Version 3

Cloud, APIs and Alerts > Social Media of Things

Coding with Tweepy to Post alerts

Let's build the code to post the tweets whenever the temperature crosses a threshold.

Step 1: Install the Tweepy library on your DigitalOcean droplet with the following command.

```
sudo pip3 install tweepy
```

Step 2: Store all the secret credentials for accessing the Twitter App we obtained in the previous lesson and then save the file with name `conf.py` . This file contains all our App configuration data. You can find all your Twitter App related credentials here (https://apps.twitter.com). Select the Twitter App you created in the previous lesson. You can view the credentials required below under the `Keys and Access Tokens` . We store all the credentials in a separate file since it is sensitive data which should not be shared with anyone. Hence it is a good practice to avoid using credentials in code directly. We shall also store the Bolt Device ID and Bolt Cloud API key in this file.

```
consumer_key = "Consumer Key (API Key) for Twitter App"
consumer_secret = "Consumer Secret (API Secret) for Twitter App"
access_token = "Access Token for Twitter App"
access_token_secret = "Access Token Secret for Twitter App"
bolt_cloud_api_key = "API key of your Bolt Cloud. You can find in API section on Bo
device_id = "ID of your Bolt Device"
```

Step 3: Let us write and understand the code for the connecting to Twitter and posting a tweet. The first step is to import the `tweepy` library and the `conf` file containing the credentials into our python file.

```
    import tweepy
    import conf
```

Step 4: We shall now create a dictionary in our program so that we can pass this data to the Tweepy library. A dictionary is a special data type used in python to access data based on key-value pairs. In the code below `config` is dictionary variable with credentials data. To access the consumer_key value we can just use the key "consumer_key" as `config['consumer_key']` .

```
config = {
"consumer_key" : conf.consumer_key,
"consumer_secret" : conf.consumer_secret,
"access_token" : conf.access_token,
"access_token_secret" : conf.access_token_secret
    }
```

Step 5: Before we can post a tweet, Twitter API makes it compulsory to authenticate that the request to post the tweet is coming from the owner of the account or a person having the access credentials. We can use the

`OAuthHandler` and the `set_access_token` methods provided by Tweepy to get the authenticated API object. Using this object, we can post the tweet. We shall create a method called `get_api_object` which has `cfg` dictionary as an input parameter and help us do the authentication process and return the API object.

```
def get_api_object(cfg):
    auth =tweepy.OAuthHandler(cfg['consumer_key'],
                              cfg['consumer_secret'])
    auth.set_access_token(cfg['access_token'],
                          cfg['access_token_secret'])
    return tweepy.API(auth)
```

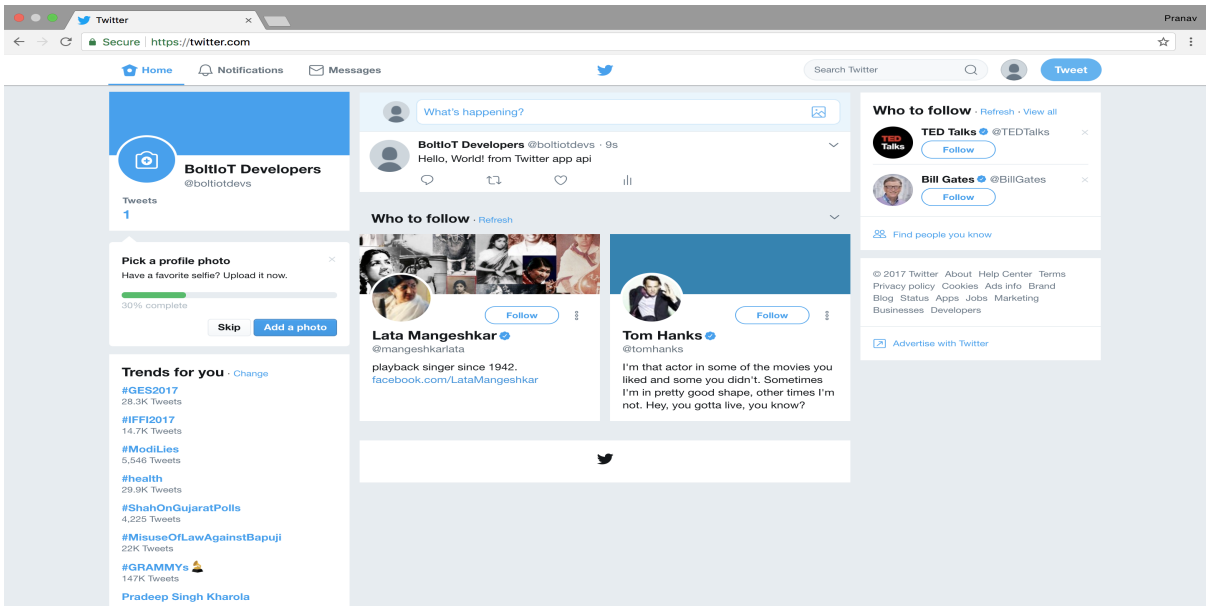
Step 6: Once we have the API object, we can use it to post a tweet on our Twitter account. We shall call the `get_api_object` method and pass the `config` dictionary to get the Twitter API object. The API object contains the `update_status` method, which accepts the tweet to be posted in a string format. It's weird that Twitter calls the tweet as an update in their API. We shall store the tweet message in our `tweet` variable and pass this variable to the `update_status` method.

```
api_object = get_api_object(config)
tweet = "Hello, world! from Tweeter app api"
status = api_object.update_status(status=tweet)
```

Step 7: Here is how the code will look when combined. Save the code in a file with the `.py` extension i.e. `post-tweet.py` .

```
import tweepy
import conf

    # Dictionary to store credentials as key-value pairs.
config = {
"consumer_key"        : conf.consumer_key,
"consumer_secret"     : conf.consumer_secret,
"access_token"        : conf.access_token,
"access_token_secret" : conf.access_token_secret
}
# Method to authenticate user via Tweepy and return API object
def get_api_object(cfg):
    auth =tweepy.OAuthHandler(cfg['consumer_key'],
                              cfg['consumer_secret'])
    auth.set_access_token(cfg['access_token'],
                          cfg['access_token_secret'])
    return tweepy.API(auth)

# Call get_api_object to authenticate user and get the API object
api_object = get_api_object(config)

# Store the tweet message in the variable
tweet = "Hello, World! From Tweeter App Api"

# Post the tweet on your Twitter account using the update_status method.
status = api_object.update_status(status=tweet)
```

You can run the above code using the `python post-tweet.py` command and view your Twitter Profile to see the tweet.



Step 8: We shall now do the hardware connections so that we can post a tweet everytime the temperature values crosses a threshold. Follow the steps as shown in the previous topic for Hardware Connections of Temperature Sensor.

Step 9: Once you are done with the connections, we shall add the logic to check if the temperature sensor value has exceeded the threshold. We shall import the boltiot library in our code.

```
from boltiot import Bolt
```

Step 10: We shall now create an object for our Bolt Device so that we can access the temperature sensor value connected to it.

```
mybolt = Bolt(conf.bolt_cloud_api_key, conf.device_id)
```

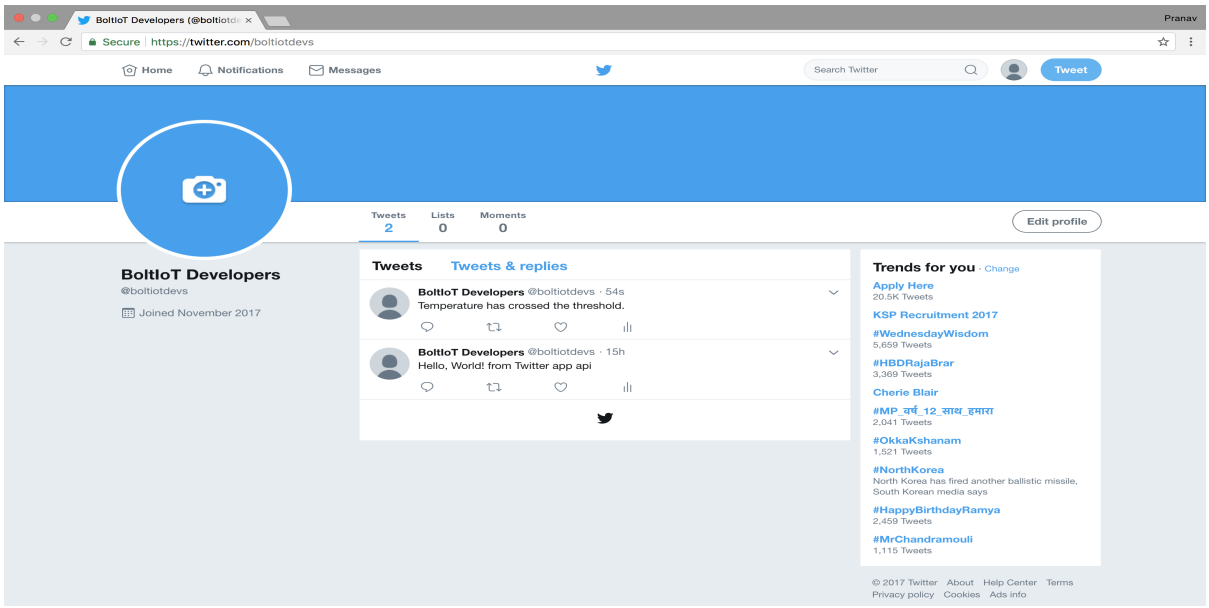
Step 11: We shall now continuously fetch the sensor value connected to pin `A0` using the `analogRead(A0)` method of the Bolt object. We have used the `time.sleep(10)` command to fetch the sensor value at an interval of every 10 seconds. We have also set the temperature threshold value to 59 using the `temperature_threshold` variable. You can modify this value as per the temperature at your place. Once we get the JSON response from `analogRead(A0)`, we load the JSON response into `data` variable so that we can access the sensor value as `data['value']`. We convert this data value into integer format and store it in the `sensor_value` variable and compare if the value is greater than the threshold. If it is greater than the threshold, we shall add the code to post the tweet here. Here's the logic for fetching the temperature data and comparing it with the threshold.

```
    temperature_threshold = 59
    while True:
        response = mybolt.analogRead('A0')
        data = json.loads(response)
        print (data['value'])
        try:
            sensor_value = int(data['value'])
            if sensor_value > temperature_threshold:
                print "Temperature has crossed the threshold."
        except Exception as e:
            print ("An error occurred ", e)
        time.sleep(10)
```

Here is how the code will look after we combine the sensor value comparison code with the twitter code. Don't forget to save the file.

```
import tweepy
import conf, json, time
from boltiot import Bolt

# Dictionary to store credentials as key-value pairs.
config = {
"consumer_key"    : conf.consumer_key,
"consumer_secret"     : conf.consumer_secret,
"access_token"        : conf.access_token,
"access_token_secret" : conf.access_token_secret
}

# Method to authenticate user via Tweepy and return API object
def get_api_object(cfg):
    auth =tweepy.OAuthHandler(cfg['consumer_key'],
                                  cfg['consumer_secret'])
    auth.set_access_token(cfg['access_token'],
                              cfg['access_token_secret'])
    return tweepy.API(auth)

mybolt = Bolt(conf.bolt_cloud_api_key, conf.device_id)
temperature_threshold = 59
while True:
    response = mybolt.analogRead('A0')
    data = json.loads(response)
    print (data['value'])
    try:
        sensor_value = int(data['value'])
        if sensor_value > temperature_threshold:
            print "Temperature has crossed the threshold."
            # Call get_api_object to authenticate user and get the API object
            api_object = get_api_object(config)
            # Store the tweet message in the variable
            tweet = "Temperature has crossed the threshold."
            # Post the tweet on your Twitter account using the update_status method
            status = api_object.update_status(status=tweet)
    except Exception as e:
        print ("An error occurred ", e)
    time.sleep(10)
```

Run the code using the `python post-tweet.py` command and monitor your Twitter feed for any tweets when the temperature crosses the threshold. The tweets will be displayed on your Twitter profile as shown below.



We have now learned how post alerts on Twitter everytime our temperature crosses a threshold. You can use this knowledge to build more interesting projects and integrate your Bolt with other Social Media sites.

PREV NEXT

Forum

Type your query here SEARCH

Q. How do I cancel the subscription on DigitalOcean, so that it does not charge my card?

Login to the DigitalOcean dashboard,
Click on the icon in the top right corner.
The Menu will drop down from the icon.
Select settings.
The settings page will open up.
Click on the deactivate account button which is the lowermost option on the page

see more ()

Q. I am not able to sign into the Twilio account, it is like I have to solve the CAPTCHA in order to sign in? What is meant by CAPTCHA?

Here is a video which will explain to you what is a CAPTCHA: https://www.youtube.com/watch?v=MWu2UiLLJI8

Q. How long does the temp_sms.py code send an sms? Does it send sms only once, or does it send sms everytime the temperature goes beyond bounds?

The code will send you an sms everytime the temperature goes beyond its bounds.
However, if you execute the command
sudo python temp_sms.py
You will need to stay logged into the digital Ocean droplet for the code to keep monitoring the temperature and send you an sms.

see more ()

Q. When I execute temp_email it is showing unexpected indent error. How do I fix this?

In python, the leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements.
The levels number of white spaces is known as an indentation level.
If you are facing unexpected indent error, it probably means that the white spaces before and after a logical grouping of instructions is different without having any function declaration or any kind of

see more ()

Q. I am getting an indentation error. How do I solve it?

This means that you have not given proper indentation i.e. spaces before the start of each line of code. Do check the code given as part of the tutorial carefully and make suitable changes. I suggest you read the course content again to be sure that there are no errors.

View more

Ask your doubts to the instructor personally if they are not already answered on forum.

Type what you want to ask the instructor

ASK THE INSTRUCTOR