IOT Lab Assignment 5

Flask restful api

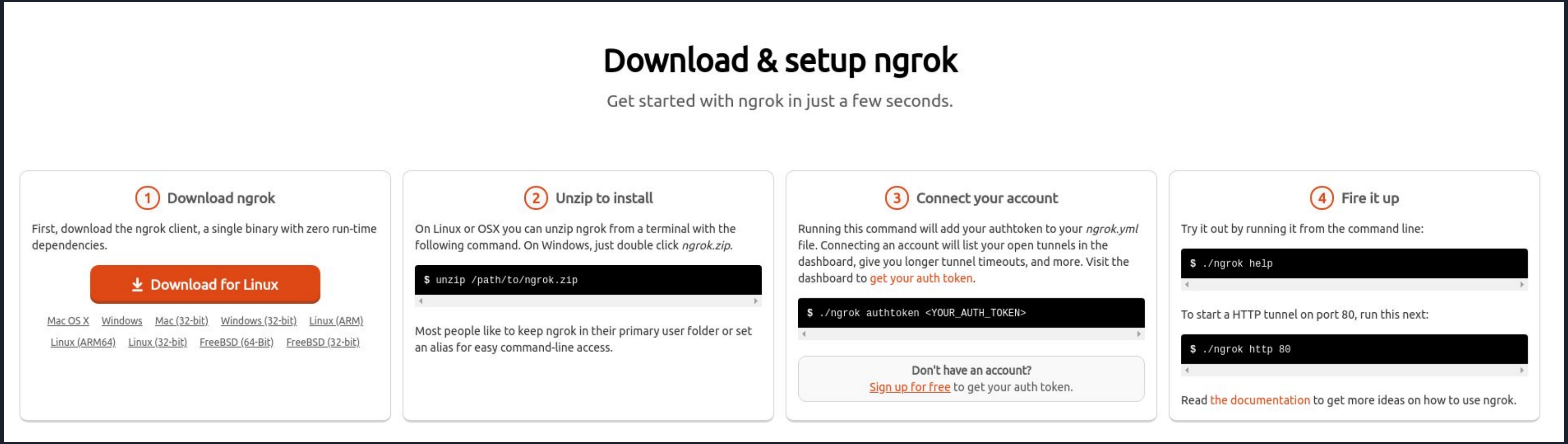
Flask (restful api)

- A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

- sudo pip3 install flask flask-restful

Ngrok (get your pi a public url for your api)

- You will need to make a free account
- https://ngrok.com



Download & setup ngrok

Get started with ngrok in just a few seconds.

1 Download ngrok

First, download the ngrok client, a single binary with zero run-time dependencies.

2 Unzip to install

On Linux or OSX you can unzip ngrok from a terminal with the following command. On Windows, just double click *ngrok.zip*.

```
$ unzip /path/to/ngrok.zip
```

Most people like to keep ngrok in their primary user folder or set an alias for easy command-line access.

3 Connect your account

Running this command will add your authtoken to your *ngrok.yml* file. Connecting an account will list your open tunnels in the dashboard, give you longer tunnel timeouts, and more. Visit the dashboard to get your auth token.

```
$ ./ngrok authtoken <YOUR_AUTH_TOKEN>
```

Don't have an account?
Sign up for free to get your auth token.

4 Fire it up

Try it out by running it from the command line:

```
$ ./ngrok help
```

To start a HTTP tunnel on port 80, run this next:

```
$ ./ngrok http 80
```

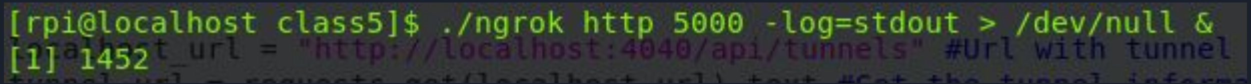
Read the documentation to get more ideas on how to use ngrok.

ngrok

1. Sudo dnf install unzip
2. curl --silent --remote-name --location -k https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-arm.zip
3. unzip ngrok-stable-linux-arm.zip
4. ./ngrok authtoken {auth token}

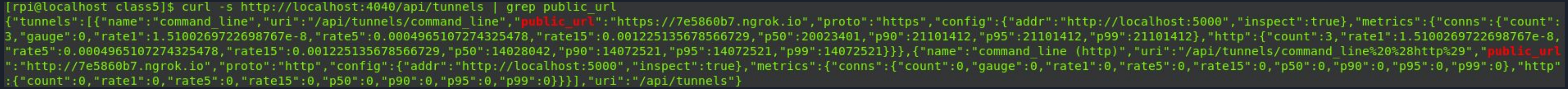
Run Ngrok, Get url, kill Ngrok

- ./ngrok http 5000 -log=stdout > /dev/null &



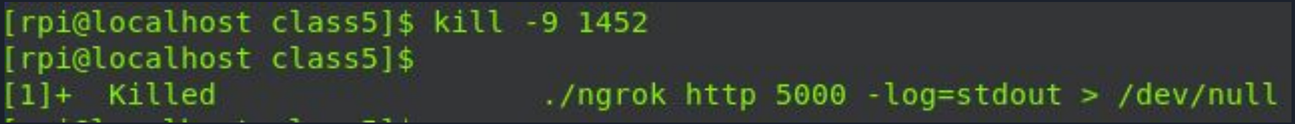
```
[rpi@localhost class5]$ ./ngrok http 5000 -log=stdout > /dev/null &
[1] 1452
```

- curl -s http://localhost:4040/api/tunnels | grep public_url (look for https)



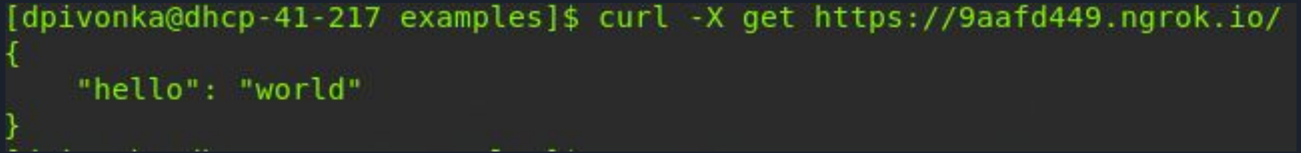
```
[rpi@localhost class5]$ curl -s http://localhost:4040/api/tunnels | grep public_url
{"tunnels":[{"name":"command_line","uri":"/api/tunnels/command_line","public_url":"https://7e5860b7.ngrok.io","proto":"https","config":{"addr":"http://localhost:5000","inspect":true},"metrics":{"conns":{"count":
3,"gauge":0,"rate1":1.5100269722698767e-8,"rate5":0.0004965107274325478,"rate15":0.001225135678566729,"p50":20023401,"p90":21101412,"p95":21101412,"p99":21101412},"http":{"count":3,"rate1":1.5100269722698767e-8,
"rate5":0.0004965107274325478,"rate15":0.001225135678566729,"p50":14028042,"p90":14072521,"p95":14072521,"p99":14072521}}},{"name":"command_line (http)","uri":"/api/tunnels/command_line%20%28http%29","public_url
":"http://7e5860b7.ngrok.io","proto":"http","config":{"addr":"http://localhost:5000","inspect":true},"metrics":{"conns":{"count":0,"gauge":0,"rate1":0,"rate5":0,"rate15":0,"p50":0,"p90":0,"p95":0,"p99":0},"http"
:{"count":0,"rate1":0,"rate5":0,"rate15":0,"p50":0,"p90":0,"p95":0,"p99":0}}}],"uri":"/api/tunnels"}
```

- Kill -9 1452 (if you want to stop ngrok)



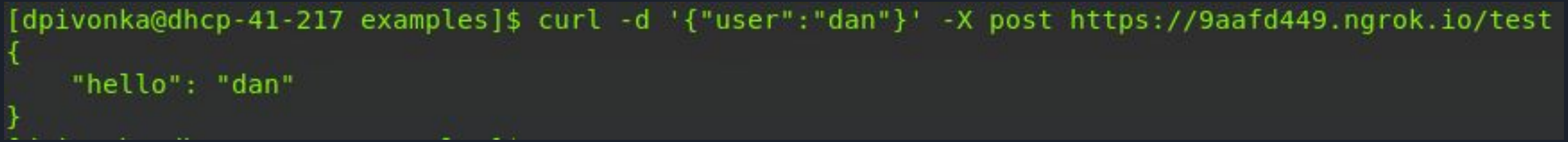
```
[rpi@localhost class5]$ kill -9 1452
[rpi@localhost class5]$
[1]+  Killed                  ./ngrok http 5000 -log=stdout > /dev/null
```

Helloworld.py



```
[dpivonka@dhcp-41-217 examples]$ curl -X get https://9aafd449.ngrok.io/
{
    "hello": "world"
}
```

Hellouser.py

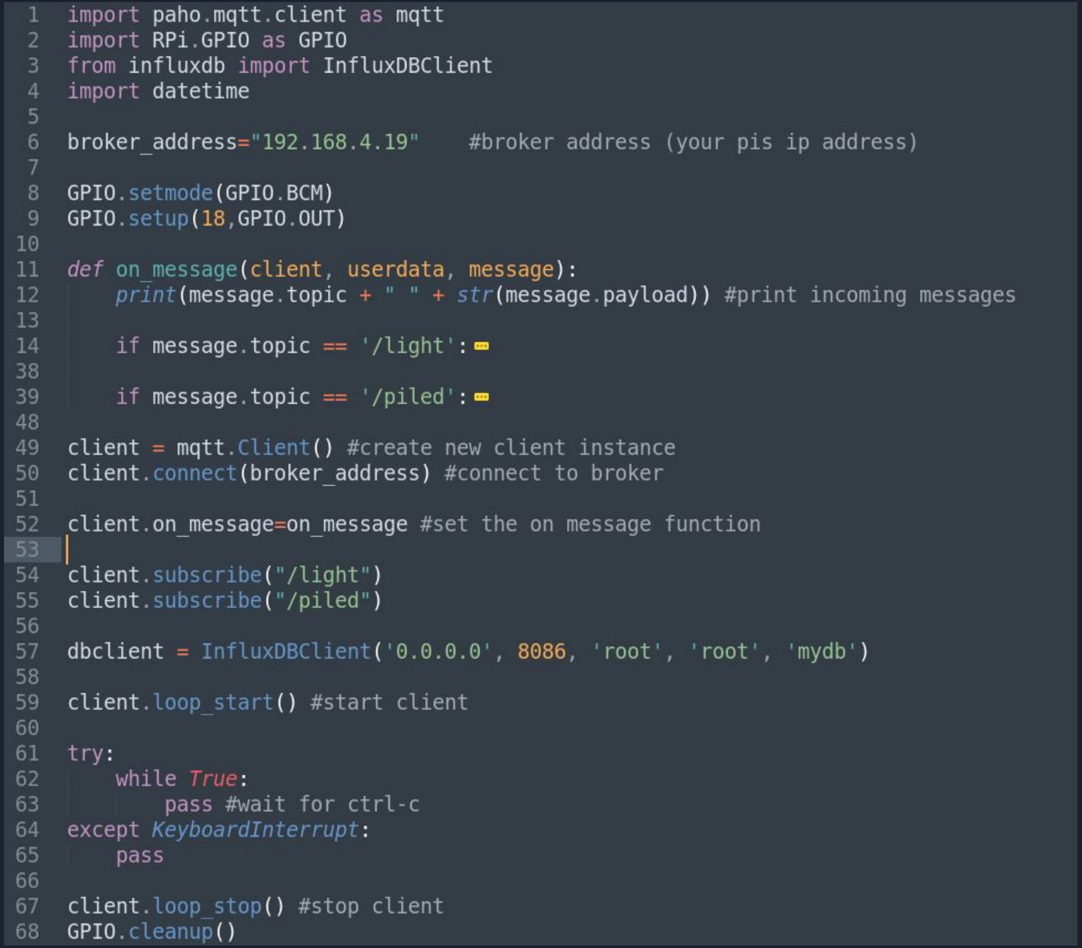


```
[dpivonka@dhcp-41-217 examples]$ curl -d '{"user":"dan"}' -X post https://9aafd449.ngrok.io/test
{
    "hello": "dan"
}
```

Broker

Broker is responsible for handling incoming light sensor data and storing it in the database

It is also responsible for handling incoming messages to turning on and off pi light

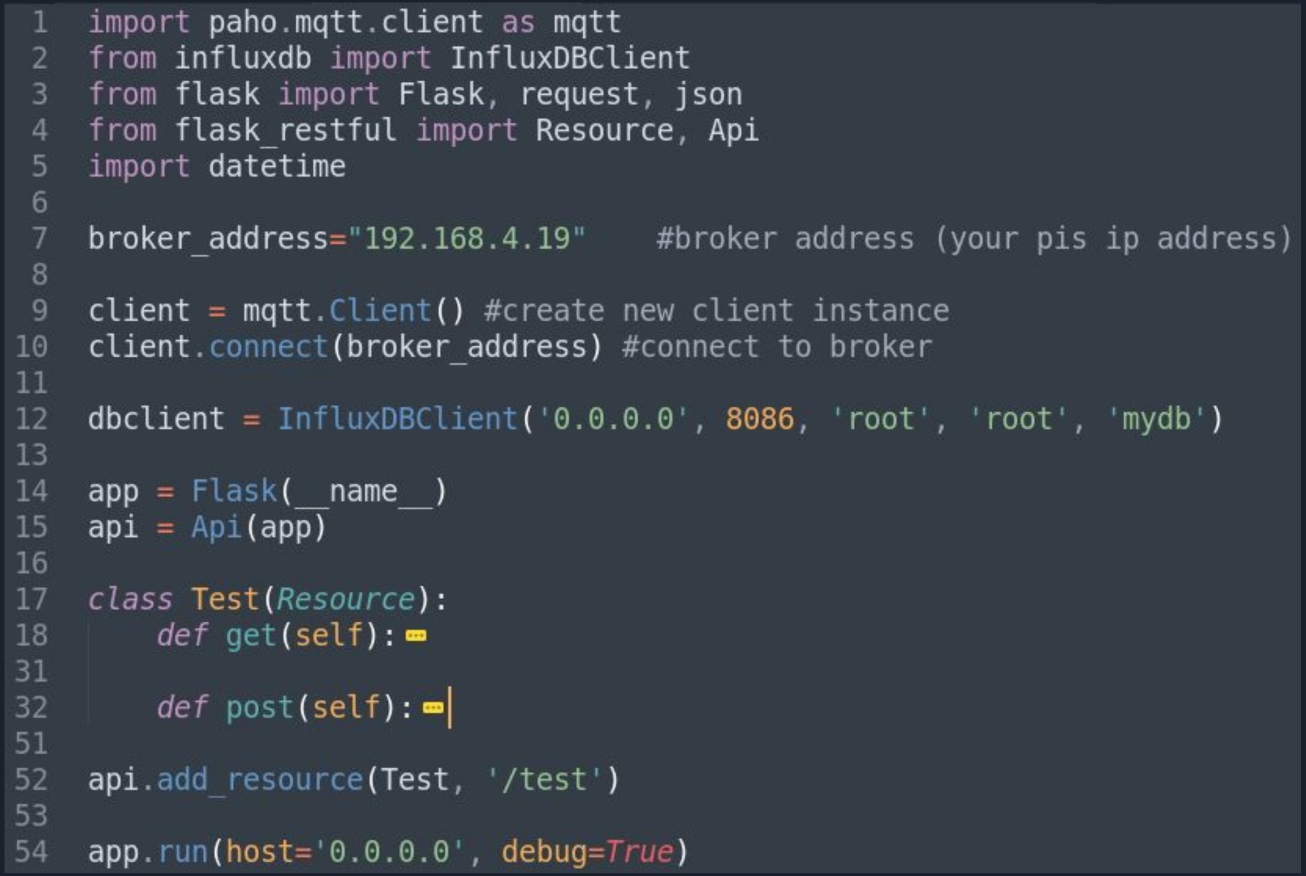


```python
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
from influxdb import InfluxDBClient
import datetime

broker_address="192.168.4.19"    #broker address (your pis ip address)

GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)

def on_message(client, userdata, message):
    print(message.topic + " " + str(message.payload)) #print incoming messages

    if message.topic == '/light':

    if message.topic == '/piled':

client = mqtt.Client() #create new client instance
client.connect(broker_address) #connect to broker

client.on_message=on_message #set the on message function

client.subscribe("/light")
client.subscribe("/piled")

dbclient = InfluxDBClient('0.0.0.0', 8086, 'root', 'root', 'mydb')

client.loop_start() #start client

try:
    while True:
        pass #wait for ctrl-c
except KeyboardInterrupt:
    pass

client.loop_stop() #stop client
GPIO.cleanup()
```

API

API is responsible for querying the db for light average and returning that value

It is also responsible for sending mqtt messages out to turn on leds (these messages go to the esp and to the broker)



```python
import paho.mqtt.client as mqtt
from influxdb import InfluxDBClient
from flask import Flask, request, json
from flask_restful import Resource, Api
import datetime

broker_address="192.168.4.19"    #broker address (your pis ip address)

client = mqtt.Client() #create new client instance
client.connect(broker_address) #connect to broker

dbclient = InfluxDBClient('0.0.0.0', 8086, 'root', 'root', 'mydb')

app = Flask(__name__)
api = Api(app)

class Test(Resource):
    def get(self):

    def post(self):

api.add_resource(Test, '/test')

app.run(host='0.0.0.0', debug=True)
```

Assignment

- Send light sensor values to pi from esp8266
- Store those values in influxdb
- Create restful api
- Post endpoint sets state of led on pi or arduino {‘device’:’pi’, ‘state’:’on”}
- Get endpoint returns avg light value over last 10 secs

Due by next class

Assignment video