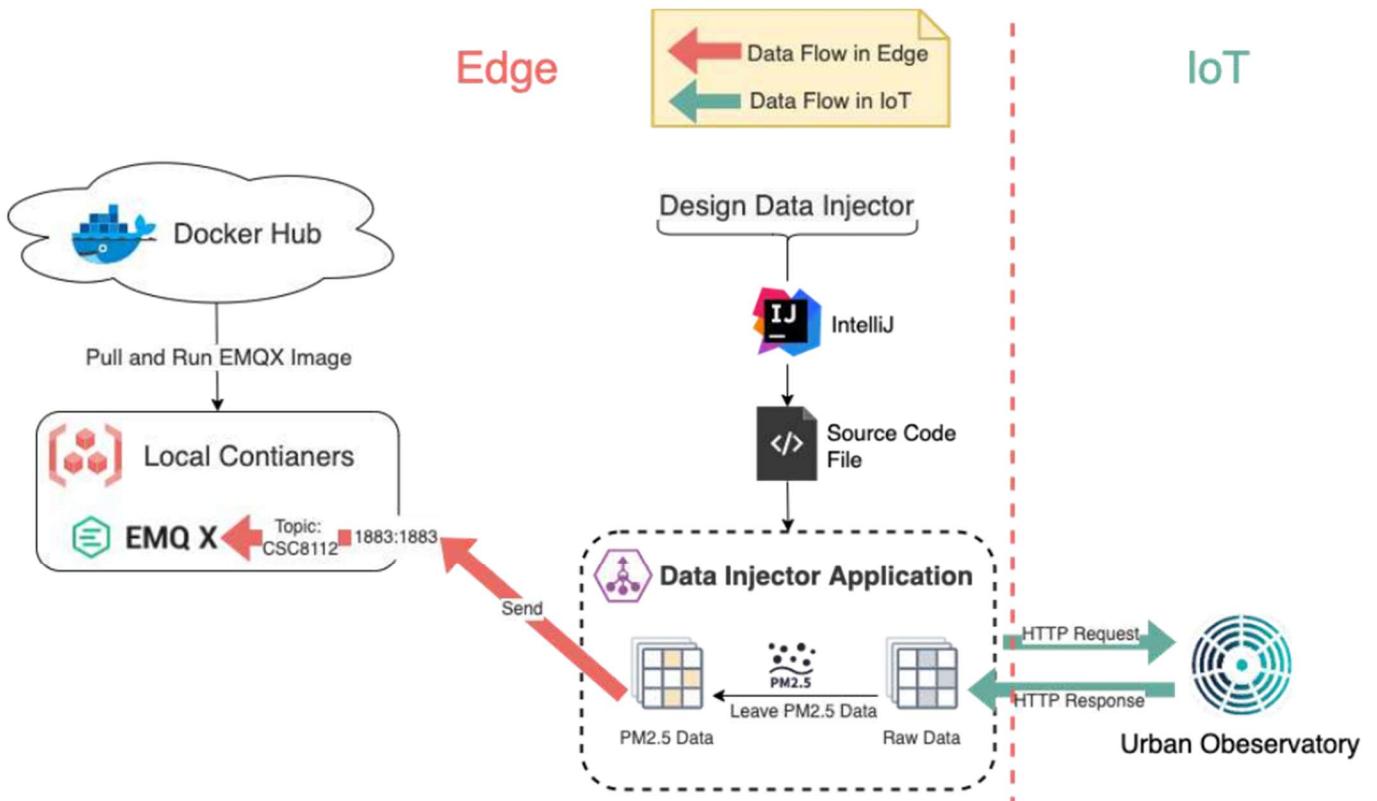
## Coursework supplements

### Overview structures

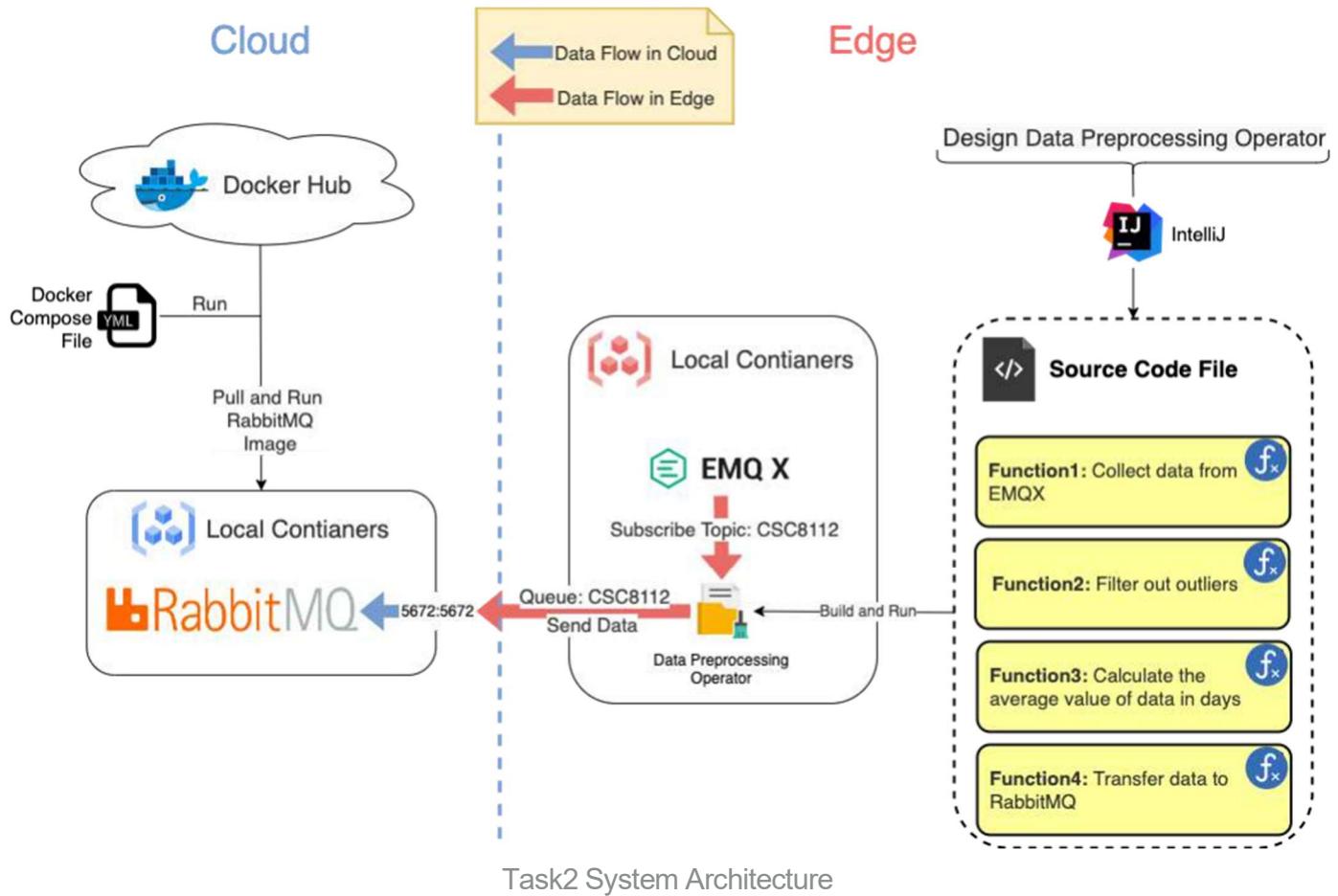




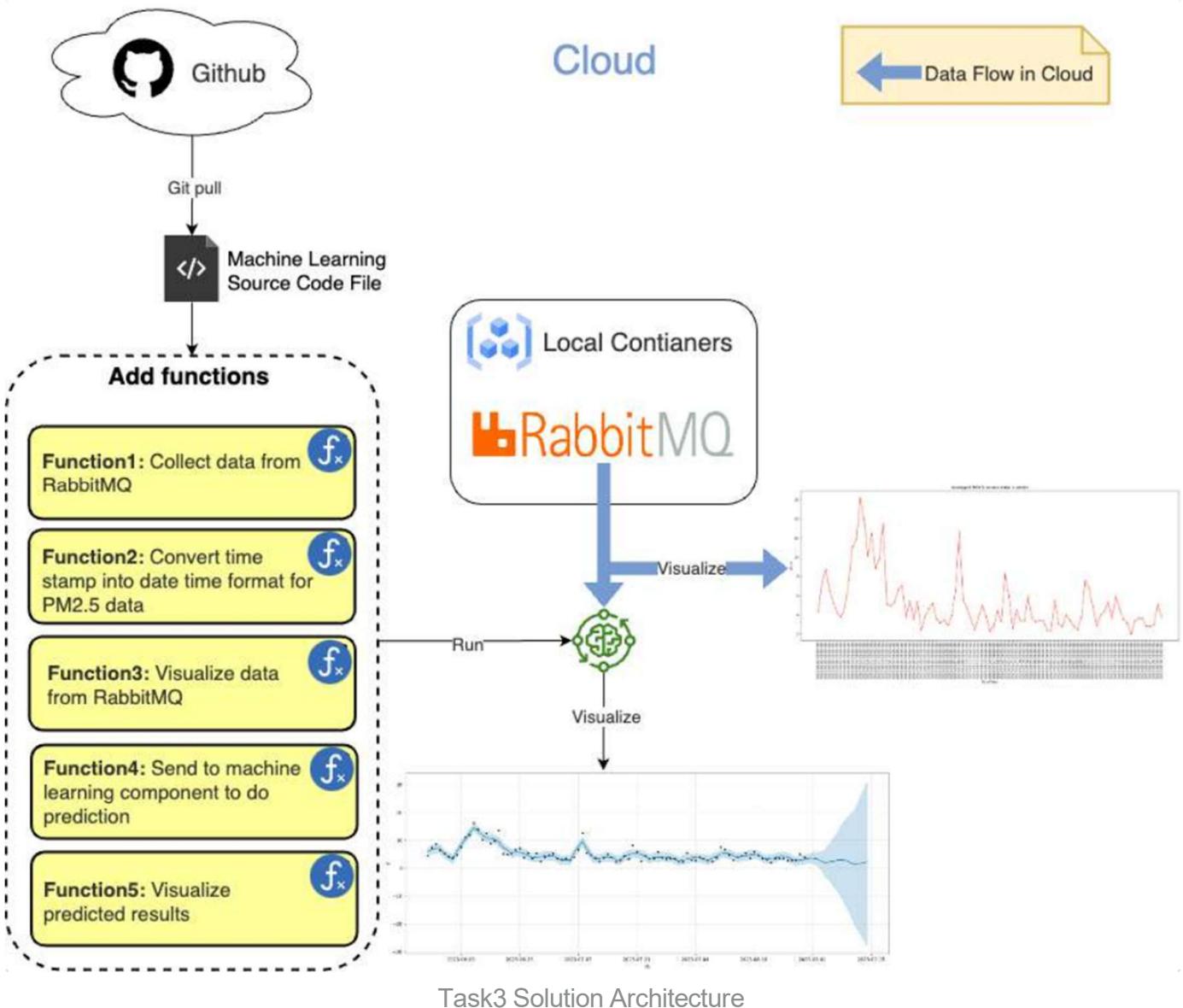
## Task 1



## Task 2



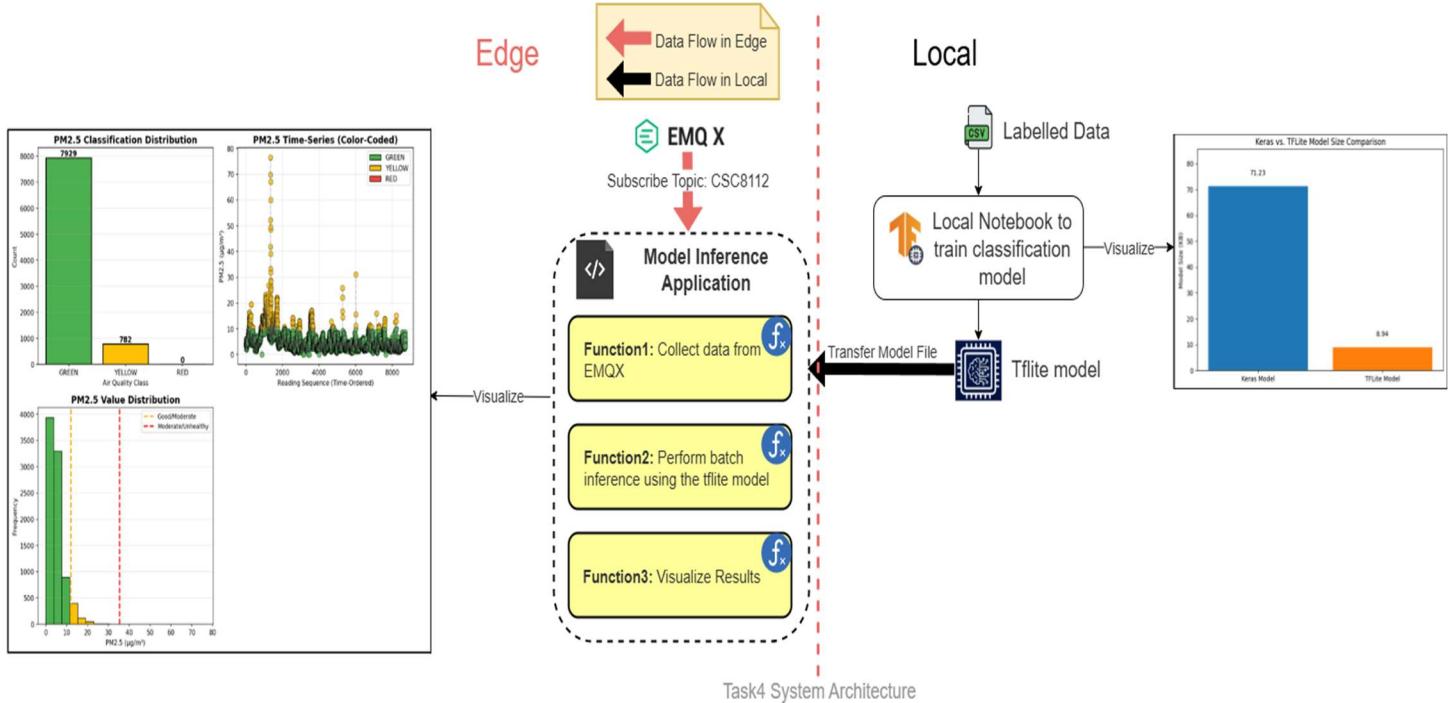
## Task 3



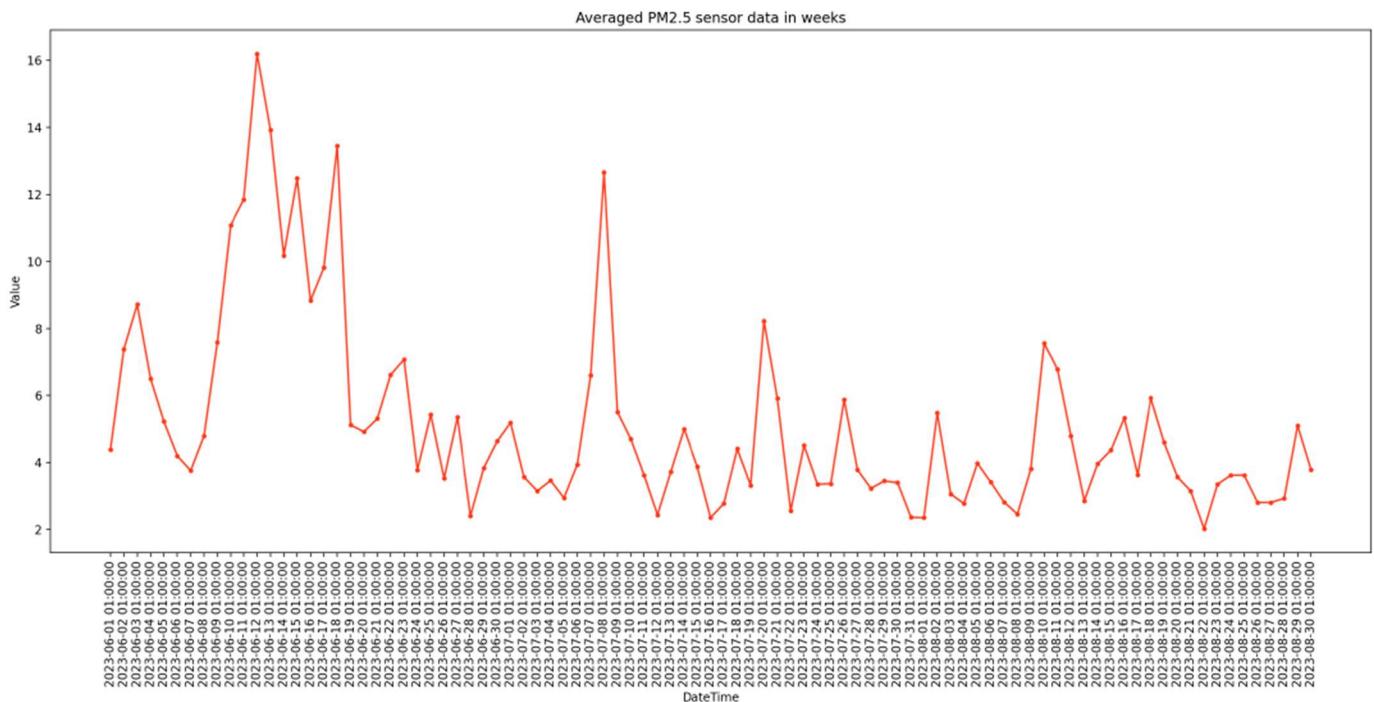
Task3 Solution Architecture

Download Machine Learning engine code: [GitHub - ncl-iot-team/CSC8112\\_MLEngine](#)

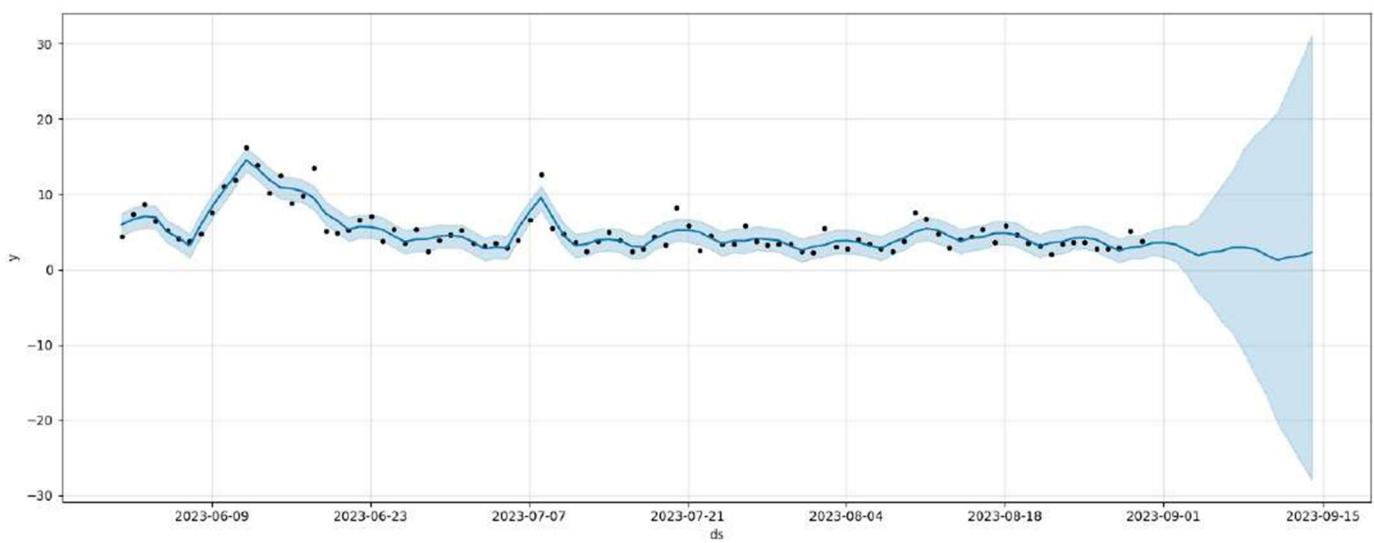
## Task 4



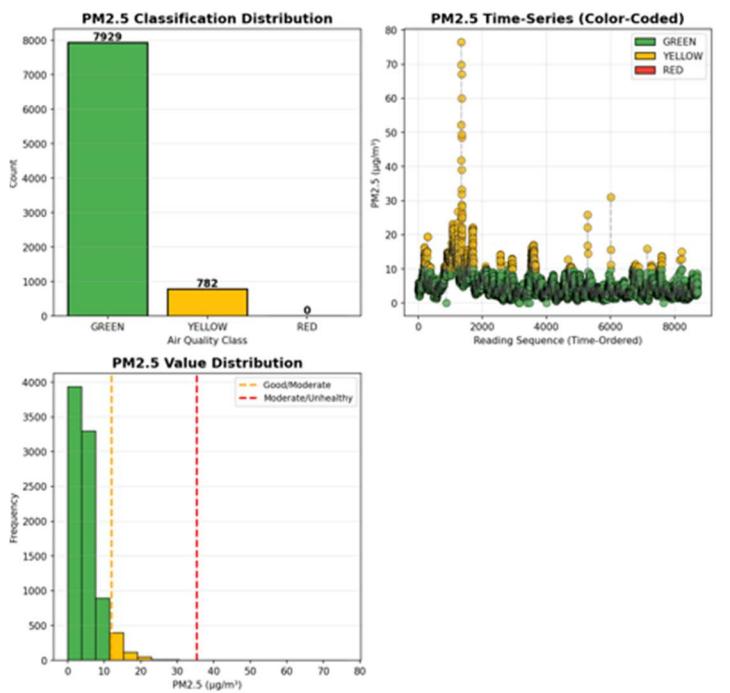
The final results in visualization look like this:



Averaged PM2.5 data in days



PM2.5 Prediction for next 15 days



### Note:

This is a sample answer. A candidate's graph trend might look different according to their model quality for Task 4.

Task 4 Graphs

## Implementation Examples

How to send HTTP request to get sensor data

## ▼ HTTP Get

Python |

```
1 import requests
2
3 if __name__ == '__main__':
4     # Target URL
5     url = "http://uoweb3.ncl.ac.uk/api/v1.1/sensors/PER_NE_CAJT_NCA189_SJB
6         _SJB2" \
7             "/data/json/?starttime=20220901&endtime=20221001"
8
9     # Request data from Urban Observatory Platform
10    resp = requests.get(url)
11
12    # Convert response(Json) to dictionary format
13    raw_data_dict = resp.json()
14
15    print(raw_data_dict)
```

## How to define MQTT subscriber in Python

## MQTT subscriber

Python |

```
1 import json
2
3 from paho.mqtt import client as mqtt_client
4
5 if __name__ == '__main__':
6     mqtt_ip = "localhost"
7     mqtt_port = 1883
8     topic = "CSC8112"
9
10    # Create a mqtt client object
11    client = mqtt_client.Client()
12
13
14    # Callback function for MQTT connection
15    def on_connect(client, userdata, flags, rc):
16        if rc == 0:
17            print("Connected to MQTT OK!")
18        else:
19            print("Failed to connect, return code %d\n", rc)
20
21    # Connect to MQTT service
22    client.on_connect = on_connect
23    client.connect(mqtt_ip, mqtt_port)
24
25    # Callback function will be triggered
26    def on_message(client, userdata, msg):
27        print(f"Get message from publisher {json.loads(msg.payload)}")
28
29    # Subscribe MQTT topic
30    client.subscribe(topic)
31    client.on_message = on_message
32
33    # Start a thread to monitor message from publisher
34    client.loop_forever()
```

## How to define MQTT publisher in Python

## MQTT publisher

Python |

```
1 import json
2 from paho.mqtt import client as mqtt_client
3
4 if __name__ == '__main__':
5     mqtt_ip = "localhost"
6     mqtt_port = 1883
7     topic = "CSC8112"
8     msg = "Hello!"
9
10    # Create a mqtt client object
11    client = mqtt_client.Client()
12
13    # Callback function for MQTT connection
14    def on_connect(client, userdata, flags, rc):
15        if rc == 0:
16            print("Connected to MQTT OK!")
17        else:
18            print("Failed to connect, return code %d\n", rc)
19
20    # Connect to MQTT service
21    client.on_connect = on_connect
22    client.connect(mqtt_ip, mqtt_port)
23
24    # Publish message to MQTT
25    # Note: MQTT payload must be a string, bytearray, int, float or None
26    msg = json.dumps(msg)
27    client.publish(topic, msg)
```

## How to define RabbitMQ consumer in Python

## ▼ RabbitMQ consumer

Python |

```
1 import json
2 import pika
3
4 if __name__ == '__main__':
5
6     rabbitmq_ip = "localhost"
7     rabbitmq_port = 5672
8     # Queue name
9     rabbitmq_queue = "CSC8112"
10
11 def callback(ch, method, properties, body):
12     print(f"Got message from producer msg: {json.loads(body)}")
13
14     # Connect to RabbitMQ service with timeout 1min
15     connection = pika.BlockingConnection(
16         pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port, so-
cket_timeout=60))
17     channel = connection.channel()
18     # Declare a queue
19     channel.queue_declare(queue=rabbitmq_queue)
20
21     channel.basic_consume(queue=rabbitmq_queue,
22                           auto_ack=True,
23                           on_message_callback=callback)
24
25     channel.start_consuming()
```

## How to define RabbitMQ producer in Python

## RabbitMQ producer

Python |

```
1 import pika
2 import json
3
4 if __name__ == '__main__':
5     rabbitmq_ip = "localhost"
6     rabbitmq_port = 5672
7     # Queue name
8     rabbitmq_queue = "CSC8112"
9     msg = "Hello!"
10    # Connect to RabbitMQ service
11    connection = pika.BlockingConnection(pika.ConnectionParameters(host=rabbitmq_ip, port=rabbitmq_port))
12    channel = connection.channel()
13
14    # Declare a queue
15    channel.queue_declare(queue=rabbitmq_queue)
16
17    # Produce message
18    channel.basic_publish(exchange='',
19                           routing_key=rabbitmq_queue,
20                           body=json.dumps(msg))
21
22 connection.close()
```

## How to use Matplotlib to visualize a line chart

## Matplotlib line chart

Python |

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3
4 if __name__ == '__main__':
5     # Prepare data
6     data = {
7         'Timestamp': ['01/09', '02/09', '03/09', '04/09', '05/09'],
8         'Value': [1, 2, 1, 10, 5]
9     }
10
11     data_df = pd.DataFrame(data)
12
13     # Initialize a canvas
14     plt.figure(figsize=(8, 4), dpi=200)
15     # Plot data into canvas
16     plt.plot(data_df["Timestamp"], data_df["Value"], color="#FF3B1D", marker='.', linestyle="--")
17     plt.title("Example data for demonstration")
18     plt.xlabel("DateTime")
19     plt.ylabel("Value")
20
21     # Save as file
22     plt.savefig("figure1.png")
23     # Directly display
24     plt.show()
```

## How to use Machine Learning Engine

```
1  from ml_engine import MLPredictor
2
3  if __name__ == '__main__':
4      # Prepare data
5      data = {
6          'Timestamp': ['2020-09-01', '2020-09-02', '2020-09-03',
7                         '2020-09-04', '2020-09-05'],
8          'Value': [1, 2, 1, 10, 5]
9      }
10     data_df = pd.DataFrame(data)
11
12     # Create ML engine predictor object
13     predictor = MLPredictor(data_df)
14
15     predictor.train()
16
17     # Do prediction
18     forecast = predictor.predict()
19
20     # Get canvas
21     fig = predictor.plot_result(forecast)
22     fig.savefig("prediction.png")
23     fig.show()
```

## How to convert to tflite model for task 4 (Sample code)

```
# Convert to TFLite with dynamic range quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model)

# Apply post-training quantization
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.float32
converter.inference_output_type = tf.float32

tflite_model = converter.convert()

# Save the quantized TFLite model
tflite_model_path = 'air_quality_model_quantized.tflite'
with open(tflite_model_path, 'wb') as f:
    f.write(tflite_model)

print(f"TFLite model saved as '{tflite_model_path}'")
```

---

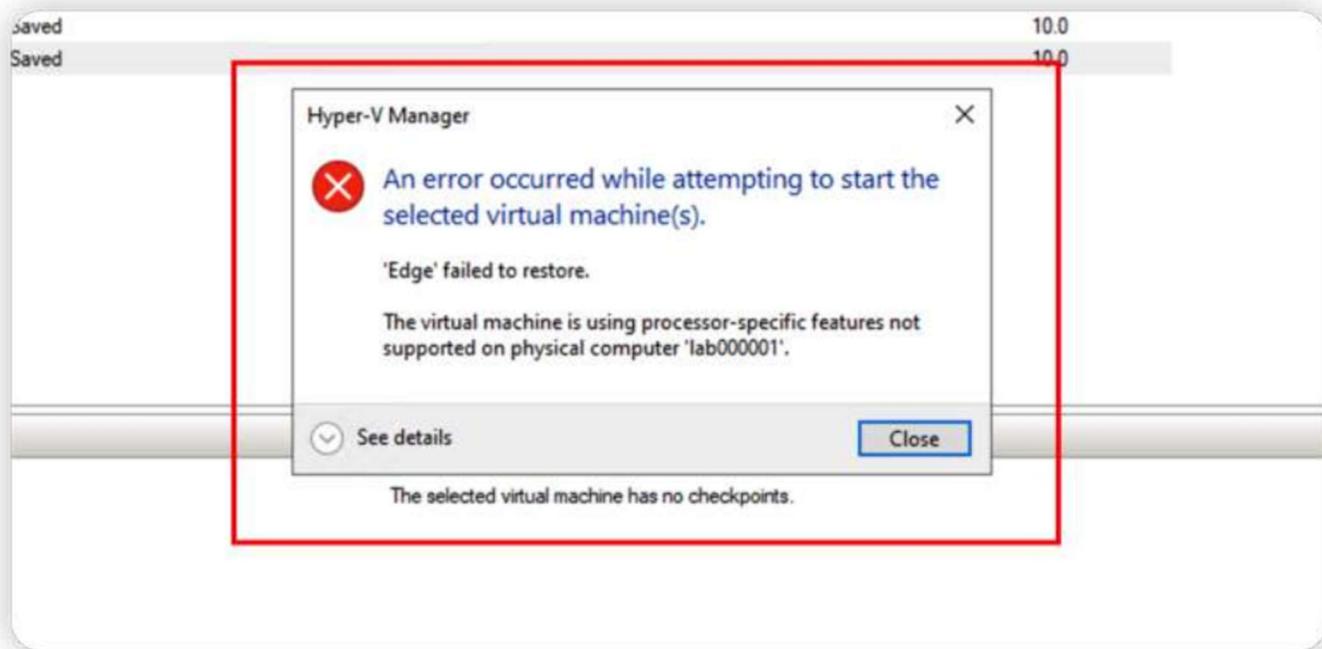
## Issues solving (Please carefully follow every step and tip in this section)

### Docker logs cannot print output

1. Please add (flush=True in your print code)

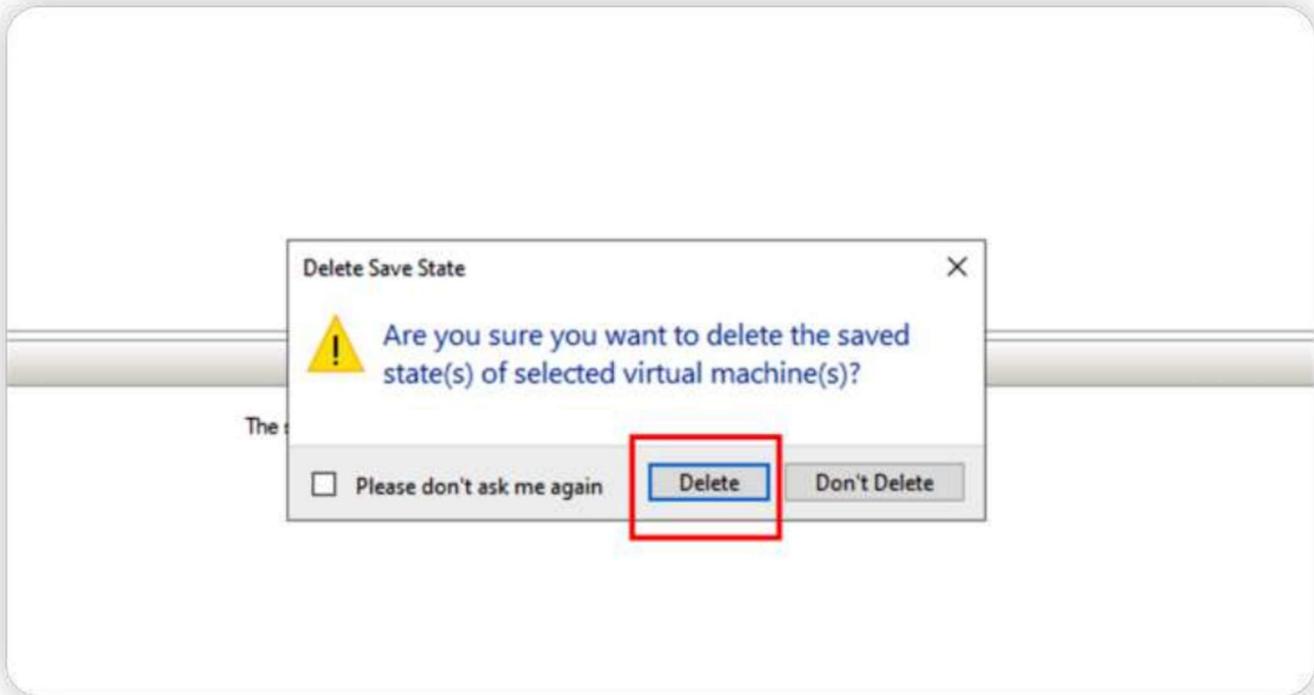
```
▼ print Python |  
1 print("hello", flush=True)
```

## An error occurred while attempting to start VM



Please delete the save status, and then try to start VM again. (Please make sure you have backup all important files before do this)

A screenshot of the Hyper-V Manager interface. The left pane shows a list of virtual machines: Edge (Saved) and Cloud (Saved). The right pane shows actions for the selected virtual machine, "LAB000001". A red box highlights the "Delete Saved State..." option in the Actions list. Other options include New, Import Virtual Machine..., Hyper-V Settings..., Virtual Switch Manager..., Virtual SAN Manager..., Edit Disk..., Inspect Disk..., Stop Service, Refresh, View, Help, and Cloud options like Connect..., Settings..., Start, Checkpoint, Move..., Export..., Rename..., and Delete... (with a red box around it).



## Cannot get sensor data from Urban Observatory

Please use this URL to replace the original one in the coursework specification.

[https://gist.githubusercontent.com/ringosham/fbd66654dc53c40bd4581d2828acc94e/raw/d5a0fcfd27ff7ea31e2aec3765eb2c5d64adb79/uo\\_data.min.json](https://gist.githubusercontent.com/ringosham/fbd66654dc53c40bd4581d2828acc94e/raw/d5a0fcfd27ff7ea31e2aec3765eb2c5d64adb79/uo_data.min.json)

## Appendix

### About target sensors information

Useful links:

1. Urban Observatory : [Urban Observatory](#)
2. City Data API v1.1 doc : [https://newcastle.urbanobservatory.ac.uk/api\\_docs/doc/sensors-csv/](https://newcastle.urbanobservatory.ac.uk/api_docs/doc/sensors-csv/)
3. Sensors search: [https://newcastle.urbanobservatory.ac.uk/#sensor\\_info](https://newcastle.urbanobservatory.ac.uk/#sensor_info)

Target sensor-s information:

## Sensor Info

JSON

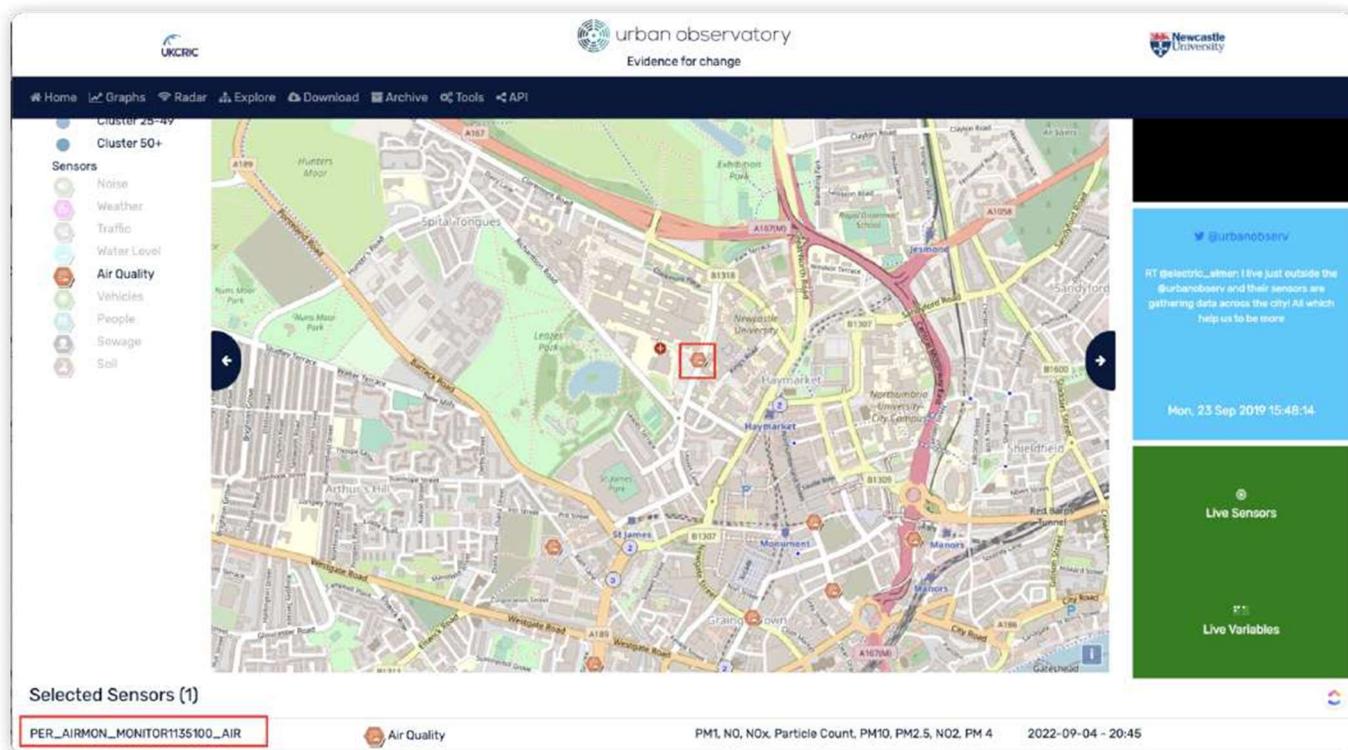
```

1 {"Third Party": false, "Location (WKT)": "POINT(-1.617676 54.979118)", "Sensor Centroid Longitude": -1.617676,
2 "Sensor Centroid Latitude": 54.979118, "Broker Name": "aq_mesh_api", "Sensor Height Above Ground": 2.0,
3 "Raw ID": "79525", "Ground Height Above Sea Level": 57.6699981689, "Sensor Name": "PER_AIRMON_MONITOR1135100"}

```

## About Raw Data:

- Date range: From 01/06/2023 to 31/08/2023.
- Data format: JSON
- Size: around 17MB(total)
  - o PM2.5 data (total data points: 8711 with timestamp in millisecond, collecting rate: every 15min (900000ms))



Geo-location of target sensor

