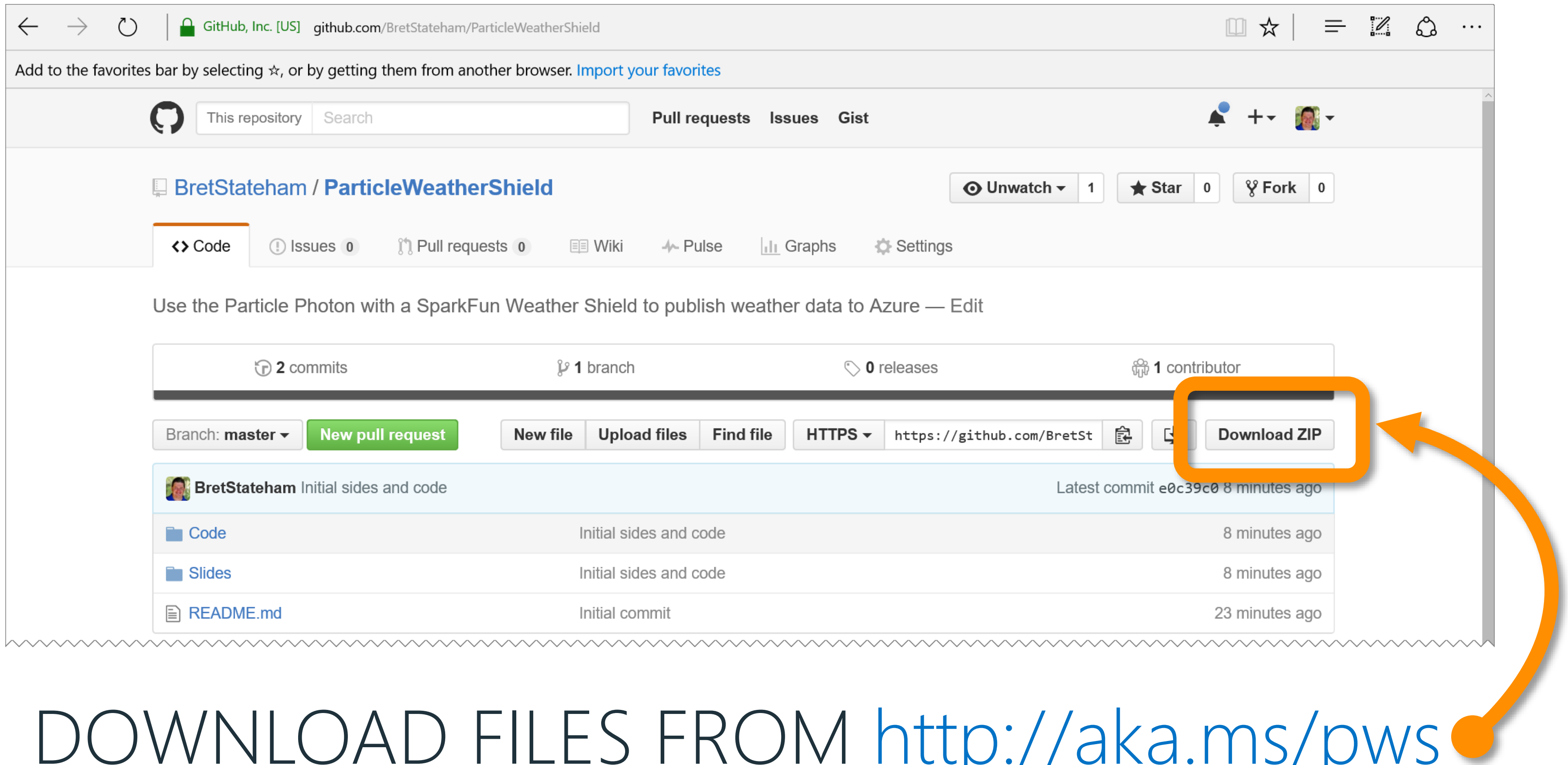
PARTICLE PHOTON, SPARKFUN WEATHER SHIELD AND MICROSOFT AZURE

http://aka.ms/pws

Microsoft

GITHUB REPO



github.com/BretStateham/ParticleWeatherShield

Add to the favorites bar by selecting ☆, or by getting them from another browser. Import your favorites

BretStateham / ParticleWeatherShield

Use the Particle Photon with a SparkFun Weather Shield to publish weather data to Azure — Edit

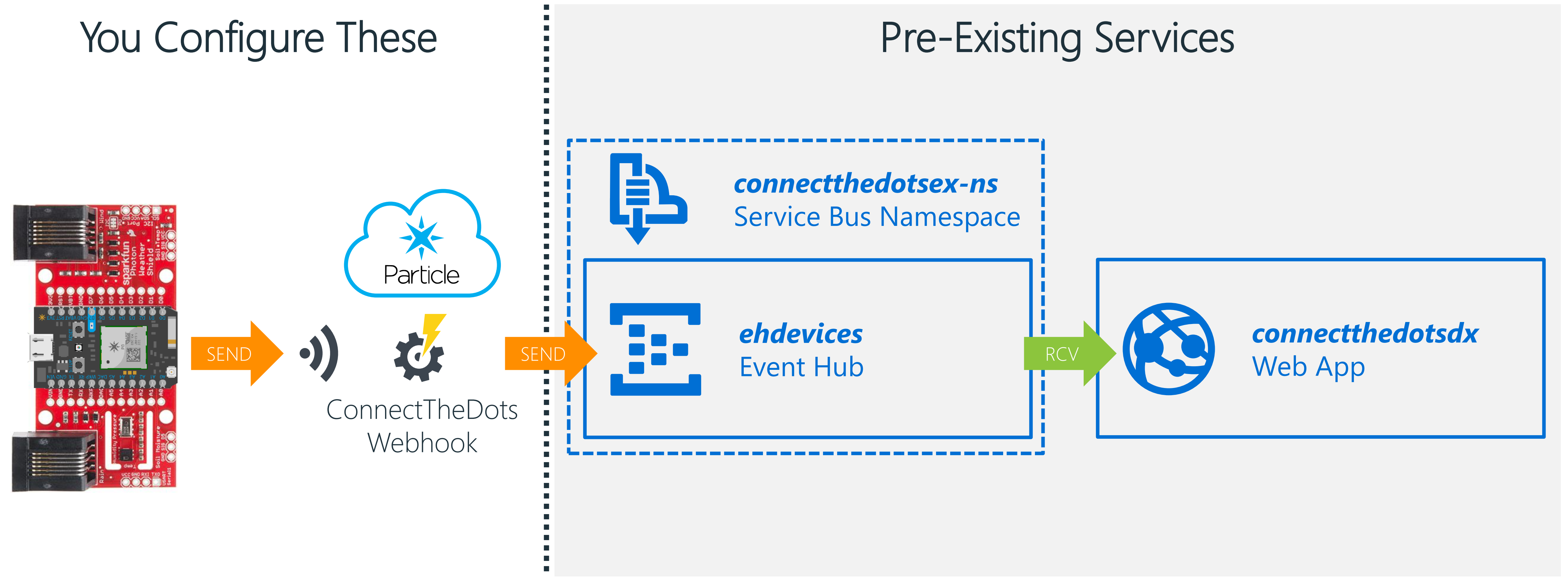
2 commits | 1 branch | 0 releases | 1 contributor

BretStateham Initial sides and code		Latest commit e0c39c0 8 minutes ago
Code	Initial sides and code	8 minutes ago
Slides	Initial sides and code	8 minutes ago
README.md	Initial commit	23 minutes ago

DOWNLOAD FILES FROM http://aka.ms/pws

Microsoft

ARCHITCTURE OVERVIEW

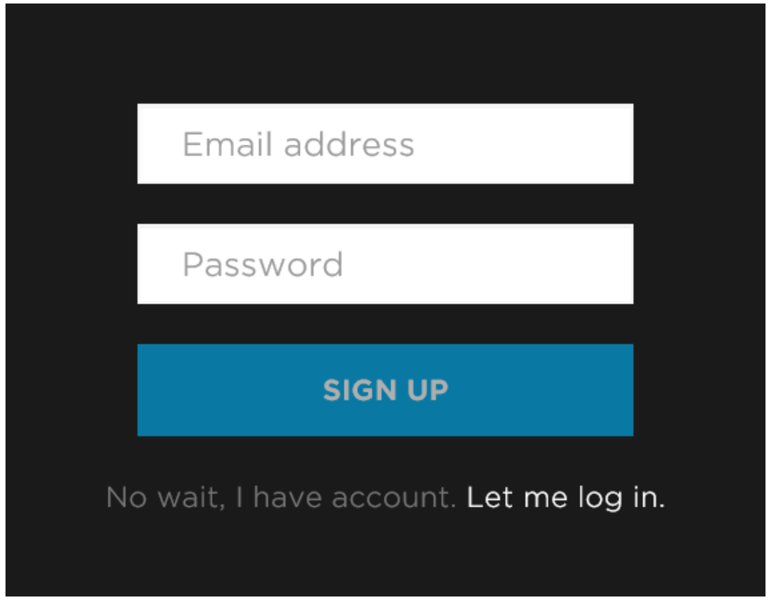


http://aka.ms/pws

INSTALLING PREREQUISITES

CREATE A FREE PARTICLE.IO ACCOUNT

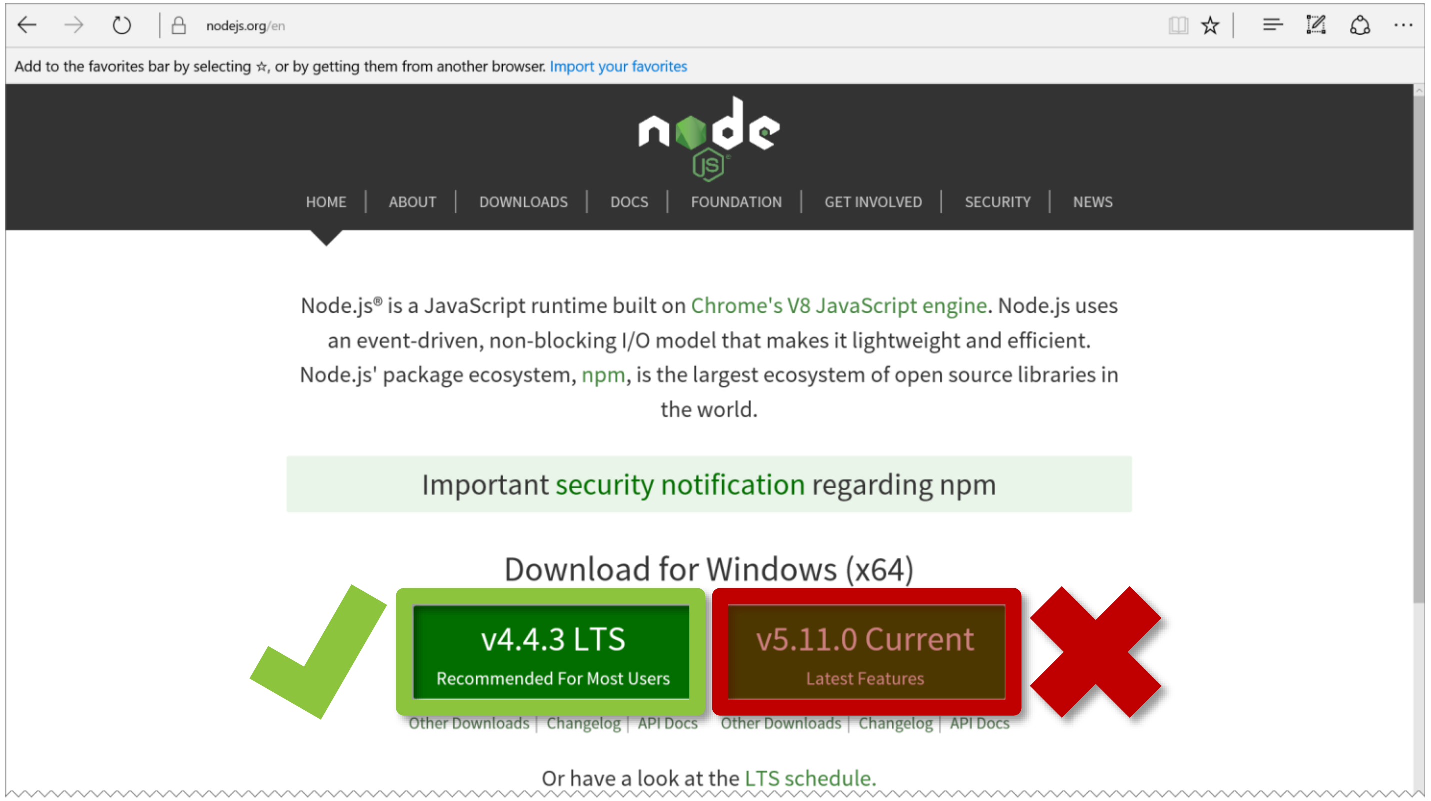
- Go to http://particle.io
- Click on the "build" link: http://build.particle.io/build
- Create an account (free)
  make sure to remember the email/pwd used



Microsoft

INSTALL NODE.JS v4 NOT v5 OR v6

https://nodejs.org/


nodejs.org/en

Add to the favorites bar by selecting ☆, or by getting them from another browser. Import your favorites

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security notification regarding npm

Download for Windows (x64)

v4.4.3 LTS
Recommended For Most Users

v5.11.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Other Downloads | Changelog | API Docs

Or have a look at the LTS schedule.

http://aka.ms/pws

Microsoft

INSTALL THE PARTICLE-CLI

Windows:	Open a command prompt
Mac:	Open a terminal window (may need to use **`sudo`**)

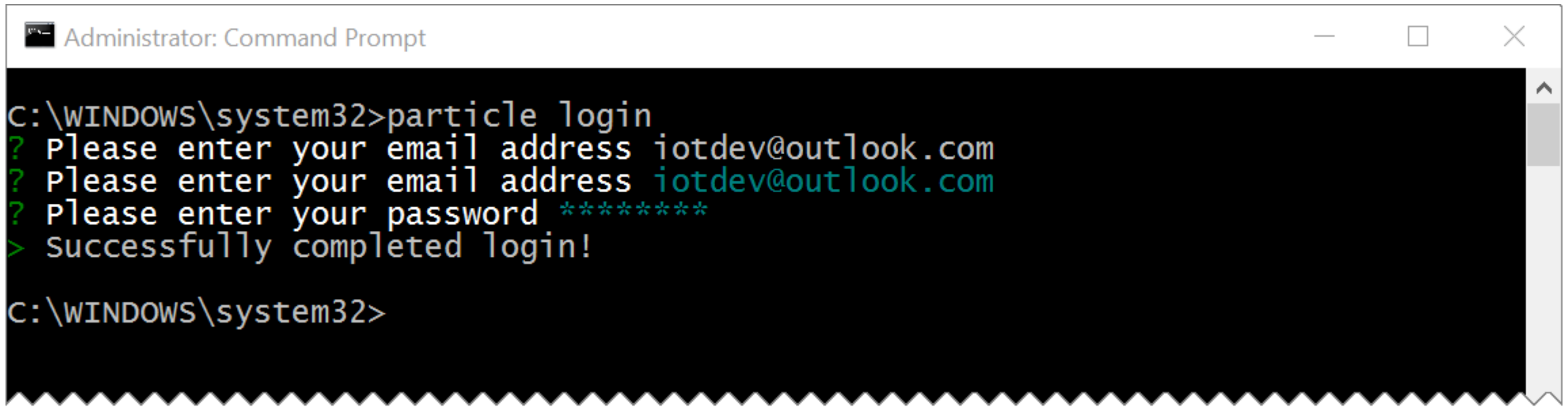
```
npm install –g particle-cli
```

LOGIN TO THE PARTICLE-CLI

- From a command prompt / terminal window:

```
particle login
```

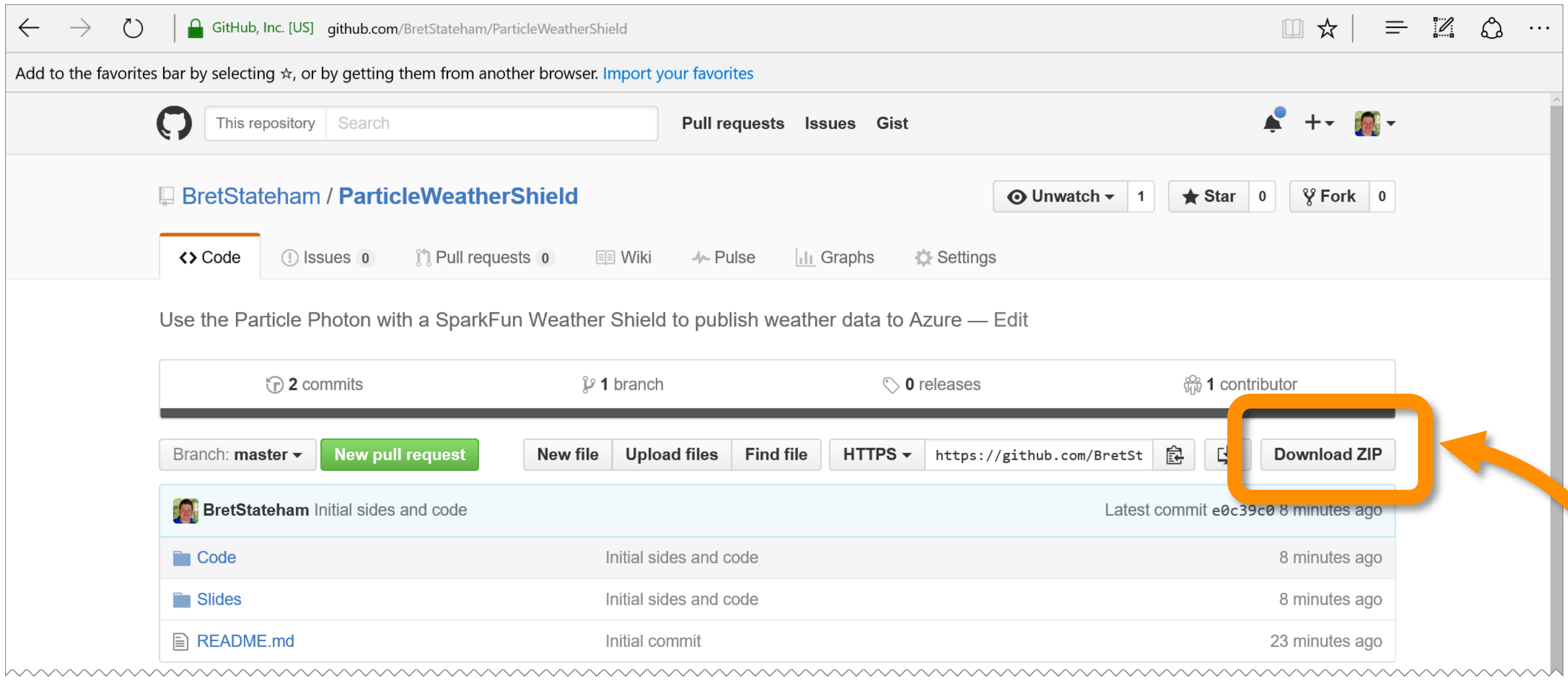
- Use the credentials for particle build account you created



```
C:\WINDOWS\system32>particle login
? Please enter your email address iotdev@outlook.com
? Please enter your email address iotdev@outlook.com
? Please enter your password ********
> Successfully completed login!

C:\WINDOWS\system32>
```

DOWNLOAD THE SOURCE


GitHub, Inc. [US] github.com/BretStateham/ParticleWeatherShield
Add to the favorites bar by selecting ☆, or by getting them from another browser. Import your favorites
This repository Search
Pull requests Issues Gist
BretStateham / ParticleWeatherShield
Unwatch 1 Star 0 Fork 0
Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings
Use the Particle Photon with a SparkFun Weather Shield to publish weather data to Azure — Edit
2 commits 1 branch 0 releases 1 contributor
Branch: master New pull request New file Upload files Find file HTTPS https://github.com/BretSt Download ZIP
BretStateham Initial sides and code Latest commit e0c39c0 8 minutes ago
Code Initial sides and code 8 minutes ago
Slides Initial sides and code 8 minutes ago
README.md Initial commit 23 minutes ago

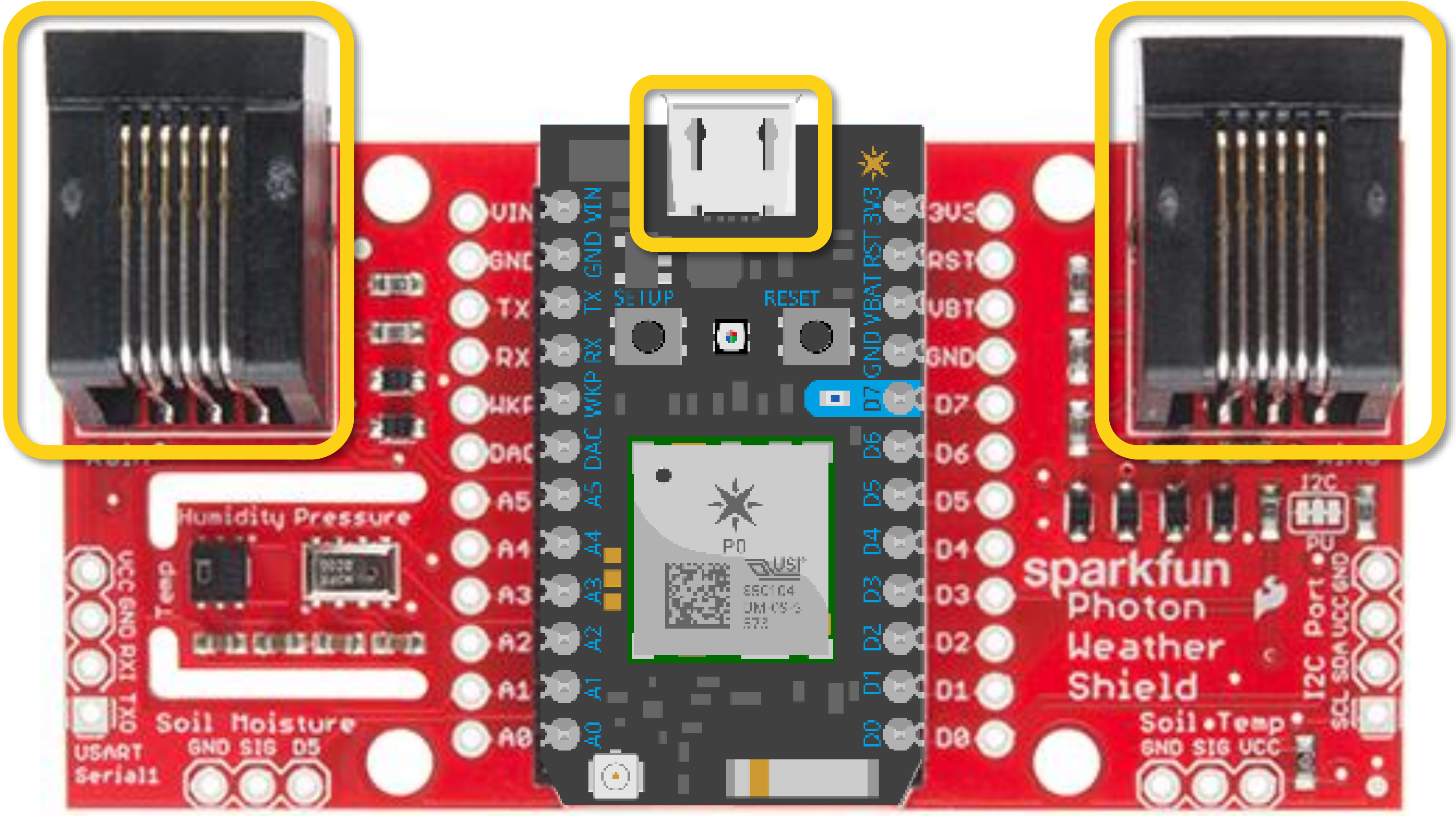
On Windows, "unblock" the .zip before you extract it:

- Right-click on .zip
- Select Properties
- Check "unblock"
- Click "OK"

DOWNLOAD FILES FROM http://aka.ms/pws

SETTING UP AND CLAIMING THE PHOTON

ASSEMBLE THE PHOTON AND WEATHER SHIELD



Photon’s USB port on the same edge as the Rain and Wind sensor jacks

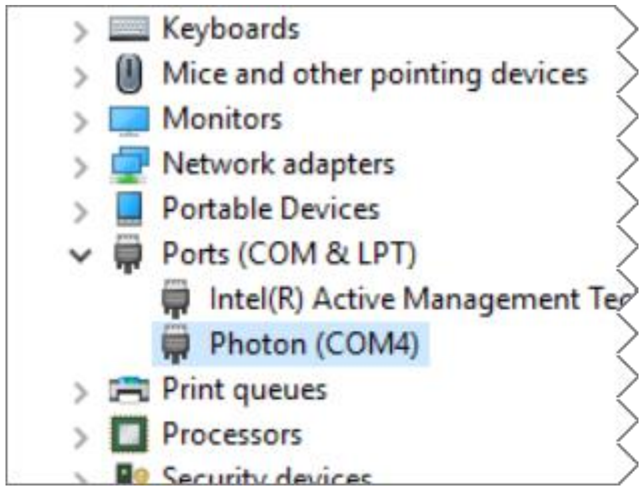
Microsoft

CONNECT THE PHOTON TO YOUR COMPUTER VIA USB CABLE

- On Windows? Install the driver:

  http://aka.ms/photondriver

- Extract the .zip file to a folder on your computer:
- Open Device Manager
- Right click-on the un-recognized device
- Select "Update Driver Software..."
- Point to the folder where you extracted the driver
- Complete the install
- Note the COM port it is attached to


Keyboards
Mice and other pointing devices
Monitors
Network adapters
Portable Devices
Ports (COM & LPT)
Intel(R) Active Management Te
Photon (COM4)
Print queues
Processors
Security devices

Microsoft

IDENTIFY THE ATTACHED PHOTON

- First find the COM port it is attached to:

```
particle serial list
```

- Then, get the device ID:

```
particle serial identify
```

- Copy the Photon ID to your clipboard and save somewhere for later

CONFIGURE THE PARTICLE WIFI

- From the command prompt:

```
particle serial wifi
```

Follow the prompts to connect the WiFi:

```
<your wifi ssid>
<your wifi security mode>
<your wifi cypher type>
<your wifi password>
```

CLAIM AND NAME YOUR PHOTON

- Then, get the device ID, copy the reported ID to your clipboard (Make sure the Photon is in Listening Mode):

```
particle serial identify
```

- Add the device to your account

```
particle device add <id>
```

- Name the device

```
particle device rename <id> <newname>
```

CREATING THE WEBHOOK

DOWNLOAD THE WEBHOOK.JSON

http://aka.ms/pwsholwebhook

- Save it to a known folder on your computer
- Open the webhook.json file to review it
- YOU DO NOT NEED TO MAKE ANY CHANGES AT THIS POINT
- Learn more about Particle Webhooks here: http://aka.ms/pwh

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

The name of the “**`event`**” that will invoke the webhook.

When you call

`Spark.publish("ConnectTheDots", payload);`

from your Photon, the Particle cloud back end will trigger this webhook and publish your **`payload`** to the Azure Event Hub, specified by the “**`url`**”

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

The "`url`" of the Azure Event Hub that the data will be published to.

In this case, we are posting to the "**`connectthedotsex-ns`**" service bus namespace into an event hub named "**`ehdevices`**"

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

The data will be sent to the url using an HTTP POST

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

The json template is used to create the data for the message that will be sent to the Azure Event Hub

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

```
{
“s”: ”Weather”,
“u”: “F”,
“m”: “Temperature”,
“v”:  79.234,
“o”: “My Organization”,
“d”: “My Device Name”
“l”: “My Location”,
}
```

Sample Payload from the Photon’s Particle.Publish() call

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

```
The values:

SPARK_PUBLISHED_AT
and
SPARK_CORE_ID

Are supplied by the Particle
cloud back end
```

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

The Shared-Access-Signature key name, and key value used to access the Azure Event Hub

The key name and key used here are from an event hub that Paul DeCarlo has setup already in the cloud for you to use.

```
{
  "event": "ConnectTheDots",
  "url": "https://connectthedotsex-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"
  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "mBLQGWxSkRHg7f2eRCLonHUpNS+DY0iPHclxjf7Cmvk="
  },

  "mydevices": true
}
```

Restricts the webhook to being triggered only from events published from your own devices

CREATE THE PARTICLE WEBHOOK

- From the command prompt, get in the same folder as the webhook.json file you downloaded, and run:

```
particle webhook create webhook.json
```

- Other particle-cli webhook commands include:

```
particle webhook list
particle webhook delete hookid
```

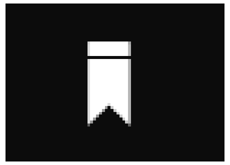
PROGRAMMING THE PHOTON

Create the ParticleWeatherShield sketch

- Create a new "App" here: https://build.particle.io/build/new
- Name it "ParticleWeatherShield"
- Download the sketch from: http://aka.ms/pwsholsketch
- Copy the text from the downloaded ParticleWeatherShield.c file
- Paste into the ParticleWeatherShield app in Particle Build

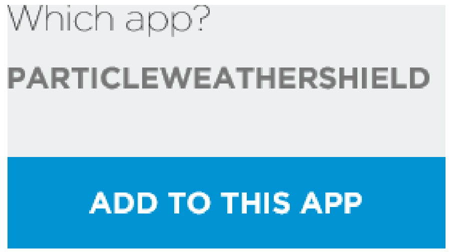
ADD THE WEATHER SHIELD LIBRARY

- In the Particle Build web interface, click the "Libraries" icon



- Search for the SparkFun_Photon_Weather_Shield_Library
- Select it, and then click the "INCLUDE IN APP" button:

- Then, select the "ParticleWeatherShield" app, and click the "ADD TO THIS APP" button



LIBRARY INCLUDE AND FIELDS

```
// This #include statement was automatically added by the Particle IDE.
#include "SparkFun_Photon_Weather_Shield_Library/SparkFun_Photon_Weather_Shield_Library.h"

Weather sensor;

char Org[] = "WearHacks LA";
char Disp[] = "Brets Demo Photon";
char Locn[] = "LA";
```

LIBRARY INCLUDE AND FIELDS

```
// This #include statement was automatically added by the Particle IDE.
#include "SparkFun_Photon_Weather_Shield_Library/SparkFun_Photon_Weather_Shield_Library.h"

char Org[] = "ORGANIZATION_NAME";
char Disp[] = "DISPLAY_NAME";
char Locn[] = "LOCATION";

//Create Instance of the Weather Shield
Weather sensor;

//The amount of time (in milliseconds) to wait between each publication of data
int sendDelay = 6000;
```

SETUP METHOD

```
void setup()
{

  //Open up the Serial port for local diagnostics
  Serial.begin(9600);

  //Initialize the I2C sensors and ping them
  sensor.begin();
```

SETUP METHOD

```
  //The following two lines tell the sensor what mode to use
  sensor.setModeBarometer();//Set to Barometer Mode
  //These are additional MPL3115A2 functions the MUST be called for the sensor to work.
  sensor.setOversampleRate(7);

  //Give the sensors some time to initialize
  delay(10000);
}
```

LOOP METHOD, AND SENSOR READING

```
void loop()

{
  //Measure Relative Humidity from the HTU21D or Si7021
  float h = sensor.getRH();

  //Measure Temperature from the HTU21D or Si7021
  float f = sensor.getTempF();
```

PUBLISH TEMPERATURE DATA

```
// Generate the temperature data payload
char payload[255];
snprintf(payload, sizeof(payload),
  "{\"s\":\"Weather\",
    \"u\":\"F\",
    \"m\":\"Temperature\",
    \"v\":%f,
    \"o\":\"%s\",
    \"d\":\"%s\",
    \"l\":\"%s\"}"
    f,Org,Disp,Locn);
```

```
{
  “s”: “Weather”,
  “u”: “F”,
  “m”: “Temperature”,
  “v”:  79.234,
  “o”: “ORGANIZATION_NAME”,
  “d”: “DISPLAY_NAME”
  “l”: “LOCATION”,
}
```

```
//Emit the payload to the serial port for monitoring purposes
Serial.println(payload);
```

```
// Send the temperature data payload
Particle.publish(“ConnectTheDots", payload);
//Wait for the specified "sendDelay" before sending the humidity data...
delay(sendDelay);
```

GROKING THE PHOTON CODE

```
// Generate the humidity data payload
snprintf(payload, sizeof(payload),
  "{\"s\":\"Weather\",
    \"u\":\"%%\",
    \"m\":\"Humidity\",
    \"v\":%f,
    \"o\":\"%s\",
    \"d\":\"%s\",
    \"l\":\"%s\"}",
    h,Org,Disp,Locn);
```

```
{
  “s”: “Weather”,
  “u”: “%”,
  “m”: “Humidity”,
  “v”:  25.345,
  “o”: “ORGANIZATION_NAME”,
  “d”: “DISPLAY_NAME”
  “l”: “LOCATION”,
}
```

```
// Emit the payload to the serial port for monitoring purposes
Serial.println(payload);
```

```
// Send the humidity data payload
Particle.publish(“ConnectTheDots", payload);
// wait for the specified "sendDelay" before looping...
delay(sendDelay);
} // End of loop()
```

MODIFY THE METADATA PROPERTIES

- Put your own values in. Make sure that the "Display Name" is appropriate and identifies your Photon clearly:

```
char Org[] = "ORGANIZATION_NAME";
char Disp[] = "DISPLAY_NAME";
char Locn[] = "LOCATION";
```

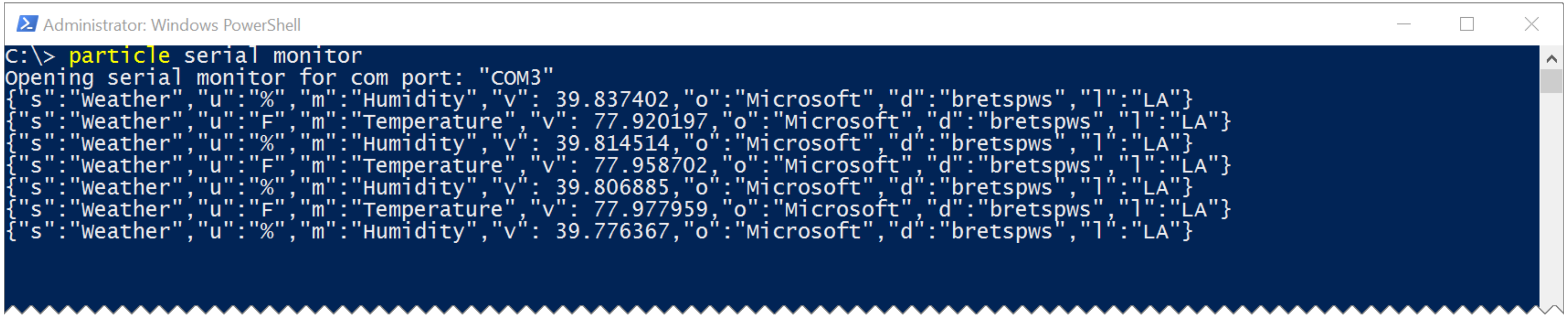
SAVE, VERIFY AND FLASH

- Click the save icon to save your sketch:
- Click the verify icon to compile and verify:
- Click the flash icon to deploy your sketch:

MONITORING YOUR PHOTON

MONITORING PHOTON VIA SERIAL

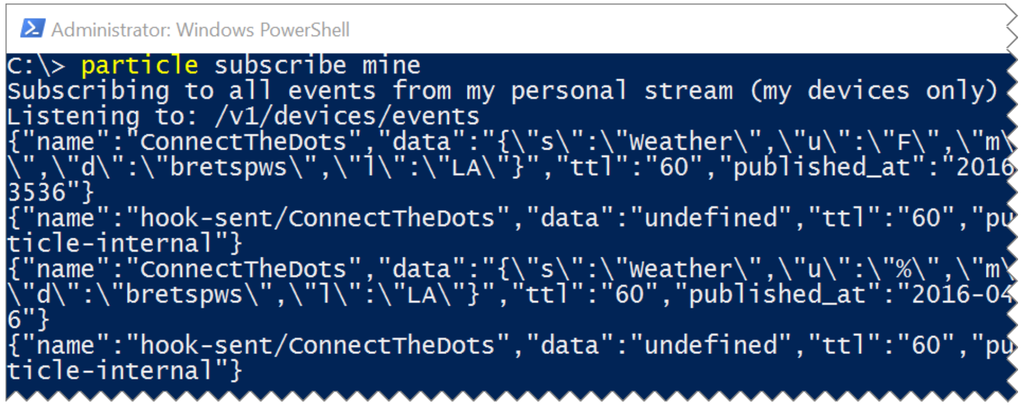
`particle serial monitor`



```
C:\> particle serial monitor
Opening serial monitor for com port: "COM3"
{"s":"Weather","u":"%","m":"Humidity","v": 39.837402,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"F","m":"Temperature","v": 77.920197,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"%","m":"Humidity","v": 39.814514,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"F","m":"Temperature","v": 77.958702,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"%","m":"Humidity","v": 39.806885,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"F","m":"Temperature","v": 77.977959,"o":"Microsoft","d":"bretspws","l":"LA"}
{"s":"Weather","u":"%","m":"Humidity","v": 39.776367,"o":"Microsoft","d":"bretspws","l":"LA"}
```

MONITORING WEBHOOK

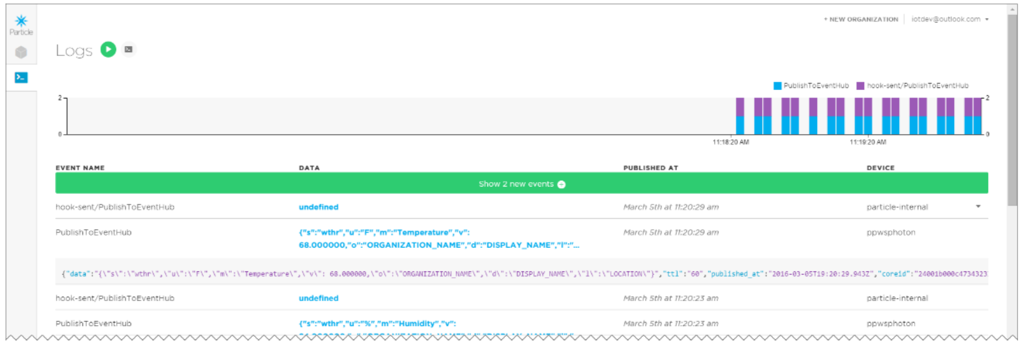
- Particle-cli:
  `particle subscribe mine`



Administrator: Windows PowerShell

```
C:\> particle subscribe mine
Subscribing to all events from my personal stream (my devices only)
Listening to: /v1/devices/events
{"name":"ConnectTheDots","data":"{\"s\":\"Weather\",\"u\":\"F\",\"m\
\",\"d\":\"bretspws\",\"l\":\"LA\"}","ttl":"60","published_at":"2016
3536"}
{"name":"hook-sent/ConnectTheDots","data":"undefined","ttl":"60","pu
ticle-internal"}
{"name":"ConnectTheDots","data":"{\"s\":\"Weather\",\"u\":\"%\",\"m\
\"d\":\"bretspws\",\"l\":\"LA\"}","ttl":"60","published_at":"2016-04
6"}
{"name":"hook-sent/ConnectTheDots","data":"undefined","ttl":"60","pu
ticle-internal"}
```

- Particle Dashboard:
  http://dashboard.particle.io/user/logs



EVENT NAME	DATA	PUBLISHED AT	DEVICE
hook-sent/PublishToEventHub	undefined	March 5th at 11:20:29 am	particle-internal
PublishToEventHub	{"s":"wthr","u":"F","m":"Temperature","v": 68.000000,"o":"ORGANIZATION_NAME","d":"DISPLAY_NAME","l":"...	March 5th at 11:20:29 am	ppwsphoton

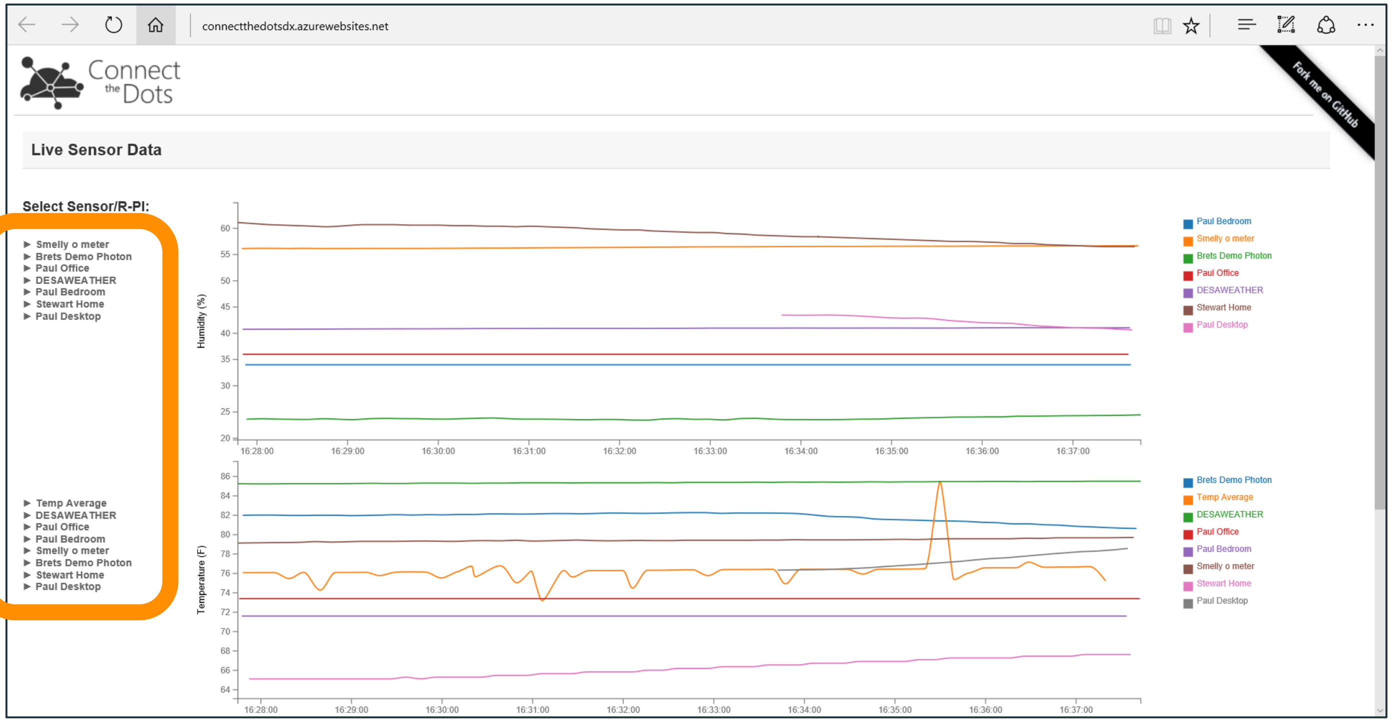
```
{"data":"{\"s\":\"wthr\",\"u\":\"F\",\"m\":\"Temperature\",\"v\": 68.000000,\"o\":\"ORGANIZATION_NAME\",\"d\":\"DISPLAY_NAME\",\"l\":\"LOCATION\"}","ttl":"60","published_at":"2016-03-05T19:20:29.943Z","coreid":"24001b000c4734323
```

EVENT NAME	DATA	PUBLISHED AT	DEVICE
hook-sent/PublishToEventHub	undefined	March 5th at 11:20:23 am	particle-internal
PublishToEventHub	{"s":"wthr","u":"%","m":"Humidity","v":	March 5th at 11:20:23 am	ppwsphoton

MONITOR VIA THE WEB

http://aka.ms/pwsholweb

Click device names along the left to show or hide them in the graphs



Microsoft

WHERE TO GO FROM HERE?

- Publish other sensor values. There are TON's of other sensors you can use. As an example, check out:

  http://aka.ms/sparkfunsensors

- Don't want to use Paul's existing event hub & web site? Make your own event hub and publish to Power BI, or consume the data from a client app! Check out my step by step walkthrough at:

  http://aka.ms/ppws

- Don't want to be limited by the Particle Cloud's rate limits? Try publishing to your own API:

  http://aka.ms/iotree