Creating Whatsapp Weather Forecast with Twilio Function and Assets

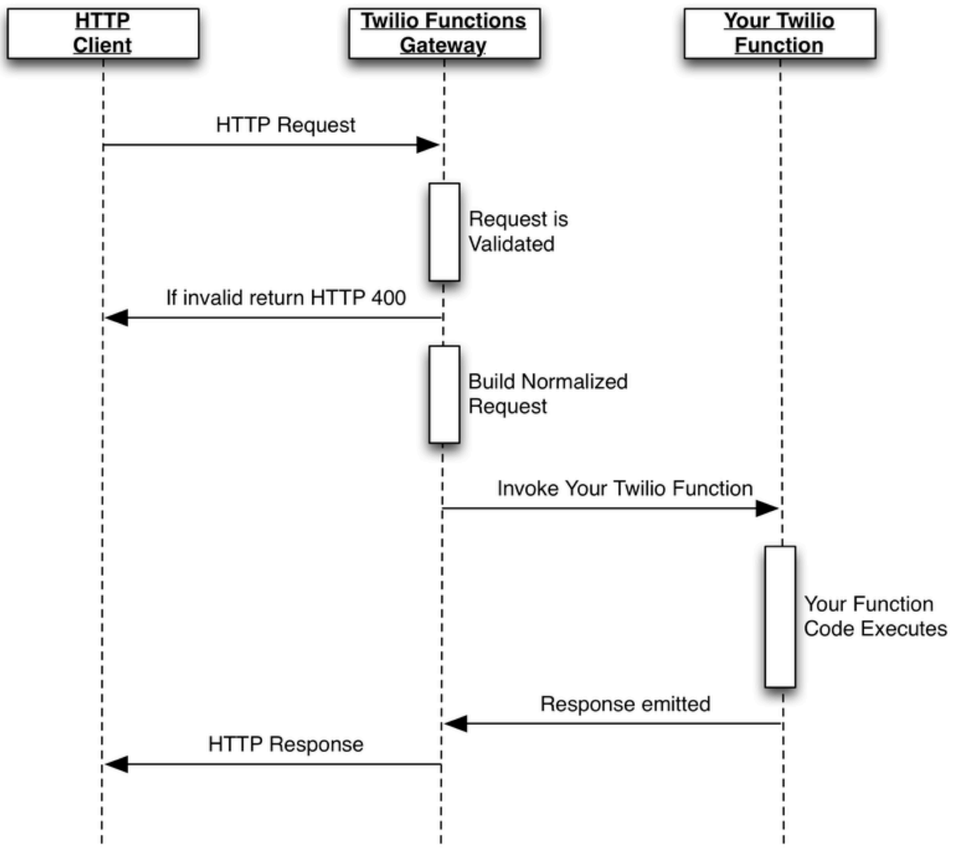
Created by Max NG – 13 Jan 2019

(Please note, at the time of writing, I do not work for Twilio)

Creating a Whatsapp Weather Forecast

Extending Whatsapp messenging with the ability to obtain weather forecast based on Area. The app extend Twilio Whatsapp communication API with Function and Assets. Function is the runtime application logic using nodejs, and Assets stores the daily weather datasource. The weather datasource is obtained from published NEA website. At the time of writing, Twilio Assets does not support API for updating the files and this can only be done via the Twilio Console.

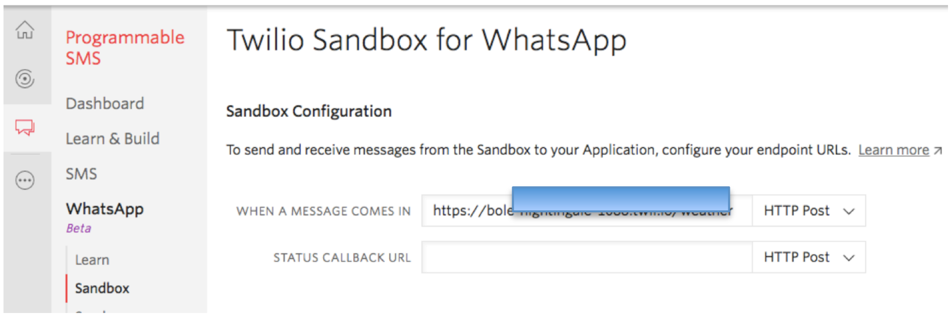
The same application flow can be replicated on other Cloud Platform that have storage and runtime container. Twilio Whatsapp communication API has a webhook that can perform event trigger to external HTTP URL. The below flow diagram taken from Twilio documentation illustrates the flows of the application.



The application logic is hosted in Twilio Function, that can be invoked whenever there is a HTTPS request. Using either a direct HTTPS request or via Twilio programmable Whatsapp or SMS messenging to invoke the Function.
The datasource of the weather report is hosted on Twilio Assets not shown on this diagram.

Twilio current pricing at the time of writing this document for Function is based on 10,000 free invocations per month. Check out the pricing on the website.

The rest of this document assumes familiarity with Twilio console and how to setup a programmable SMS/Whatsapp. Using the same account that was setup previously, in Twilio Sandbox for WhatsApp, change the sandbox configuration to point the WhatsApp account to a new URL when a new message arrives.



Twilio Sandbox for WhatsApp

Sandbox Configuration

To send and receive messages from the Sandbox to your Application, configure your endpoint URLs. Learn more ↗

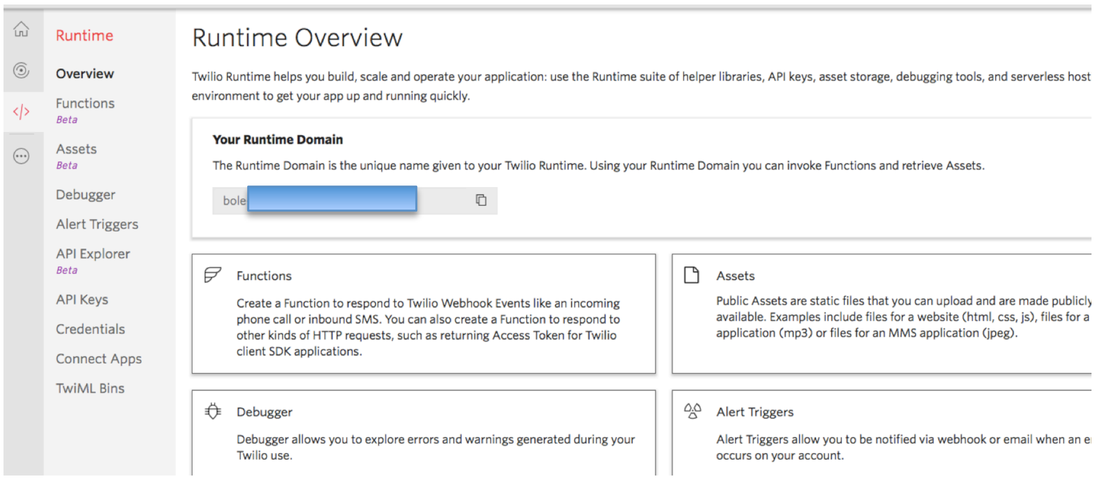
WHEN A MESSAGE COMES IN https://bole[redacted] HTTP Post

STATUS CALLBACK URL HTTP Post

The HTTPS URL will be the location of the Twilio Function that will host the application logic.

Note that to use WhatsApp message, your device's WhatsApp number will have to be registered with the Twilio Sandbox. If this has not already been done, you can do so following the instruction listed under the WhatsApp console.

Now, toggle to Twilio Function. On the left panel, select ***All Products***, and then ***Runtime***. You will be presented with a menu of all available Runtime developer tools.



Runtime Overview

Twilio Runtime helps you build, scale and operate your application: use the Runtime suite of helper libraries, API keys, asset storage, debugging tools, and serverless host environment to get your app up and running quickly.

Your Runtime Domain

The Runtime Domain is the unique name given to your Twilio Runtime. Using your Runtime Domain you can invoke Functions and retrieve Assets.

bole[redacted]

Functions

Create a Function to respond to Twilio Webhook Events like an incoming phone call or inbound SMS. You can also create a Function to respond to other kinds of HTTP requests, such as returning Access Token for Twilio client SDK applications.

Assets

Public Assets are static files that you can upload and are made publicly available. Examples include files for a website (html, css, js), files for a application (mp3) or files for an MMS application (jpeg).

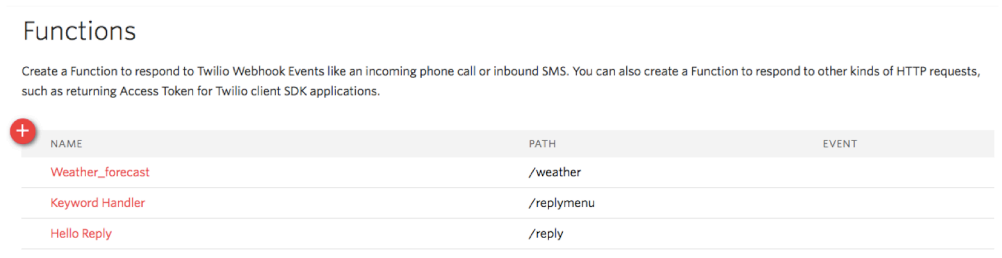
Debugger

Debugger allows you to explore errors and warnings generated during your Twilio use.

Alert Triggers

Alert Triggers allow you to be notified via webhook or email when an e occurs on your account.

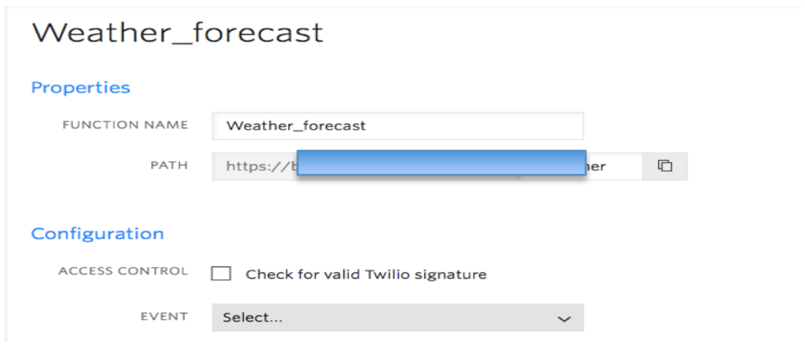
Select Functions from the menu. Add a new Function, and select a blank template. Note the domain of this new Function, this will be the URL that you will input in the earlier WhatsApp console for the webhook. Add a unique path for this particular function. The complete domain and path name will be the URL to invoke the Function. I called my Function "***Weather_forecast***".


Functions

Create a Function to respond to Twilio Webhook Events like an incoming phone call or inbound SMS. You can also create a Function to respond to other kinds of HTTP requests, such as returning Access Token for Twilio client SDK applications.

NAME	PATH	EVENT
Weather_forecast	/weather	
Keyword Handler	/replymenu	
Hello Reply	/reply	

For testing purposes, unchecked the ticked box to Access Control – Check for valid Twilio signature. For quick testing, you can send a cURL message to the same URL to emulate a whatsapp message send to Twilio. This can be easier to trigger when Twilio platform does not check the valid signature token by sending a https request direct to the function as shown in earlier diagram.


Weather_forecast

Properties

FUNCTION NAME: Weather_forecast

PATH: https://[redacted]er

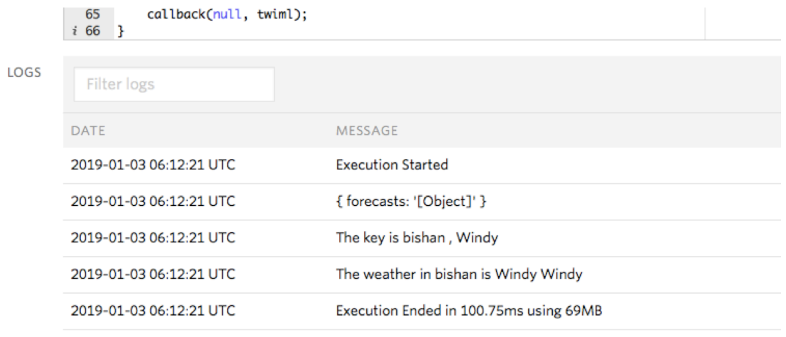
Configuration

ACCESS CONTROL: ☐ Check for valid Twilio signature

EVENT: Select...

The full nodejs code for this function is in weather.js, you can download it off Github.

Few things to take notes, console.log, will output the message to console below the Twilio Function console. This can be helpful for any debugging needed. It would be nice if the log can be stored and retrieve externally.


```
    callback(null, twiml);
}
```

LOGS

DATE	MESSAGE
2019-01-03 06:12:21 UTC	Execution Started
2019-01-03 06:12:21 UTC	{ forecasts: '[Object]' }
2019-01-03 06:12:21 UTC	The key is bishan , Windy
2019-01-03 06:12:21 UTC	The weather in bishan is Windy Windy
2019-01-03 06:12:21 UTC	Execution Ended in 100.75ms using 69MB

At the time of writing this document, Twilio Function has a maximum of 5seconds runtime. If your function runs for more than 5seconds, it will be auto terminated. Calling a function within a function is supported but the primary function still has the 5second window.

This sets up the WhatsApp messenging to trigger the ***Weather_forecast*** Function via webhook (when a message arrives). From the code, we have a data source of the weather forecasts. This is stored in Twilio Assets.

```
  let file = Runtime.getAssets()['weather.json'].open(); //Opening the JSON
                       file stored in Twilio Assets
```

The source of this data source file is a JSON file. This is obtained from NEA public website that published the weather forecast in different areas in Singapore. API to obtain this supported and can details can be obtained from NEA website developer page.

Using cURL a dump of the JSON data can be obtained

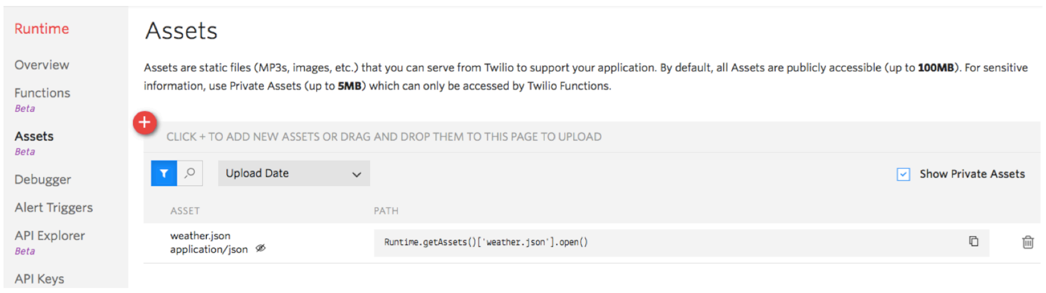
```
curl -X GET -o weather1.json
https://api.data.gov.sg/v1/environment/2-hour-weather-forecast -H
"accept: application/json"
```

This will output the complete JSON data from NEA website, which has more datasets than is required for this project. For the purpose of this document, this is currently retrieved, reformatted and upload to Twilio Assets. Uploading of the JSON file currently is only supported from Twilio Console.

I called this file weather.json and the format of this file looks like the below

```
{"forecasts":[
        {"area":"ang mo kio","forecast":"Windy"},
        {"area":"bedok","forecast":"Windy"},
        {"area":"bishan","forecast":"Windy"},
        {"area":"boon lay","forecast":"Windy"},
        {"area":"bukit batok","forecast":"Windy"},
        {"area":"bukit merah","forecast":"Windy"},
        {"area":"bukit panjang","forecast":"Windy"},
        {"area":"bukit timah","forecast":"Windy"},
```

When you download the JSON data from NEA, it is in JSON string, I've reformat in JSON format. You could write a parser to perform this. To validate the file in JSON format, you can check it using https://jsonlint.com



Assets

Assets are static files (MP3s, images, etc.) that you can serve from Twilio to support your application. By default, all Assets are publicly accessible (up to **100MB**). For sensitive information, use Private Assets (up to **5MB**) which can only be accessed by Twilio Functions.

CLICK + TO ADD NEW ASSETS OR DRAG AND DROP THEM TO THIS PAGE TO UPLOAD

Upload Date

[✓] Show Private Assets

ASSET	PATH
weather.json application/json	`Runtime.getAssets()['weather.json'].open()`

Once you have the data source uploaded to Twilio Assets, you are set. We have made this asset to be private, so you will not be able to access it outside of Twilio function. If this is a public asset, you can access it using HTTPS GET. The domain is similar to your Twilio Function with paths having /assets/file-name

Testing your Weather Forecast Application

Now that we have the application setup, its time to perform a quick test. Remember the application flow diagram. We can invoke the Function either by sending a HTTPS request or a WhatsApp message to the Twilio Sandbox. We will test both in stages.

Sending a HTTPS request direct to Twilio Function.

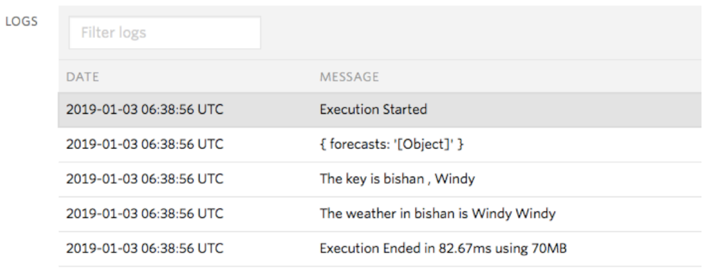
Remember our Twilio Function URL (domain and path). We will use that to invoke the function. Our code examines the HTTP body and matches it against the JSON key-value array.

Sending a HTTPS POST request with JSON data with "***Bishan***" as the key

```
curl -v -X POST  https://bole-nightingale-xxxx.twil.io/weather-H "Content-Type: application/json" -d
"{\"Body\":\"Bishan\"}"

  Trying 18.210.141.xx...
* Connected to bole-nightingale-xxxx.twil.io (18.210.141.xx) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
* Server certificate: *.twil.io
* Server certificate: Amazon
* Server certificate: Amazon Root CA 1
* Server certificate: Starfield Services Root Certificate Authority - G2
> POST /weather HTTP/1.1
> Host: bole-nightingale-xxxx.twil.io
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 17
>
* upload completely sent off: 17 out of 17 bytes
< HTTP/1.1 200 OK
< Date: Thu, 03 Jan 2019 06:38:21 GMT
< Content-Type: text/xml; charset=utf8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Shenanigans: none
< Strict-Transport-Security: max-age=15768000
< Server: Twilio Assets v0.1
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
<
* Connection #0 to host bole-nightingale-xxxx.twil.io left intact
<?xml version="1.0" encoding="UTF-8"?><Response><Message>The weather in bishan is Windy</
Message></Response>
```

From our Twilio Function console, you can also observe what has been sent to console log.

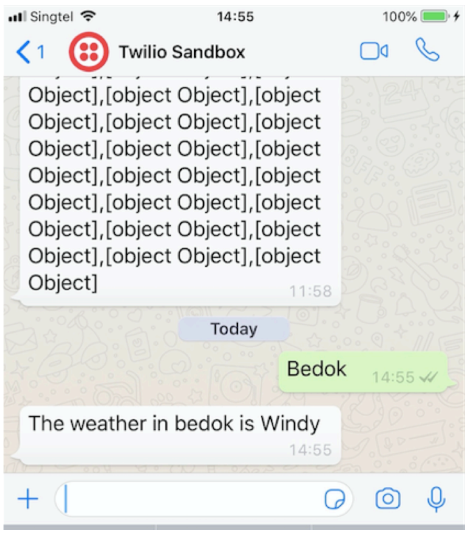


```
LOGS
DATE                      MESSAGE
2019-01-03 06:38:56 UTC   Execution Started
2019-01-03 06:38:56 UTC   { forecasts: '[Object]' }
2019-01-03 06:38:56 UTC   The key is bishan , Windy
2019-01-03 06:38:56 UTC   The weather in bishan is Windy Windy
2019-01-03 06:38:56 UTC   Execution Ended in 82.67ms using 70MB
```

Now that we have successfully received a response with the corresponding weather forecast, lets tried it using Whatsapp.

Sending a Weather forecast request in WhatsApp

Use the device that has already been registered with Twilio Sandbox. Send the area where you want to query the weather forecast. Let's use area ***Bedok.***



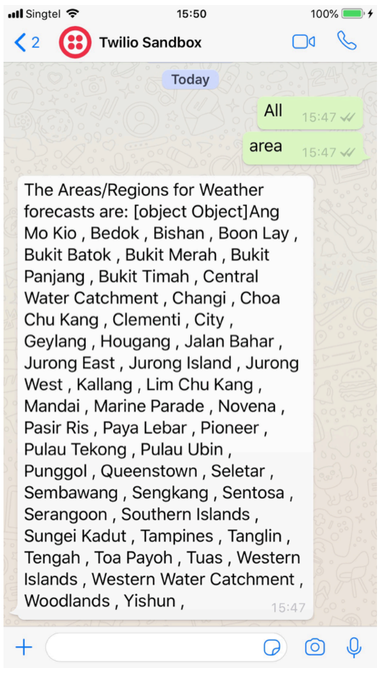
Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[object Object] 11:58

Today

Bedok 14:55

The weather in bedok is Windy 14:55

Asking for the list of Areas available, by sending message “area”



All 15:47

area 15:47

The Areas/Regions for Weather forecasts are: [object Object]Ang Mo Kio , Bedok , Bishan , Boon Lay , Bukit Batok , Bukit Merah , Bukit Panjang , Bukit Timah , Central Water Catchment , Changi , Choa Chu Kang , Clementi , City , Geylang , Hougang , Jalan Bahar , Jurong East , Jurong Island , Jurong West , Kallang , Lim Chu Kang , Mandai , Marine Parade , Novena , Pasir Ris , Paya Lebar , Pioneer , Pulau Tekong , Pulau Ubin , Punggol , Queenstown , Seletar , Sembawang , Sengkang , Sentosa , Serangoon , Southern Islands , Sungei Kadut , Tampines , Tanglin , Tengah , Toa Payoh , Tuas , Western Islands , Western Water Catchment , Woodlands , Yishun , 15:47

Can you guess where did Twilio host the Functions, Assets and TwilioCDN?