WELCOME TO

INITECH

http://aka.ms/hcicsource

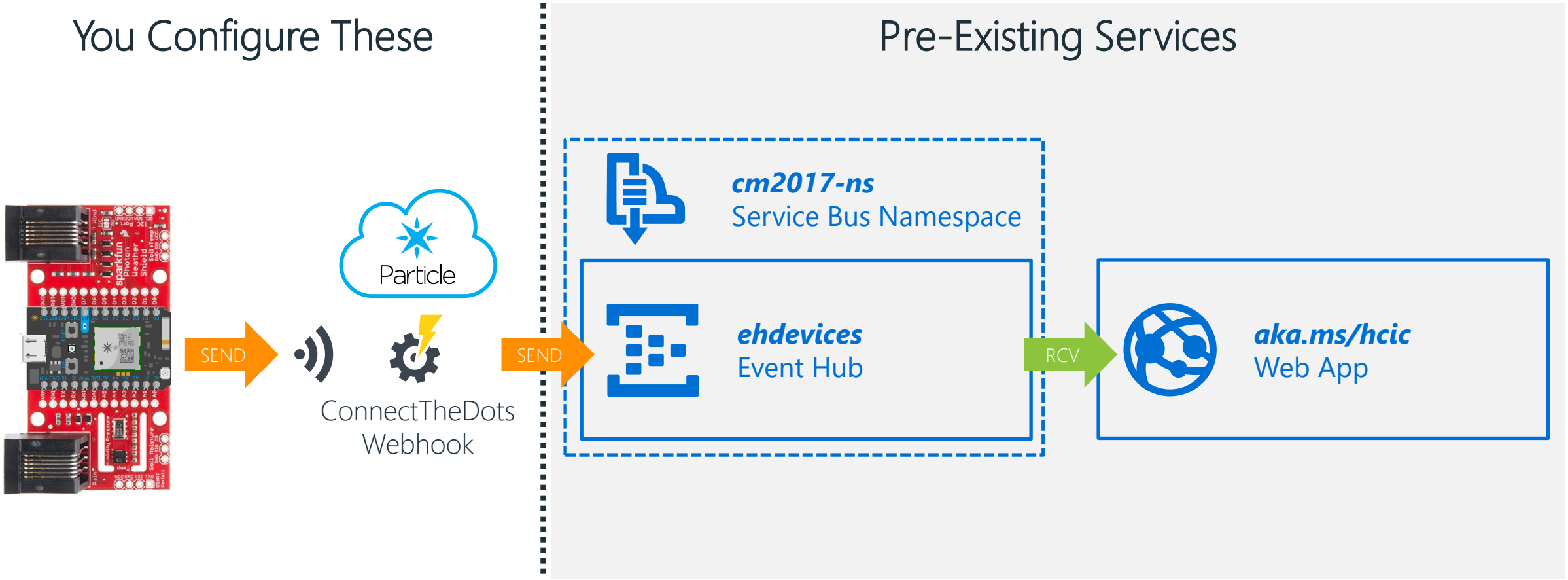
Microsoft

How cold is codemash?

http://aka.ms/hcicsource

Microsoft

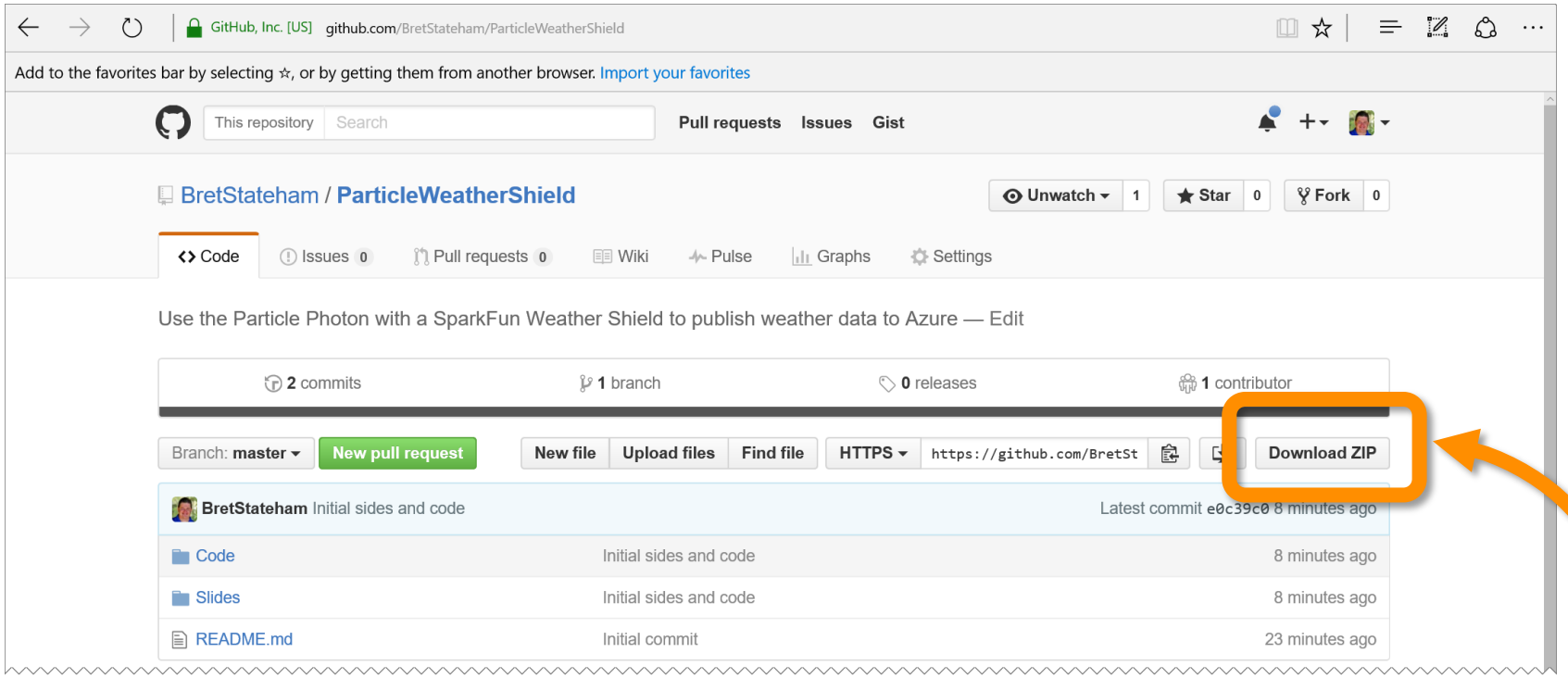
ARCHITECTURE OVERVIEW



http://aka.ms/hcicsource

Microsoft

GITHUB REPO



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

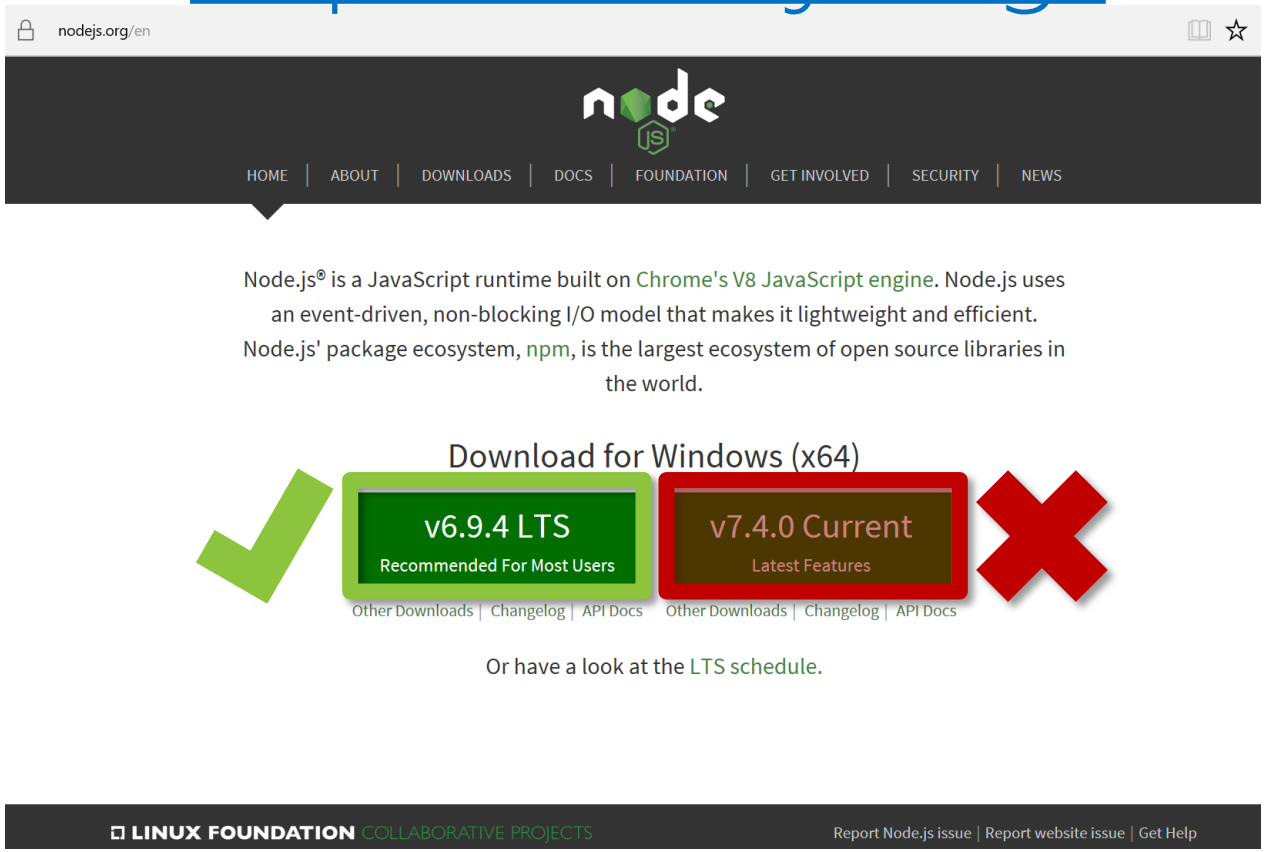
Microsoft

Installing prereqs, just a moment!

INSTALLING PREREQUISITES

INSTALL NODE.JS v6.9.4 NOT v7.4.0

https://nodejs.org/



http://aka.ms/hcicsource

Microsoft

INSTALL PREREQUISITES

- Install Node v6.9.4 LTS (NOT v7.4.0)
  https://nodejs.org/

  Need a version manager for node? Check out
  https://github.com/creationix/nvm for Mac OSx or Linux or
  http://github.com/coreybutler/nvm-windows for Windows

- Download and install Python 2.x (not 3+)
  https://www.python.org/downloads/

- On Windows? You *may* need Visual Studio C++ tools:
  http://aka.ms/msbuild2013

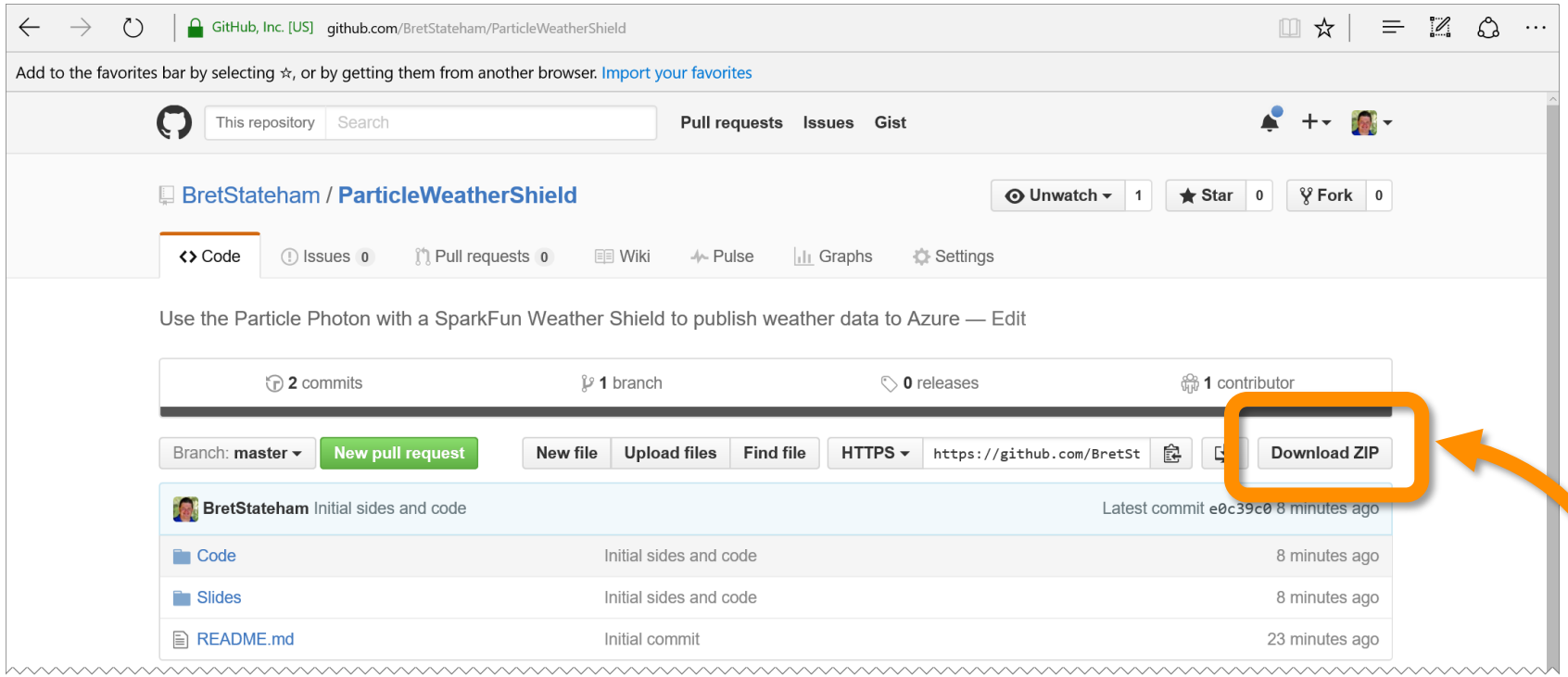
INSTALL THE PARTICLE-CLI

Windows: Open a command prompt

Mac: Open a terminal window (may need to use **sudo**)

```
npm install -g particle-cli
```

DOWNLOAD THE SOURCE

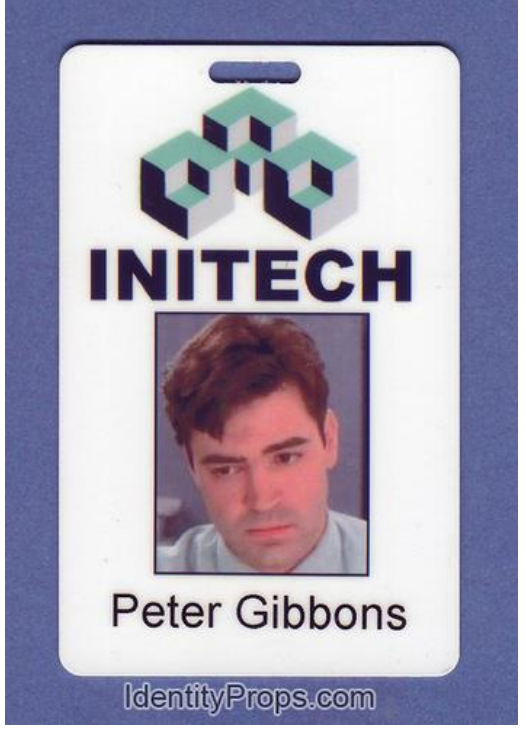


On Windows, "unblock" the .zip before you extract it:

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

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

Microsoft

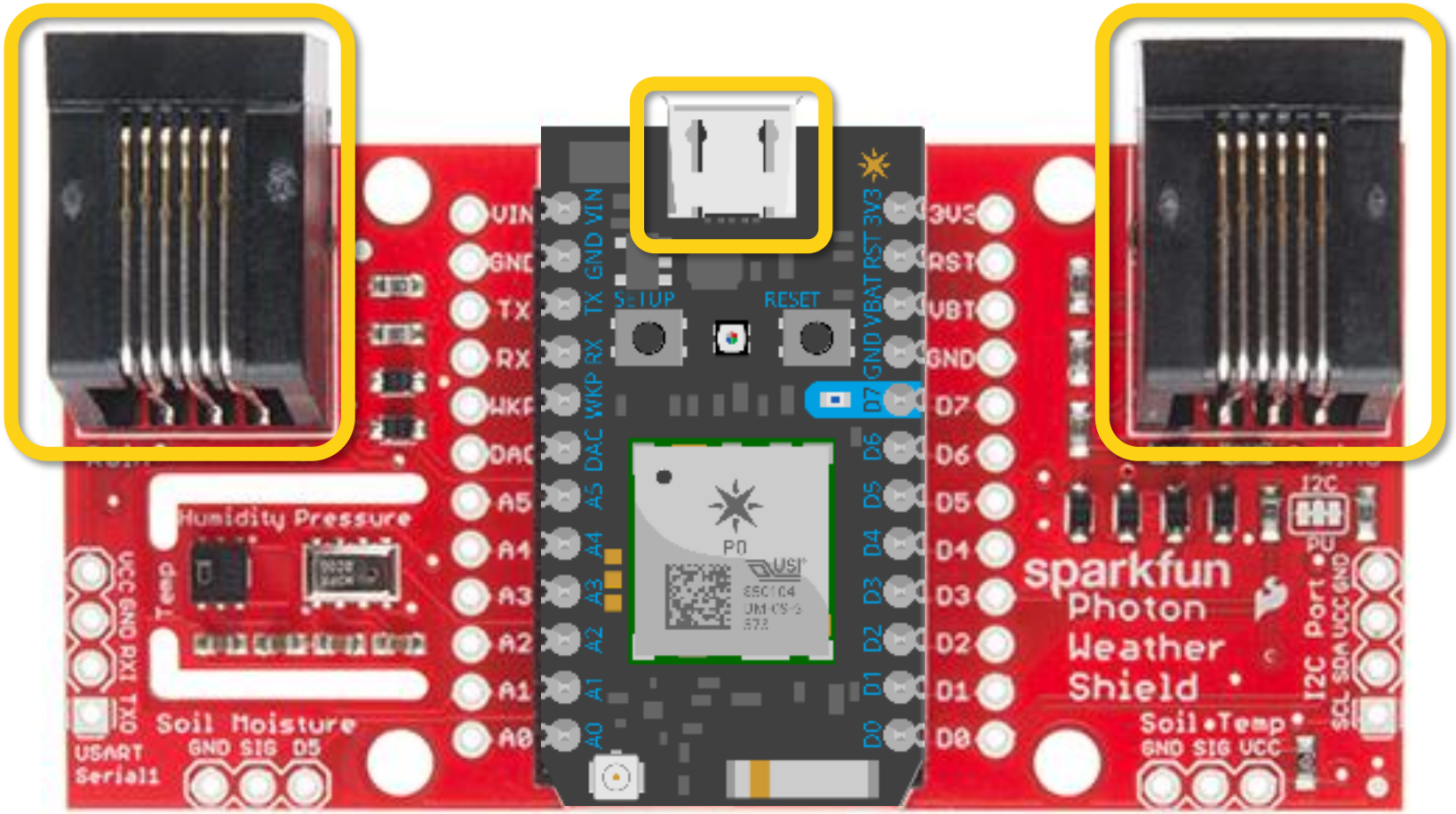


SETTING UP AND CLAIMING THE PHOTON

http://aka.ms/hcicsource

Microsoft

ASSEMBLE THE PHOTON AND WEATHER SHIELD



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

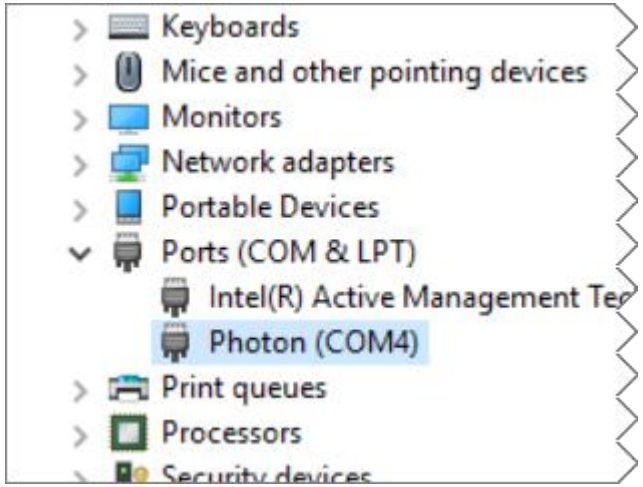
Microsoft

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



IDENTIFY THE ATTACHED PHOTON

- First find the COM port it is attached to:

```
particle serial list
```

- Then, get the device ID:

```
particle serial identify
```

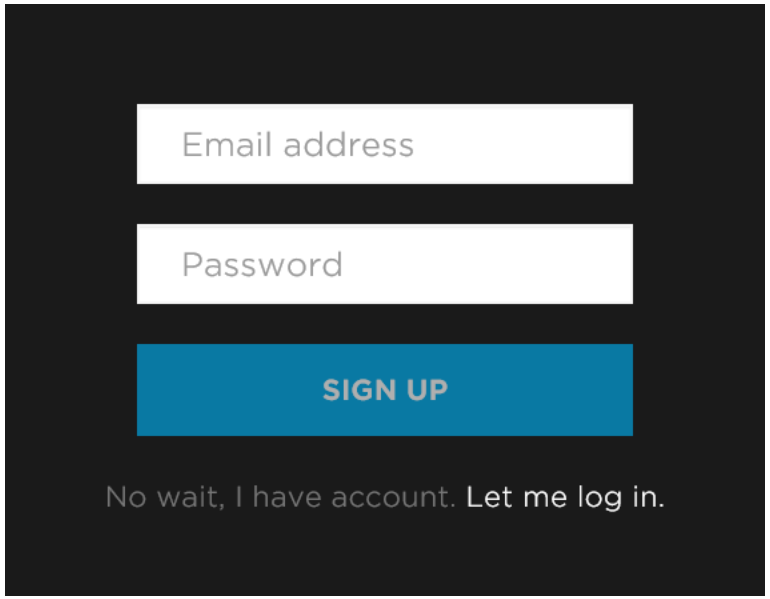
- Copy the Photon ID to your clipboard and save somewhere for later

http://aka.ms/hcicsource

Microsoft

CREATE A FREE PARTICLE.IO ACCOUNT

- Go to http://particle.io
- Click on the "build" link: http://build.particle.io/build
- Create an account using
  - U: <id>@mailinator.com
  - P: <id>



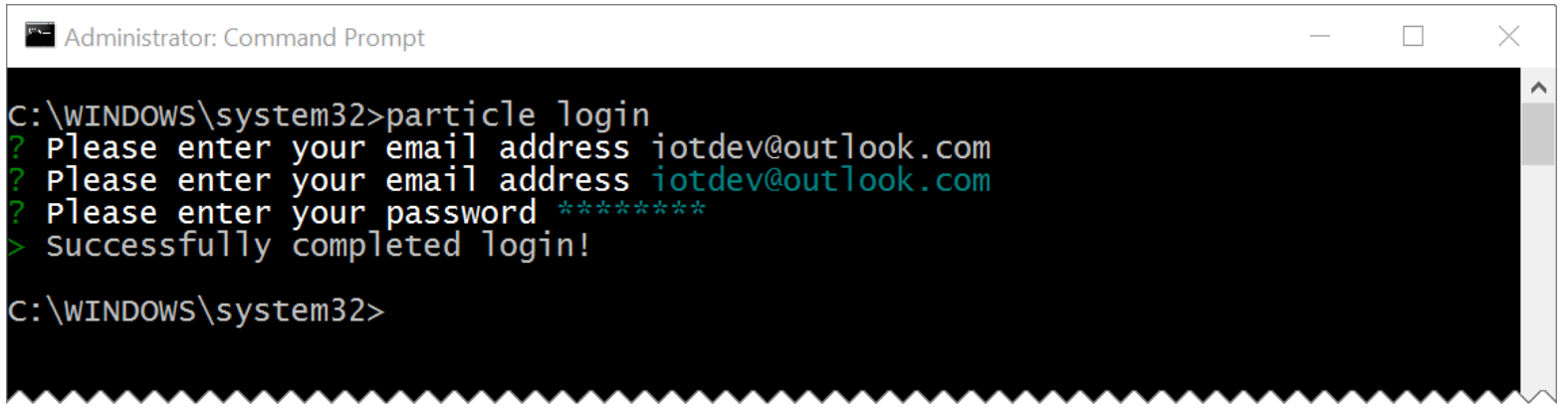
Microsoft

LOGIN TO THE PARTICLE-CLI

- From a command prompt / terminal window:

```
particle login
```

- Use the credentials for particle build account you created



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>
```

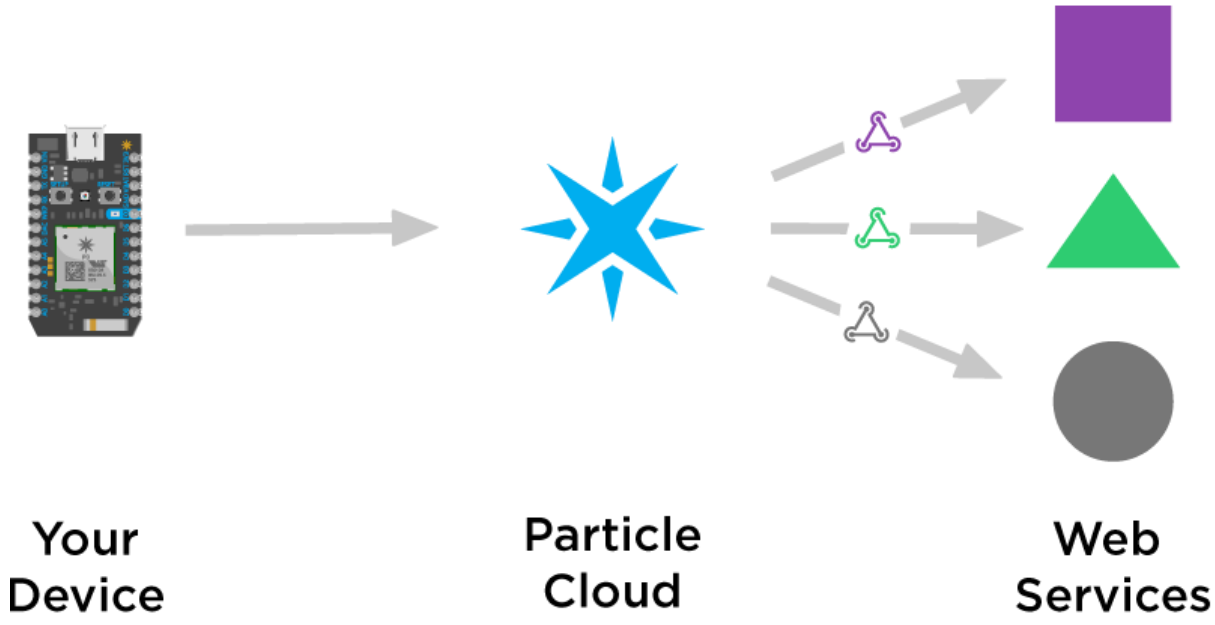
CONFIGURE THE PARTICLE WIFI

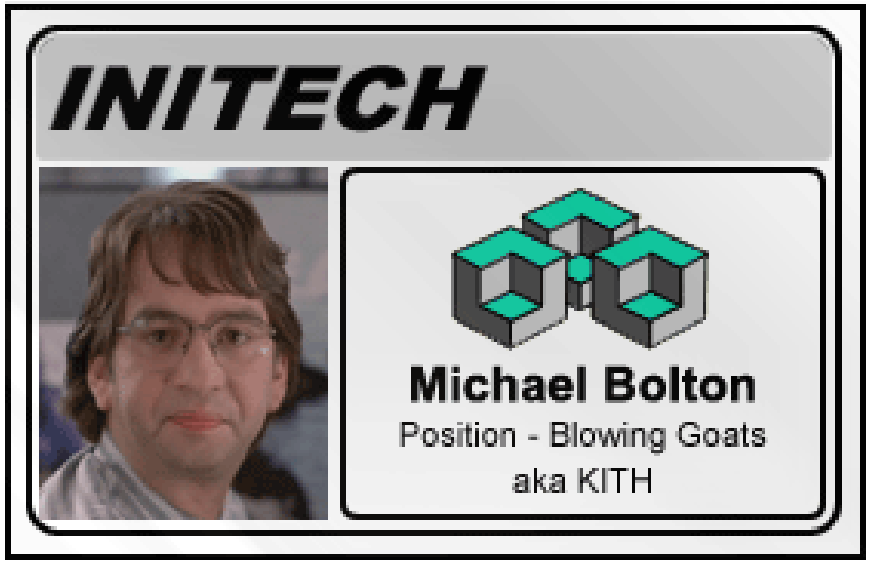
- From the command prompt:

```
particle serial wifi
```

Follow the prompts to connect the WiFi:

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





CREATING THE WEBHOOK

Microsoft

```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```

The name of the "**event**" that will invoke the webhook.

When you call

```
Spark.publish("HowColdIsCodemash",
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": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```

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

In this case, we are posting to the "**cm2017-ns**" service bus namespace into an event hub named "**ehdevices**"

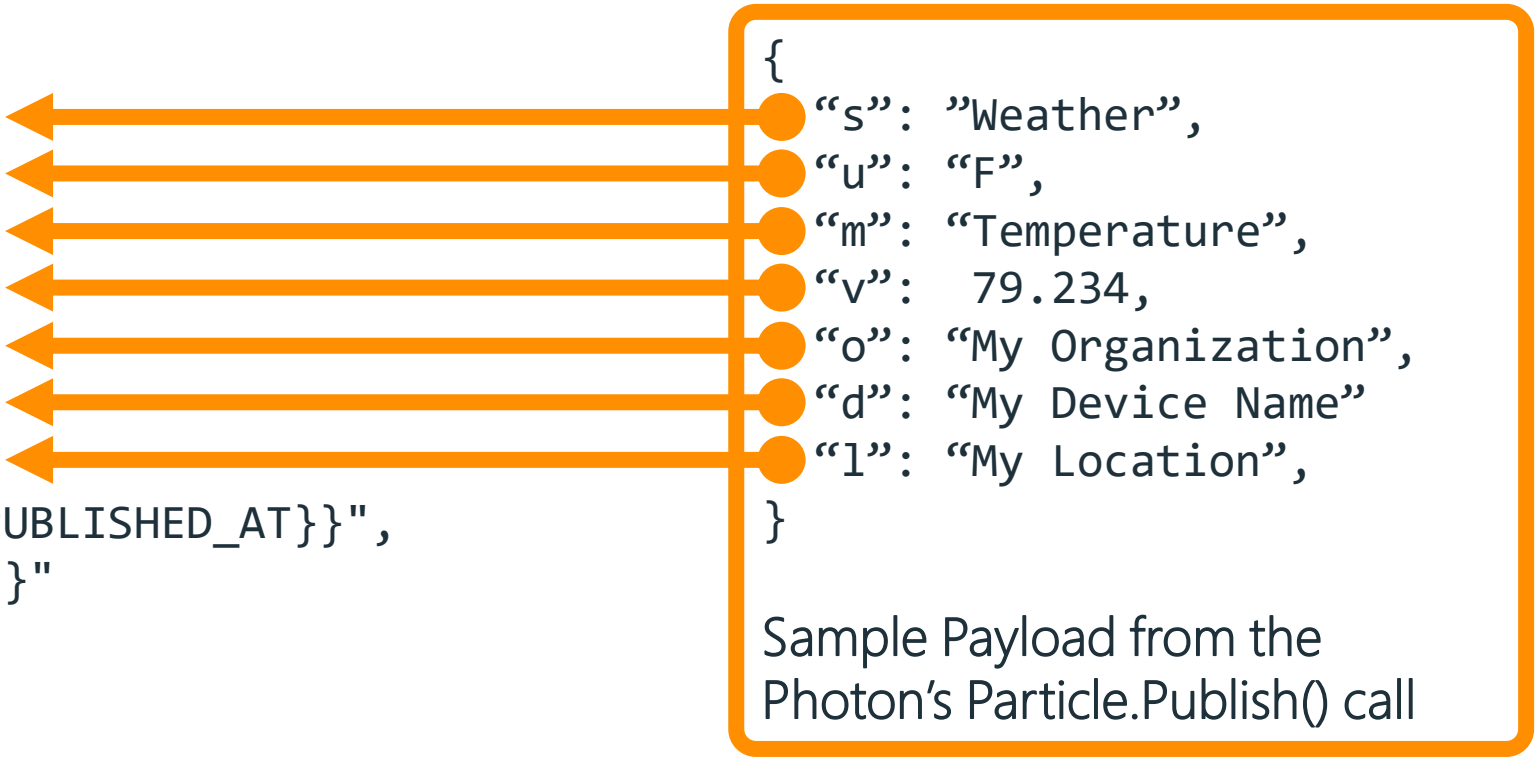
```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-ns.servicebus.windows.net/cmdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"

  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```

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

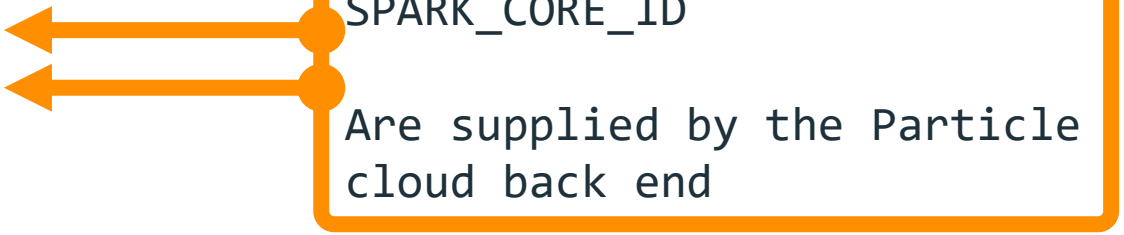
```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-ns.servicebus.windows.net/ehdevices/messages",
  "requestType": "POST",
  "json": {
    "subject": "{{s}}",
    "unitofmeasure": "{{u}}",
    "measurename": "{{m}}",
    "value": "{{v}}",
    "organization": "{{o}}",
    "displayname": "{{d}}",
    "location": "{{l}}",
    "timecreated": "{{SPARK_PUBLISHED_AT}}",
    "guid": "{{SPARK_CORE_ID}}"

  },

  "azure_sas_token": {
    "key_name": "D1",
    "key": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```

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

```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```



```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "mydevices": true
}
```



```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  },
  "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 Michael Bolton has setup already in the cloud for you to use.

```
{
  "event": "HowColdIsCodemash",
  "url": "https://cm2017-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": "30fpGKkPgzpKYXrl5nxYbb6OwIN/2h2ulVHXCsWxrpU="
  }
  "mydevices": true
}
```

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

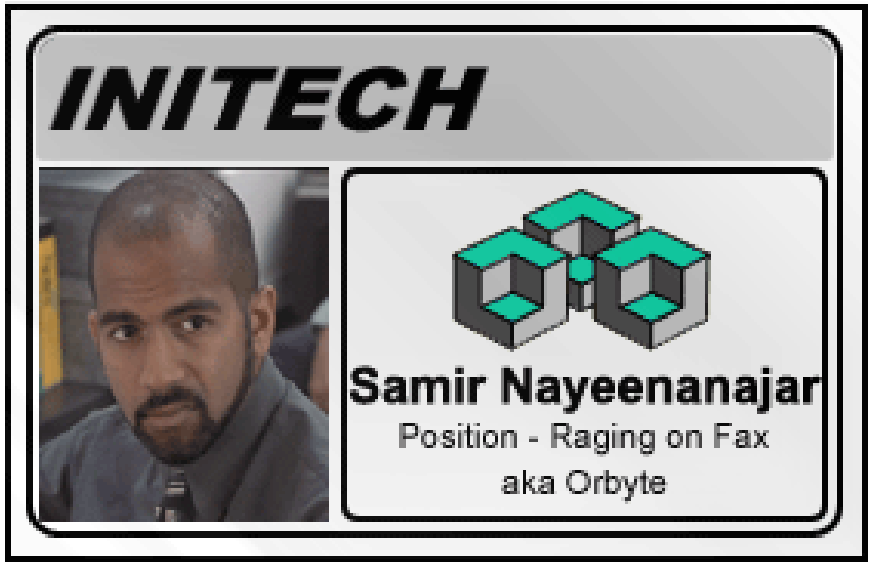
CREATE THE PARTICLE WEBHOOK

- 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

http://aka.ms/hcicsource

Microsoft

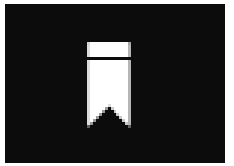
CREATE THE PARTICLEWEATHERSHIELD SKETCH

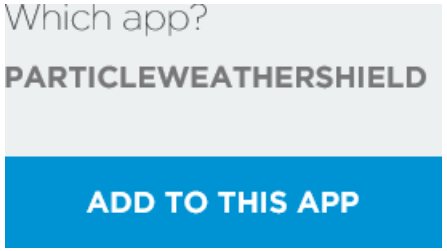
- Create a new "App" here: https://build.particle.io/build/new
- Name it "**ParticleWeatherShield**"
- Download the sketch from: aka.ms/hcicsketch
- Copy the text from the downloaded **ParticleWeatherShield.c** file
- Paste into the **ParticleWeatherShield** app in Particle Build

http://aka.ms/hcicsource

Microsoft

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[] = “Initech";
char Disp[] = “Samirs Printer";
char Locn[] = “OH";
```

Microsoft

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

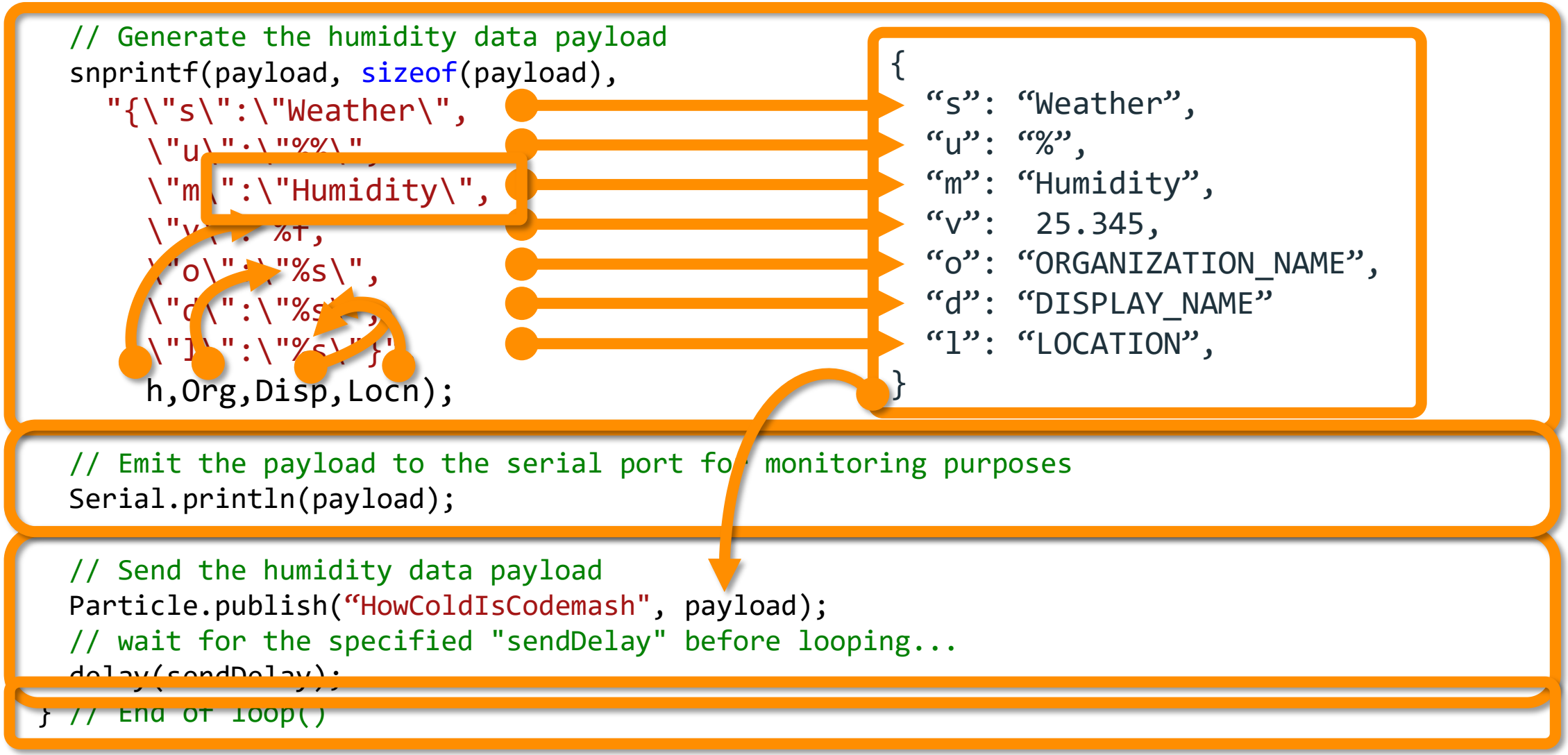
```
// Generate the temperature data payload
char payload[255];
snprintf(payload, sizeof(payload),
  "{\"s\":\"Weather\",
    \"u\":\"F\",
    \"m\":\"Temperature\",
    \"v\":%f,
    \"o\":\"%s\",
    \"d\":\"%s\",
    \"l\":\"%s\"}"
    f,Org,Disp,Locn);
```

```
{
  “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(“HowColdIsCodemash", payload);
//Wait for the specified "sendDelay" before sending the humidity data...
delay(sendDelay);
```

GROKING THE PHOTON CODE



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";
```

HAS ANYBODY SEEN A RED STAPLER?

SAVE, VERIFY AND FLASH

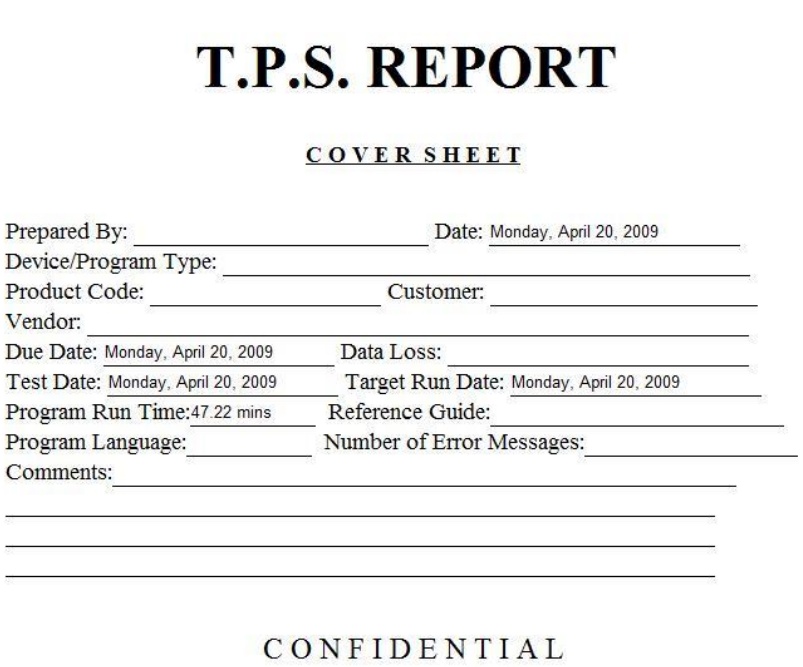
- Click the save icon to save your sketch:

- Click the verify icon to compile and verify:

- Click the flash icon to deploy your sketch:

http://aka.ms/hcicsource

Microsoft



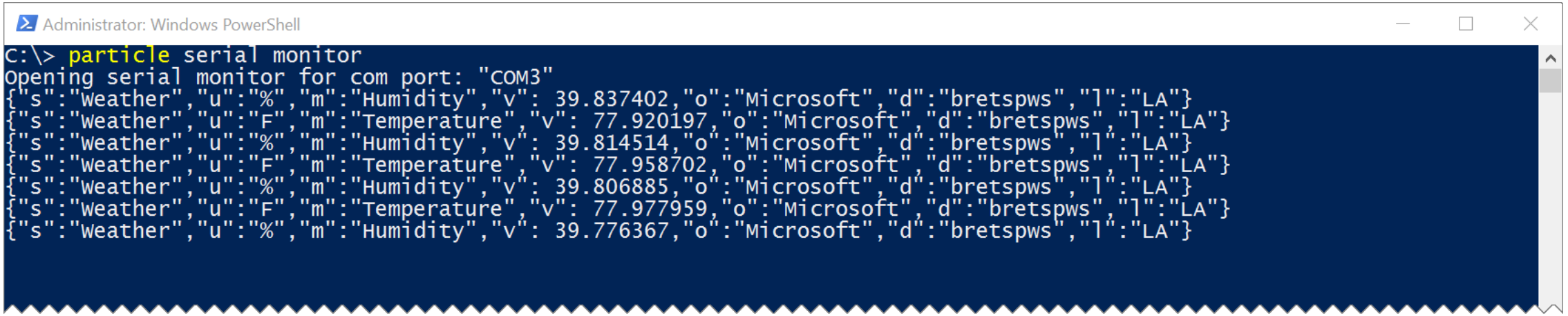
MONITORING YOUR PHOTON

http://aka.ms/hcicsource

Microsoft

MONITORING PHOTON VIA SERIAL

particle serial monitor



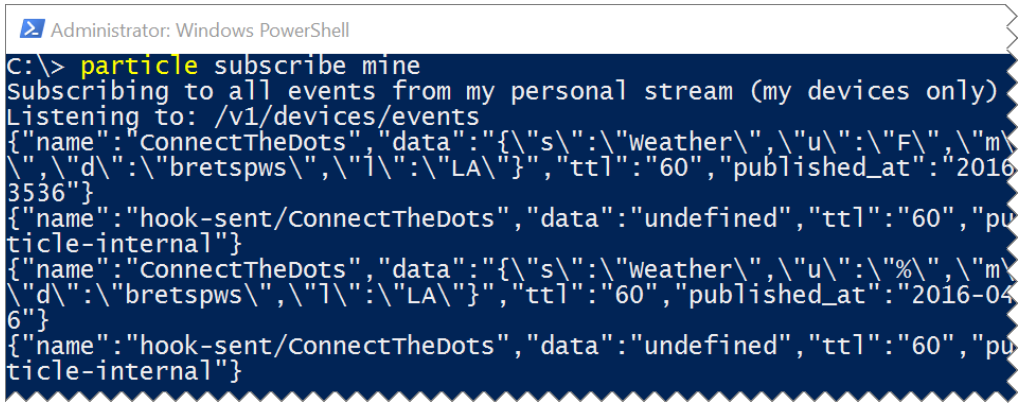
http://aka.ms/hcicsource

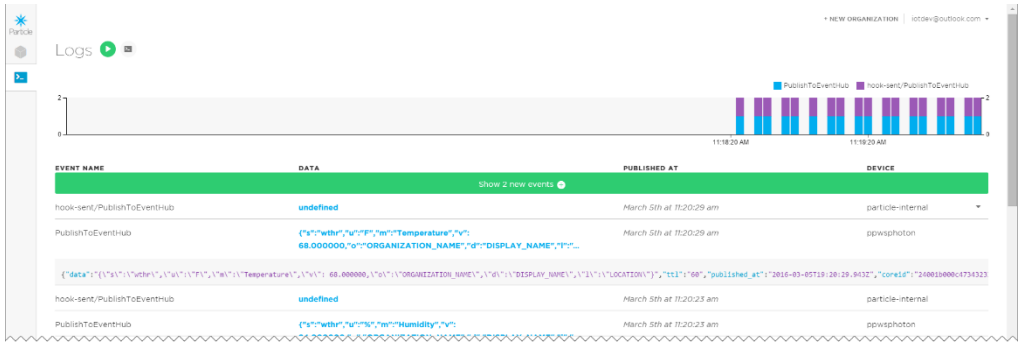
Microsoft

MONITORING WEBHOOK

- Particle-cli:
  `particle subscribe mine`

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



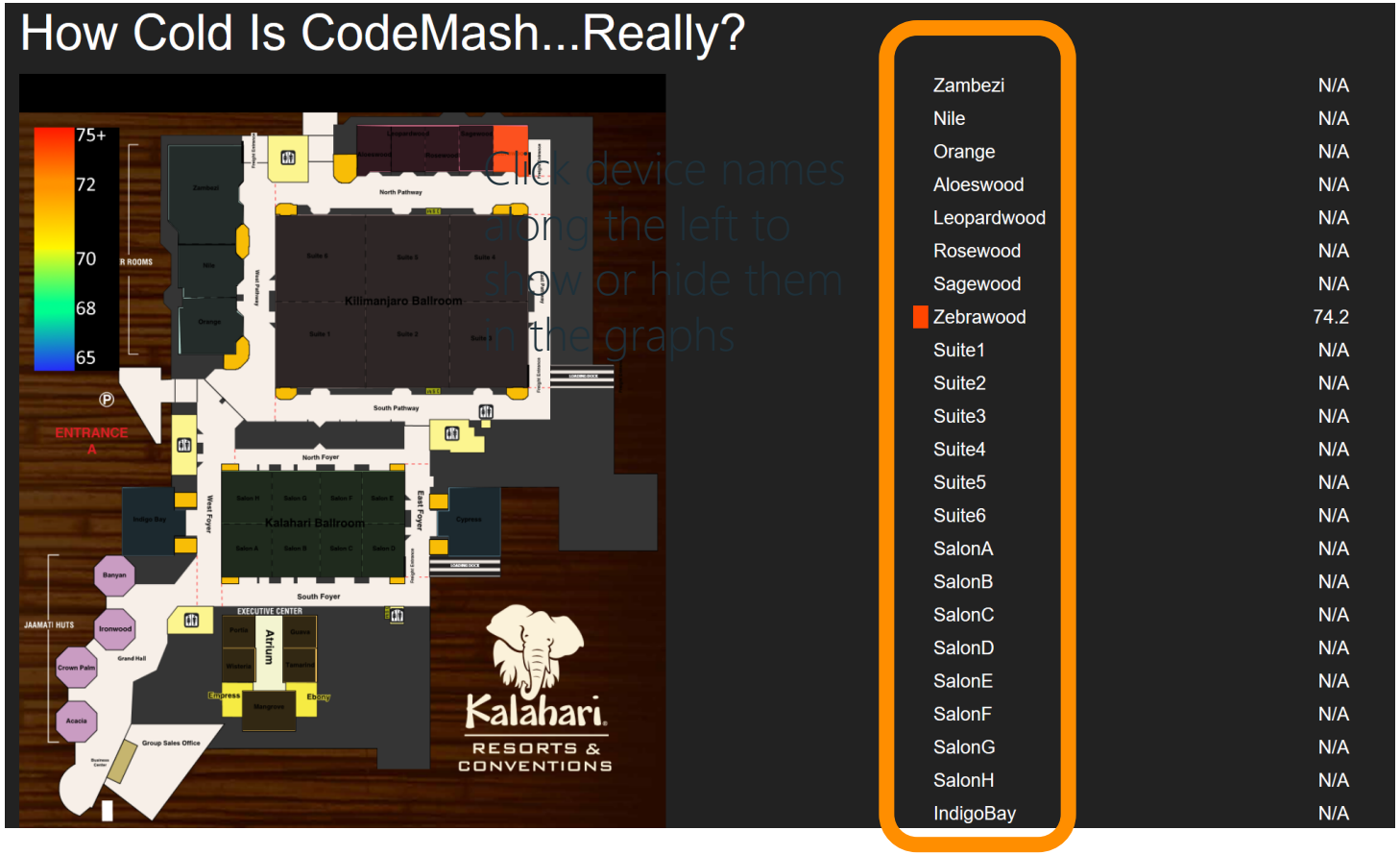


http://aka.ms/hcicsource

Microsoft

MONITOR VIA THE WEB

http://aka.ms/hcic



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

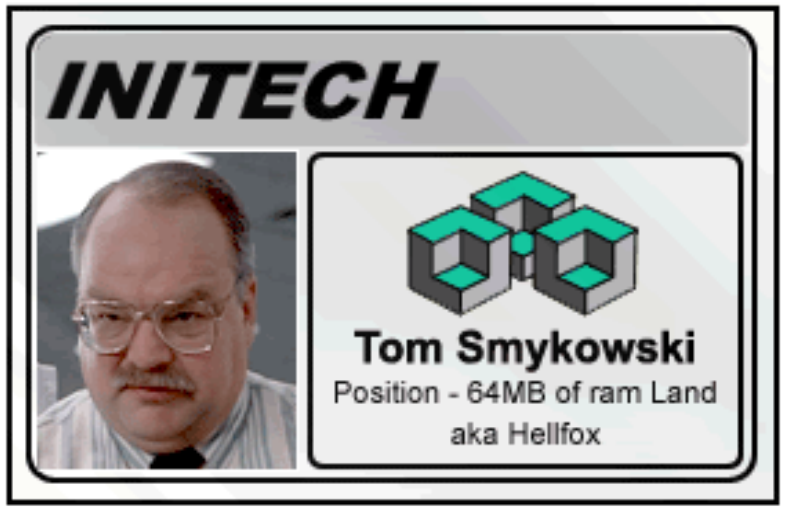
http://aka.ms/hcicsource

Microsoft

FIRE!!!!!!!!

http://aka.ms/hcicsource

Microsoft



JUMPING TO CONCLUSIONS

http://aka.ms/hcicsource

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 our 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

http://aka.ms/hcicsource

Microsoft

ADVANCE YOUR AZURE SKILLS

Build your expertise and your career with training and certification!

Whether you're new to Azure or already a cloud professional, training is one of the best investments you can make in your career. We're offering a free online training option and great deals on certification.

Go to https://aka.ms/cmazure to get started today!