CSC8112 IoT Module Coursework

Ringo Sham, Amrit Kumar

Outline

Objectives

To utilise IoT data to finish a specific task in a real application, which involves:

- Data collection
- Data preprocessing
- Time-series sensor data prediction
- PM2.5 sensor data classification
- Visualisation

To understand the pipeline implementation with data flow and deploying Docker-based application hosting environment

Final report needs to be submitted to NESS by 15:00 pm on November 14, 2025
Live demonstration session will be organised on either November 10, 2025 (30min per student)

Outline

Microservices

What is Microservices

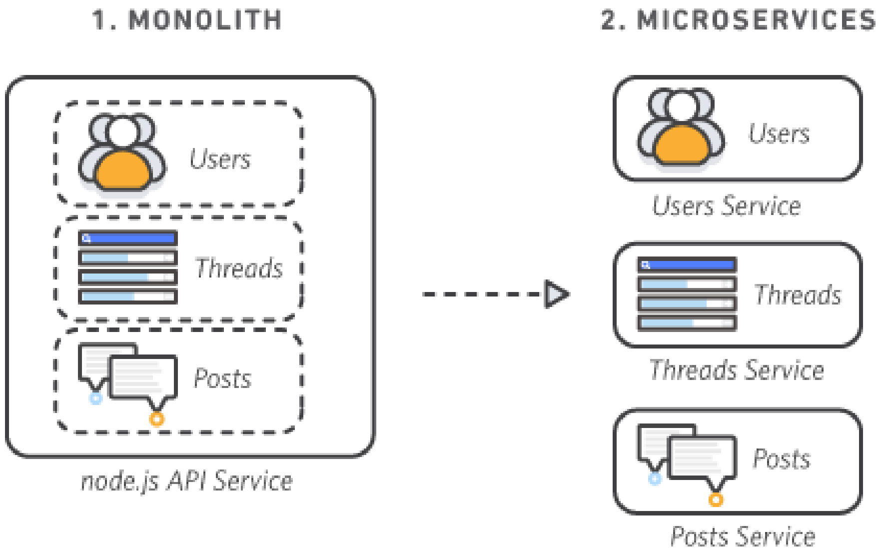
Microservices –
also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications.

https://aws.amazon.com/microservices/

What is Microservices?



Monolithic vs. Microservices Architecture

With monolithic architectures, all processes are tightly coupled and run as a single service.

With a microservices architecture,
an application is built as independent components that run each application process as a service

https://aws.amazon.com/microservices/

Docker

Docker

What is Docker:

- An open platform to help developers build, share, and run modern applications
- Separate applications from the infrastructure to deliver software quickly

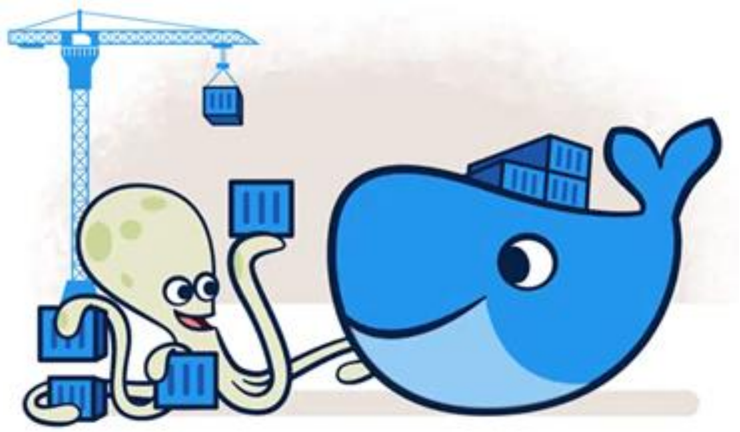
Docker makes development efficient and predictable

Download: https://www.docker.com/

Docker

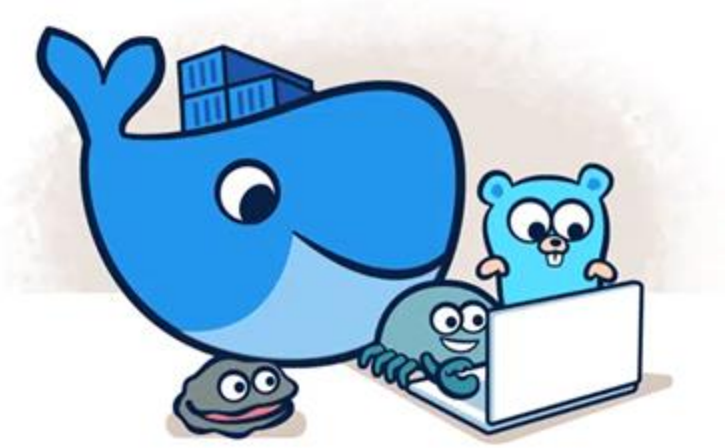
Build:

- Get a head start on your coding by leveraging Docker images to efficiently develop your own unique applications on Windows and Mac. Create your multi-container application using Docker Compose.
- Integrate with your favorite tools throughout your development pipeline – Docker works with all development tools you use including VS Code, CircleCI and GitHub.
- Package applications as portable container images to run in any environment consistently from on-premises Kubernetes to AWS ECS, Azure ACI, Google GKE and more.



https://www.docker.com/

Docker



Share:

- Leverage Docker Trusted Content, including Docker Official Images and images from Docker Verified Publishers from the Docker Hub repository.
- Innovate by collaborating with team members and other developers and by easily publishing images to Docker Hub.
- Personalize developer access to images with roles based access control and get insights into activity history with Docker Hub Audit Logs.

https://www.docker.com/

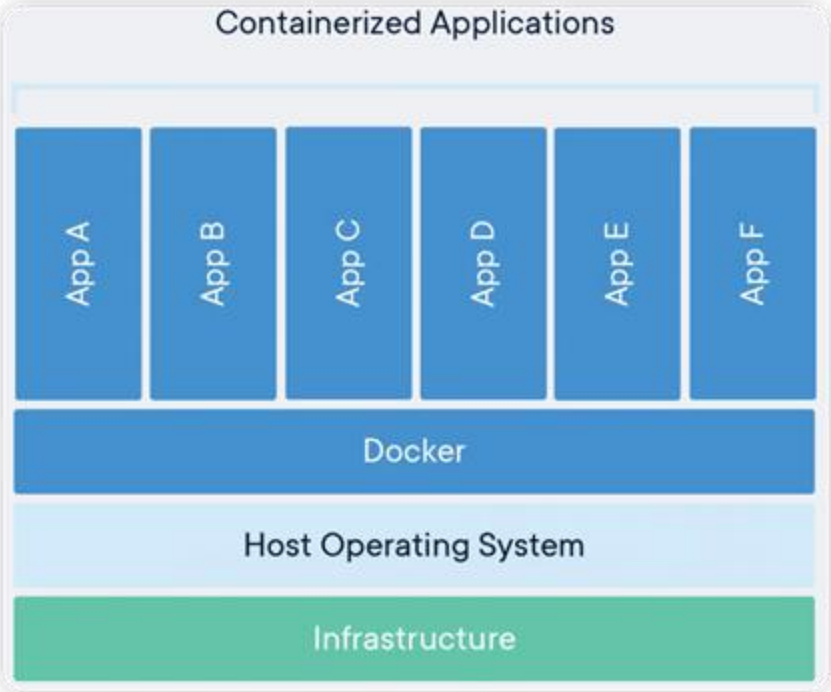
Docker

Run:

- Deliver multiple applications hassle free and have them run the same way on all your environments including design, testing, staging and production – desktop or cloud-native.
- Deploy your applications in separate containers independently and in different languages. Reduce the risk of conflict between languages, libraries or frameworks.
- Speed development with the simplicity of Docker Compose CLI and with one command, launch your applications locally and on the cloud with AWS ECS and Azure ACI.

https://www.docker.com/

Docker container



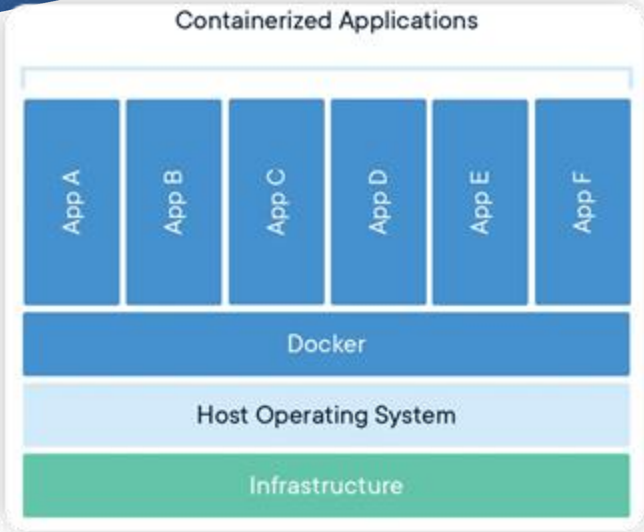
What is docker container:

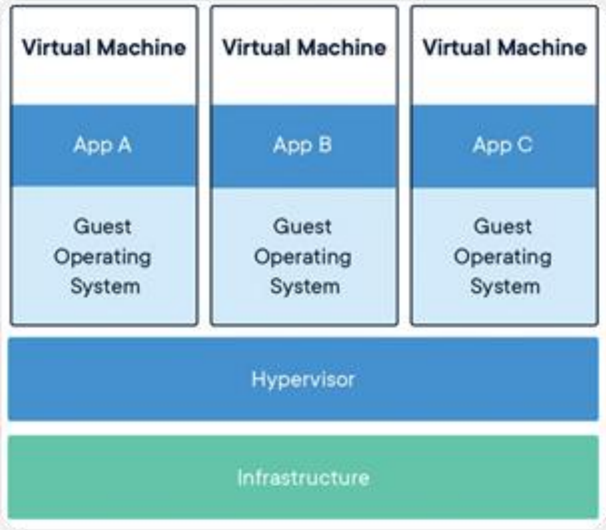
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

- **Standard:** Docker created the industry standard for containers, so they could be portable anywhere
- **Lightweight:** Do not require an OS per application and no licensing costs.
- **Secure:** Strongest default isolation capabilities in the industry.

https://www.docker.com/resources/what-container/

Docker vs. Virtual Machines





Containers

1. packages code and dependencies together.
2. Multiple containers can share OS kernel.
3. isolated processes.
4. less space (tens of MBs).
5. can handle more applications and require fewer VMs and OS.

Virtual Machines

1. Physical hardware turns one server into many servers.
2. The hypervisor allows multiple VMs to run on a single machine.
3. Full copy of an OS, the application, necessary binaries and libraries (tens of GBs).
4. Slow to boot.

https://www.docker.com/resources/what-container/

Containerized Microservices with IoT devices

Why IoT Development needs Microservices and Containerization?

- Easy developing, integrating and scaling applications in large-scale IoT system
- Microservices and containerization enable efficient and faster development by breaking down IoT functionalities into small, modular and independent units that work in isolation without affecting the overall performance of the IoT ecosystem
- Docker container is lightweight and friendly with IoT devices
- Application update and iterate faster

https://www.docker.com/blog/from-edge-to-mainstream-scaling-to-100k-iot-devices/
https://www.einfochips.com/blog/why-iot-development-needs-microservices-and-containerization/

Docker container

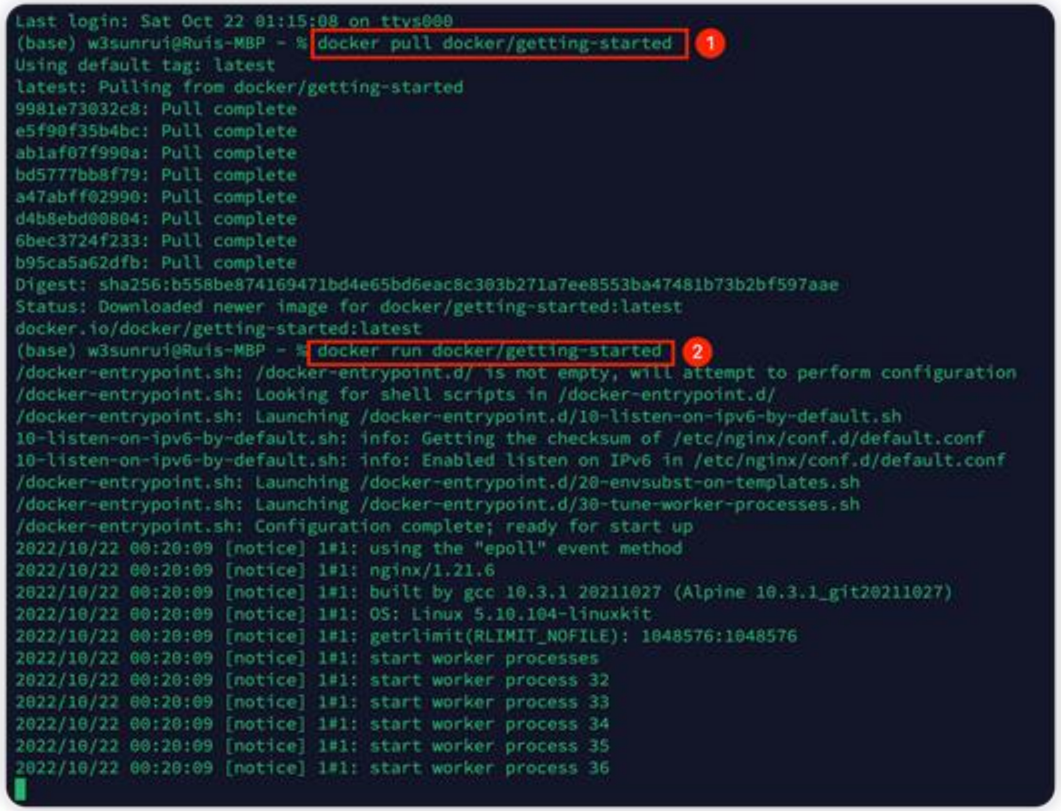
How to run a Docker container from a public image:

1. Pull a docker image:

$ docker pull docker/getting-started

2. Run image as container:

$ docker run docker/getting-started



```
Last login: Sat Oct 22 01:15:08 on ttys000
(base) w3sunrui@Ruis-MBP ~ % docker pull docker/getting-started
Using default tag: latest
latest: Pulling from docker/getting-started
9981e73032c8: Pull complete
e5f90f35b4bc: Pull complete
ab1af07f990a: Pull complete
bd5777bb8f79: Pull complete
a47abff02990: Pull complete
d4b8ebd00804: Pull complete
6bec3724f233: Pull complete
b95ca5a62dfb: Pull complete
Digest: sha256:b558be874169471bd4e65bd6eac8c303b271a7ee8553ba47481b73b2bf597aae
Status: Downloaded newer image for docker/getting-started:latest
docker.io/docker/getting-started:latest
(base) w3sunrui@Ruis-MBP ~ % docker run docker/getting-started
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/10/22 00:20:09 [notice] 1#1: using the "epoll" event method
2022/10/22 00:20:09 [notice] 1#1: nginx/1.21.6
2022/10/22 00:20:09 [notice] 1#1: built by gcc 10.3.1 20211027 (Alpine 10.3.1_git20211027)
2022/10/22 00:20:09 [notice] 1#1: OS: Linux 5.10.104-linuxkit
2022/10/22 00:20:09 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/10/22 00:20:09 [notice] 1#1: start worker processes
2022/10/22 00:20:09 [notice] 1#1: start worker process 32
2022/10/22 00:20:09 [notice] 1#1: start worker process 33
2022/10/22 00:20:09 [notice] 1#1: start worker process 34
2022/10/22 00:20:09 [notice] 1#1: start worker process 35
2022/10/22 00:20:09 [notice] 1#1: start worker process 36
```


https://www.docker.com/resources/what-container/

Docker Compose File

Docker compose file:

The Compose file is a YAML file defining services, networks, and volumes for a Docker application.

YAML file:

YAML is a data serialization language that is often used for writing configuration files.

```json
{
  "doe": "a deer, a female deer",
  "ray": "a drop of golden sun",
  "pi": 3.14159,
  "xmas": true,
  "french-hens": 3,
  "calling-birds": [
     "huey",
     "dewey",
     "louie",
     "fred"
  ],
  "xmas-fifth-day": {
  "calling-birds": "four",
  "french-hens": 3,
  "golden-rings": 5,
  "partridges": {
    "count": 1,
    "location": "a pear tree"
  },
  "turtle-doves": "two"
  }
}
```

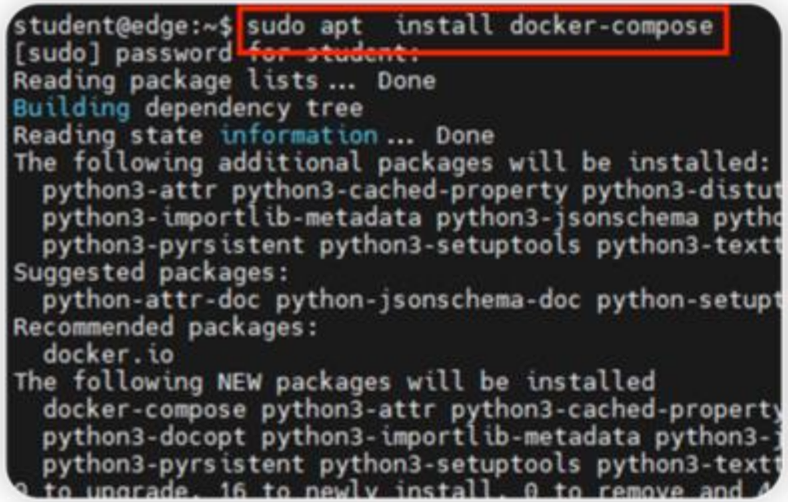
JSON

```yaml
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
  - huey
  - dewey
  - louie
  - fred
xmas-fifth-day:
  calling-birds: four
  french-hens: 3
  golden-rings: 5
  partridges:
    count: 1
    location: "a pear tree"
  turtle-doves: two
```

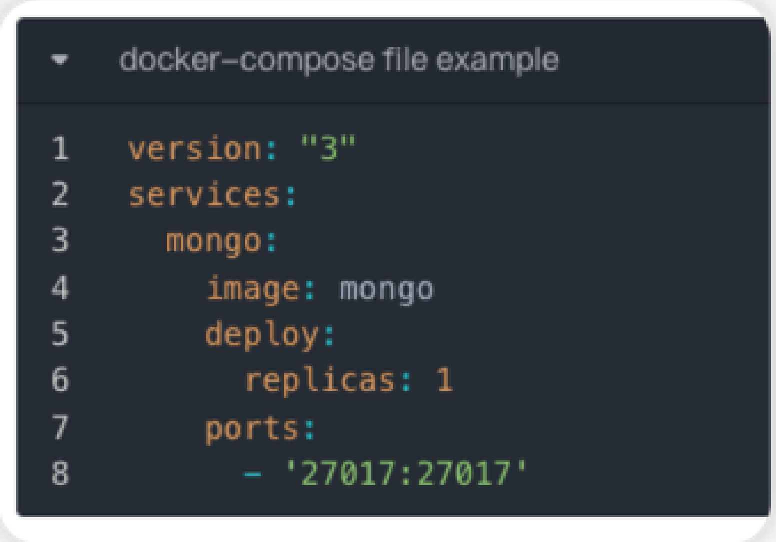
YAM
L

Docker Compose File

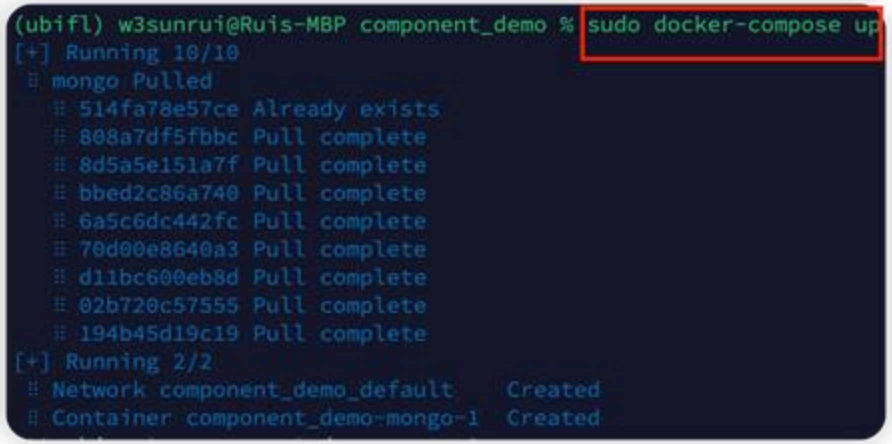
How to automatically pull and start multiple services by defining a Docker Compose file.



1. Install Docker-Compose tool



2. Define docker-compose file



3. Run docker-compose file

How to build your code

1. Define DockerFile

Dockerfile

```
# Base on image_full_name (e.g., ubuntu:18.04) docker image
FROM python:3.8.12

# Switch to root
USER root

# Copy all sources files to workdir
ADD <your project directory> /usr/local/source

# Change working dir
WORKDIR /usr/local/source

# Prepare project required running system environments
#   requirements.txt is a document that pre-define any
#   python dependencies with versions required of your code
RUN pip3 install -r requirements.txt

# Start task
CMD python3 <your main .py file>
```

2. Run DockerFile to build your code into image

Build code

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

3. Define docker-compose file

docker-compose configuration file

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

4. Run docker-compose file to start your code as a container

Start a image

```
sudo docker-compose up
```

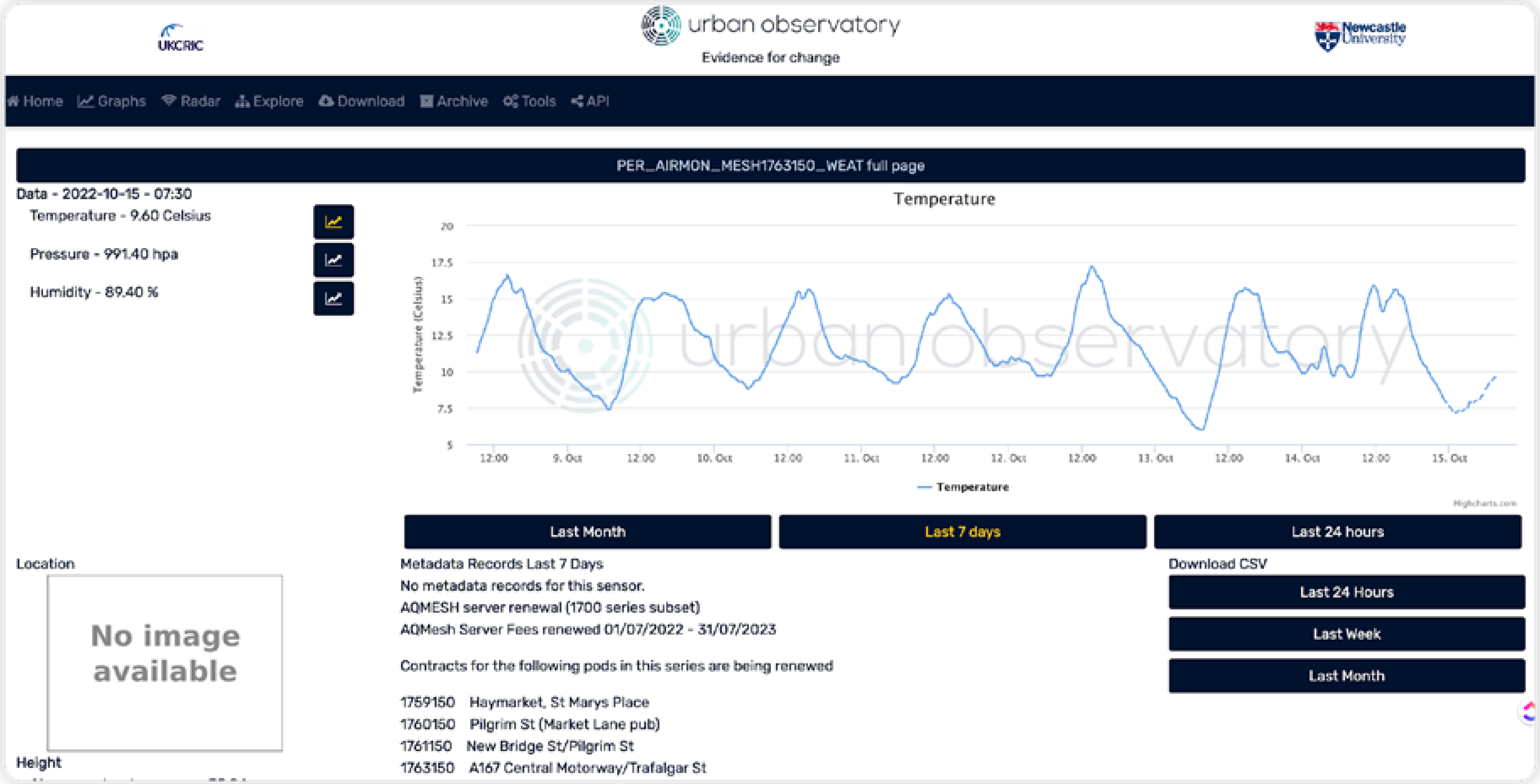
Dockerfile https://docs.docker.com/engine/reference/builder/

Urban Observation Platform

Urban Observatory Platform

urban observatory https://urbanobservatory.ac.uk/

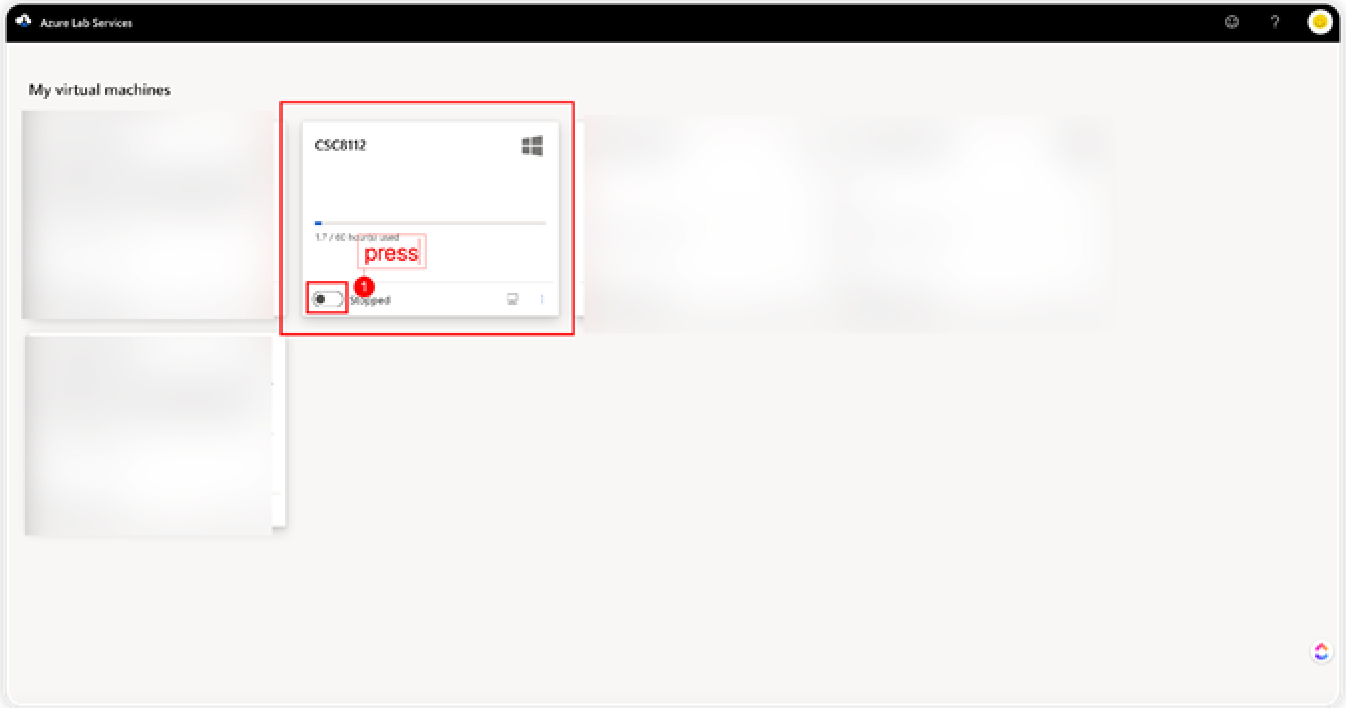
Urban Observatory Platform



urban observatory https://urbanobservatory.ac.uk/

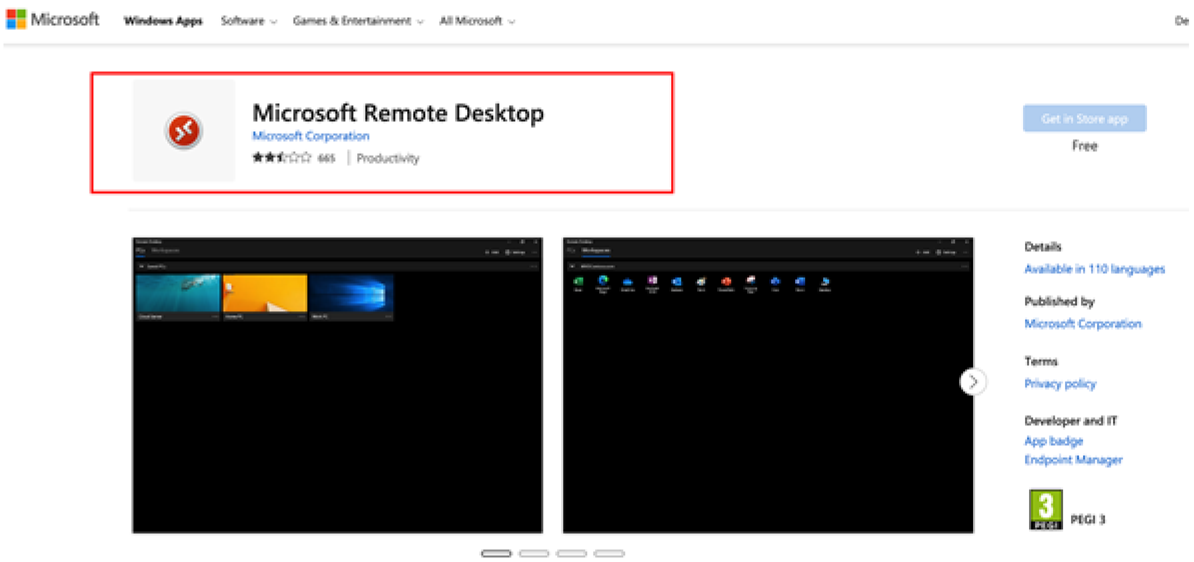
Azure Lab

How to access Azure Lab

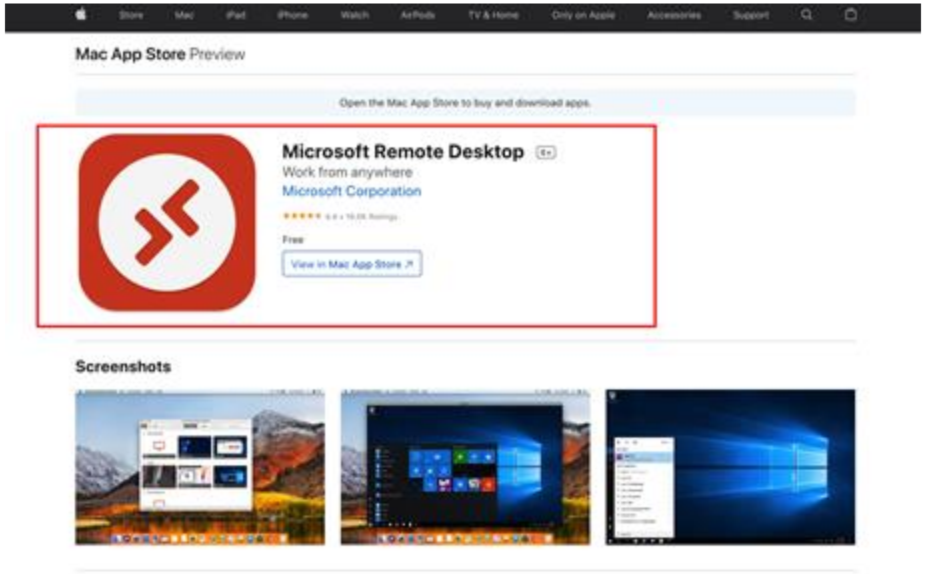


Login via:
https://labs.azure.com/virtualmachines

How to access Azure Lab

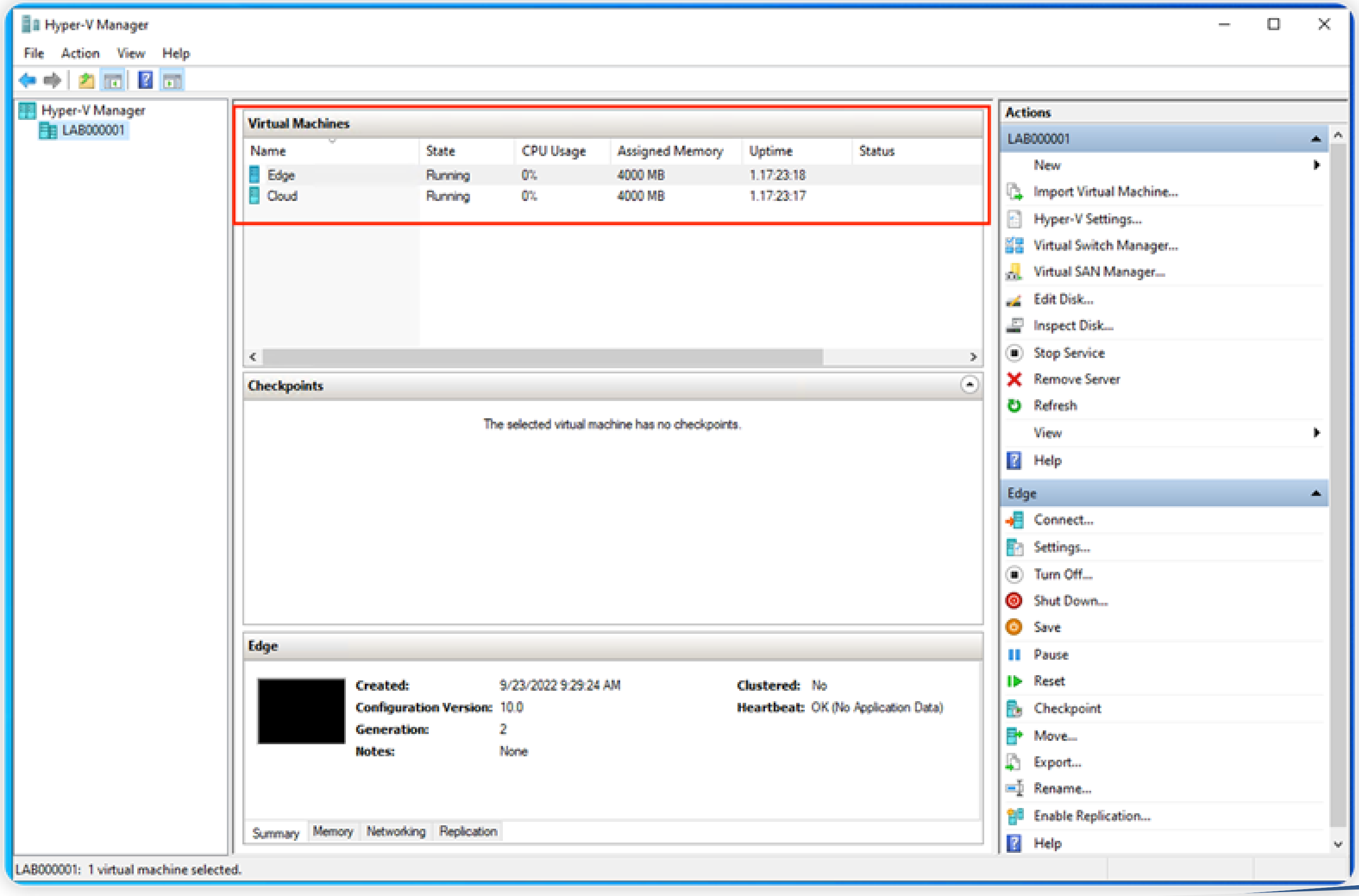


Windows



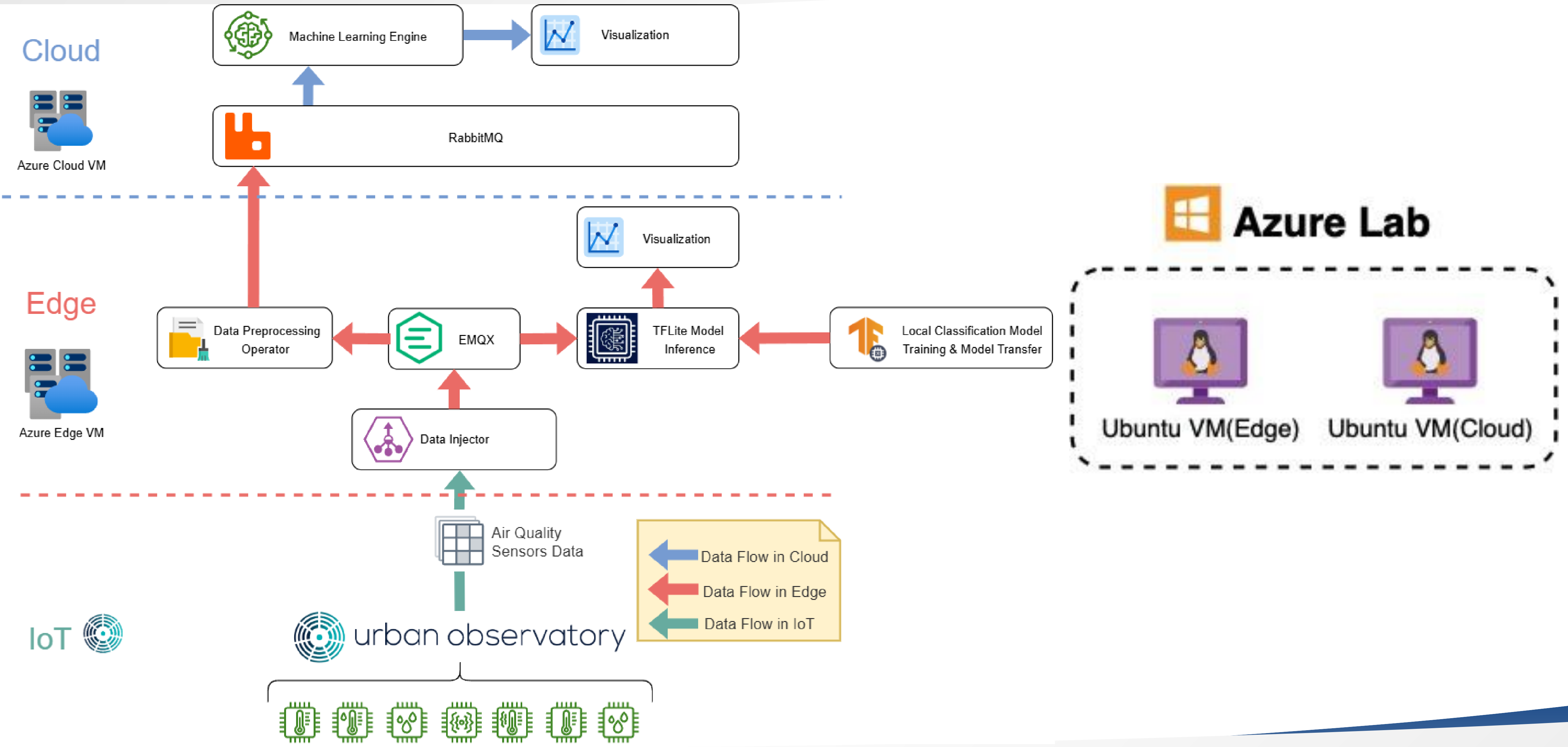
Mac OS

How to access Azure Lab



Outline

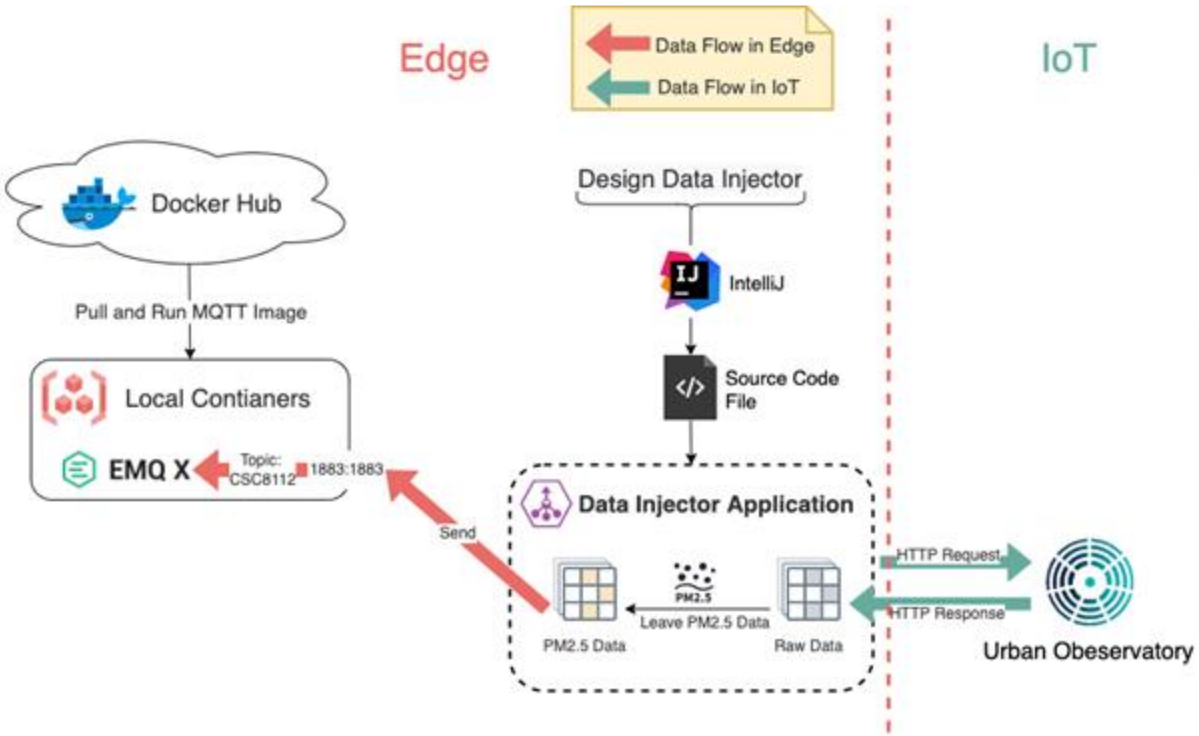
Overview Architecture



Outline

Task 1

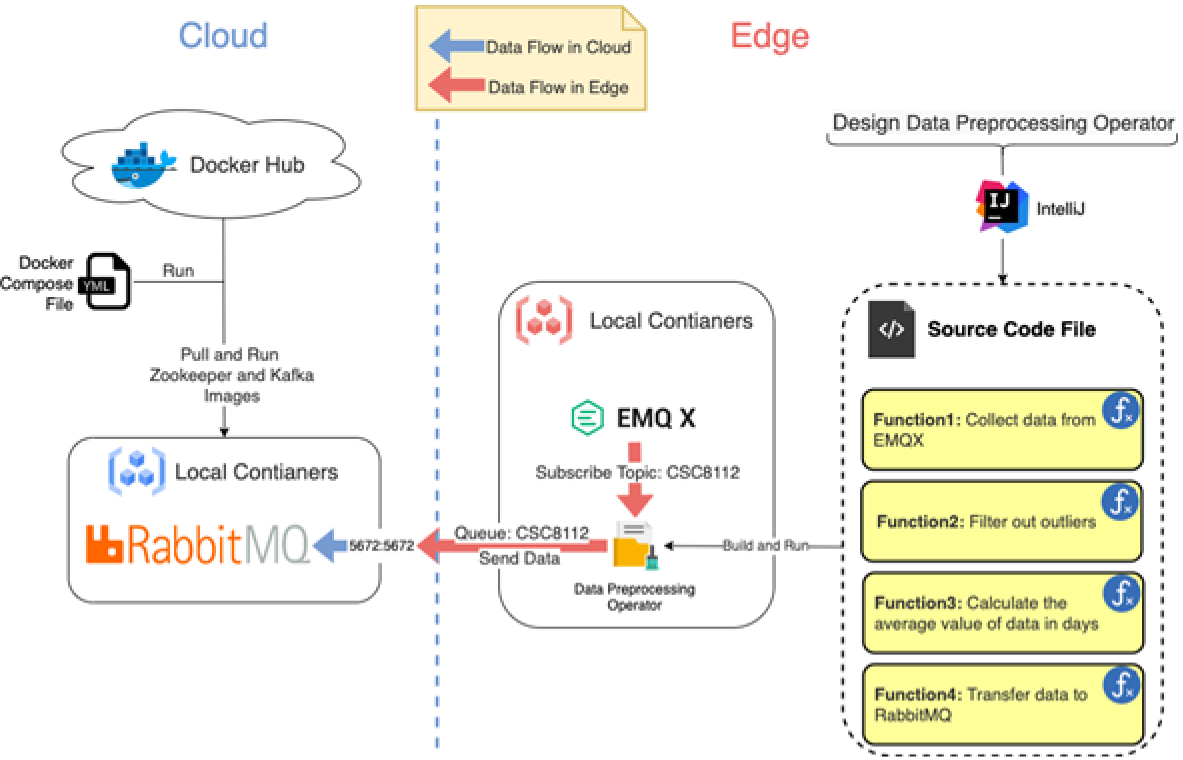
Task 1: structure



1. Pull and run the Docker image "emqx/emqx" from Docker Hub in the virtual machine running on Azure lab (Edge). Perform this task first using the command line interface (CLI).
2. Develop a data injector component with the following functions (Code) in Azure Lab (Edge) or the Azure Lab localhost:
   - (a) Collect data from Urban Observatory platform by sending HTTP request to the following url ([https://github.com/ncl-iot-team/CSC8112/raw/refs/heads/main/data/uo_data.min.json]). Following that, please print out the raw data streams that you collected on the console.
   - (b) Although the raw air quality data you collected from the Urban Observatory API contains many metrics including NO_2, NO, CO_2, PM2.5, and PM10, among others, for the purpose of this coursework you only need to store and analyze PM2.5 data. While many meta-data are available for PM2.5 data, such as sensor name, timestamp, value, and location, you only need to store the metrics related to the Timestamp and Value meta-data fields.
   - (c) Send all PM2.5 data to be used by Task 2.2 (a) and Task 4.5 (b) via MQTT of Azure lab (Edge).
   - (d) Package the data injector as a Docker image. Each run on the Docker image should trigger the send operation once.

Task 2

Task 2: structure



1. Define a *Docker compose file* which contains the following necessary configurations and instructions for deploying and instantiating the following set of Docker images (as shown in Figure 1) on Azure lab (Cloud):

   (a) Download and run RabbitMQ image (rabbitmq:management);

2. Design a data preprocessing operator with the following functions (code) in *Azure Lab (Edge)*:

   (a) Collect all PM2.5 data published by Task 1.2 (c) from EMQX service, and please print out the PM2.5 data to the console (this operator will run as a Docker container, so the logs can be seen in the docker logs console automatically).

   (b) Filter out outliers (the value greater than 50), and please print out outliers to the console (docker logs console).

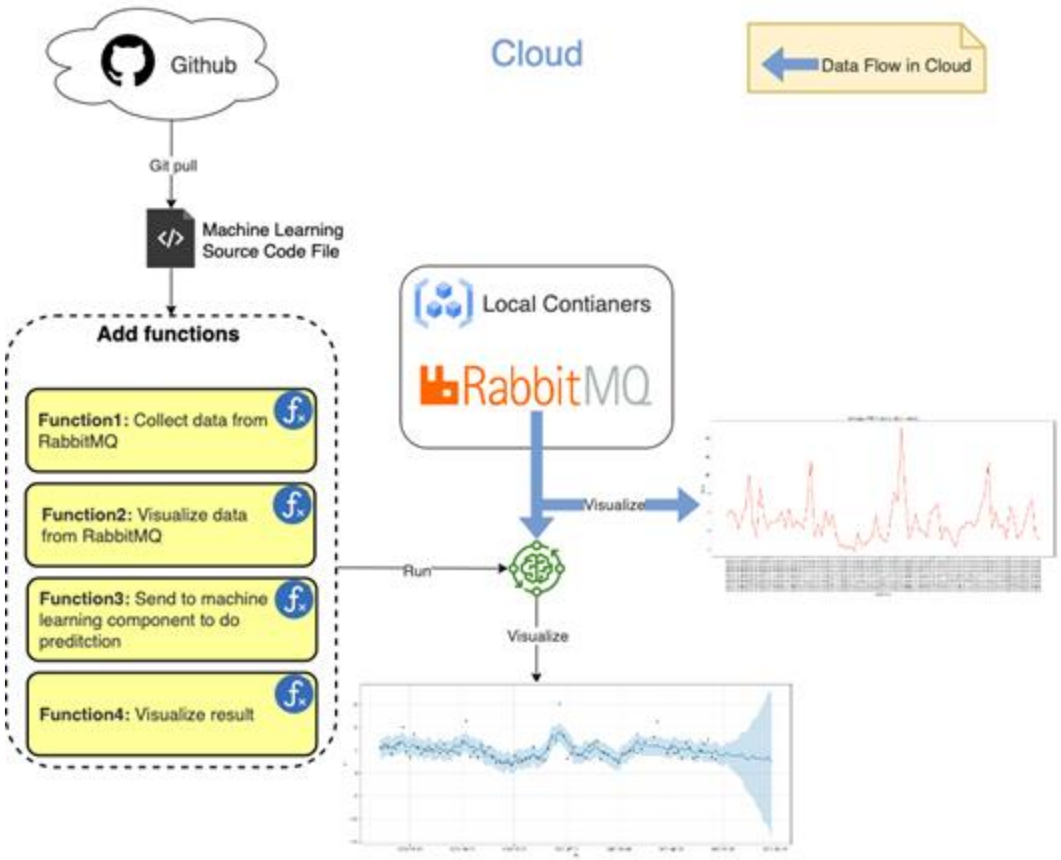
   (c) Average the PM2.5 data over 24-hour periods. Ensure that each daily average aligns correctly with the start and end of the day based on the Unix timestamp, and print the results to the console (docker logs console).

   (d) Transfer all results (averaged PM2.5 data) to be used by Task 3.2 (a) via AMQP to *Azure lab (Cloud)*.

3. Define a Dockerfile to migrate your "data preprocessing operator" source code into a Docker image and then define a *docker-compose file* to run it as a container locally on the Azure lab (Edge).

Task 3

Task 3: structure



1. Download a pre-defined Machine Learning (ML) engine code from [https://github.com/ncl-iot-team/CSC8112_MLEngine].

2. Design a PM2.5 prediction operator with the following functions (code) in Azure Lab (Cloud) or the Azure Lab localhost:

   (a) Collect all averaged daily PM2.5 data computed by Task 2.2 (d) from RabbitMQ service, and please print out them to the console.

   (b) Convert timestamp to date time format (year-month-day hour:minute:second), and please print out the PM2.5 data with the reformatted timestamp to the console.

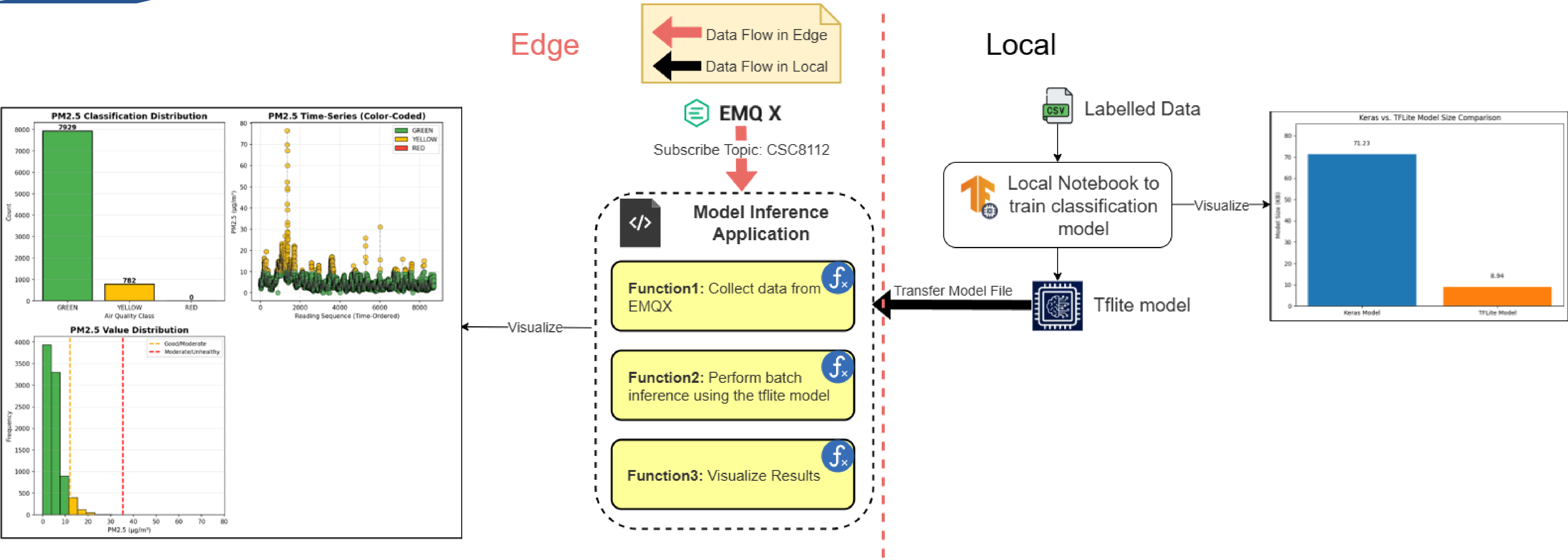
   (c) Use the line chart component of matplotlib to visualize averaged PM2.5 daily data, directly display the figure or save it as a file.

   (d) Feed averaged PM2.5 data to machine learning model to predict the trend of PM2.5 for the next 15 days (this predicted time period is a default setting of provided machine learning predictor/classifier model).

   (e) Visualize predicted results from Machine Learning predictor/classifier model, directly display the figure or save as it a file (pre-defined in the provided Machine Learning code).

Task 4

Task 4: structure



Task4 System Architecture

Task 4: specification

1. Use the provided labeled air quality dataset with PM2.5 values [https://github.com/ncl-iot-team/CSC8112/blob/main/data/PM2.5_labelled_data.csv], and train a classification model locally in your machine or google colab or kaggle environment with cpu/gpu(optional) using TensorFlow into corresponding 3 classifications as Red, Yellow, and Green. You can perform label-encoding, normalization, sampling to the labeled data if needed to improve model quality.

2. Convert the trained model to TensorFlow Lite format (.tflite), apply quantization to reduce model size and save it.

3. Evaluate model performance (accuracy, precision, recall, confusion matrix) and visualize the model size comparison graph(original vs TFLite optimized).

4. Transfer the .tflite model and any other necessary file(if needed) to Azure Lab (Edge VM) using copy-paste or file upload.

5. Create a Python script on Edge VM that:

    (a) Loads the TensorFlow Lite model.

    (b) Subscribe to the MQTT broker

    (c) Perform batch-inference on the json data from the MQTT broker.

    (d) Classifies each PM2.5 reading as Green/Yellow/Red using .tflite model file.

    (e) Logs classification results with the PM2.5 value to console.

    (f) Visualize results with a bar-chart showing count of Green/Yellow/Red classifications, a Time-series plot of PM2.5 values with color-coded classification points, PM2.5 value frequency distribution, and a summary of classification.

Task 5

Task 5

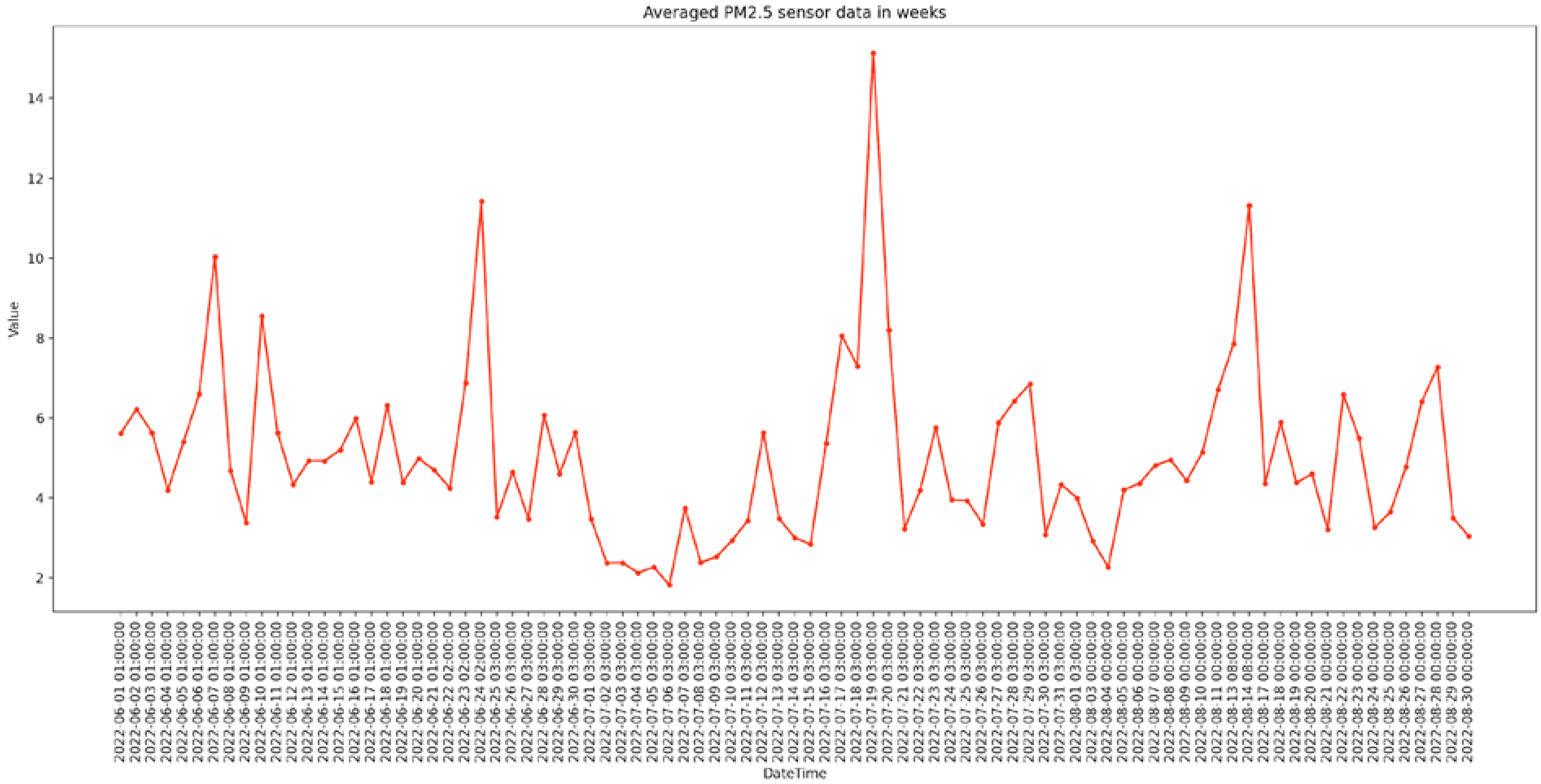
Report

Prepare the Final Report in plain English. The report should consist of:

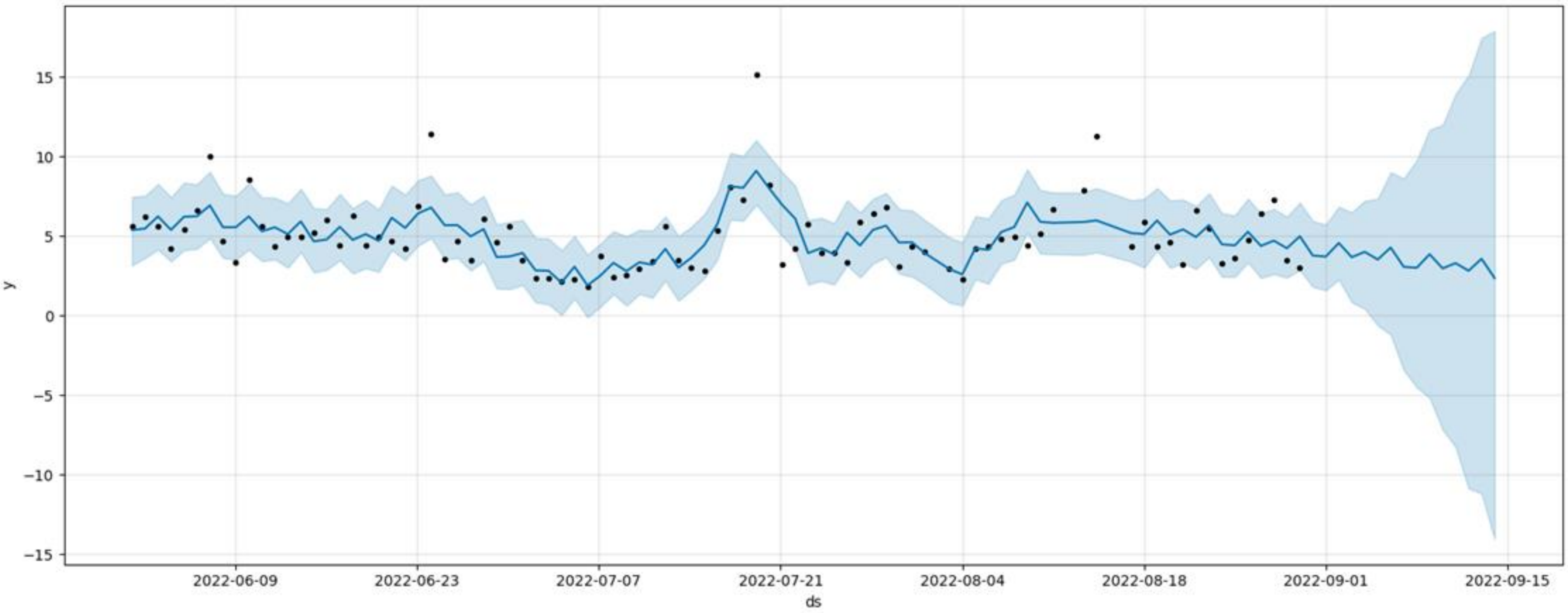
1. Detailed response to each task and related sub-tasks.
2. Screenshots of running services in the Docker Environment.
3. Screenshots of Code Snippets and/or Docker console;
4. Plots of data and prediction results by using Matplotlib.
5. Analytical discussion of the results and related conclusions.

Expected Results of Figures

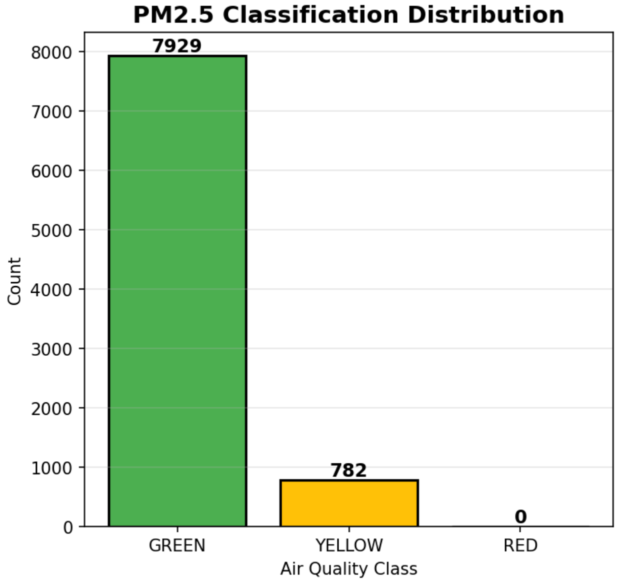
Visualization – Averaged PM2.5 Data

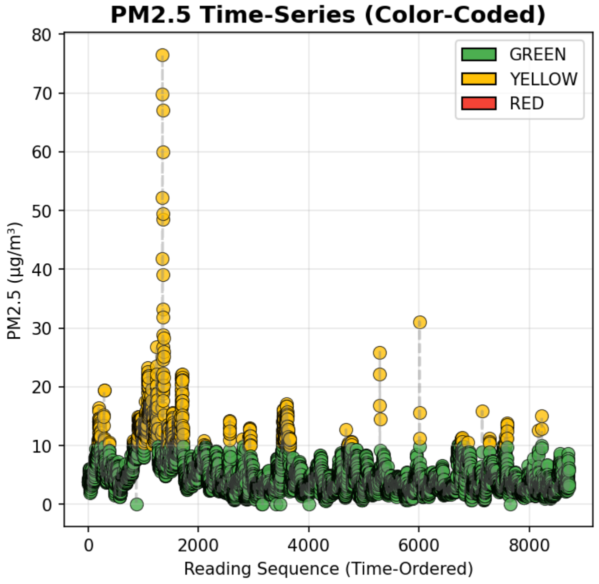


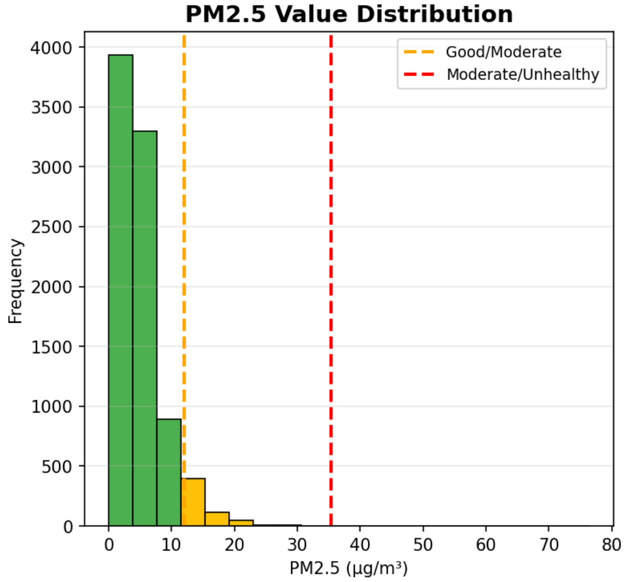
Visualization - Prediction



Visualization – Classification







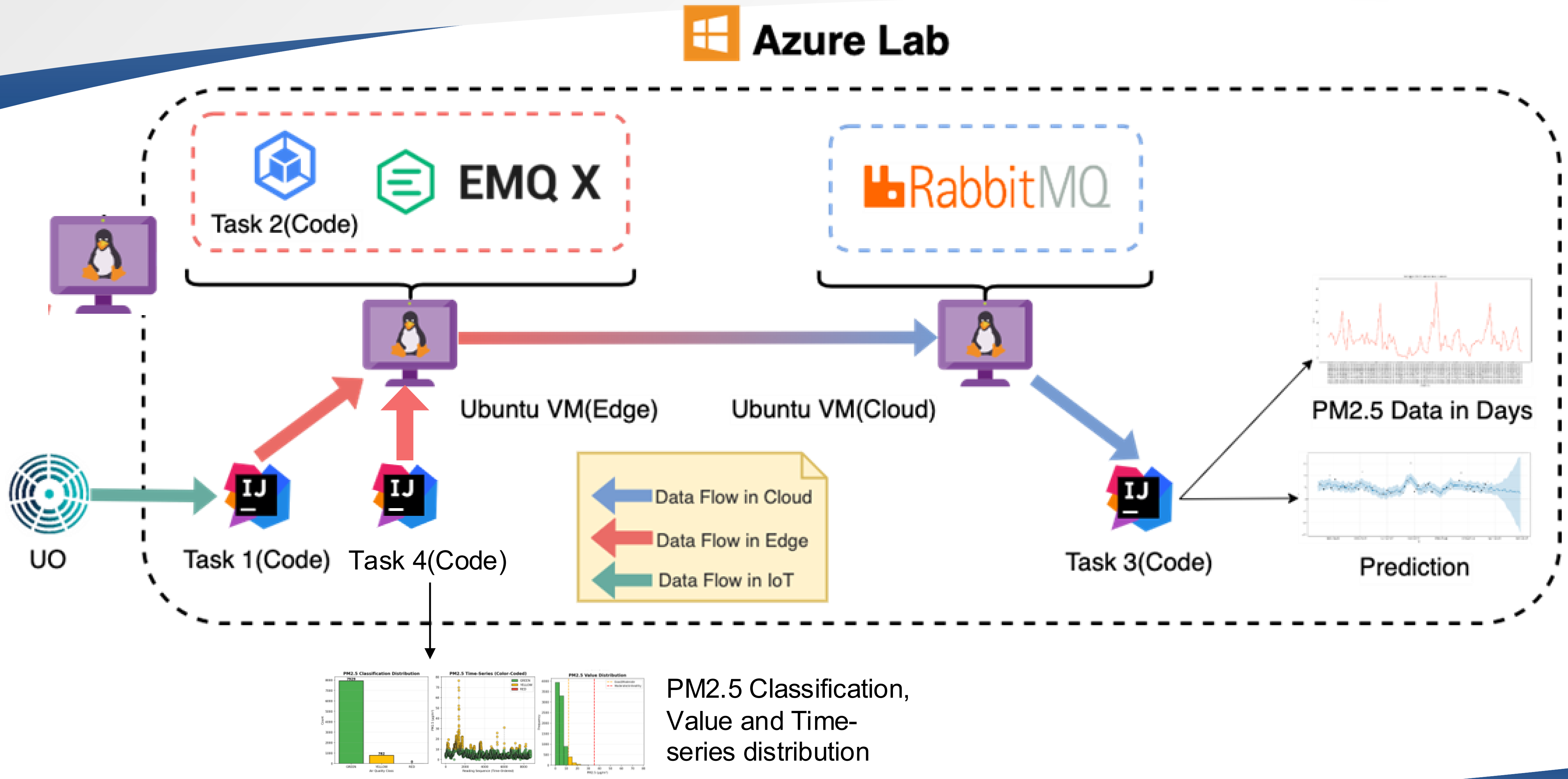

```
STATISTICS SUMMARY
==============================

Total PM2.5 Readings: 8711
Active Sensors: 0

PM2.5 Statistics:
  • Mean:       5.18 µg/m³
  • Median:     4.08 µg/m³
  • Min:        0.00 µg/m³
  • Max:       76.49 µg/m³
  • Std:        3.86 µg/m³

Classification Breakdown:
  • GREEN:  7929 ( 91.0%)
  • YELLOW:  782 (  9.0%)
  • RED:       0 (  0.0%)
```

Note:

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

Full Workflow in Azure Lab

Summarized structure



Newcastle University

Thanks

For Your Listening