Network Softwarization: Technologies and Enablers (Winter 2019)

Module 2.4: IoT, Smart Cities, and 5G Use Cases

Over the past two decades, the emergence of IoT technologies and the pervasive use of small electronic devices in monitoring the urban infrastructure and services have transformed many aspects of modern human societies and set the foundation of "smart framework". Connected devices generate large volumes of data which require sophisticated data collection and analysis methods to be effectively used. This lab builds upon the use of Open Data APIs and effective open source software components to enable you to collect environmental data and deploy a basic application to perform some data analytics tasks.

Two achieve this, we can use two different approaches. The first one is "**pull-based approach**" in which we use web protocols (HTTP or HTTPS) to connect to public API endpoints and collect data. The second mechanism is to use "**push-based approach**" in which we develop an IoT gateway that notifies applications and users whenever there is an update in the data.

This lab will use the Python programming language for the implementation requirements. The following links may provide related information to help you with your programming challenges. To complete the programming tasks, you only need to set up your computer to use Python.

- https://www.tutorialspoint.com/python/
- https://www.learnpython.org/

Part 1: Traffic Flow Analysis (Pull-based Approach)

For this part of the lab, you will use the following APIs from the Connected Vehicles and Smart Transportation (CVST) IoT application platform to obtain real-time data about the traffic flow in GTA. To check the platform visualization tools and learn more about the current platform APIs, please navigate to portal.cvst.ca.

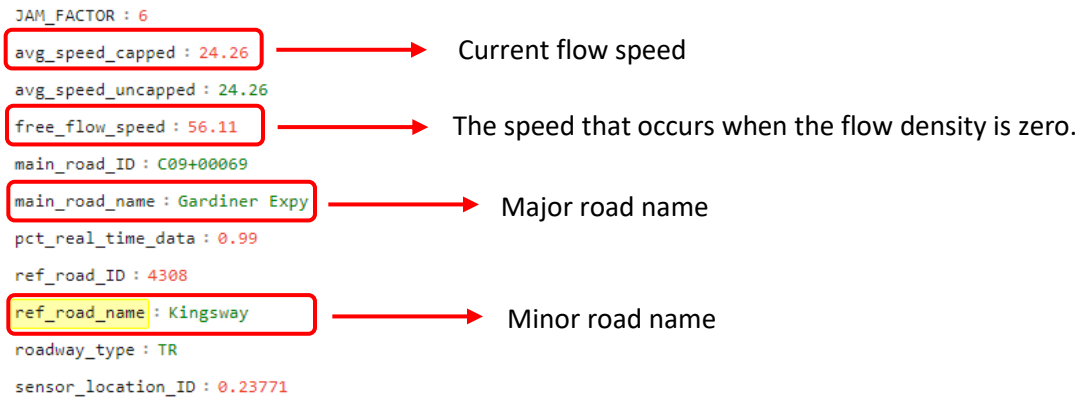
- CVST APIs that we use in this part of the lab:
  - **Traffic Sensor API:** http://portal.cvst.ca/api/0.1/HW_speed
  - **Bixi Bicycle API:** http://portal.cvst.ca/api/0.1/bixi

Note 1: Any API call or data analytics tasks (including the data visualization) should be performed within your Python application.

Part 1.1: Flow Speed Analysis

For this part of the lab, you will use "Traffic Sensor" API to collect real-time traffic flow data for major roads, highways, and streets in GTA. To access to the API data, you need to call the API first. Upon calling, you will receive a JSON document that has four variables. The variables description are as follows:

- **_id** → This is a platform specific ID which refers to the document ID in our fast Indexing service.
- **data[]** → Contains a list of all the GTA traffic sensors and the associated traffic flow measurements. Currently, we are collecting measurements for 5448 sensors. The following shows the set of variables and measurements for a sample traffic sensor in the list.



- **date_time** → Provides the date and time under which the measurements have been collected.

- **timestamp**: Provides the number of seconds since January 1st, 1970 in UTC.

Note 1: You can use the online JSON viewer available at https://jsoneditoronline.org/ to see the JSON document and its structure before you use it in your code or you can install a JSON formatter extension on your browser.

Tasks to Complete

1. You need to write a Python code to collect the Traffic Flow data from "Traffic Sensor" API and extract the following fields from it.
    - **data[]**: For each sensor object data in the list, extract the items that are indicated by a red rectangle around them as shown in the figure above and eliminate the rest. The fields that need to be kept are "avg_speed_capped", "free_flow_speed", "main_road_name", and "ref_road_name".
    - **date_time**: keep this item to stamp your data with the time and date at which they are collected.
2. You need to combine the "main_road_name" and "ref_road_name" together to create a single unique field to indicate the exact location of the Traffic Sensor. To do so, name the new field as "road_name" and use the following pattern to combine the two fields.
    - road_name = main_road_name + "/" + ref_road_name

Note 2: The Python code you developed in step 1 and step 2 is called a **Parser**. The main role of the Parser module is to retrieve data from an external source and transform it into your platform specific format.

3. You need to run your Parser code for a **continuous duration of 24 hours and call the "Traffic Sensor" API every 1 hour**. Each time you call the API, you need to perform steps 1 and 2 as well. Feel free to use any methods you are familiar with to save the data you collected.

Note 3: Please avoid calling the platform APIs more than the required frequency.

Important Point: We only need a subset of the collected data to perform the following analysis. Please choose the first ten Traffic Sensors (roads name) from the collected data and eliminate the rest. Each time you call the API, you receive a JSON document that contains a variable called "data". The variable is a list that contains

data for all the traffic sensors in the GTA. To keep some of the traffic sensors data and eliminate the rest, you need to apply the changes in this part of the JSON document. **For efficiency and better performance achievement of your application**, you can first select the first ten traffic sensors and then proceed with Steps 1 to 3.

4. After collecting the data in Step 3, you need to perform the following analysis on the data.
   4.1. Create a graph and plot the "avg_speed_capped" for all the selected traffic sensors independently from each other as a function of time (24 hours). Use vertical axis for the speed and horizontal axis for the time. Can you find any specific pattern in the data?

Note 4: You can plot all the sensors data in one graph or you can plot them separately.

In your report, include the graph(s) and explain your answer.

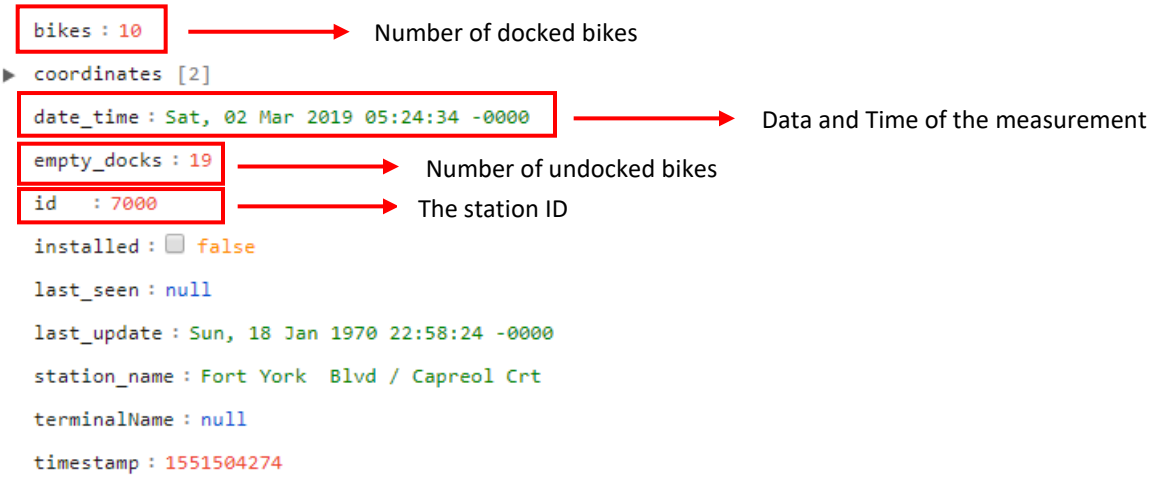
4.2. Divide "avg_speed_capped" by "free_flow_speed" and name it as "ρ". Create a graph and plot the parameter "ρ" as a function of time (24 hours) for the same roads you used in step 4.1 and answer the following questions?

- Can you explain what does the graph specify?
- Can you find any specific trends in the data?
- Can you explain what does it mean if the value of "ρ" is bigger than one?
- Can you also find an example of this event in your graph?

In your report, include the graph(s) and explain your answers.

Part 1.2: Bixi Data Analysis

For this part of the lab, you will use the "Bixi Bicycle" API to collect real-time data about the Bicycles in GTA from the CVST platform. You need to call the API first to retrieve the data. Upon calling the API, the platform will provide you with a JSON document that contains the following data items.



Important Point: The above image shows state data for a single Bixi station in the GTA. Each time you call the API, you will get update for all the installed stations.

Note 5: It is possible that you will receive data for changing number of stations in each API call. For accuracy of your analysis, **keep the data for the first 30 stations and eliminate the rest** each time you collect data.

Note 6: You can use the online JSON viewer available at https://jsoneditoronline.org/ to see the JSON document and its structure before you use it in your code or you can install a JSON formatter extension on your browser.

Tasks to Complete

1. You need to write a Parser code by using Python and collect the API data **for a continuous duration of 24 hours with a frequency of one API call per hour**. Each time you receive the JSON digest from the API, you need to keep the data items that are circled in red from the above picture and eliminate the rest. You need to do this for every single station you selected (first thirty). Feel free to use whatever mechanisms you like to keep your data for future steps.

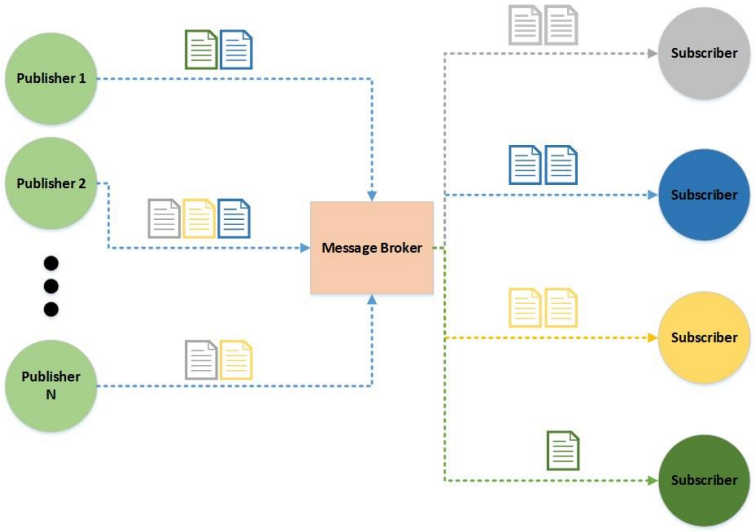
Note 7: The date_time variable denotes the exact time when the station reported its status to the platform. Stations are not sending their data at the same time and that's why you may see a time variation of (+- 45mins) between the stations reported time. However, every single station reports its status at least once per hour. Therefore, you can ignore the time variation and assume that each time you call the API, you get new updates.

2. The sum of the values for “bikes” and “empty_docks” denote the maximum capacity of each station. The “empty_docks” value also shows the number of bikes which are taken from the station. For each station, create a new parameter and name it as “station_capacity”. You need to calculate the value for “station_capacity” according to the formula below.
    - station_capacity = empty_docks + bikes
3. After preparing your data in step 1 and 2, you need to write a Python code that performs the following simple analysis.
    3.1. For each time you collect the data (every hour) calculate the total number of Bixi bicycles and the total number of bicycles which are currently in use in the whole GTA.
    3.2. Divide the total number of bicycles in use by the total number of bicycles in GTA for every time you receive data (every hour) and name it as “bicycle_load”.
    3.3. Create a graph and plot the “bicycle_load” as a function of time (24 hours). Can you explain what does the graph specify? Can you find any pattern in the data?

In your report, include the graph and explain your answers.

Part 2: Create an Event-based IoT Gateway (Push-based Approach)

In this lab, you will create an event-based IoT gateway that collects data from multiple data sources and notifies the interested users if an event occurs in the system. To achieve this, we will take the advantage of a publish subscribe messaging system that uses MQTT protocol and enables multiple users to subscribe for event notifications. Upon the occurrence of an event, the message broker informs the subscribers of that event. The following event demonstrates a very high-level architecture of what a publish subscribe system may look like.



The publishers and subscribers use a naming scheme to communicate with each other. In this lab you will use a hierarchical URI-based naming scheme. For instance, **"utoronto/GB119/sensor1/temperature"** is a hierarchical name that specifies the temperature data which belongs to a geographical location. Names also refer as Topics in such systems. Each document and subscriber with a different color in the above figure refers to a specific topic in the system.

In this lab, we will collect the Air Pollution and Weather Information data from two open IoT data platform APIs.

- Air Quality API accessible at: https://aqicn.org/api/
- Current Weather API accessible at: https://openweathermap.org/

To access to these APIs, you need to get a token. For Air Quality API, navigate to https://aqicn.org/data-platform/token/#/ and fill your email and preferred name to get your token.

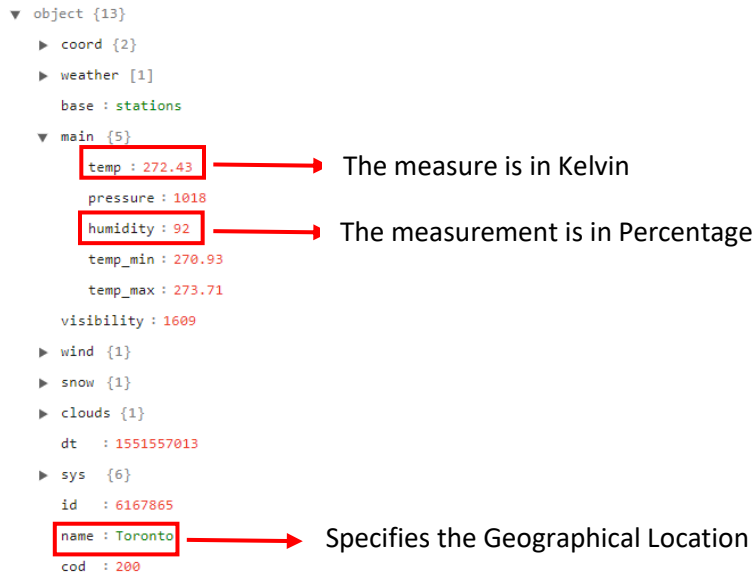
For Current Weather API navigate to https://openweathermap.org/appid and click on Sign up and create an account to get your token. You can always have access to your tokens and create more by navigating to the token page available at https://home.openweathermap.org/api_keys.

After successfully receiving your tokens you can use the following API calls to retrieve data from the specified data sources.

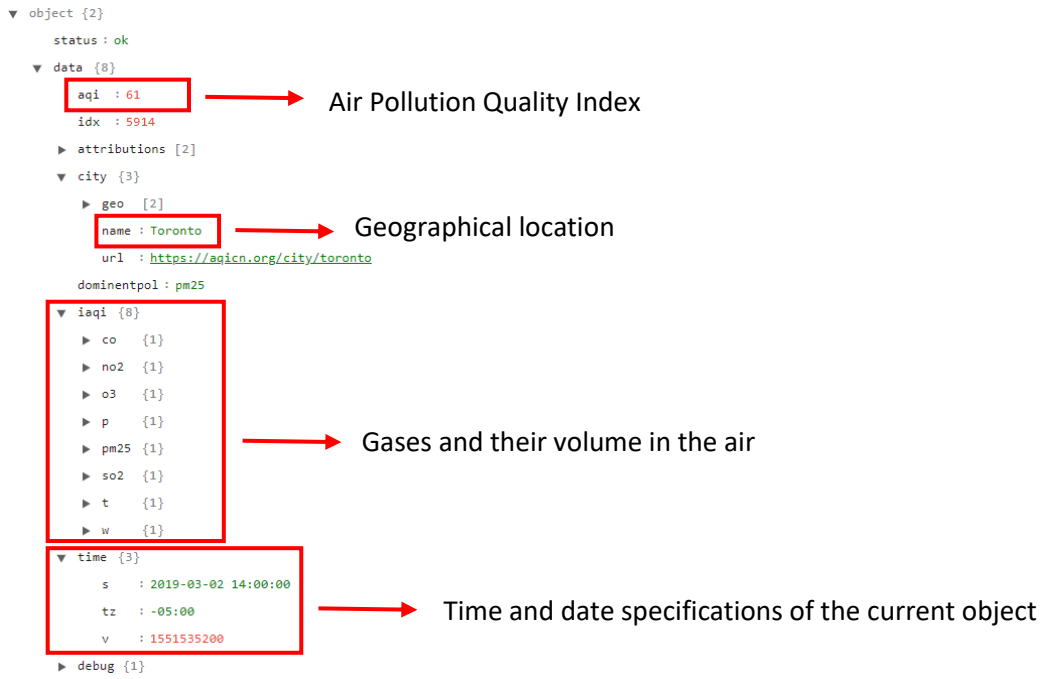
- Air Quality API:
  - http://api.waqi.info/feed/"The City Name"/?token="Your Token"
    - Please replace the items in quotation mark with your values.
- Current Weather API:
  - https://api.openweathermap.org/data/2.5/weather?q="City Name","Country Code"&appid="Your Token"
    - Please replace the items in quotation mark with your values.
    - For instance, you can use “ca” to refer to Canada.

Upon calling each API, you will receive a JSON document that provides you with the latest update of the platform data with reference to your input parameters. The followings show an example of the two APIs digest.

- Current Weather API:



- Air Quality API:



Note 8: Please refer to http://aqicn.org and https://api.openweathermap.org for more information about the APIs.

Tasks to Complete

1. For this part of the lab, you need to create two publishers.

- **Air Quality Publisher:** Your publisher performs two important tasks.
  - **Data collection**: Your publisher calls the Air Quality API for the following cities in the world and collect their Air Quality Index every hour for a continues duration of 24 hours. Cities are: {Toronto, Vancouver, Newyork, Shanghai, Tehran, London, Seoul, Jakarta, Istanbul, and Tokyo}.
    Example: http://api.waqi.info/feed/newyork/?token="Your Token"
  - **Event notification**: Your publisher must publish each cities Air Quality Index with its associated city name as the message topic to the message broker. Use the following naming scheme to publish Air Quality data for each city.
    - ece1508/city_name/aqi → Example: ece1508/toronto/aqi

Note 9: You must publish the data for every city once you call the API every hour. **Do not save** the whole data and send them at once.

- **Current Weather Publisher**: Your publisher performs three important tasks.
  - **Data Collection**: Your publisher calls the Current Weather API for the following cities in the world and collect their temperature and humidity every hour for a continuous duration of 24 hours. Cities are: {Toronto, Vancouver, Newyork, Shanghai, Tehran, London, Seoul, Jakarta, Istanbul, and Tokyo}.
    Example:https://api.openweathermap.org/data/2.5/weather?q=toronto,ca&appid="Your Token"
  - **Calculating the Heat Index**: You need to calculate the Heat Index according to the following formula for each time you collect data (every hour) and for each city separately. The Heat Index indicates what we know as **"feels like"**.

    $$\mathrm{HI} = c_1 + c_2T + c_3R + c_4TR + c_5T^2 + c_6R^2 + c_7T^2R + c_8TR^2 + c_9T^2R^2$$

    In the above formula use the following values for the variables.
    T = Ambient dry-bulb temperature (in degrees of Celsius) → You need to convert the temperature from Kelvin (API-Specific measurement) to Celsius according to the following formula.

    $$T_{(°C)} = T_{(K)} - 273.15$$

    R = Relative humidity in percentage (between 0-100)
    HI = Heat index in degrees of Celsius
    For the values of co-efficient parameters please use the followings:

    c_1 = −8.78469475556
    c_2 = 1.61139411
    c_3 = 2.33854883889
    c_4 = -0.14611605
    c_5 = -0.012308094
    c_6 = -0.0164248277778
    c_7 = 0.002211732
    c_8 = 0.00072546
    c_9 = -0.000003582

- **Event notification**: Your publisher must publish each cities Heat Index with the city name as the message topic to the message broker. Use the following naming scheme to publish Heat Index for each city.
    - ece1508/city_name/hi → Example: ece1508/toronto/hi

Note 10: You must publish the data for every city once you call the API every hour and calculate the Heat Index. **Do not save** the whole data and send them at once.

2. For this part of the lab, you need to create two subscribers to receive the publisher's data and **plot the hourly changes as a function of time (24 hours) for each city independently**.
    - **Air Quality Subscriber**: The subscriber must subscribe for the same topics that you specified to publish your data. In the event of an update, the subscriber receives the data from the message broker. Can you find any specific pattern in the graphs?

    In you report: Include your graphs and explain your answer.

    - **Current Weather Subscriber**: The same as above, you need to create a subscriber to receive the Heat Index of each city. In the event of an update, the subscriber gets notified by the message broker and will receive the data. Can you find any specific pattern in the graphs?

    In you report: Include your graphs and explain your answer.
3. Can you see any specific cross-relationship in the movement pattern of Heat Index and Air Pollution Index for the specified cities?

    In your report: Explain your answer.

Note 11: In addition to your answers and graphs that you provided for each section, you must include your Publishers and Subscribers code. For verification, we run our publishers with your subscribers to test the outcomes.

Important Point: For this lab, you need to use the following pieces of information to connect publishers and subscribers to the message broker.

- **MQTT Message Broker IP address**: 142.150.208.252
- **Port Number**: TCP 1883

Note 12: Please run your subscribers before the publishers.

What to Submit

This lab includes two major parts. Each part includes multiple tasks. Please look for "**In your report**" key phrase in each part and sub parts of the lab and provide the material which we request for. The following describes the structure of your submission and any additional contents that you need to submit.

To submit your materials, you need to create a compressed file (e.g., zip, rar, or tar) that contains the following items.

- **A .pdf document** that contains the answer to all the questions in the lab.
- **The code you developed** for the publishers and subscribers. Please note that you must use Python to create the publishers and subscribers. Your final submission should include two publishers and two subscribers with the following names with reference with the API endpoints they work with.
    - **Air Quality API:**
        - Publisher name → aqi_pub.py
        - Subscriber name → aqi_sub.py
    - **Current Weather API:**
        - Publisher name → hi_pub.py
        - Subscriber name → hi_sub.py

Note 13: Please provide informative comments on your code, if it is necessary to understand a part of it.

Important Note: The implementation of MQTT protocol for a sample publisher and subscriber will be provided to you. You can use them as a function in your code to establish communication with the MQTT message broker.

Useful Material

Since we use the MQTT protocol on publishers and subscribers to communicate with the message broker, it might be useful to know how to work with the protocol. The following links provide additional information that may help you while you are writing your code.

- https://github.com/mqtt/mqtt.github.io/wiki
- http://www.steves-internet-guide.com/into-mqtt-python-client/

During this lab, you may need to visualize data and create plots in your program. The "matplotlib" is a powerful plotting library which you can use to create plots in your program. The following links may help you to use the library and its features.

- https://matplotlib.org/users/pyplot_tutorial.html
- https://swcarpentry.github.io/python-novice-gapminder/09-plotting/

The following link may also be helpful in setting you up to work with the external APIs in Python. You may find more useful resources on Internet as well.

- https://www.dataquest.io/blog/python-api-tutorial/

Note 14: You are not limited to use any libraries in this lab. Feel free to take the advantage of any library that you are familiar with to do this assignment.