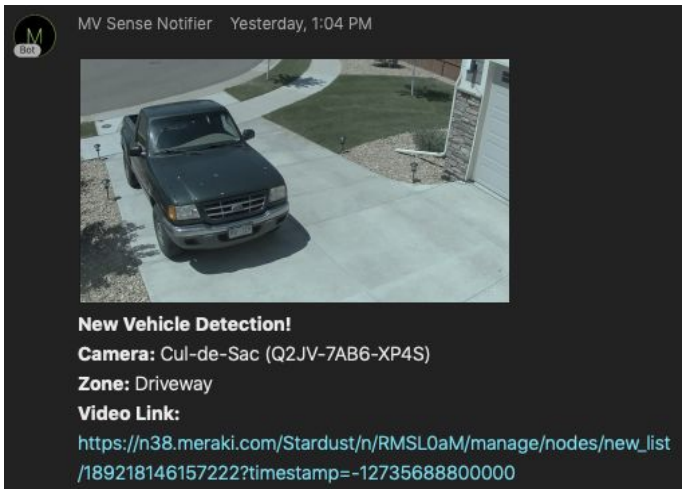
Meraki MV WebEx Teams Bot

People/Vehicle Detection and Analysis

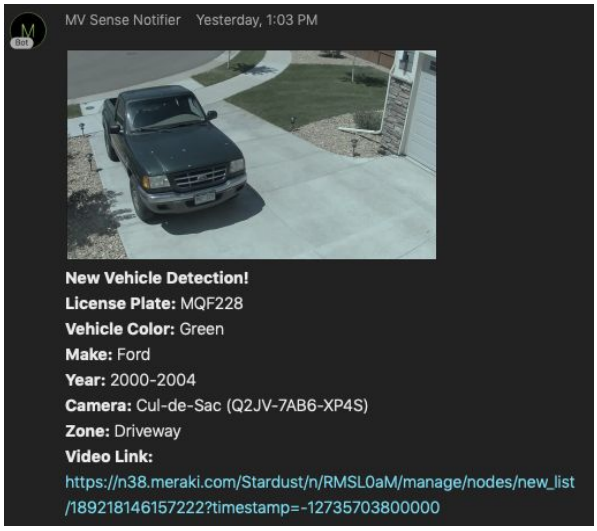
Summary

Without external integrations, the MV Bot will detect people and/or vehicles, and will send a message to WebEx Teams or Telegram with a snapshot, detail of the detection, and a video link. The resulting alert looks like this:



With optional external integrations, license plate recognition (LPR), vehicle analysis, face analysis, and image analysis with description can be layered on. Note that these services come with a cost, but there are free trials for all three that can be used for a POC/lab.

A sample vehicle analysis/LPR alert looks like this:



Server Setup

In order to run the app, you'll need to have Python3 installed, along with some of the dependencies. Installation instructions for Python can be found here: https://realpython.com/installing-python/

It also never hurts to make sure your packages are up to date. Assuming an "apt"-based linux distro:

```
apt-get update && apt-get -y upgrade
```

App Setup

Extract the ZIP, then navigate to that directory from the command line or other python-enabled interface.

To install the required dependencies, run:

```
pip3 install -r requirements.txt
```

Integrations Setup

For the following sections, open the **config.py** file as we'll be adding to it based on the desired integrations. OpenALPR and Microsoft Cognitive services are optional, and can be enabled/disabled in the **config.py** file.

Meraki

Obtain a Meraki API key with write access to your organization. For instructions, see here: https://documentation.meraki.com/zGeneral_Administration/Other_Topics/The_Cisco_Meraki_Dashboard_API

Paste your Meraki API key in **config.py** as **MERAKI_API_KEY**.

Use the Meraki API to grab the Network ID containing your cameras:
First, get your Organization ID:

```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' -H 'Content-Type:
application/json' -X GET
'https://api.meraki.com/api/v0/organizations'
```

Next, use the Organization ID to find your Network ID:

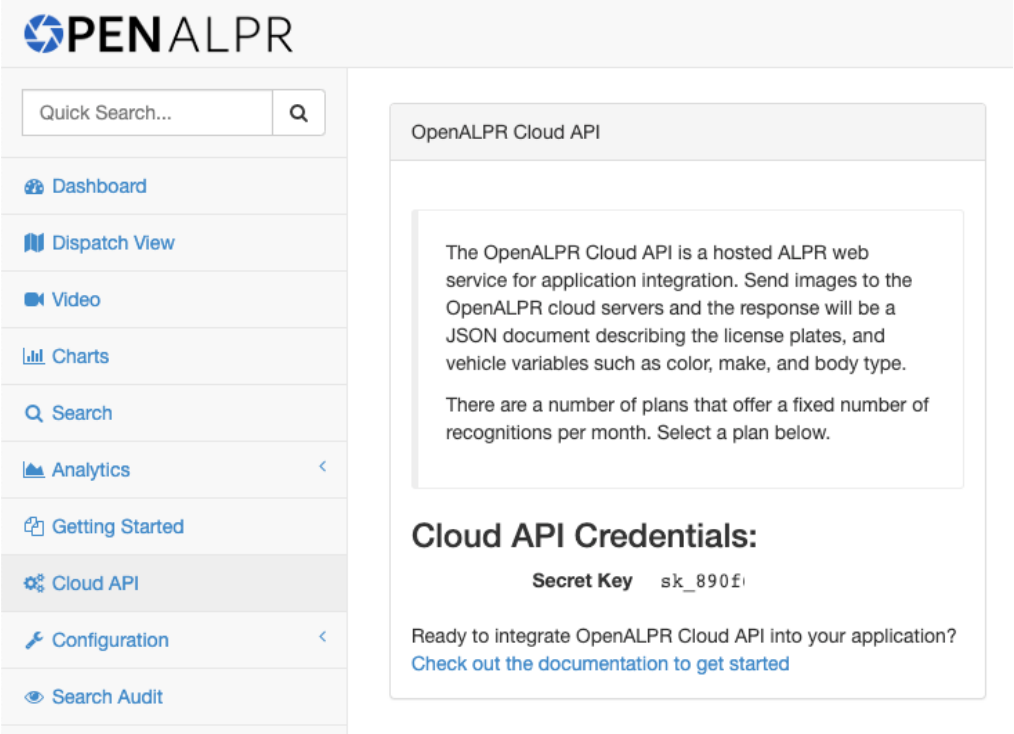
```
curl -L -H 'X-Cisco-Meraki-API-Key: <key>' -H 'Content-Type:
application/json' -X GET
'https://api.meraki.com/api/v0/organizations/{organizationId}/network
s'
```

Paste your Network ID in **config.py** as **NETWORK_ID**.

OpenALPR

OpenALPR is used for vehicle detection and license plate recognition (LPR). You get 1,000 image analyses per month for free.

Go to https://www.openalpr.com/ and sign up for a free account. Once done, head to the “Cloud API” page, copy your **Secret Key**, and paste it as the **openalprsecret** variable in config.py.

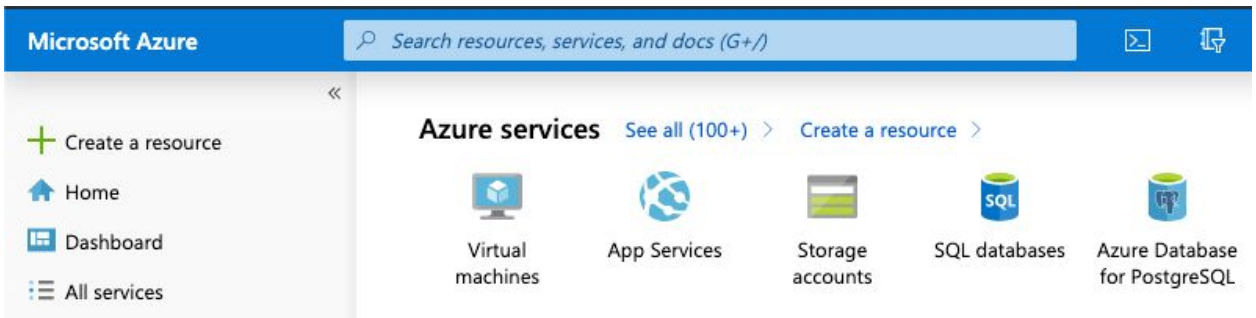


Microsoft Azure Cognitive Services

Azure Cognitive Services is used for the rest of the image analysis.

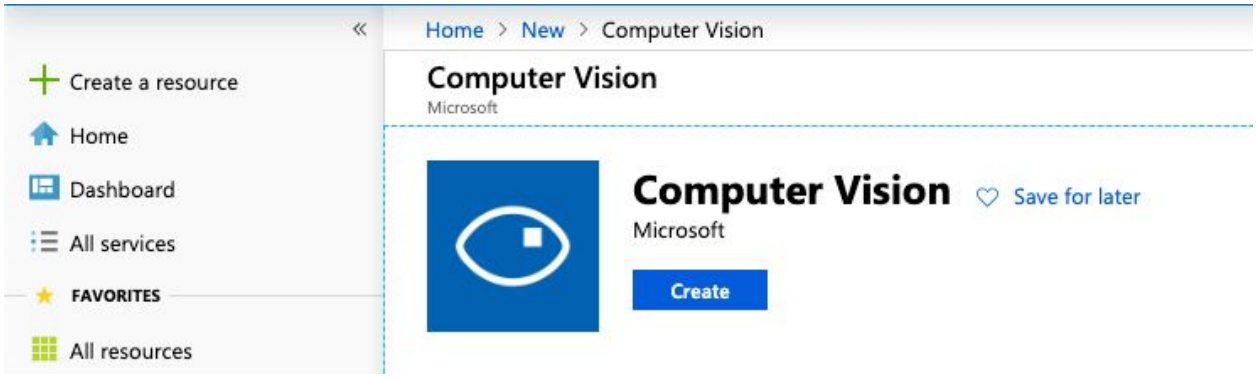
Computer Vision is used for the image analysis and object detection, and Face is used for face analysis.

Go to https://azure.microsoft.com/en-us/try/cognitive-services/ and log in with any existing Microsoft account, or create a new one for free. Once done, log in and you should be directed to the Azure portal.



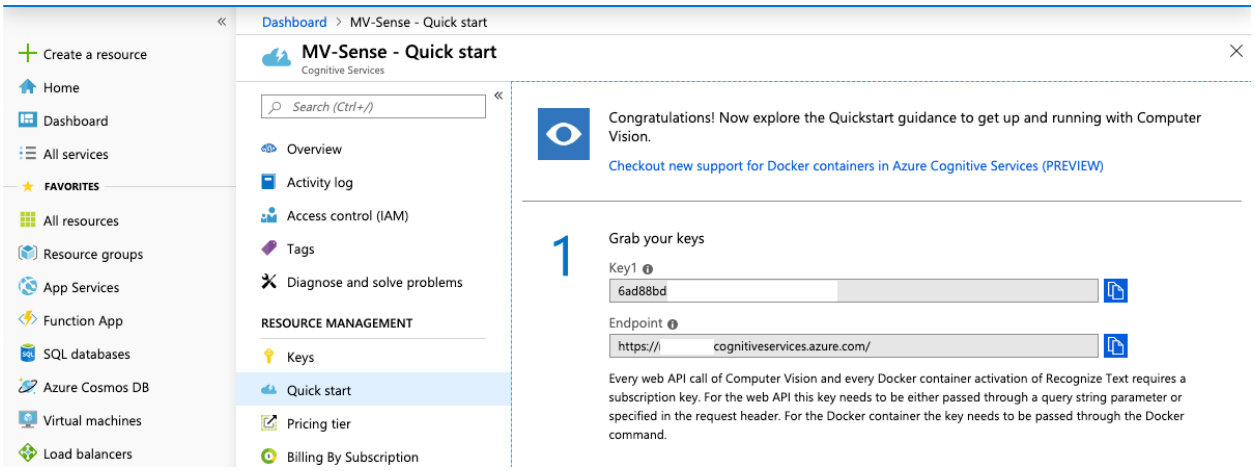
Computer Vision

Click “Create a resource”, then search for “Computer Vision” and click “Create”



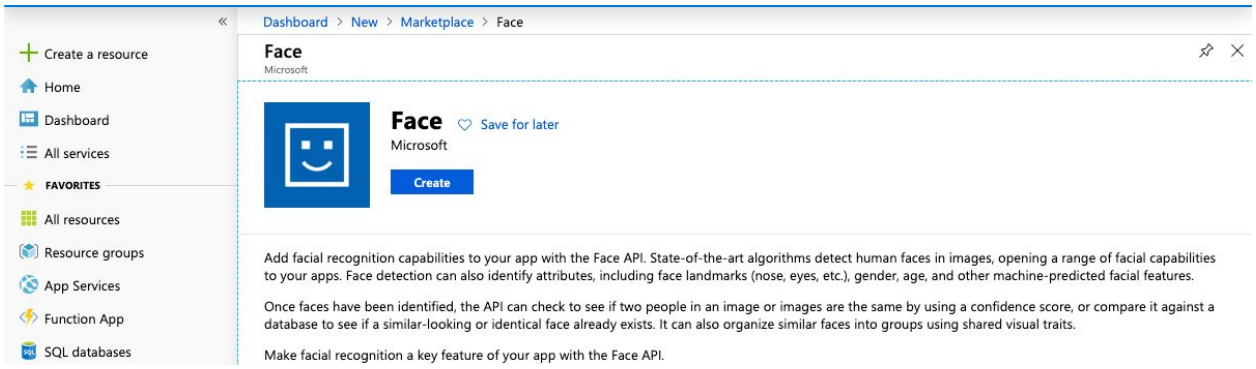
Give the new resource whatever name you'd like, make sure you're using "Free Trial" as the subscription and "F0" as the pricing tier. Pick a Location geographically close to you. Create a new Resource Group and give it a generic name. It should look like this:

Once done, navigate to Dashboard > (Your Computer Vision Resource) > Quick Start. Copy **Key1** and paste it in config.py as the **microsoftapikey** variable. Copy **Endpoint** and paste it in as **computervision_endpoint**.



Face

Similar to the last steps, click “Create a resource”, then search for “Face” and click “Create”. You can reuse the Resource Group created in the previous section.



Once done, navigate to Dashboard > (Your Face Resource) > Quick Start. Copy **Key1** and paste it in config.py as the **microsoftfaceapikey** variable. Copy **Endpoint** and paste it in as **face_endpoint**.

MQTT Setup

This app uses MQTT for the real-time person and vehicle detection, and therefore you'll need an MQTT broker configured.
***Note that your cameras must have L3 reachability to your MQTT broker*

Below are the setup commands for Ubuntu, for other operating systems see here: https://mosquitto.org/download/

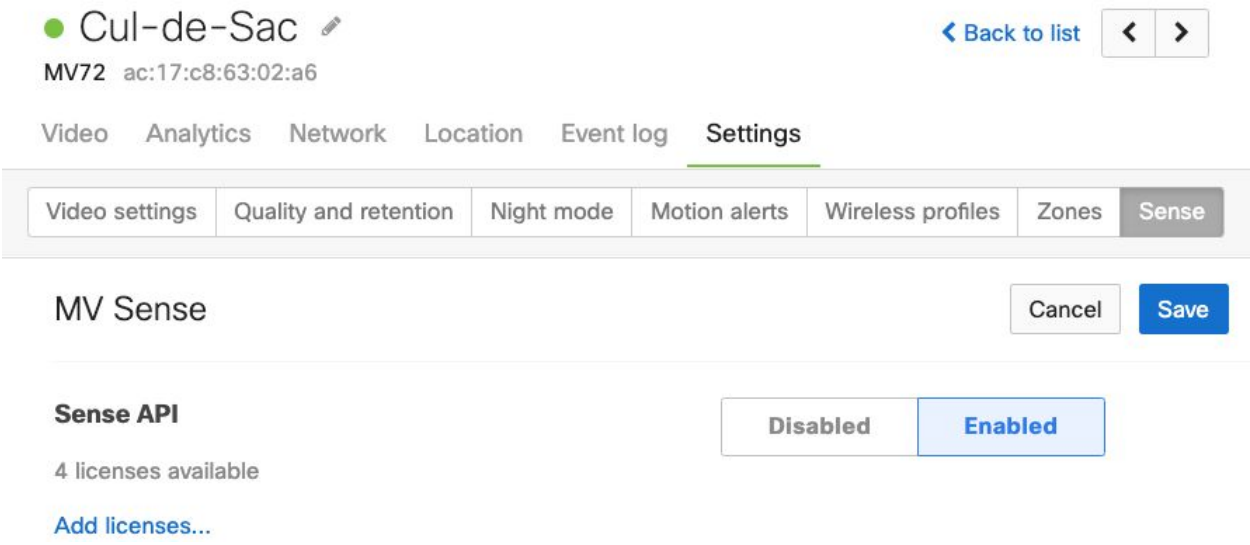
```
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients
```

Add the IP address of your MQTT server in **config.py** as **MQTT_SERVER**.

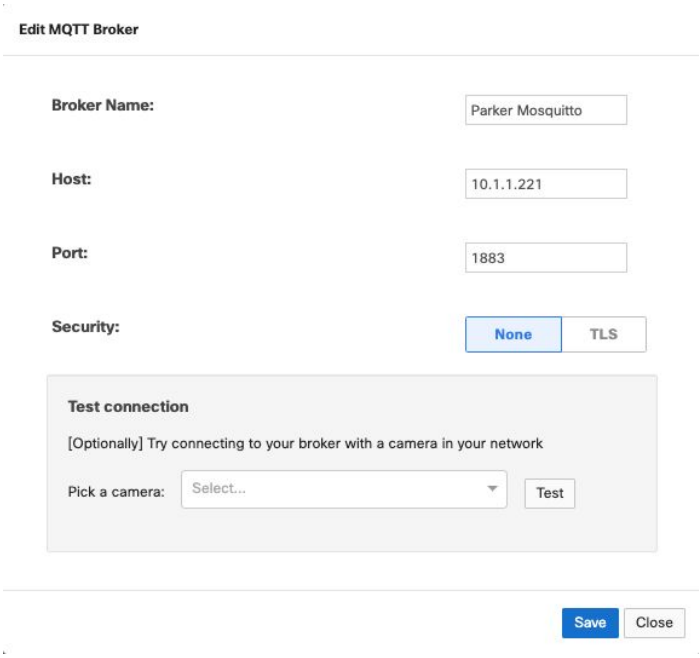
Camera Setup

For the cameras you'd like to use, first create at least one **Zone** for each under **Settings > Zones**.

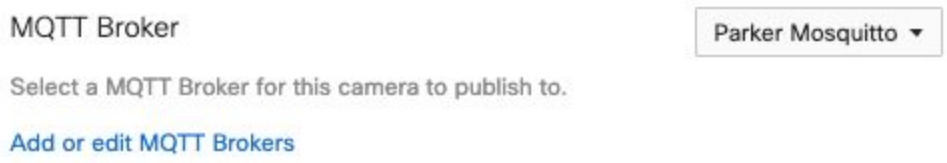
Next, enable MV Sense under **Settings > Sense**.



For the first camera, add an MQTT Broker, give it a friendly name, add the IP address of your MQTT server, and leave the rest as default:



For subsequent cameras, simply choose the MQTT broker you created from the Dropdown:



Finally, copy the Serial Number for the camera and add it to the **COLLECT_CAMERAS_SERIAL_NUMBERS** array, and add the Zone to the **COLLECT_ZONE_IDS array** in **config.py**

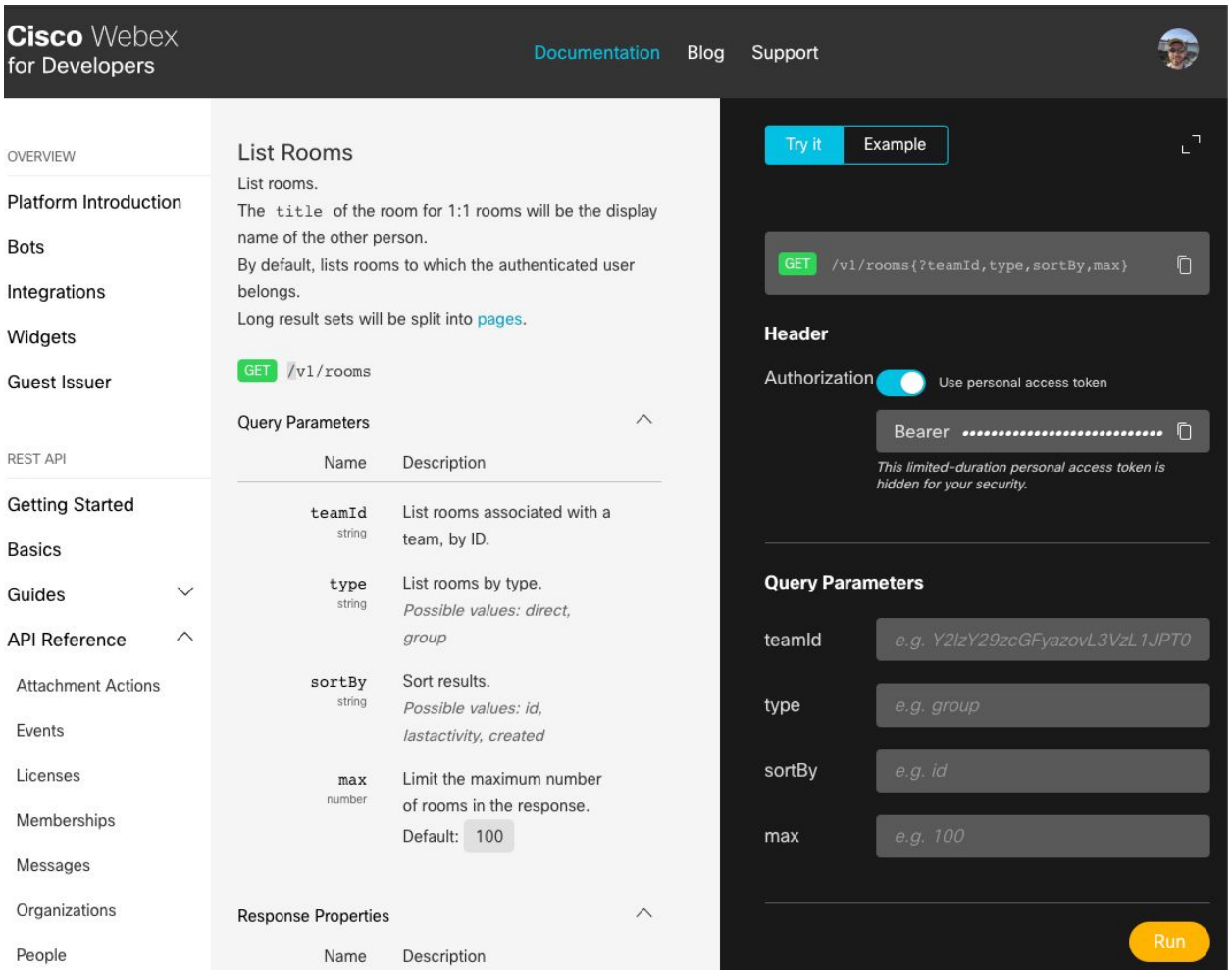
Once you've saved your MV Sense configuration, you can verify that messages are being received by your MQTT broker by running the following command on your MQTT server:

```
mosquitto_sub -t "#"
```

WebEx Teams Setup

In the WebEx Teams Client, create a new 1:1 or use an existing room, and add **mvnotify@webex.bot** as a participant.

Next, we need the Room ID so the script knows where to publish messages. Go to:
https://developer.webex.com/docs/api/v1/rooms/list-rooms
Log in with the same credentials you use for WebEx Teams, and run the “Try It” query:



Find your desired room, copy the ID and paste it in **config.py** as **ROOM_ID**. Leave the **WEBEXTEAMKEY** as is.

Final App Config

There are a few final things we need to tweak in **config.py**:

Enable or disable the integrations/features you want to use by setting the following values to “True” or “False”:

vehicle_detect
people_detect
image_detect
face_detect
lpr

Finally, set the **MOTION_ALERT_PAUSE_TIME** variable to the interval on how frequently you’d like to receive notifications. This is a very important variable, as it affects the rate at which you’ll receive messages in WebEx Teams, as well as the rate that you’ll consume image analyses in Azure or OpenALPR. For example, if a vehicle is parked in a given zone, you’ll get repeated notifications until that vehicle is moved or the script is exited.

Run the App

Now that all of the hard work is done, running the app is easy. From your Python computer, navigate to the folder with the app, and run:

```
python3 app.py
```

Miscellaneous Notes

- In order to reduce false positives, the script waits until 4 MQTT messages are received with people/vehicles detected before sending a notification.

- In addition, the MV Snapshot API returns the image URL before the image is actually available, so the script pauses for 5 seconds after grabbing the snapshot before sending the notification. Because of this, you should expect a 5-7 second delay between an object detected and the notification being received.

- The script will run forever until quit, which can be forced with Ctrl + C on most operating systems.

- A Docker container is available, which simplifies the installation, dependency requirements, and start/stop of the script. For setup instructions, please contact me at nwiens@cisco.com