SIOTLAB | SANTA CLARA UNIVERSITY
Internet of Things Research Lab

INTERNET OF THINGS | COEN 243
DEPARTMENT OF COMPUTER ENGINEERING, SANTA CLARA UNIVERSITY, CALIFORNIA
WEI XIAO, INSTRUCTOR (WXIAO@SCU.EDU)
SHIVAKUMAR MATHAPATHI, GUEST INSTRUCTOR (SMATHAPATHI@SCU.EDU)

Internet of Things

Database, Cloud And Edge Solutions

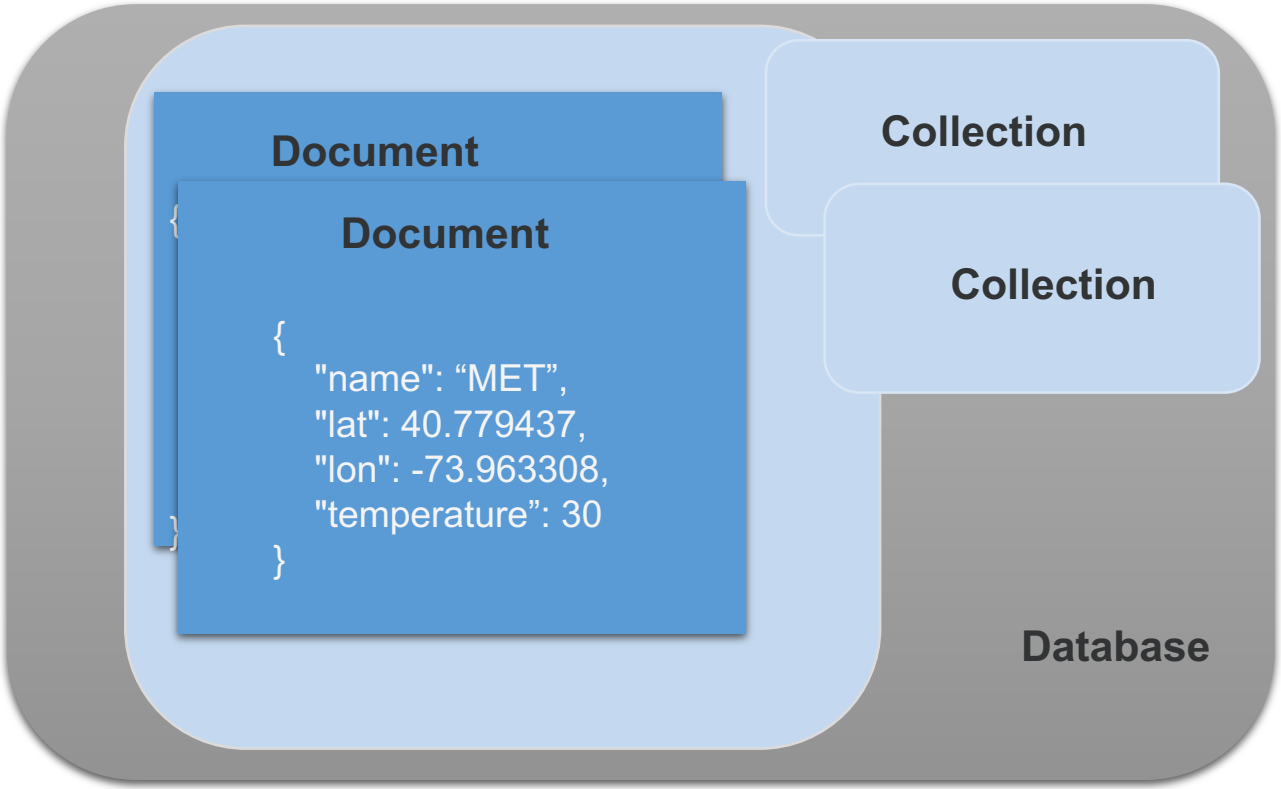
Slide Set: 1

Database Integration

MongoDB

Document, Collection, Database

- Open-source NoSQL, non-relational database
- Dynamic Schema



BSON

- Example:

```
{
        name: "Alan",
        age: 25,
        groups: [ "news", "sports”]
 }
```

- Example:

```
{
        name: { first: "Alan", last: "Turing" },
        birth: new Date('Jun 23, 1912'),
        death: new Date('Jun 07, 1954'),
        contribs: [ "Turing machine", "Turing test", "Turingery" ],
        views : NumberLong(1250000)
 }
```

BSON Types

Type	Number	Alias	Notes
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.
ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	
DBPointer	12	“dbPointer”	Deprecated.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Deprecated.
JavaScript (with scope)	15	“javascriptWithScope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Decimal128	19	“decimal”	New in version 3.4.

MongoDB Shell

- The mongo shell is an interactive JavaScript interface to MongoDB.
- You can use the mongo shell to query and update data as well as perform administrative operations.

MongoDB Host

- MongoDB Atlas https://www.mongodb.com/cloud/atlas
- mLab https://mlab.com

Other DB

- InfluxDB `https://www.influxdata.com/`
- Couchbase `https://www.couchbase.com/`

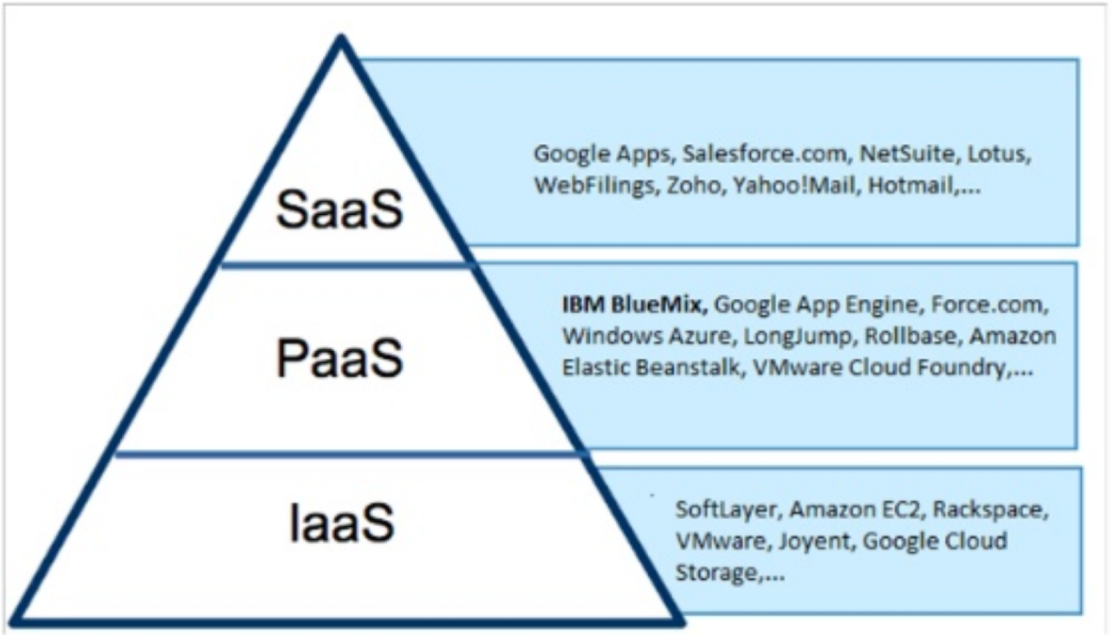
Cloud Computing

Cloud Computing

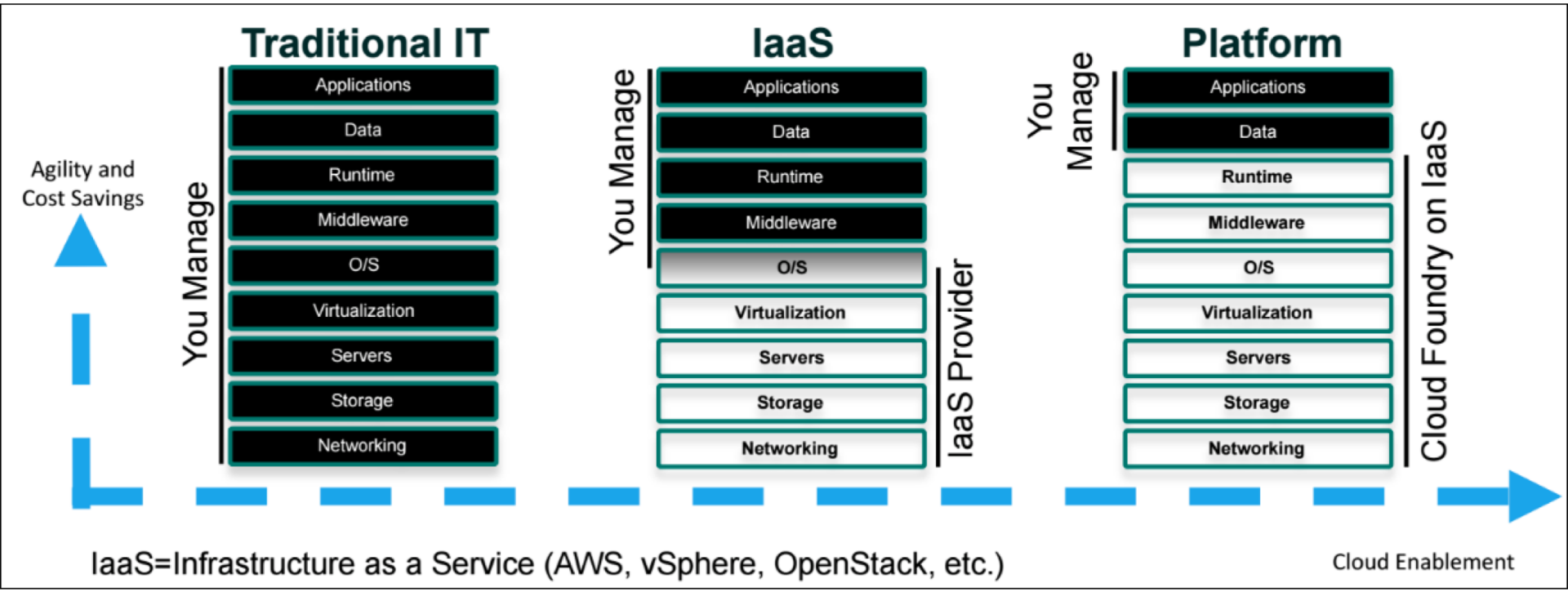
Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources(such as networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction

Cloud Computing

- **SaaS (Software as a Service)** is something you simply sign up for and generally start using with little or no customization.
- **PaaS (Platform as a Service)** is something which you use to build or run your own applications —e.g. a managed environment for running code or a DB.
- **IaaS (Infrastructure as a Service)** is core IT infrastructure, you either rent a virtual server, or to provide solutions for file storage, load balancing or some other “network type” problem.



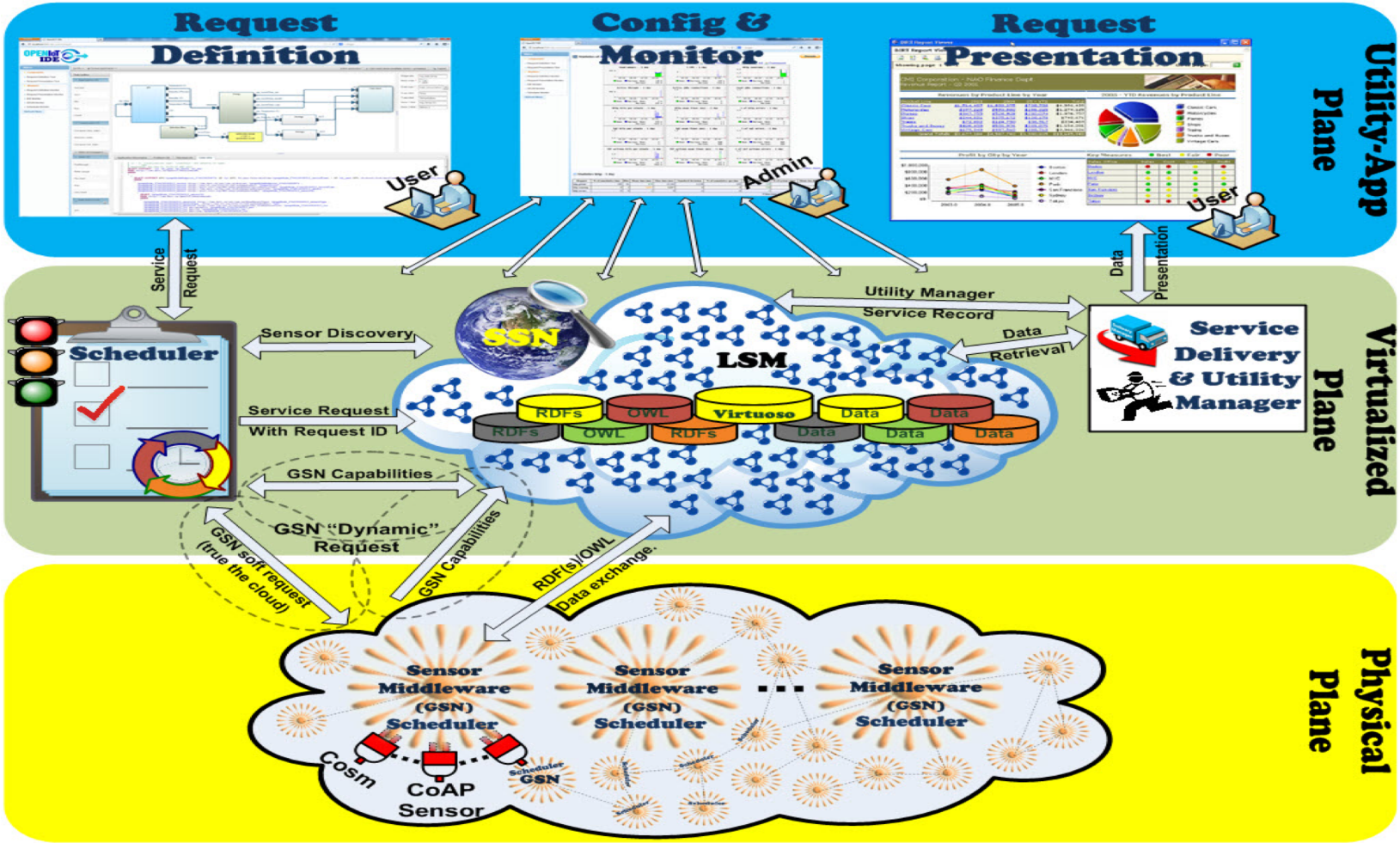
Cloud Computing



Source: CloudFoundry.com

Cloud Platforms

Open IoT



AWS IoT

AWS IoT is a fully managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.

1

Securely connect and manage any physical device across multiple networks and protocols

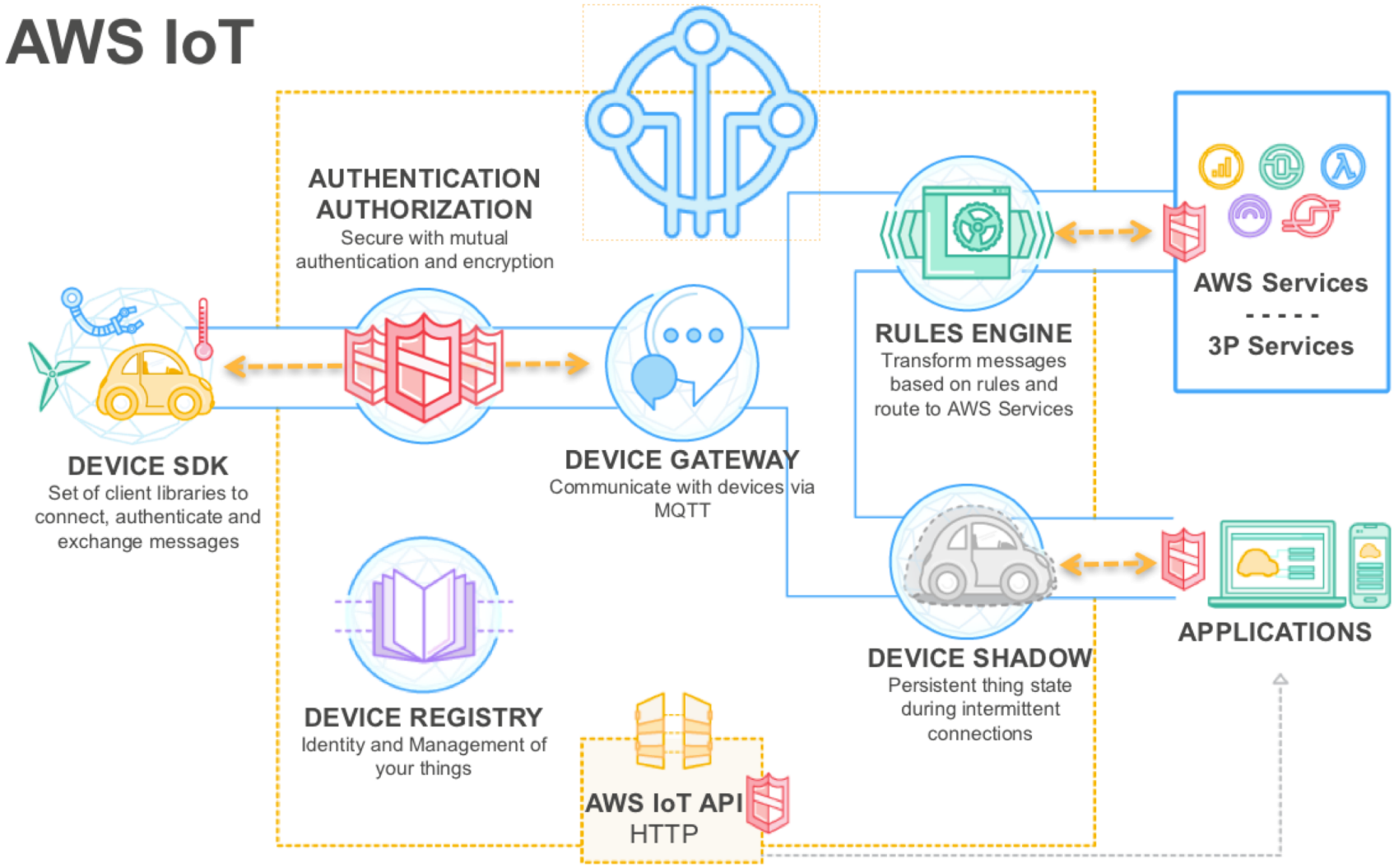
2

Extract and Filter data from your devices and take action with custom Rules

3

Create Web and Mobile Applications that interact with Devices reliably at any time

AWS IoT



Source: aws.amazon.com

Device Gateway

- Standard protocols – MQTT / HTTP
- AWS front-end for devices
- Scalable, long-lived connections
- Secure by default – X509 certs, TLS 1.2

Topic Based Architecture:

❑Example: `scu/computerengineering/coen243`

Device Registry

- Device meta-data

❑ Example:

```
$ aws –region us-east-1 iot describe-thing –thing-name pump1
{
   “attributes”: {
          “MN”:”AB776”,
          “install_data”: “12-29-2016”,
          “SN”: “2542678921”
    },
   “thingName”: “pump1”,
   “defaultClientId”: “pump1”
}
```

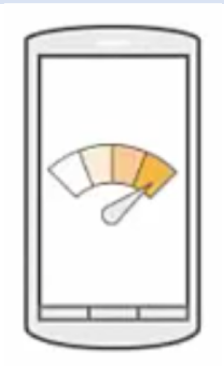
Device Shadows

- Report its current state to one or multiple shadow
- Retrieve its desired state from shadow

- Shadow reports delta, desired and reported states along with metadata and version

- Set the desired state of a device
- Get the last reported state of the device
- Delete the shadow

Device Shadows



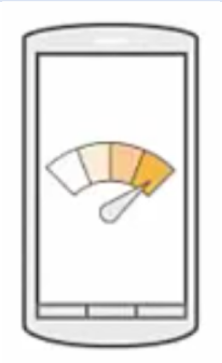
"engine": "ON"

"engine": "OFF"

"engine": "ON"

```
{
"state": {
  "desired": {
    "engine": "ON",
    "light": "OFF"
  }
  "reported": {
    "engine": "OFF",
    "light": "OFF"
  }
  "delta": {
    "engine": "ON",
 }
 }
"version": 1
}
```

Device Shadows

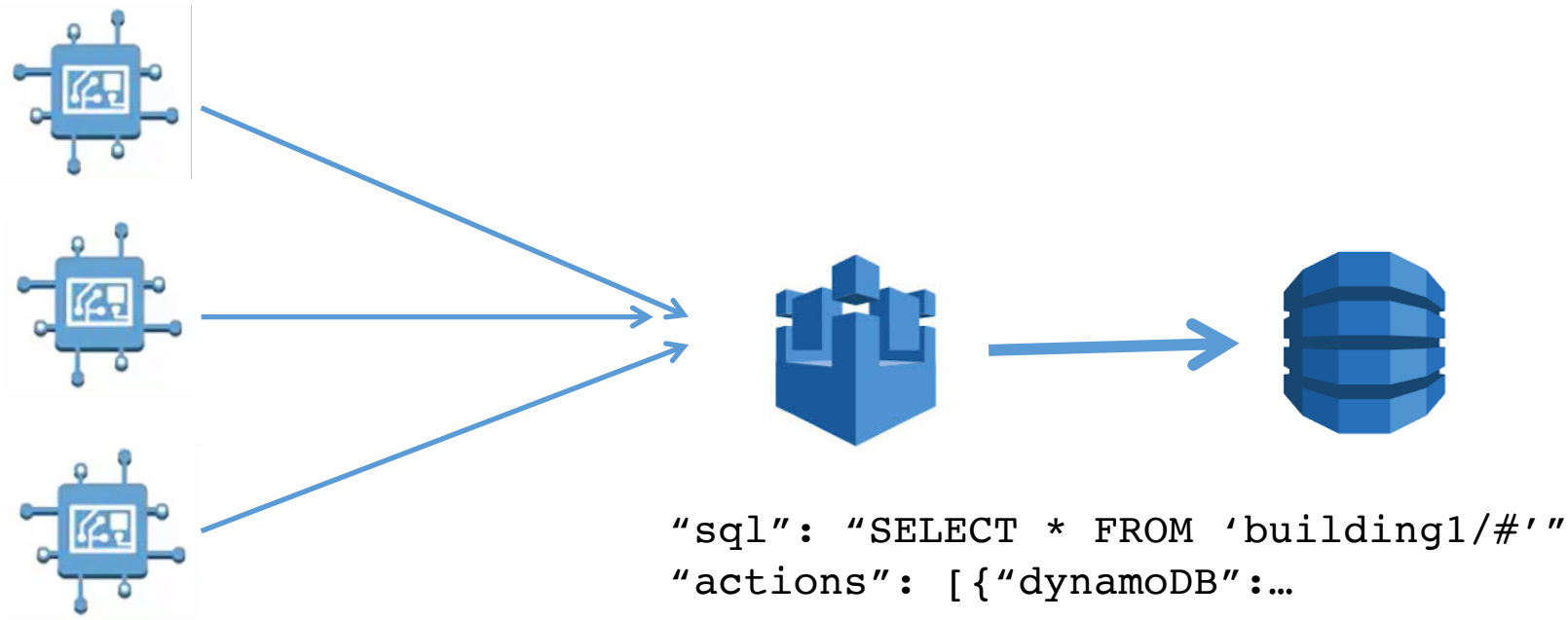


"engine": "ON"

```
{
"state": {
  "desired": {
    "engine": "ON",
    "light": "OFF"
  }
  "reported": {
    "engine": "ON",
    "light": "OFF"
  }
}
"version": 1
}
```

Rules Engine

- Transform message based on rules and route to AWS Services

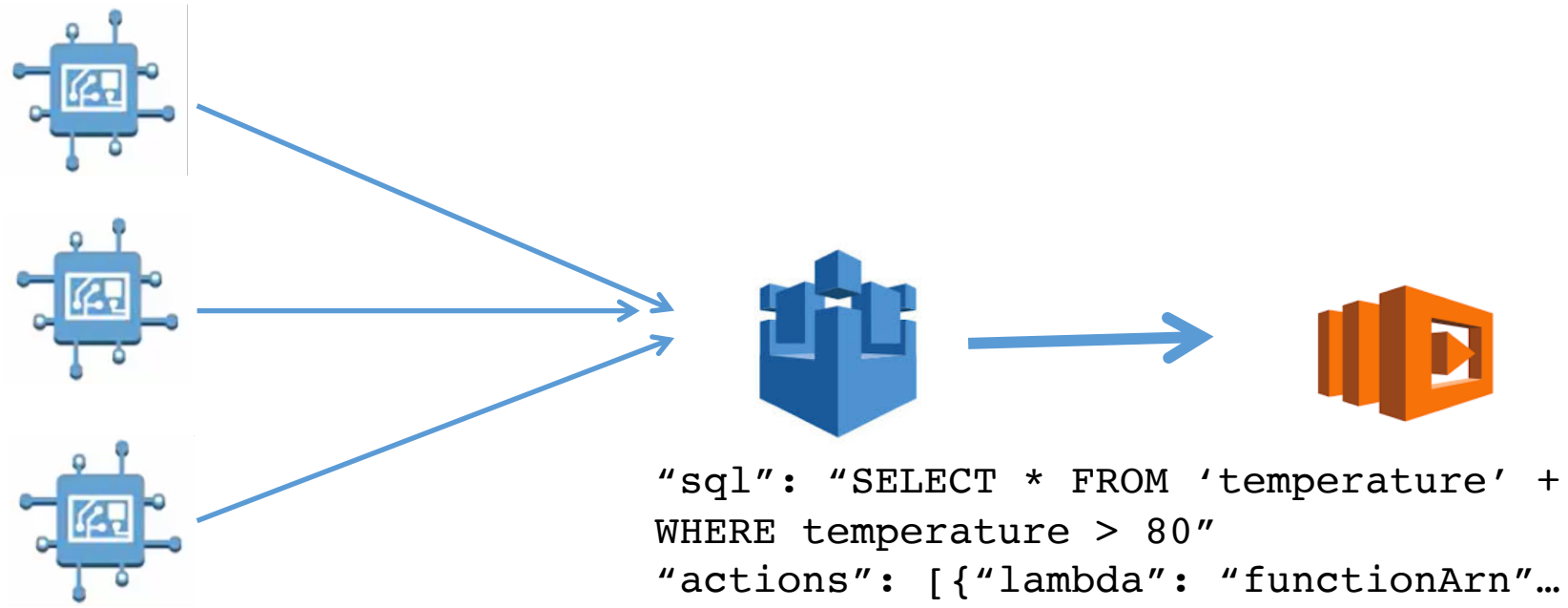


```
"sql": "SELECT * FROM 'building1/#'"
"actions": [{"dynamoDB":…
```

```
building1/story1/temperature
building1/story2/temperature
building1/story1/humidity
building2/story1/humidity
building2/story2/temperature
```

Rules Engine

- Transform message based on rules and route to AWS Services



```
"sql": "SELECT * FROM 'temperature' +
WHERE temperature > 80"
"actions": [{"lambda": "functionArn"…
      {"sns":{"topicArn":....
```

```
building1/story1/temperature
building1/story2/temperature
building1/story1/humidity
building2/story1/humidity
building2/story2/temperature
```

Rules Engine

Integration:

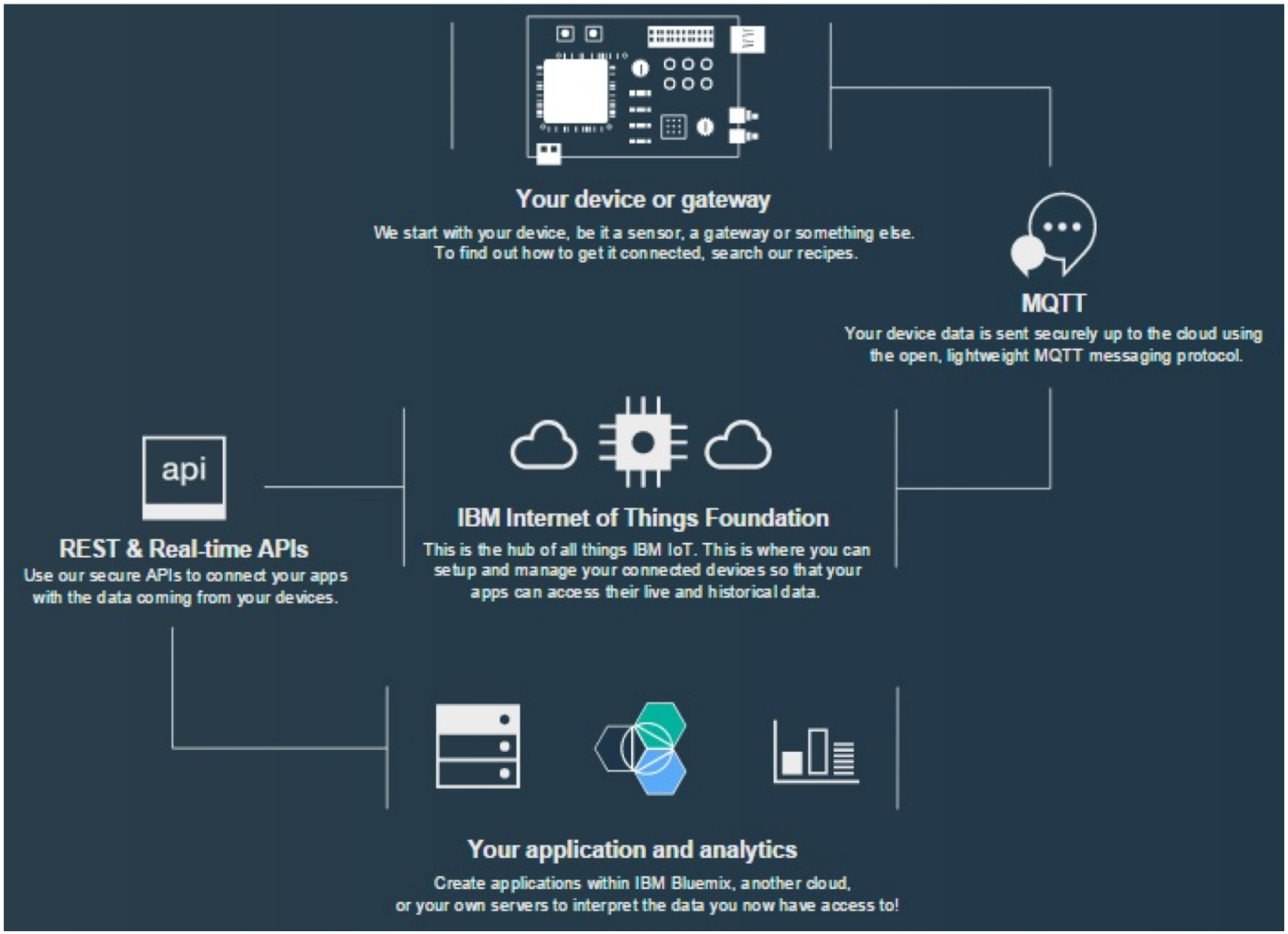
- DynamoDB
- Kinesis
- Lambda
- S3
- SNS
- Firehose
- SQS
- Republish(AWS IoT)

Device SDKs

- C-SDK: RTOS, MCU
- JS-SDK: Embedded Linux Platforms
- Arduino:
- Mobile SDK: Android and iOS

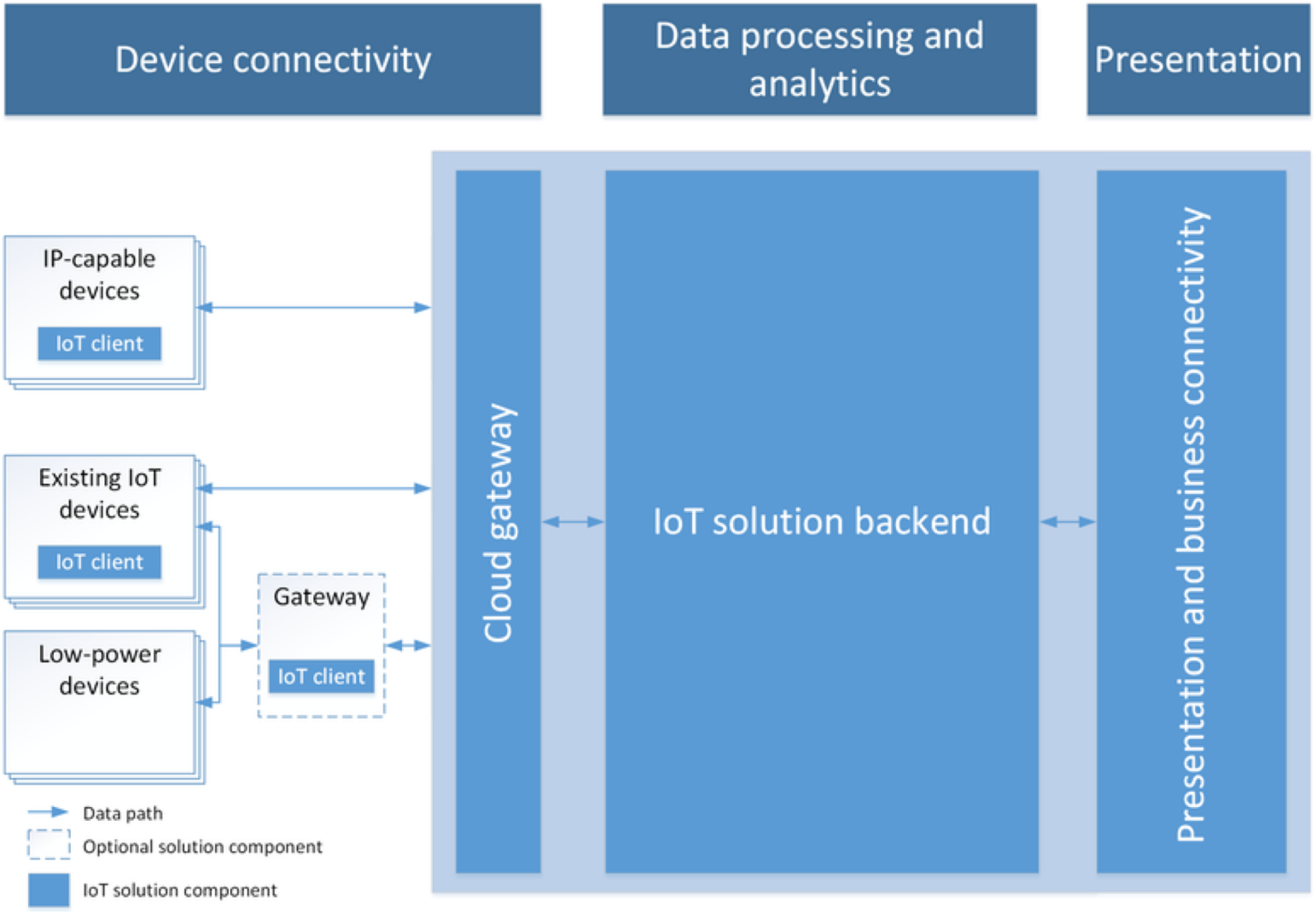
Cloud Platforms

IBM Bluemix



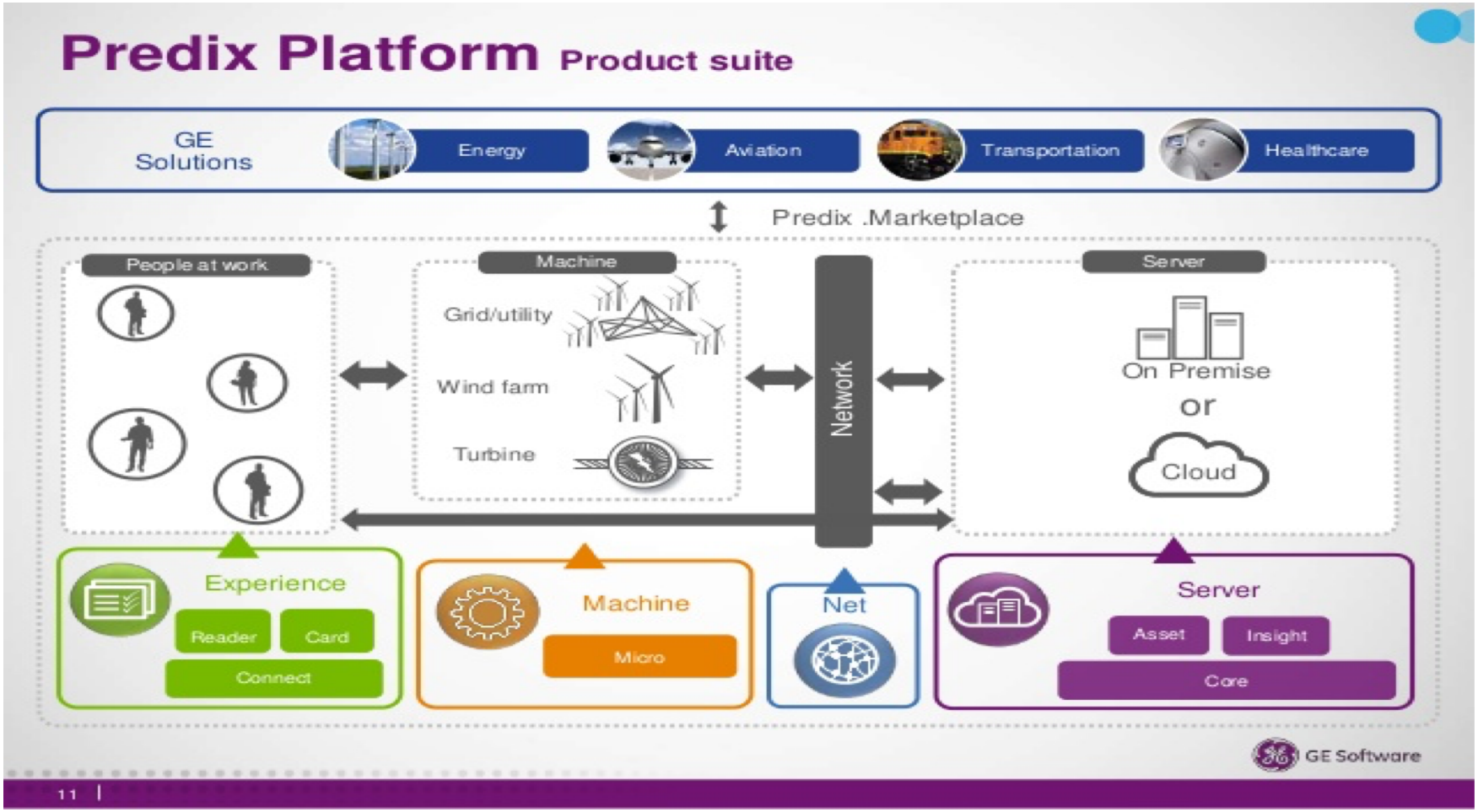
Cloud Platforms

Microsoft Azure



Cloud Platforms

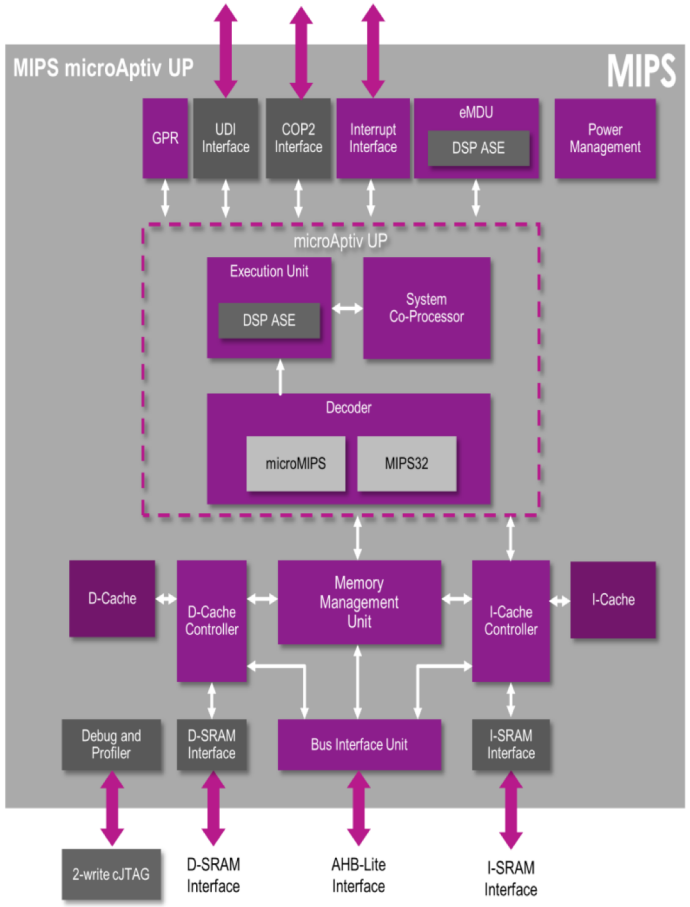
GE Predix

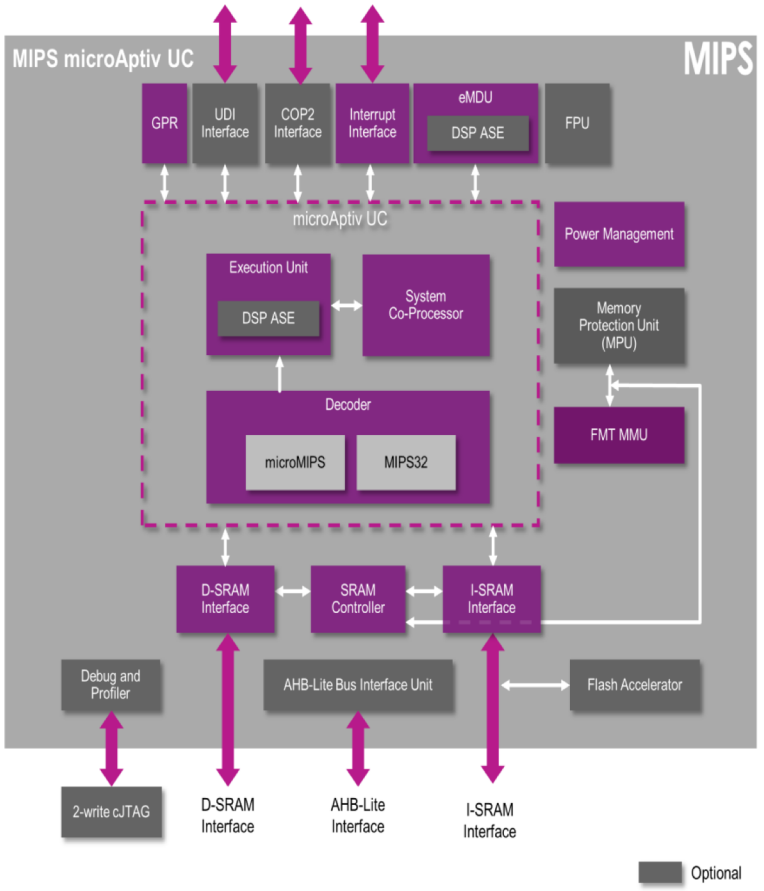


Cloud Platforms

Texas Instrument Exosite

Samsung ARTIK





ARTIK Cloud

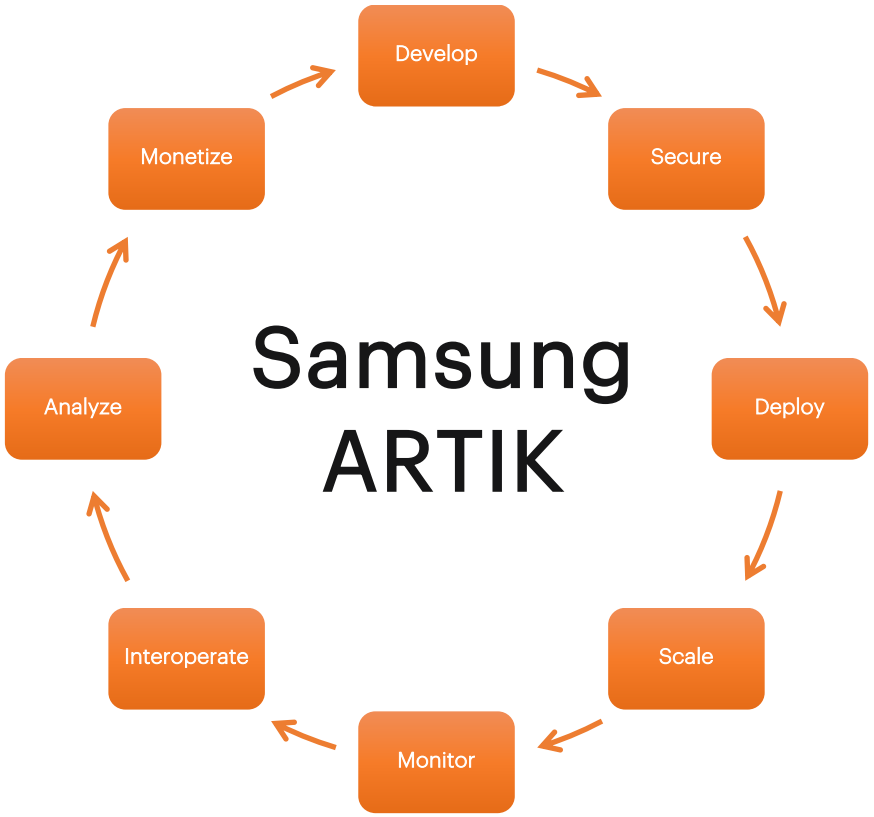
ARTIK makes it easy to manage the

entire IoT lifecycle

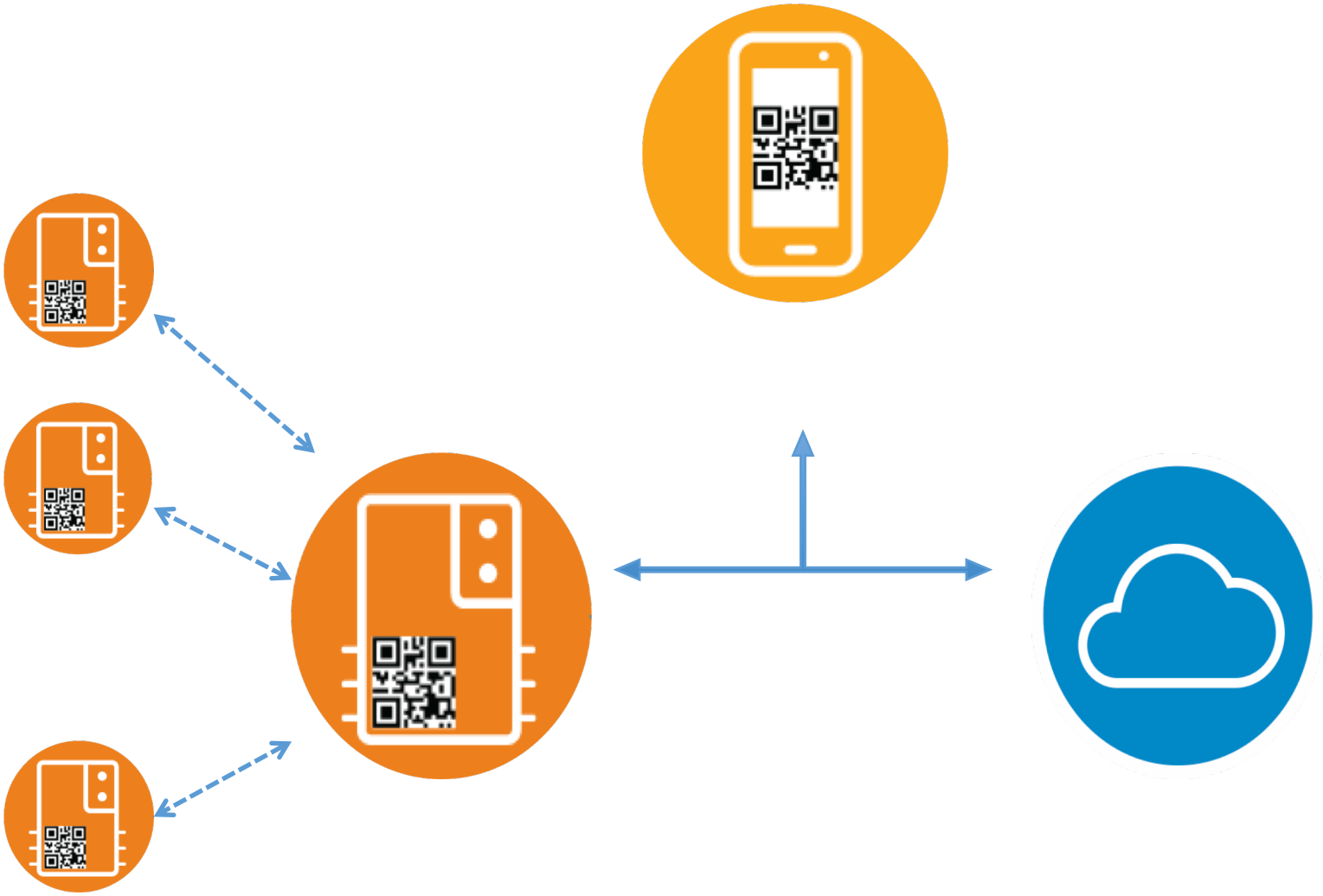
from things

to secure and interoperable

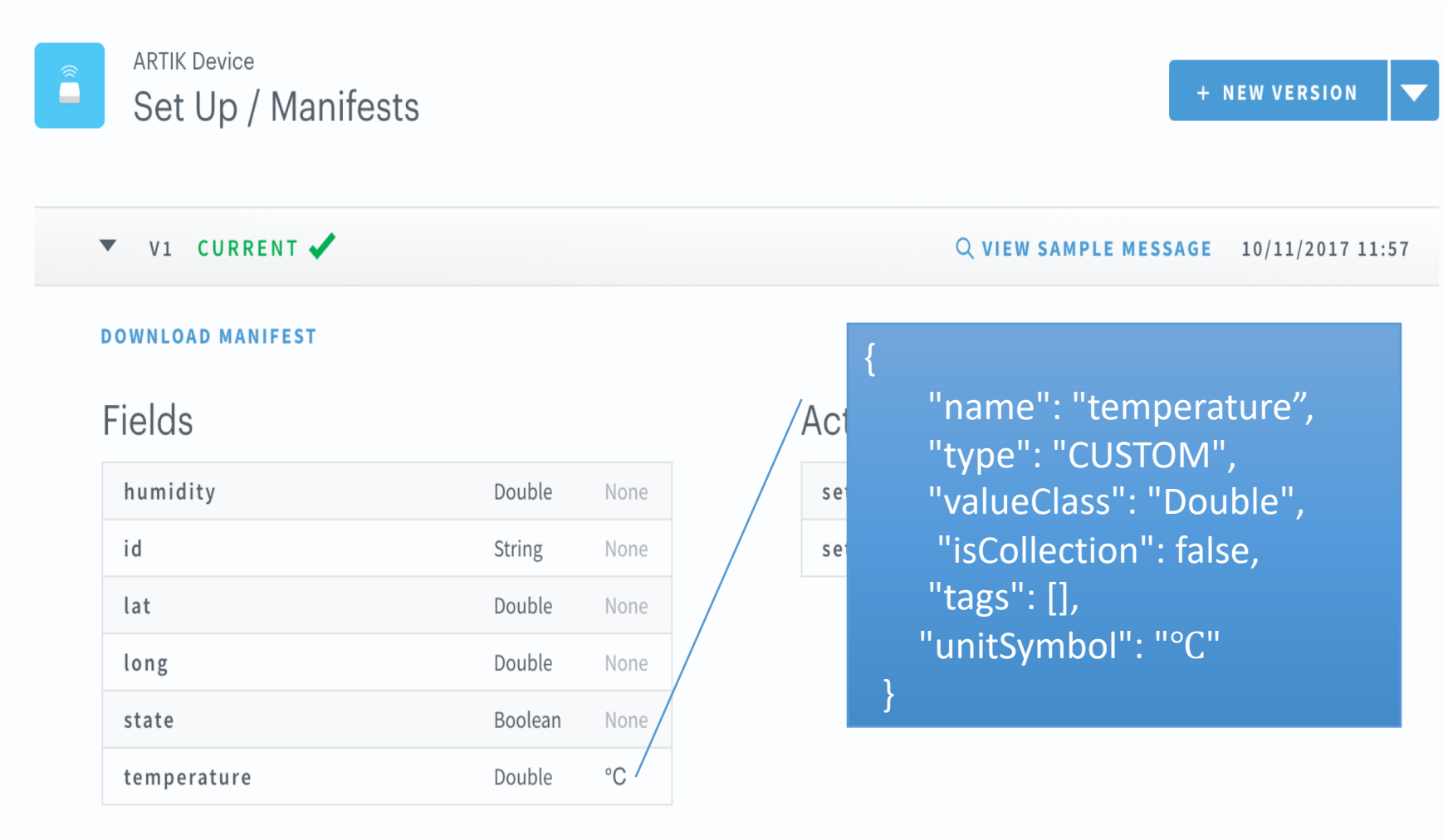
IoT products and solutions



Easy onboarding



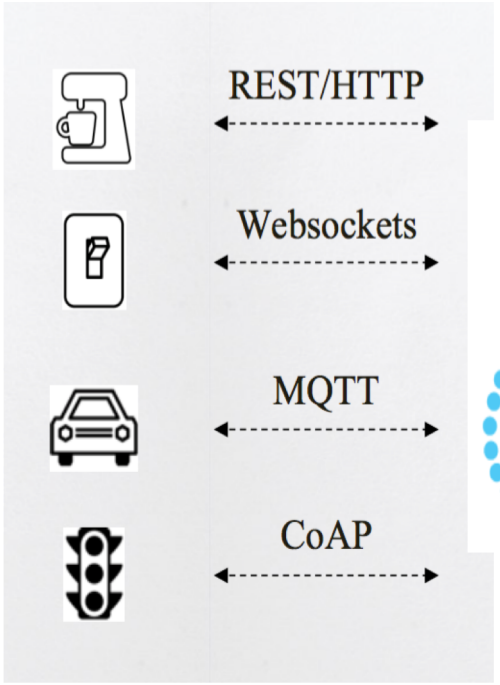
Device Type, Manifest



Platform / Cloud Software Development

Platform Development:

- 05x/5/7: ARTIK SDK / IDE
- Other Options for Gateway devices



Cloud Development:

Easy to use open APIs

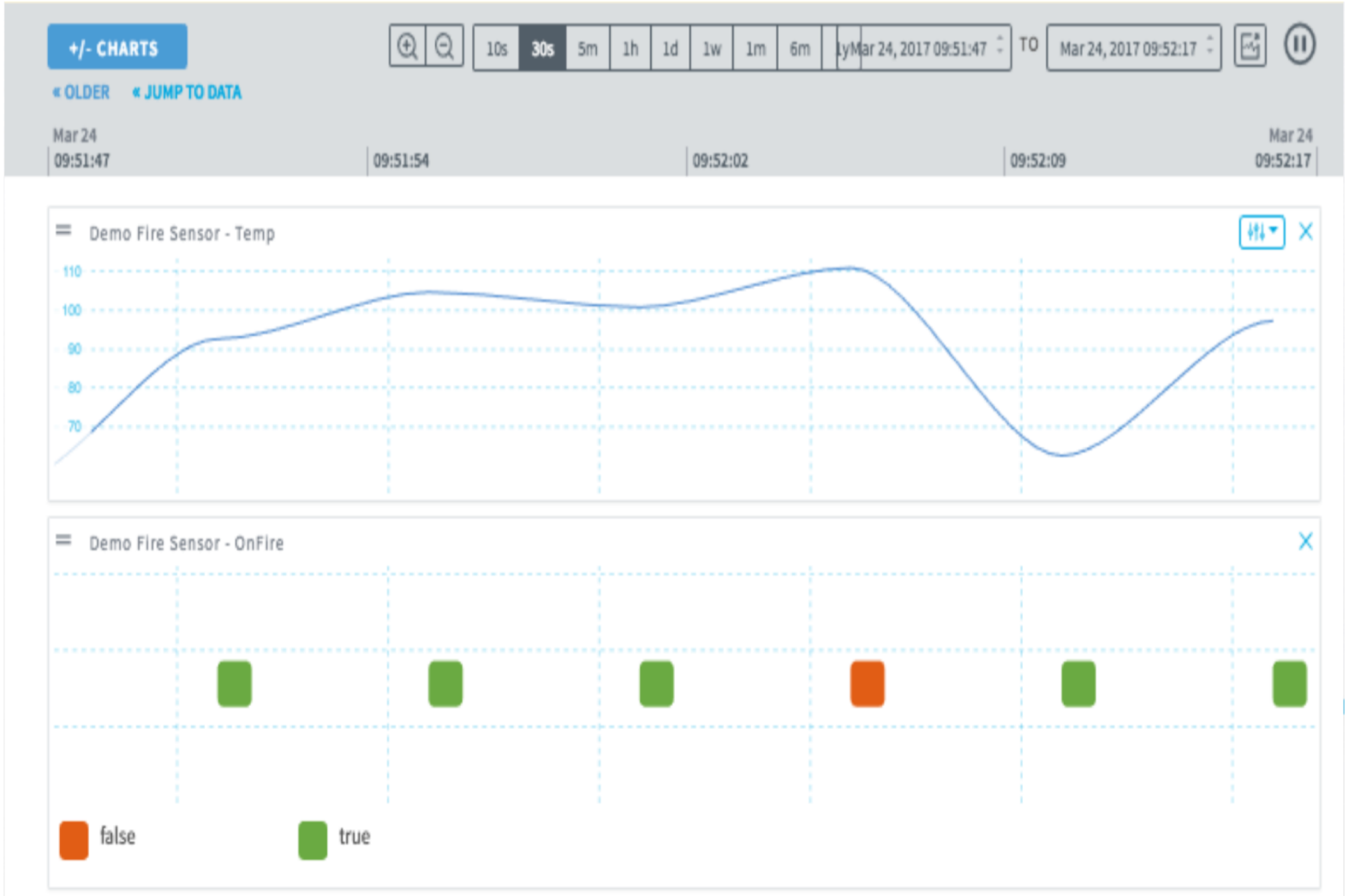
10 Mobile & platform SDKs

Developer Portal

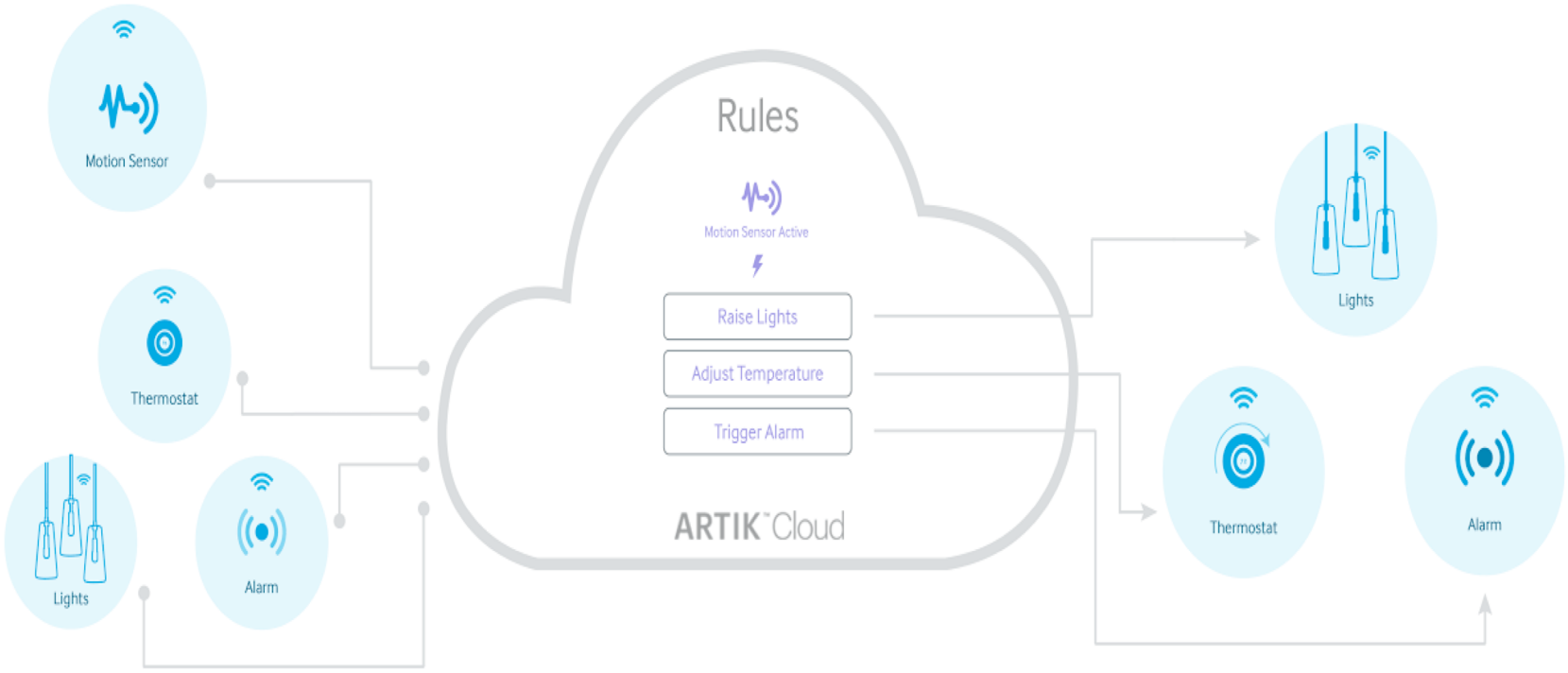
API Console

Device Simulator

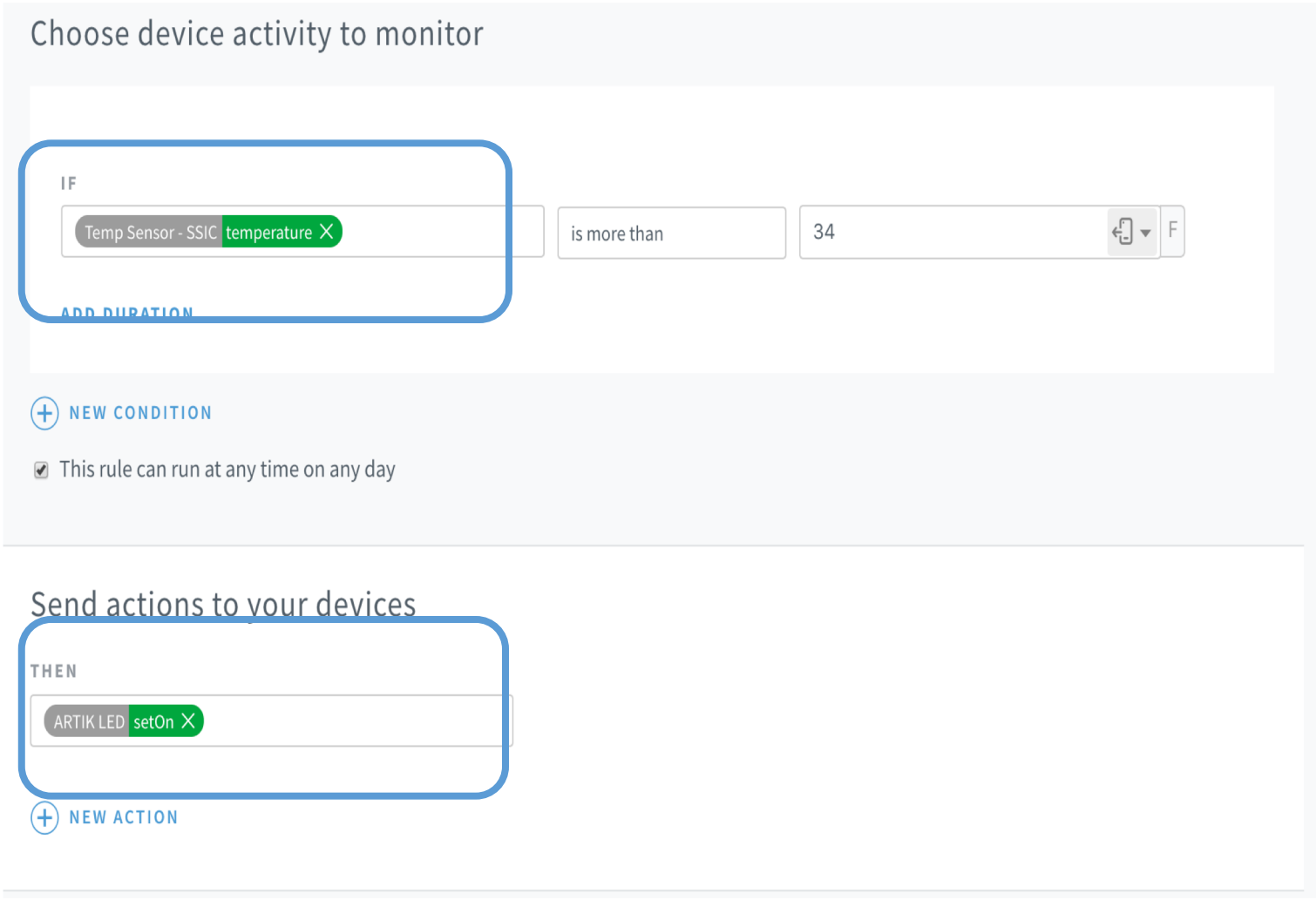
Data Visualization



Orchestration Engine

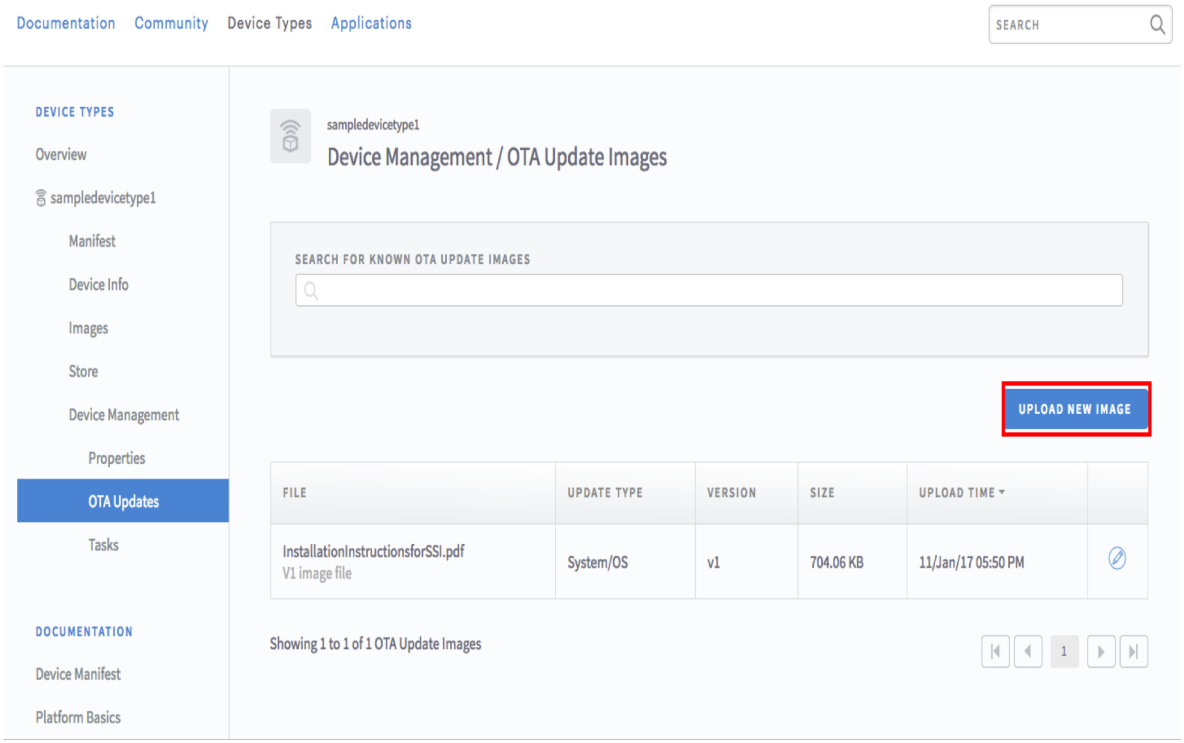


Rules


Choose device activity to monitor
IF
Temp Sensor - SSIC
temperature
is more than
34
F
ADD DURATION
NEW CONDITION
This rule can run at any time on any day
Send actions to your devices
THEN
ARTIK LED
setOn
NEW ACTION

Fleet Management & Over the Air Update

- Based on Lightweight Machine to Machine(LWM2M)
- Monitors device presence, also properties like firmware version memory usage, battery consumption etc.
- OTA



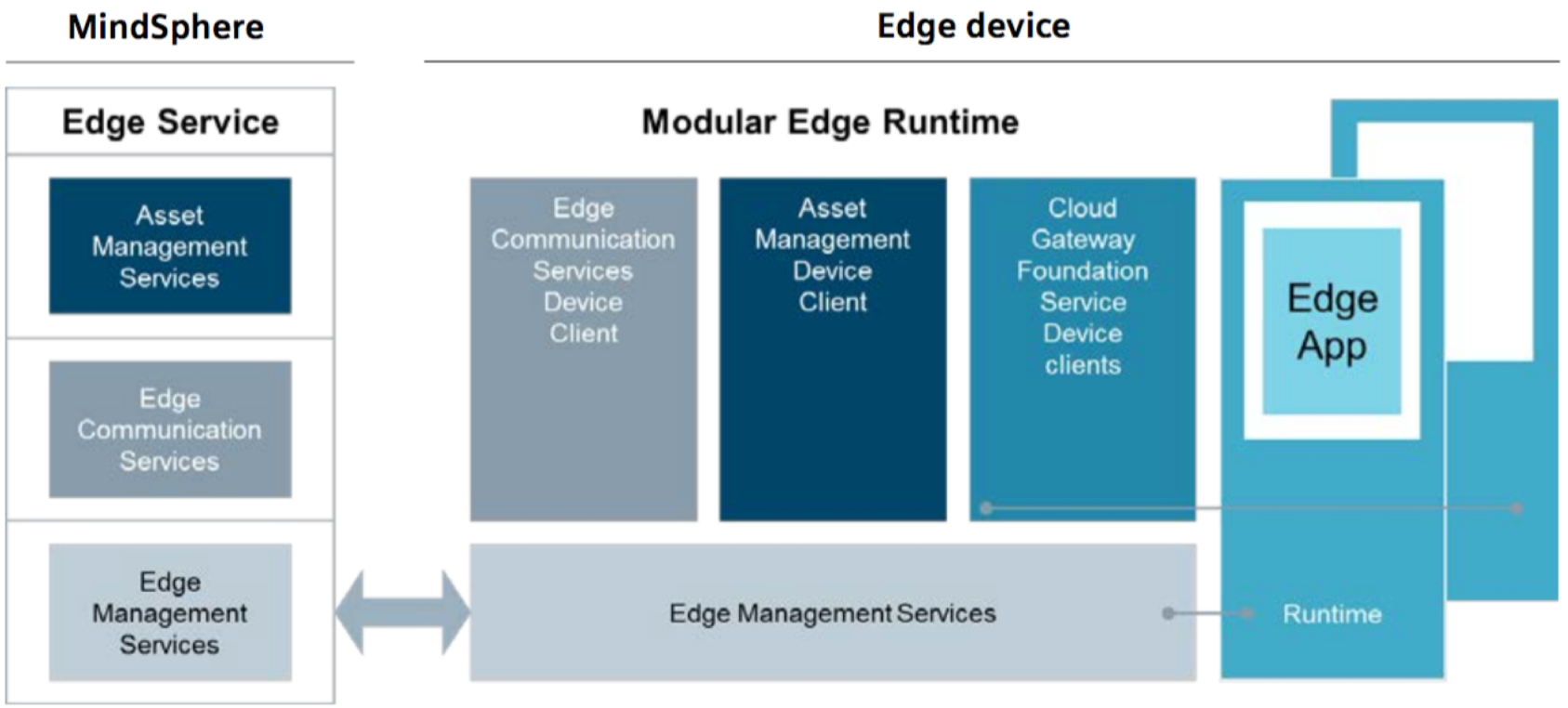
IIOT: Thingworx (PTC)

- PTC ThingWorx is an application development platform for IoT
- Including Thingworx Foundation, Analytics, Kepware
- AR integration
- Cloud connectors to AWS, Microsoft Azure

IIOT: Siemens MindSphere

- MindSphere is the cloud-based, open IoT operating system that connects real things to the digital world, and enables powerful industry applications and digital services to drive business success.
- MindSphere is based on CloudFoundry. It is a Platform as a Service (PaaS) enables a rich partner ecosystem to develop and deliver new applications.
- Open standards and interfaces: e.g, OPC Unified Architecture(OPC UA) standard, a M2M communication protocol for industrial automation interoperability.

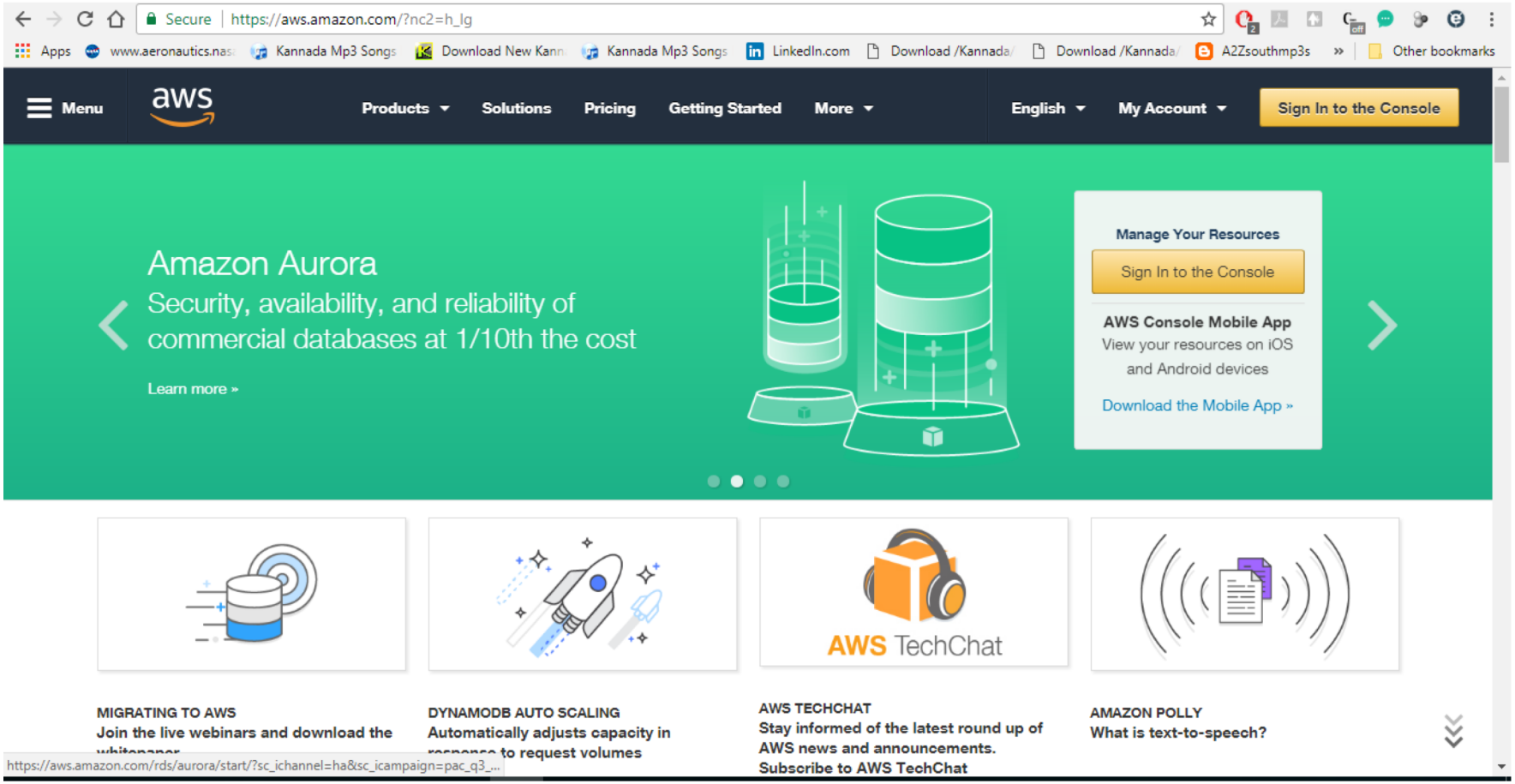
IIOT: Siemens MindSphere



Setting Up AWS IoT

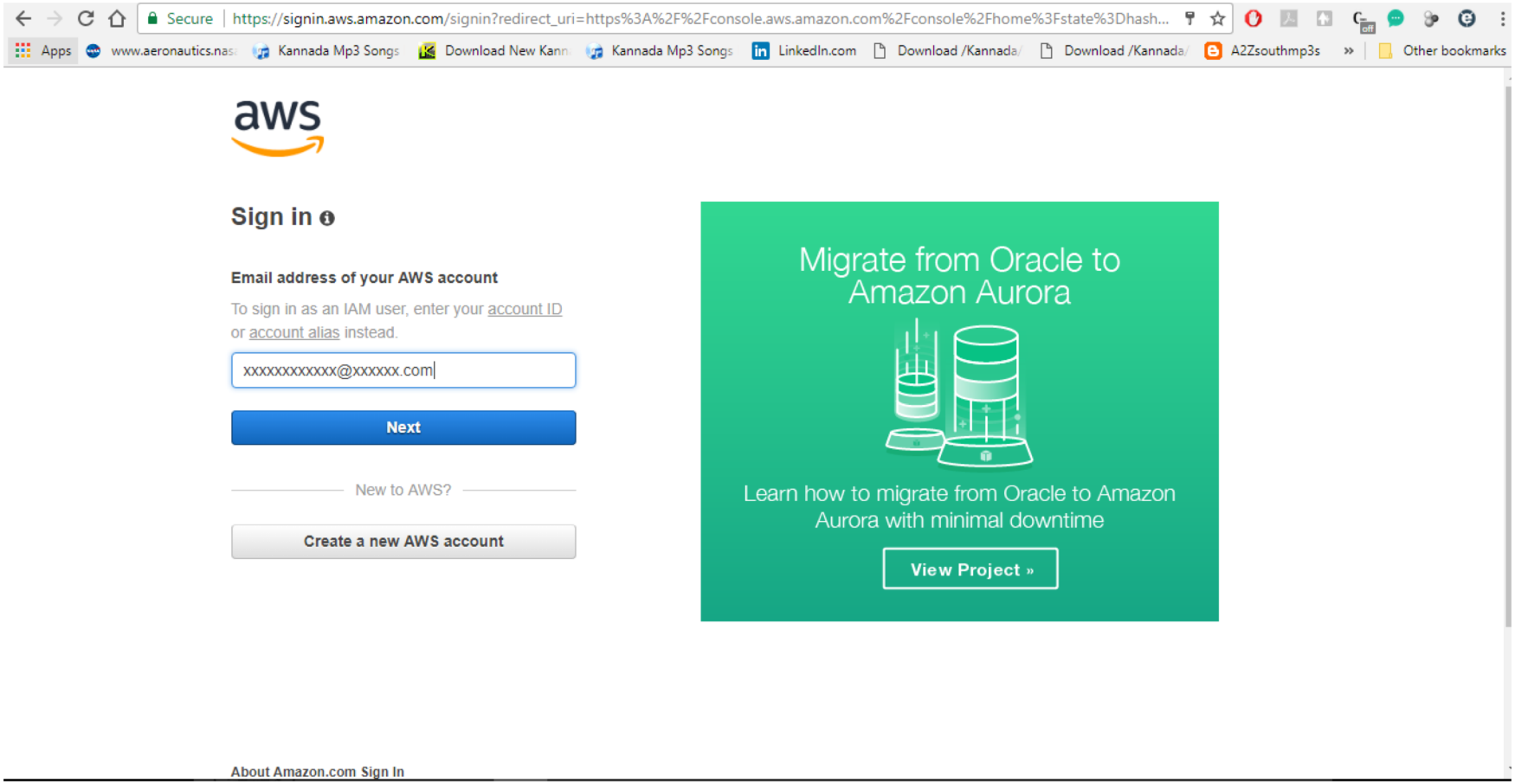
Setting Up AWS IoT

1. Open AWS console and click on Sign In to the Console.



Setting Up AWS IoT

2. If you are a new user then select I am a new user and click on Sign in using our secure server or use the existing credentials.



Setting Up AWS IoT

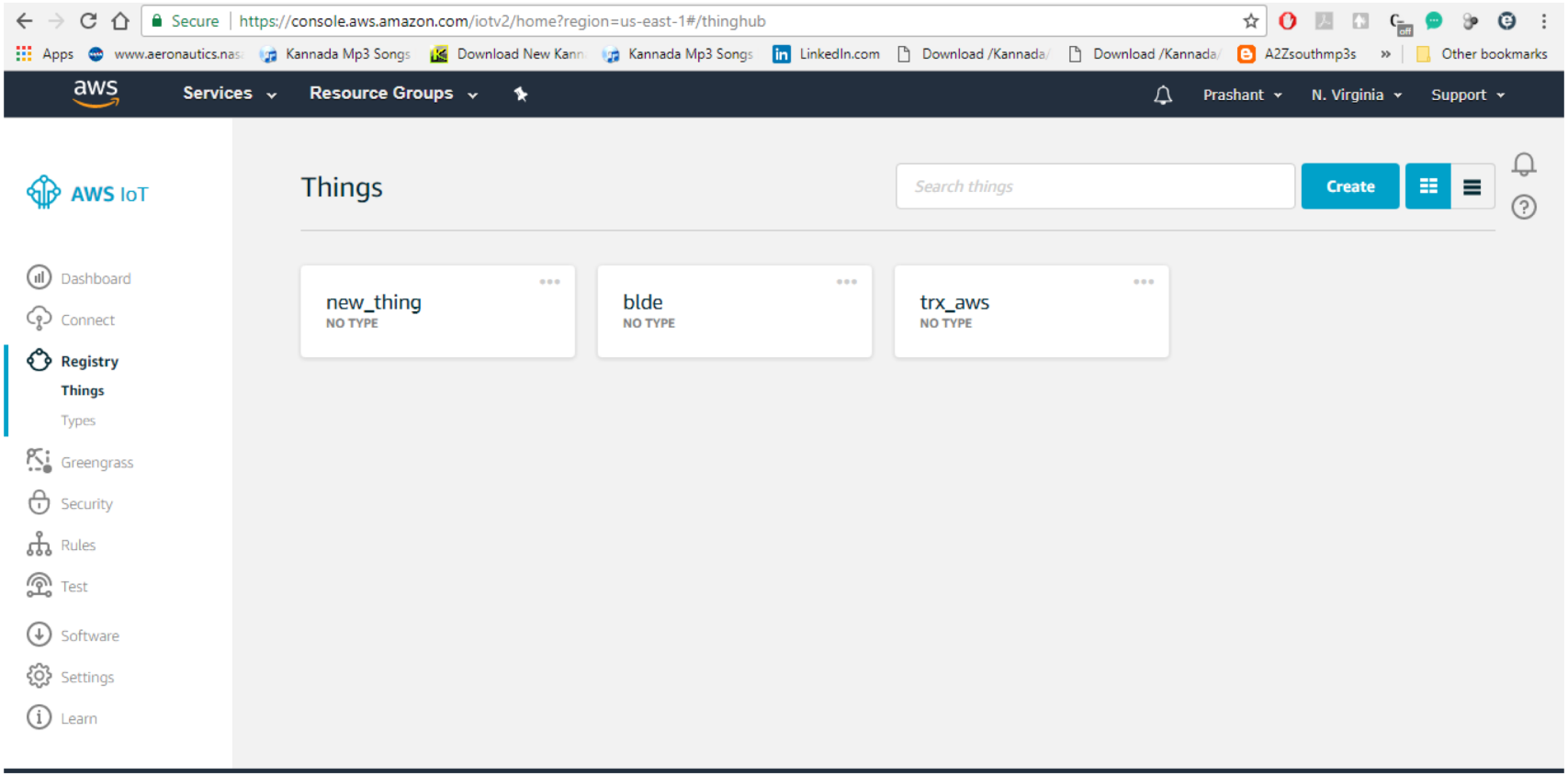
3. Then fill up the form and create and verify your account.

4. Sign in to your aws account and click on AWS IoT in the management console.

5. Now start creating thing as follows.

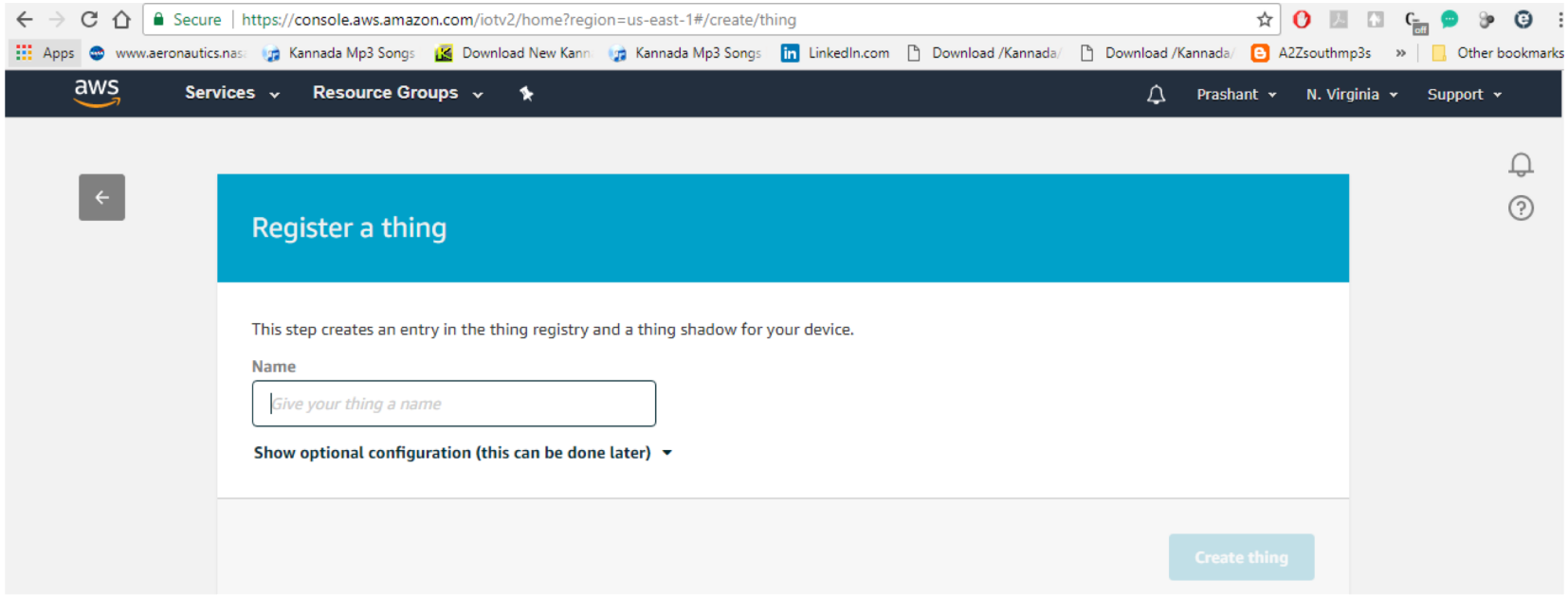
Configuring your AWS IoT Service

6. Go to AWS IoT and open up the AWS IoT Dashboard. Now let us create a Thing on the AWS IoT dashboard by clicking on the “Registry” and “Create” button.



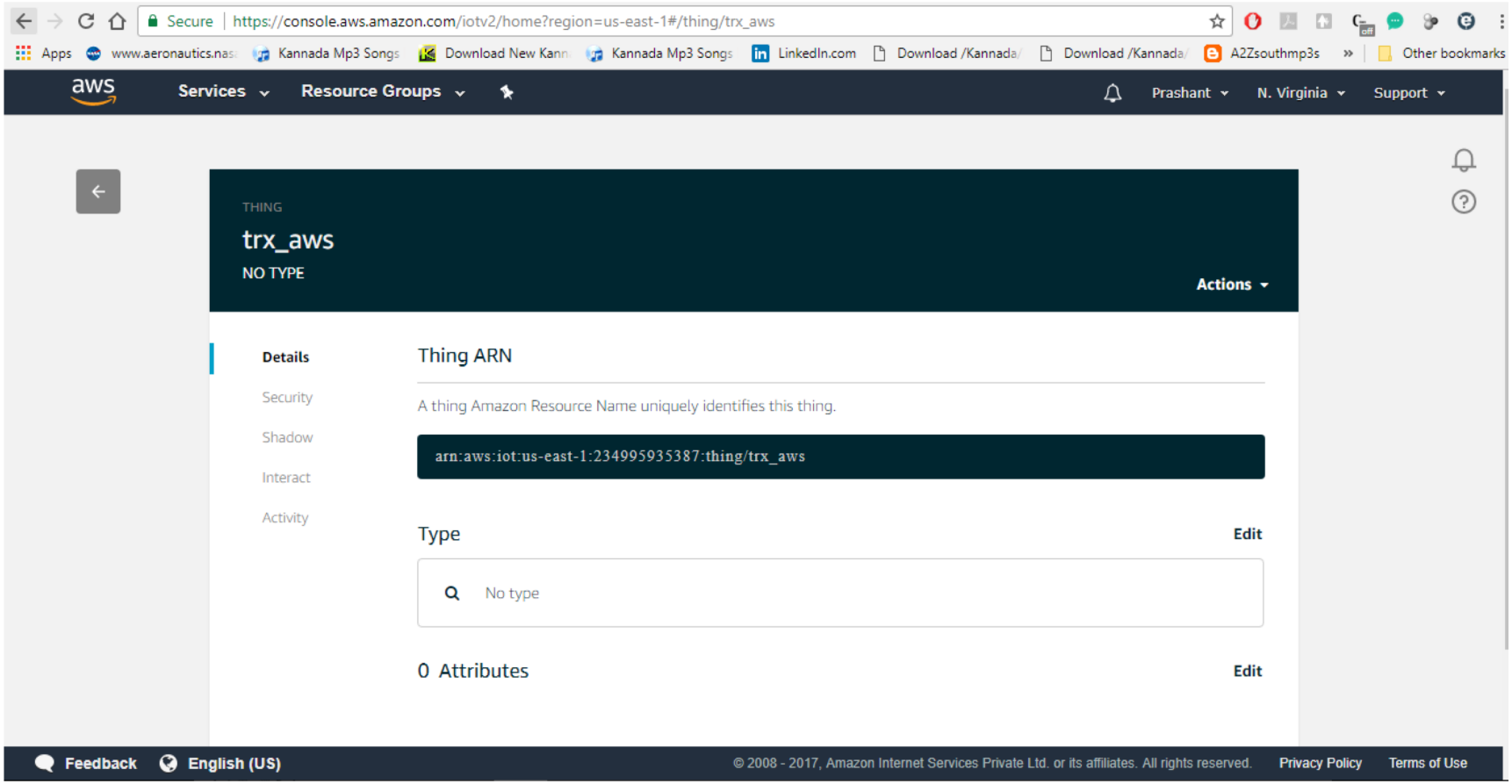
Register a Thing

7. Please make a note of this thing-name, we will have to configure the starter kit with this exact same name later on.



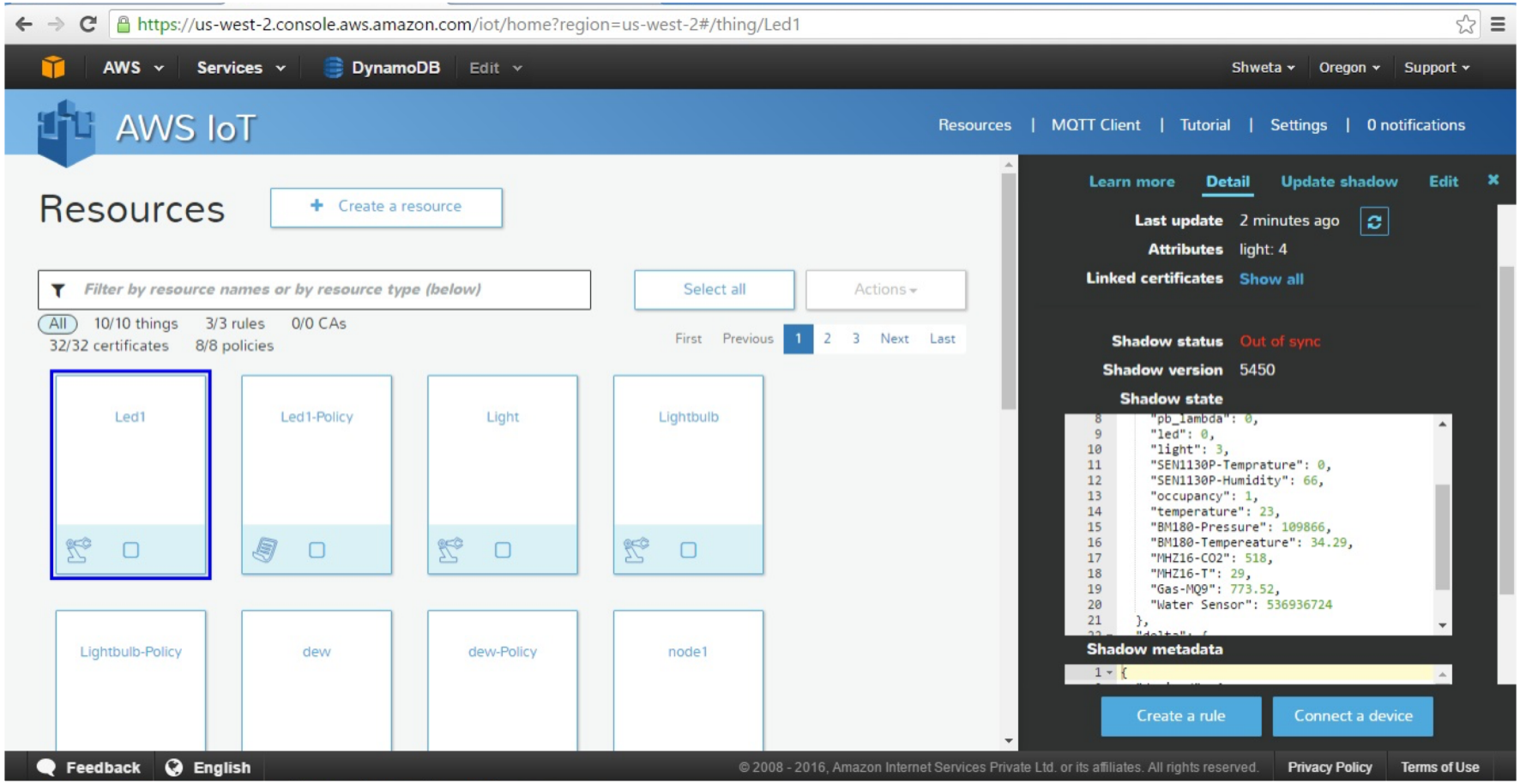
Register a Thing

8. The thing is now created on the AWS IoT service. Clicking on Thing, will show you the contents of this thing.



Shadow

9. Click on “View Thing” and select “Thing name” in AWS IoT page



Connecting Raspberry Pi to Ubidots

Components

- Raspberry Pi 3 model B
- 8 GB Micro SD card (Suggested)
- Micro USB power adapter
- Display monitor
- Keyboard
- Mouse

Raspberry Pi Setup

If you have already done a basic setup of your Raspberry Pi you can jump this section, otherwise follow these steps:

1. Download the Raspberry Pi operating system, we recommend Raspbian Wheezy.
2. Unzip the file and you'll end up with a large IMG file: DON'T COPY AND PASTE THIS FILE INTO YOUR SD CARD, because it won't work. You'll need to write it properly to the SD card by following these steps according to your operating system:
   1. MacOS: http://ivanx.com/raspberrypi/
   2. Linux: http://elinux.org/RPi_Easy_SD_Card_Setup#Using_Linux_.28including_on_a_Pi.21.29
   3. Windows: http://elinux.org/RPi_Easy_SD_Card_Setup#Using_Windows_7_or_Windows_XP
3. Plug a keyboard and a monitor through the HDMI or TV/Analog ports.

Configuring Raspberry Pi to use Ubidots Python API Client

1. Let's make sure your device is up to date so that it has the latest python tools (be aware that this will take a while):
   I. *$ sudo apt-get update*
   II. *$ sudo apt-get upgrade*

2. Download the pip installer and install Ubidots' Python library
   I. *$ sudo apt-get install python-setuptools*
   II. *$ sudo easy_install pip*
   III. *$ sudo pip install ubidots*

Setup a Test Variable in Ubidots

1. As a logged in user navigate to the “Data” tab.
2. Create a Data Source by clicking on the orange icon on the right. Then create a variable within that Data Source.
3. Take note of the variable’s ID to which you want to send data. For this example we’ll use a variable with the ID: “521d792df91b2816f35c8587”
4. Take note of your API key.

Send Data to Ubidots

1. Coming back to your Raspberry Pi:
2. Create a directory called “ubidots” where you can put this and future scripts:
   I. *$ mkdir ubidots*
3. Create a python script using your favorite text-editor. We’ll use “nano” in this case:
   I. *$ cd ubidots*
   II. *$ nano ubi-test.py*

Send Data to Ubidots

Put the following code into the created file. Please note the fields where you should put your API key and your variable ID.

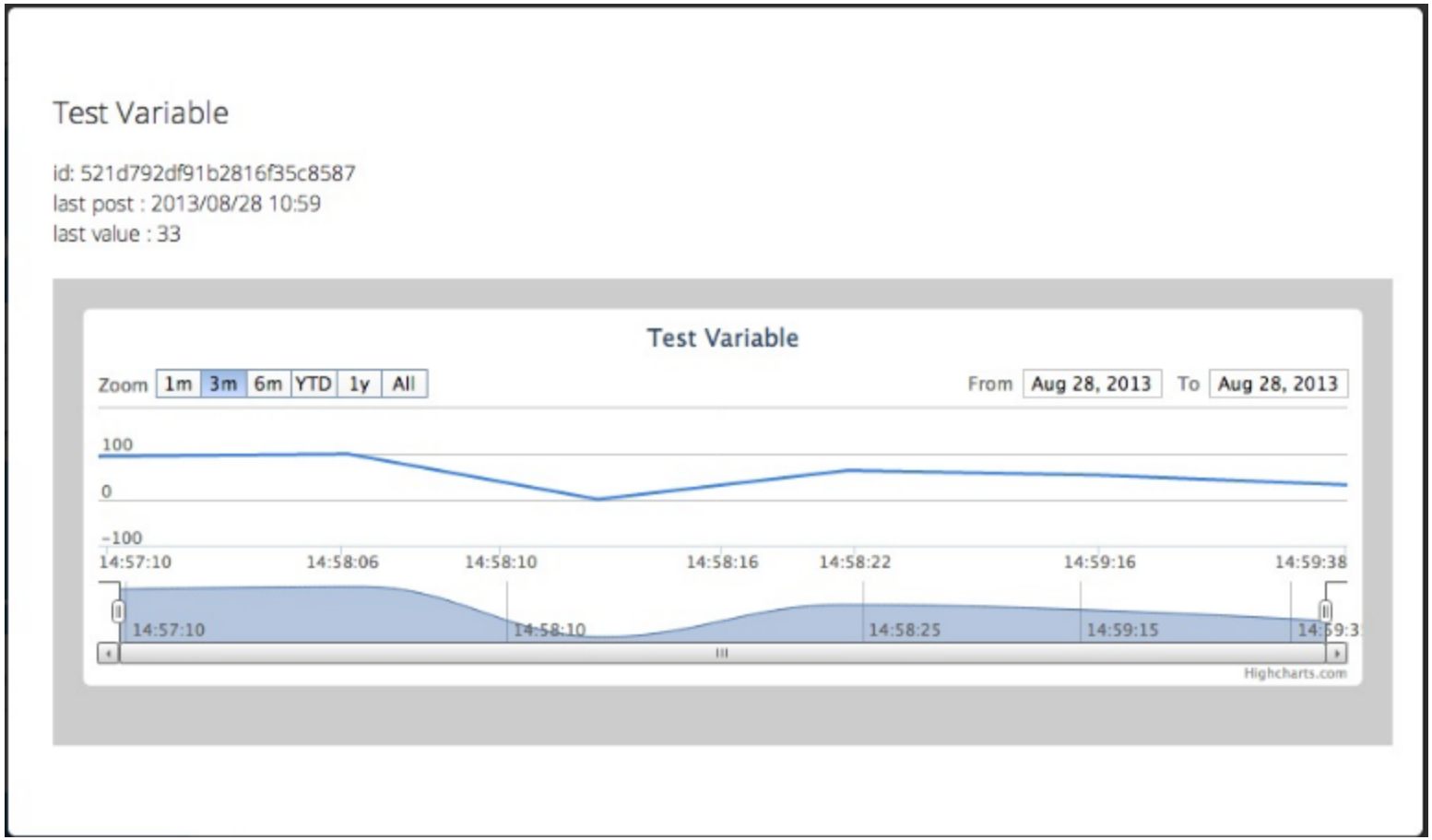
```
from ubidots import ApiClient
import random

#Create an "API" object

api = ApiClient("7fj39fk3044045k89fbh34rsd9823jkfs8323")

#Create a "Variable" object

test_variable = api.get_variable("521d792df91b2816f35c8587")

#Here is where you usually put the code to capture the data, either through your GPIO pins or as a calculation. W
e'll simply put a random value here:

test_value = random.randint(1,100)

#Write the value to your variable in Ubidots

test_variable.save_value({'value':test_value})
```

Send Data to Ubidots

Run the code several times to send some random values to the cloud:

```
$ python ubi-test.py
```

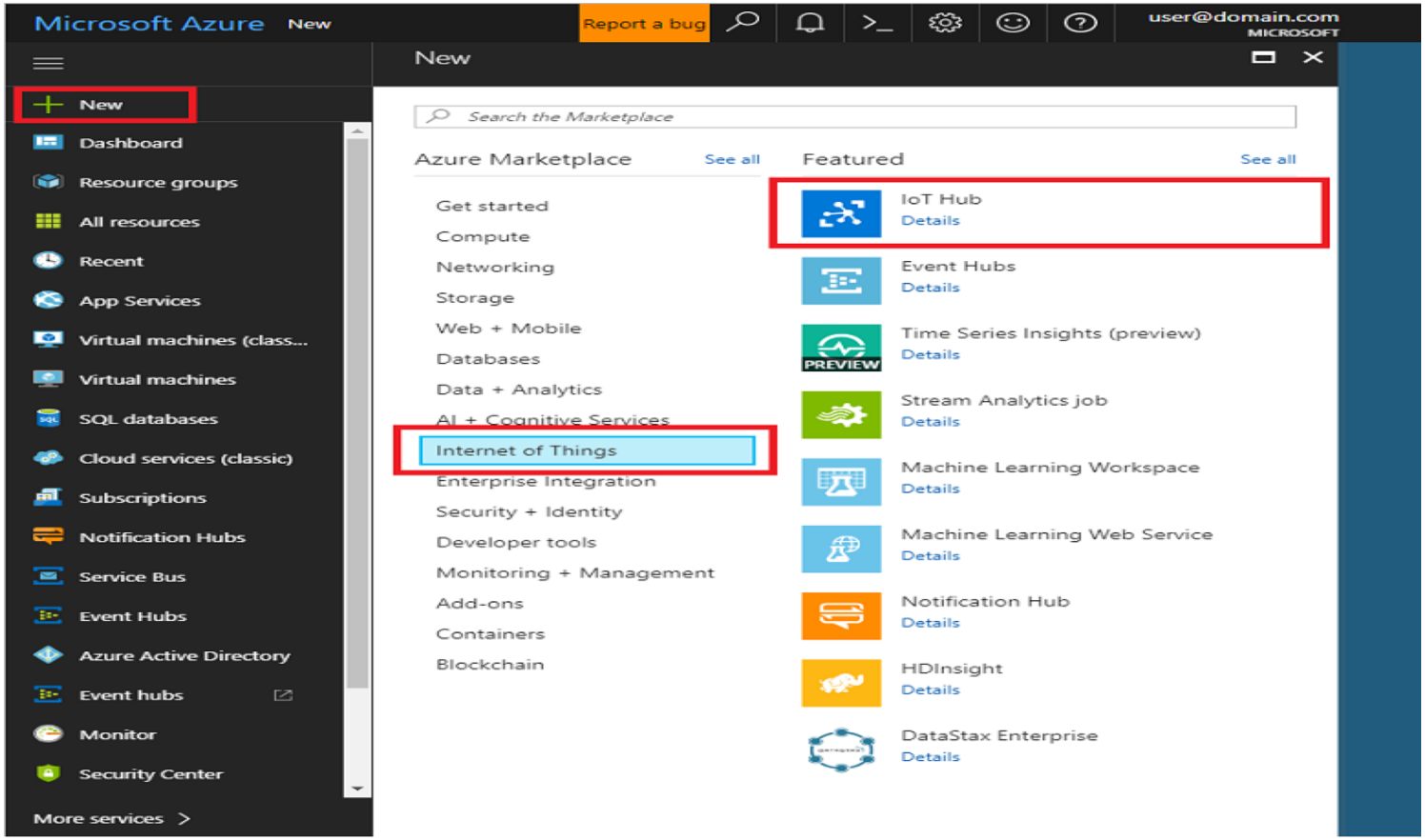
You should now see some data-points in your browser! Once your data is in the cloud, it's all down the hill. You can trigger events according to your data, create insights to share, etc.



Connect Raspberry Pi to Azure IoT Hub (Node.js)

Create an IoT hub

1. Sign in to the Azure portal.
2. Select New > Internet of Things > IoT Hub.



Create an IoT hub

3. In the IoT hub pane, enter the following information for your IoT hub:

Name: Create a name for your IoT hub. I

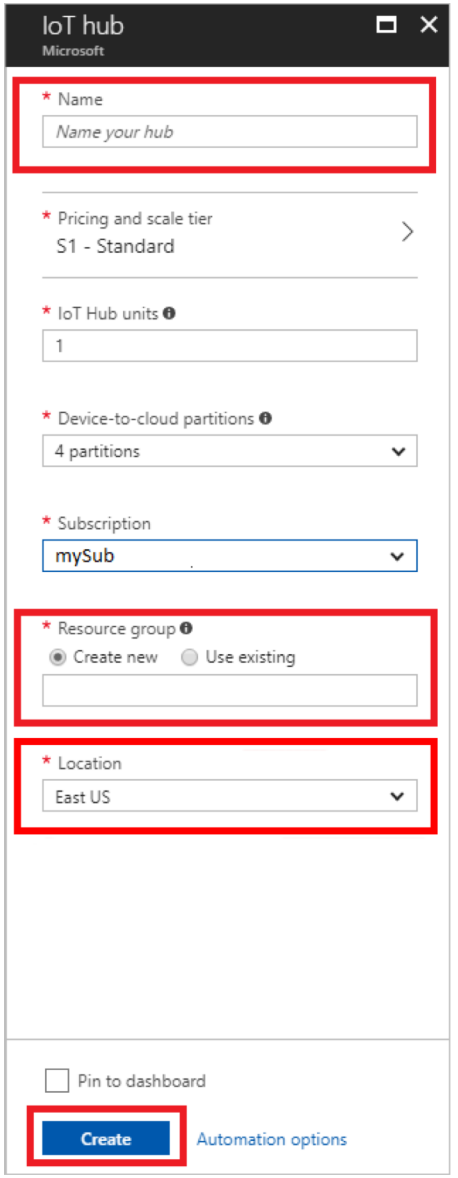
The IoT hub will be publicly discoverable as a DNS endpoint,

Pricing and scale tier: see the Pricing and scale tier.

Resource group: Create a resource group to host the IoT hub es

Location: Select the closest location to you.

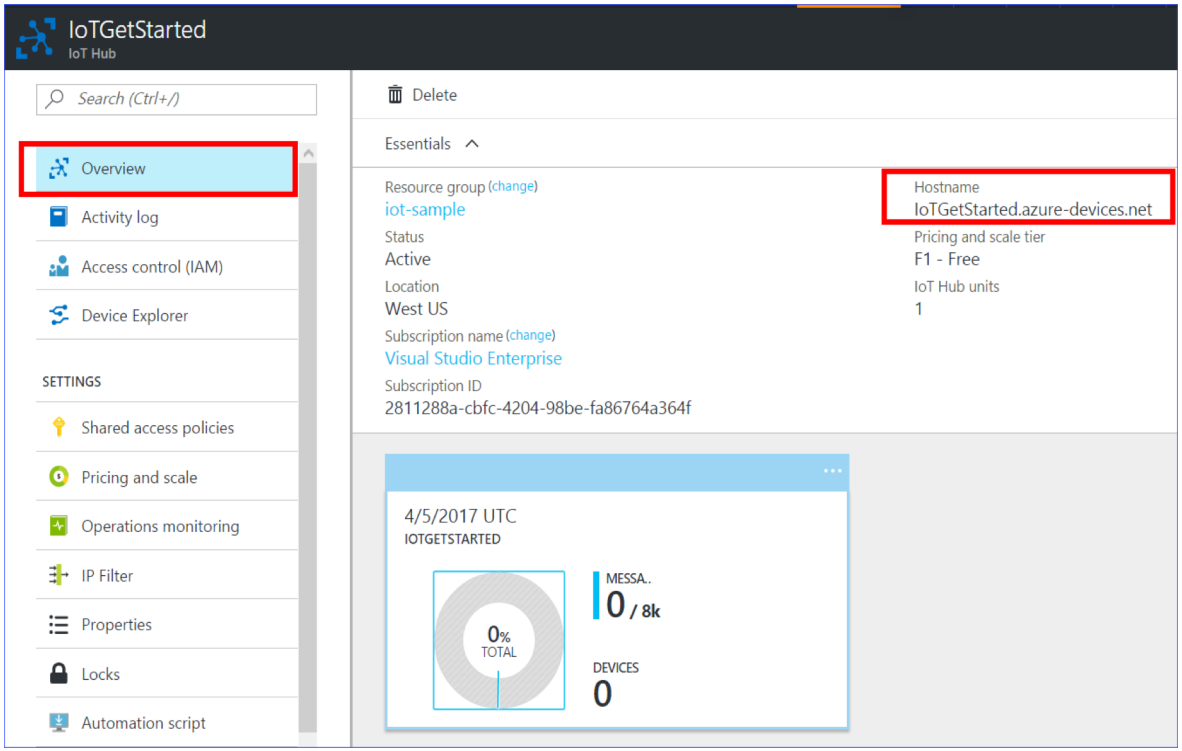
Pin to dashboard: Check this option for easy access



Create an IoT hub

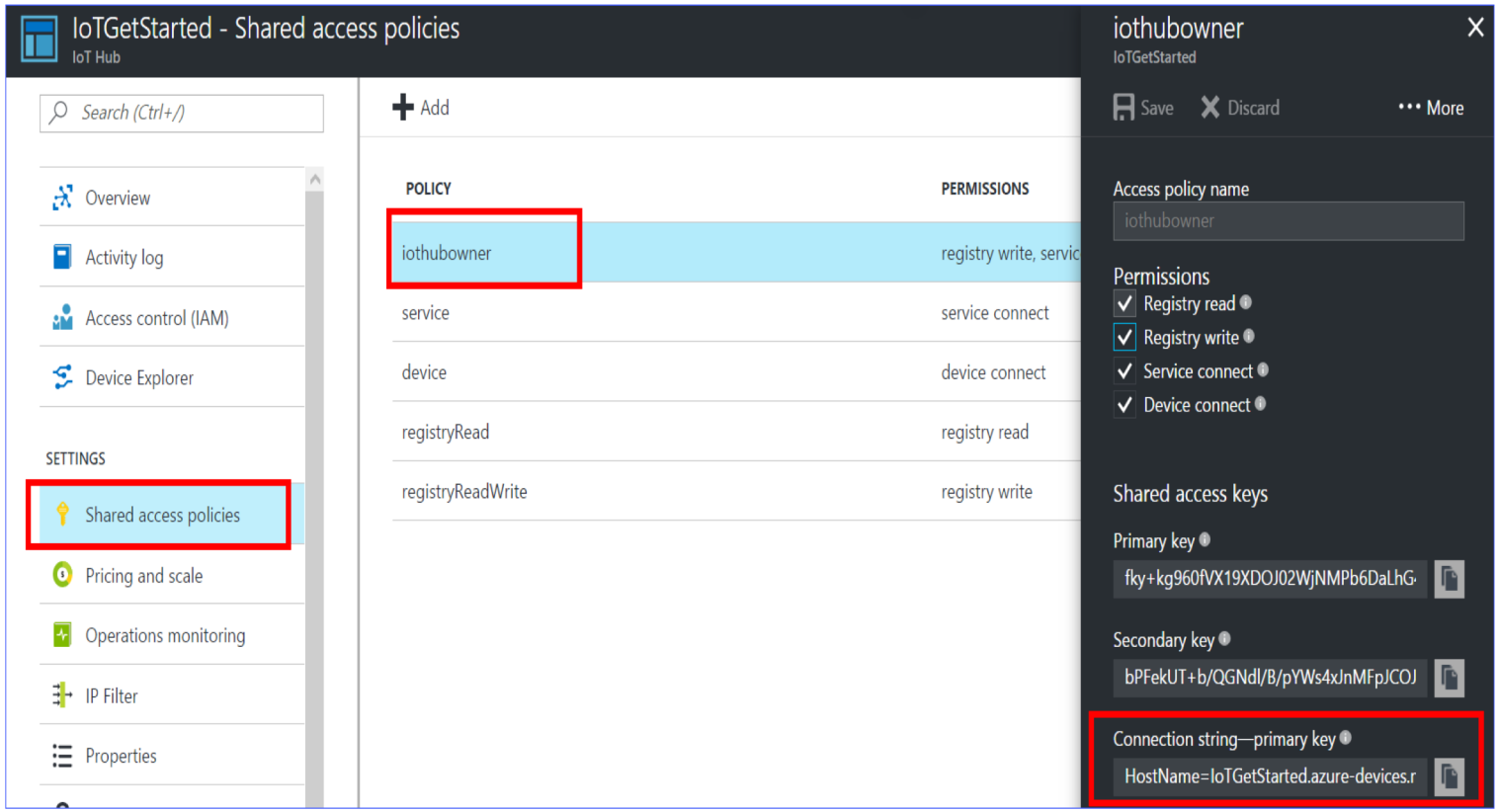
4. Click Create. Your IoT hub might take a few minutes to create. You can monitor the progress in the Notifications pane.

5.Click on the dashboard. Make a note of the Hostname, and then click Shared access policies.



Create an IoT hub

6. In the Shared access policies pane, click the iothubowner policy, and then copy and make a note of the Connection string of your IoT hub. For more information



Register a device in the IoT hub for your device

1. In the Azure portal, open your IoT hub.
2. Click Device Explorer.
3. In the Device Explorer pane, click Add to add a device to your IoT hub.
4. Then do the following:

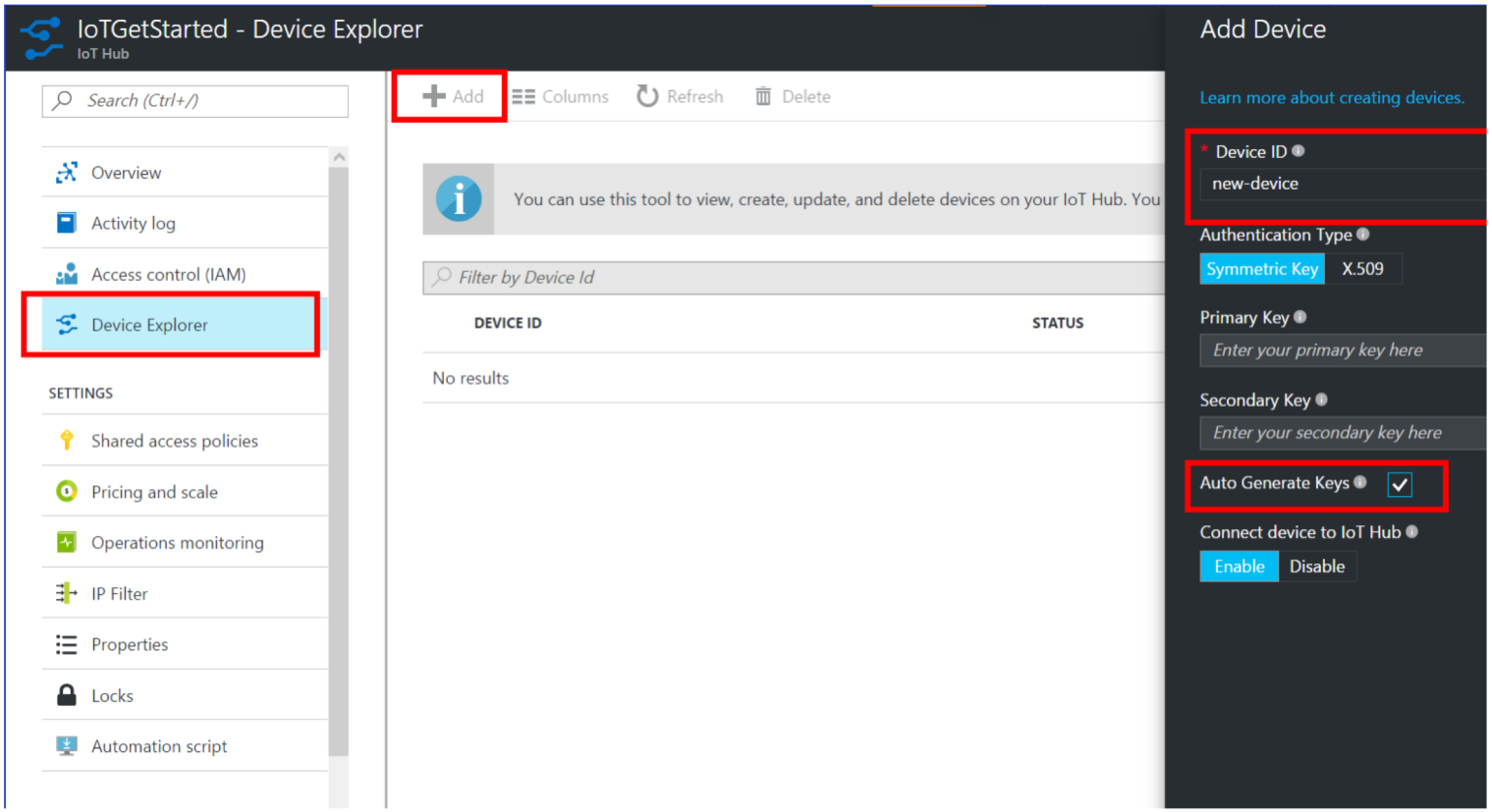
   Device ID: Enter the ID of the new device. Device IDs are case sensitive.

   Authentication Type: Select Symmetric Key.

   Auto Generate Keys: Select this check box.

   Connect device to IoT Hub: Click Enable

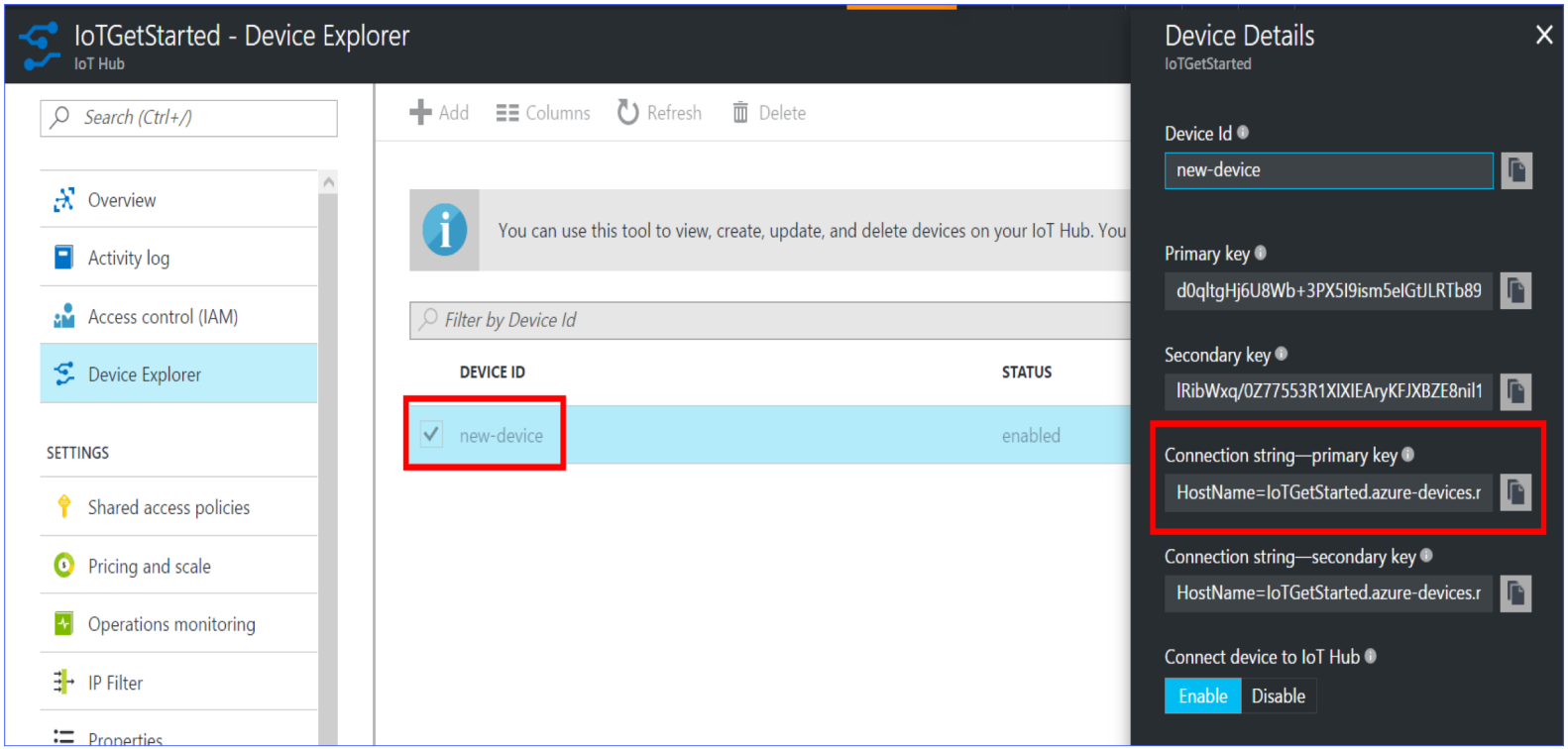
Register a device in the IoT hub for your device



- Click Save.
- After the device is created, open the device in the Device Explorer pane.

Register a device in the IoT hub for your device

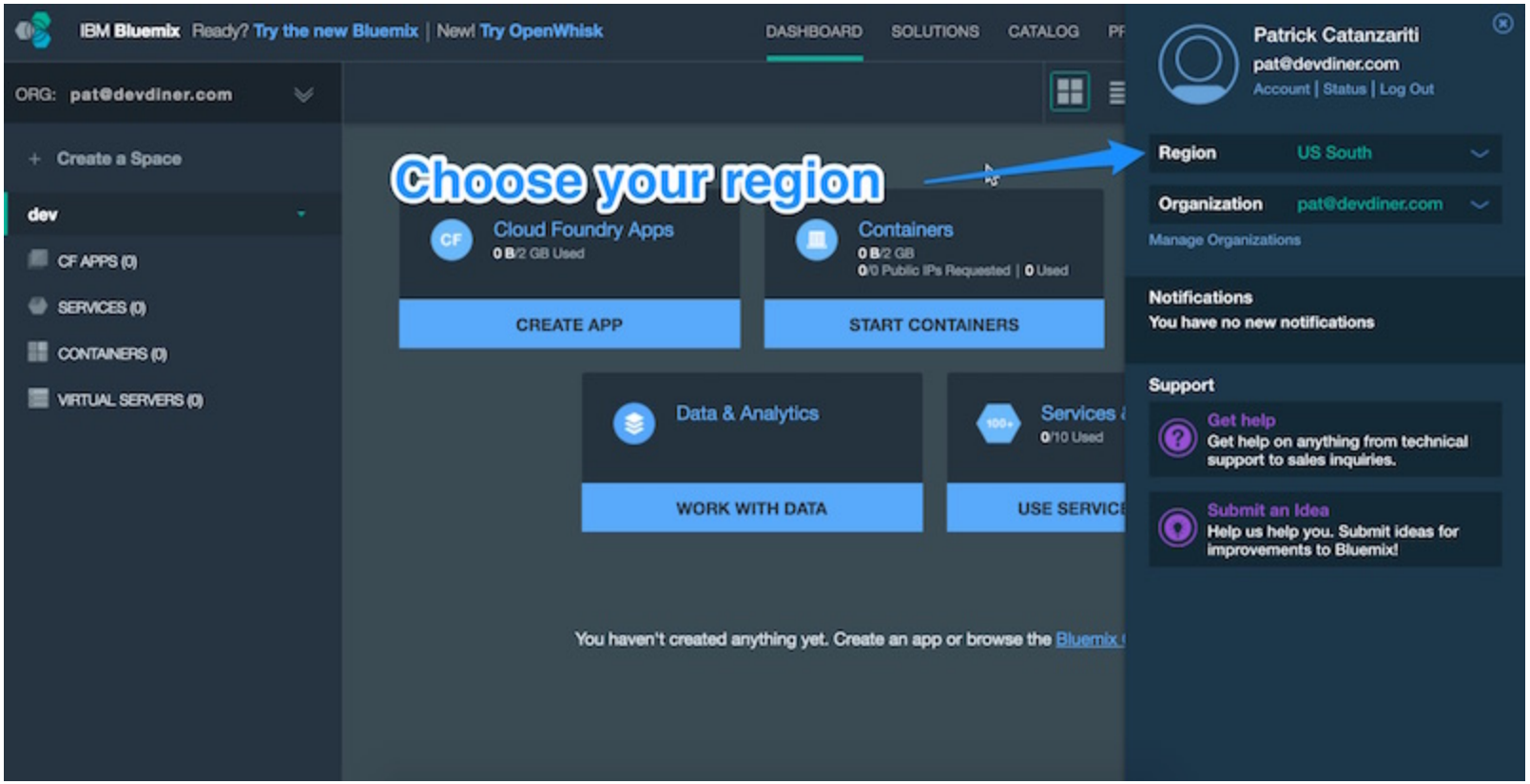
- Make a note of the primary key of the connection string.



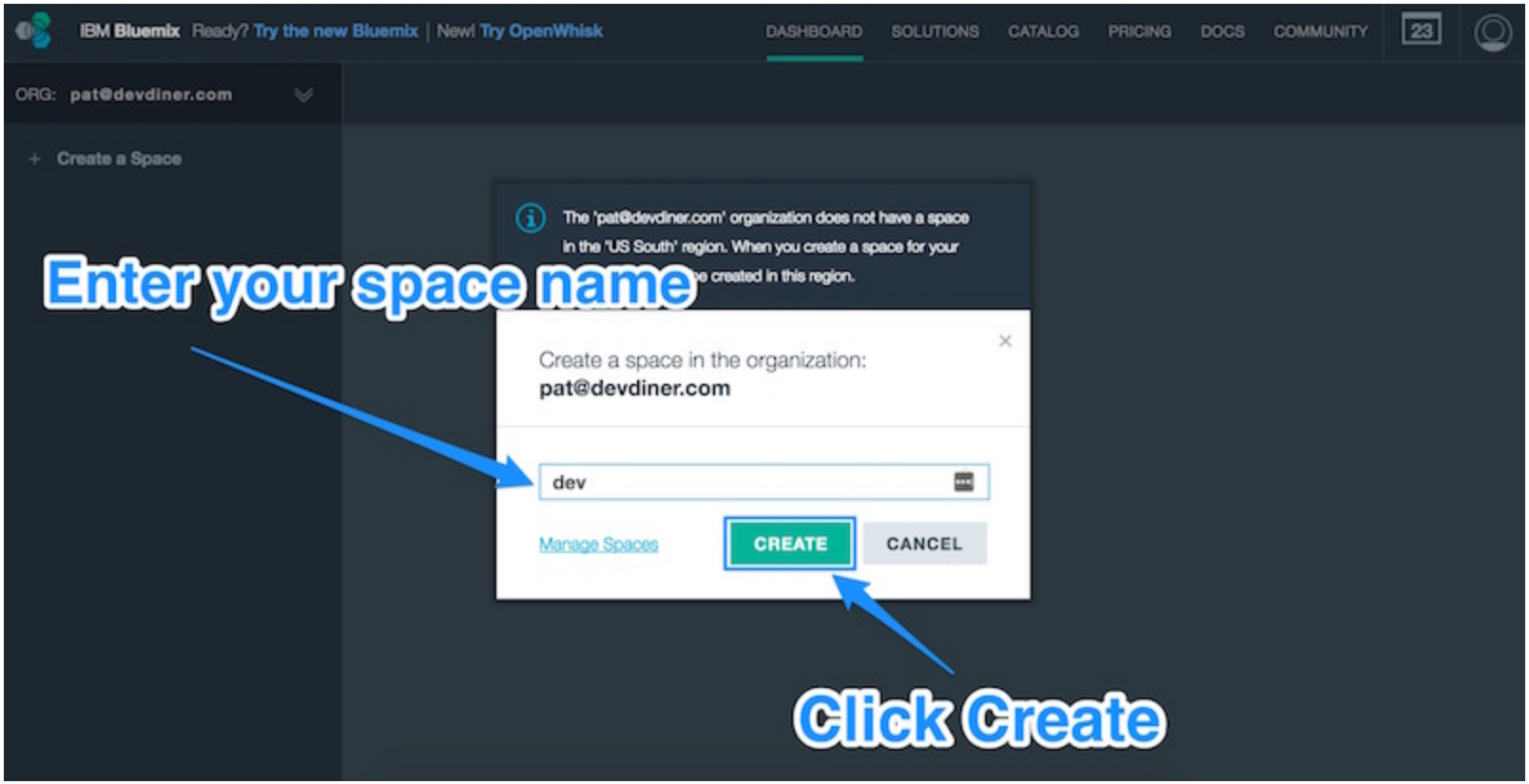
Connecting Raspberry Pi to IBM Bluemix

Connecting Raspberry Pi to IBM Bluemix

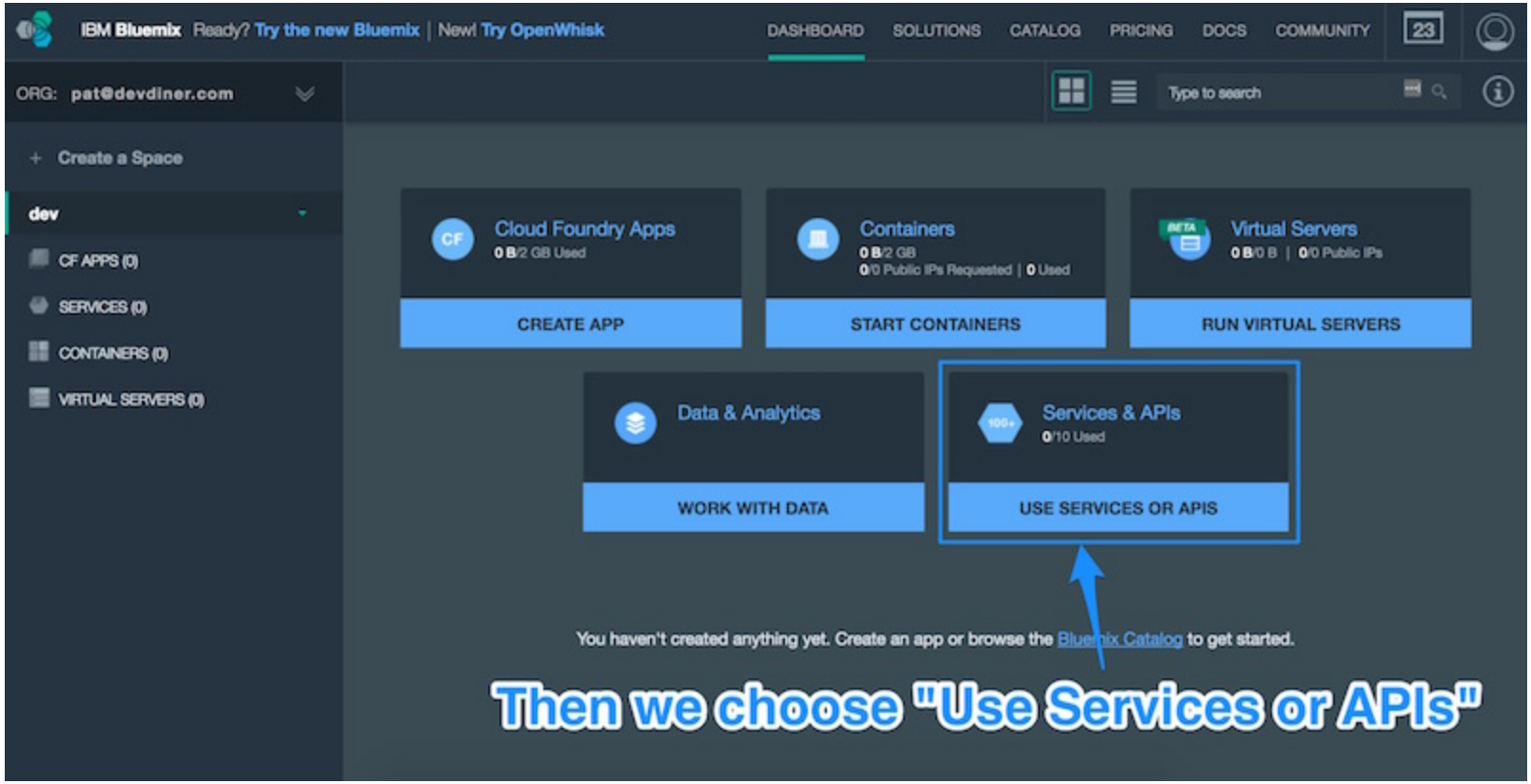
- Login to Bluemix - https://console.ng.bluemix.net/login



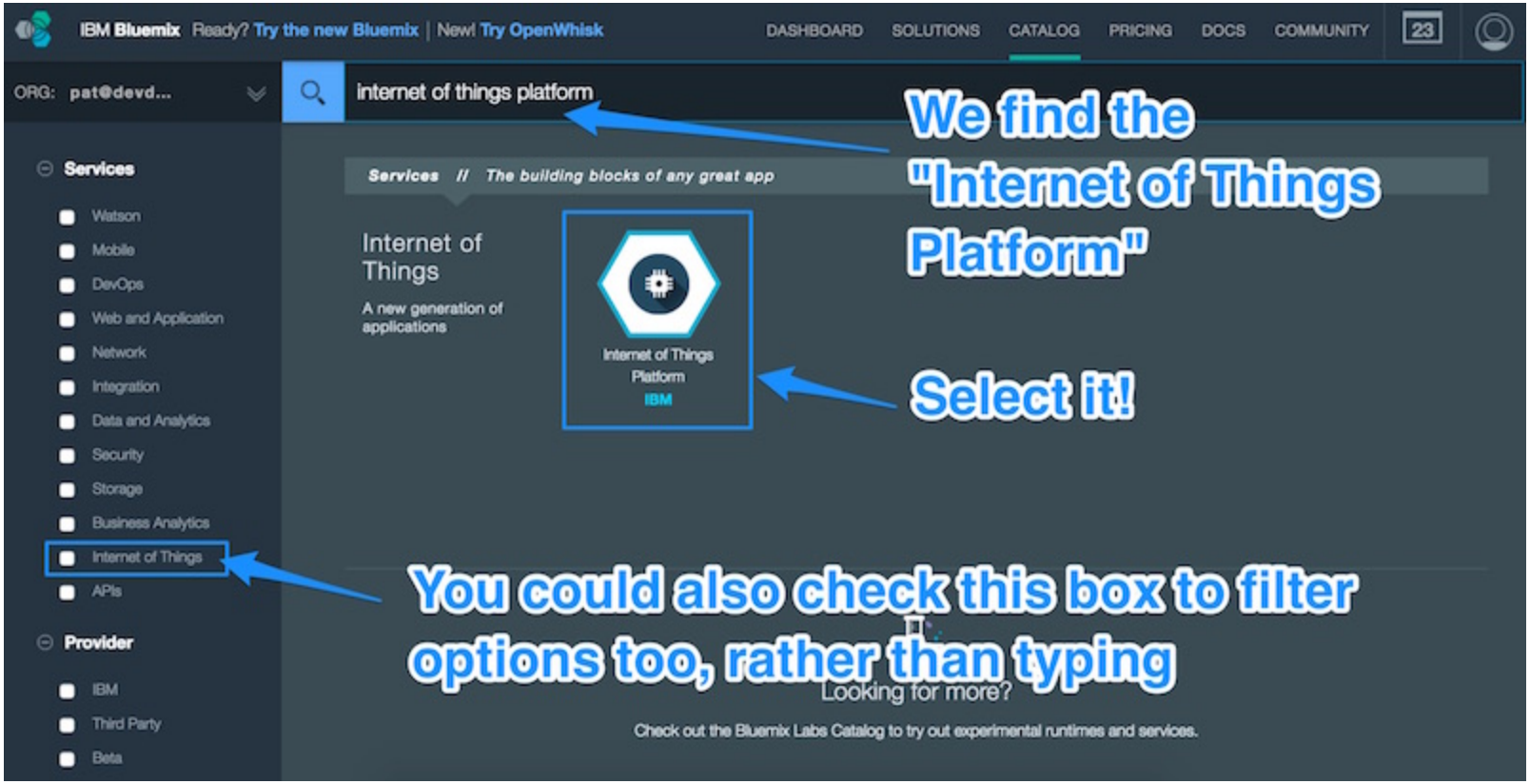
Connecting Raspberry Pi to IBM Bluemix



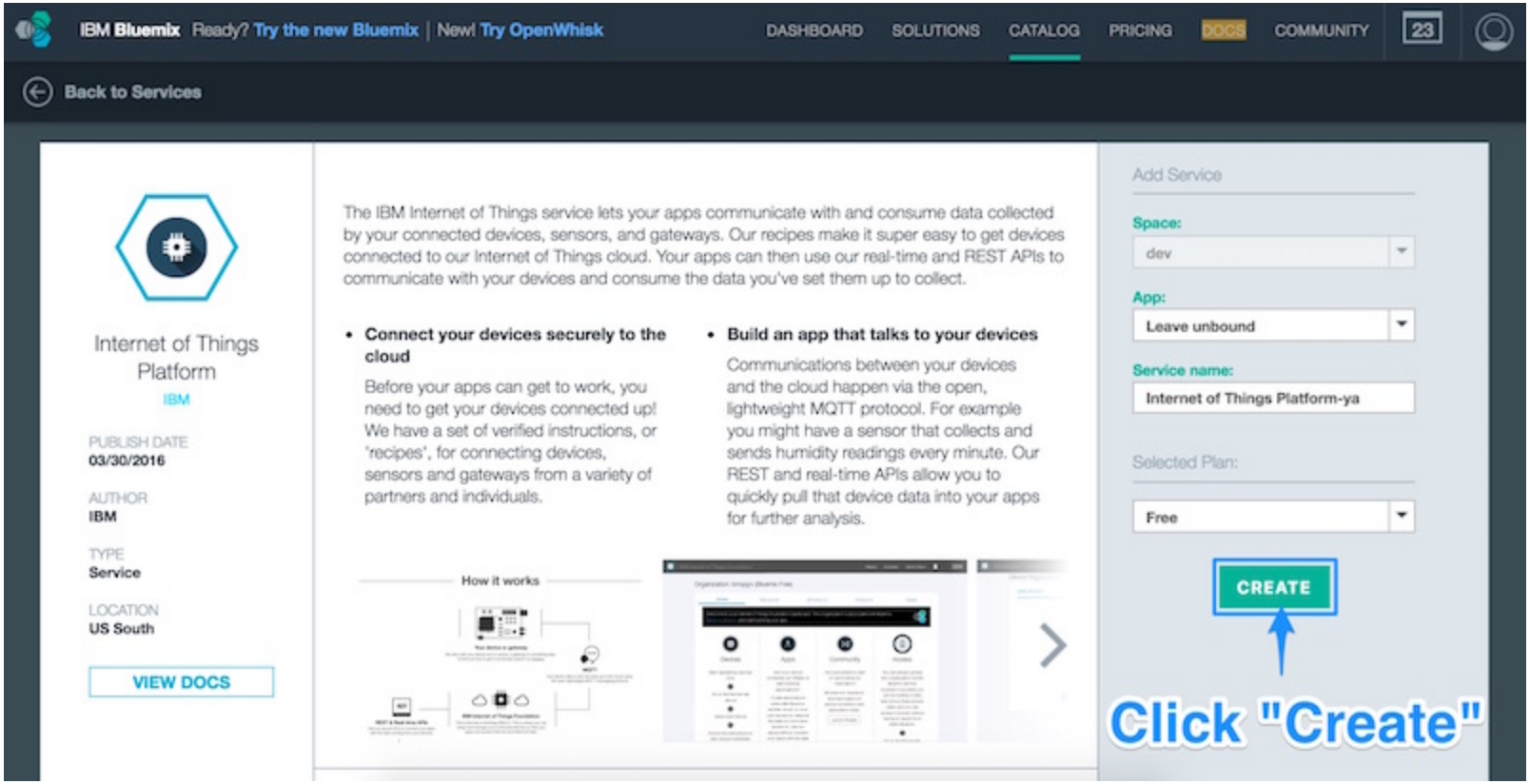
Connecting Raspberry Pi to IBM Bluemix



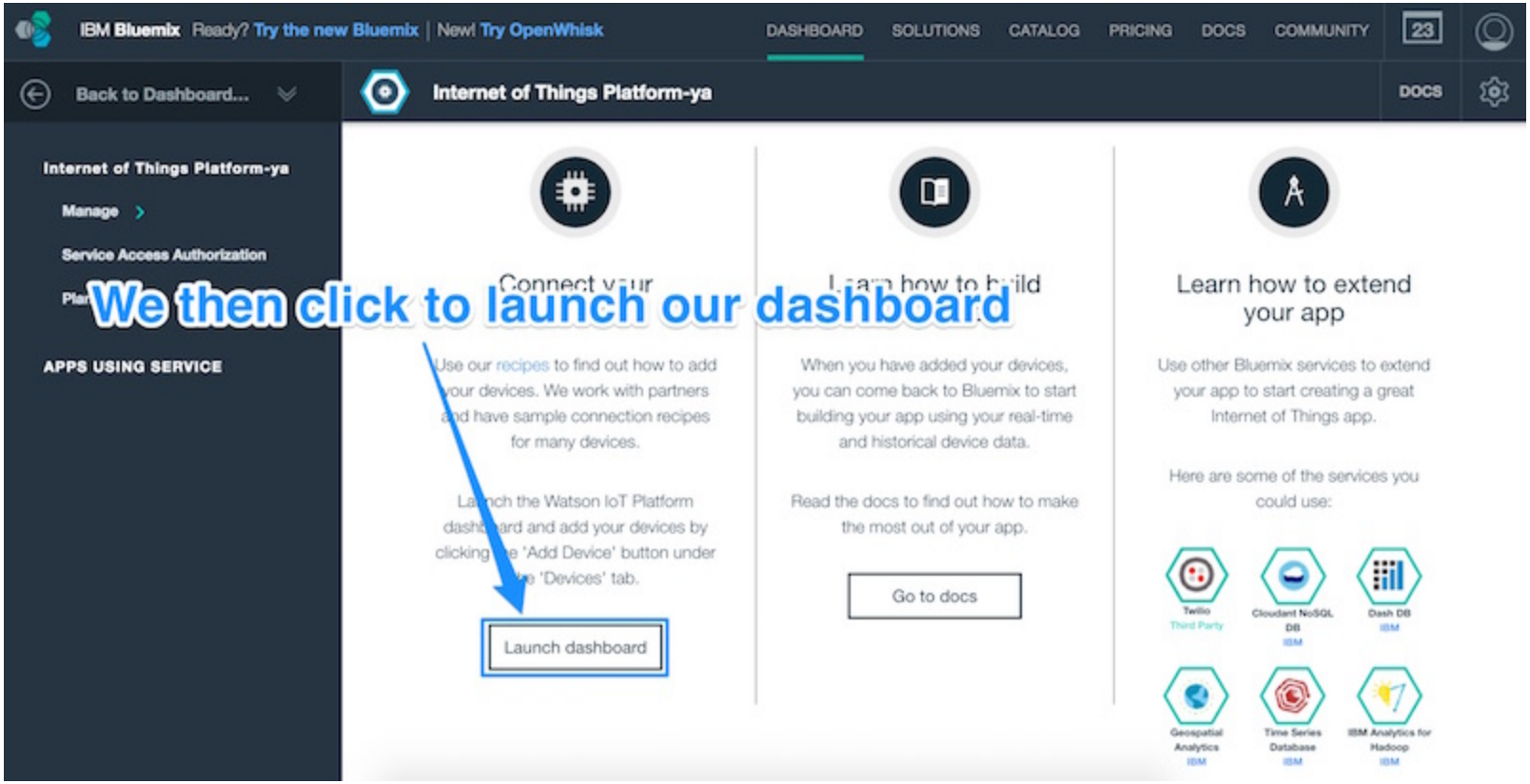
Connecting Raspberry Pi to IBM Bluemix



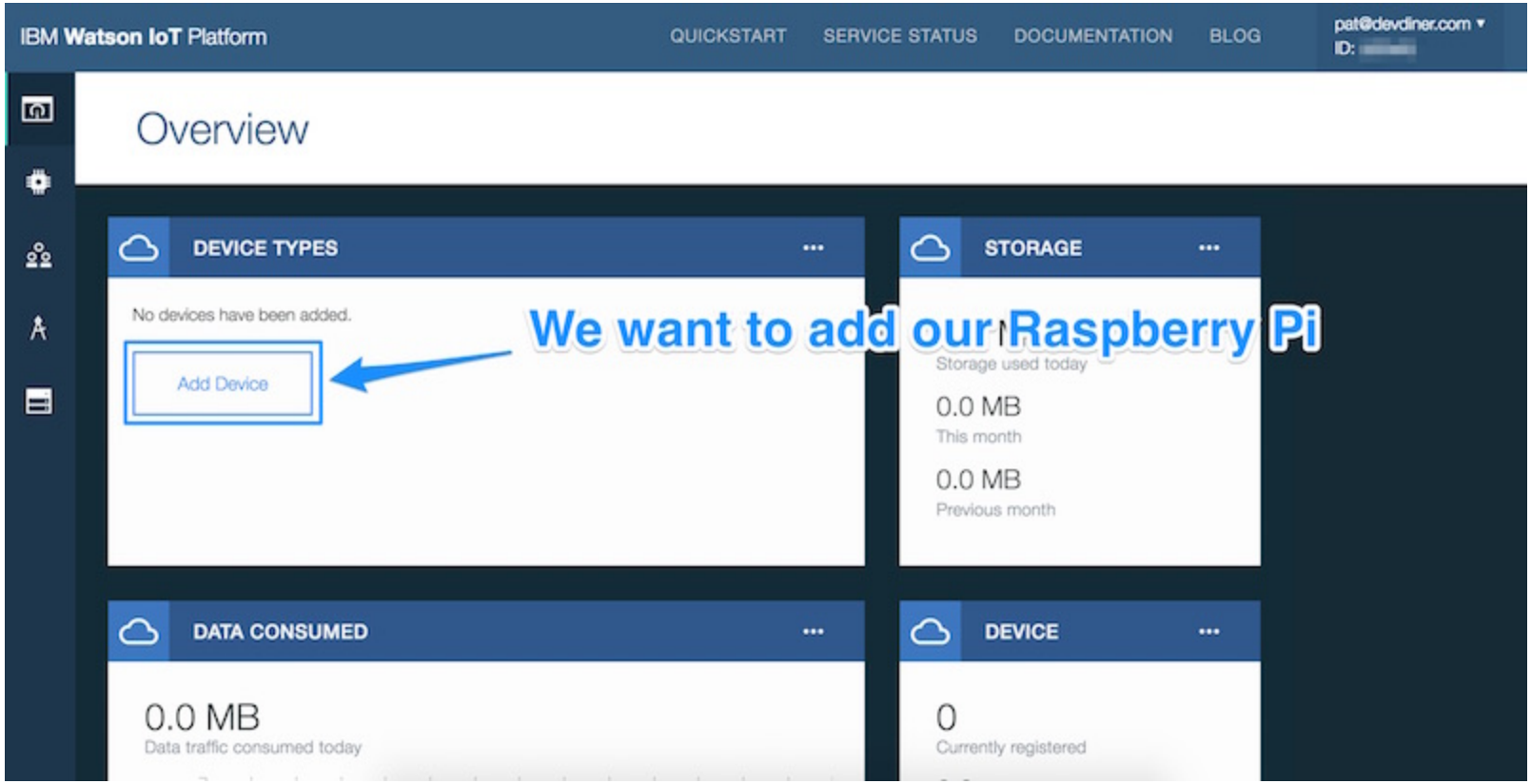
Connecting Raspberry Pi to IBM Bluemix



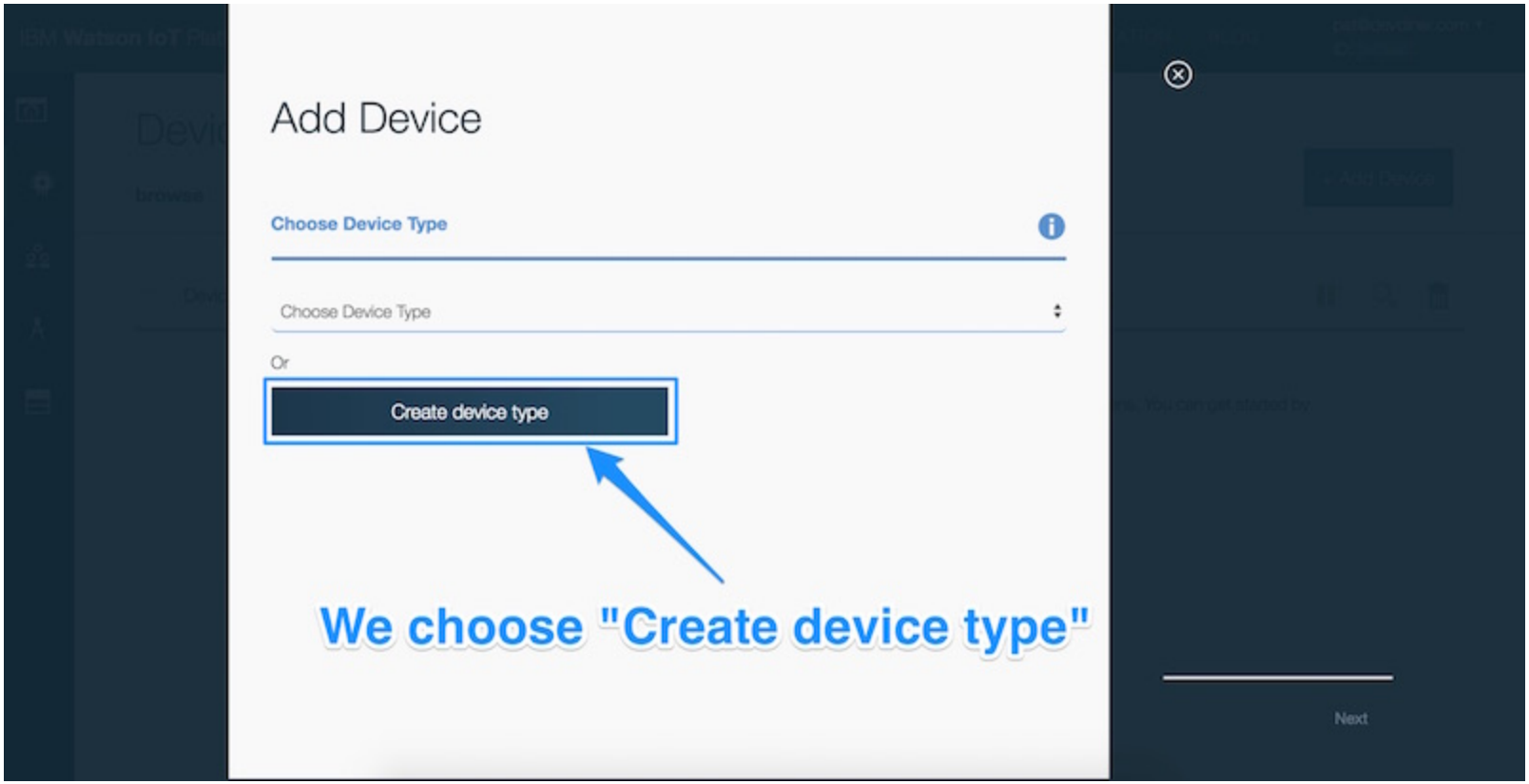
Connecting Raspberry Pi to IBM Bluemix



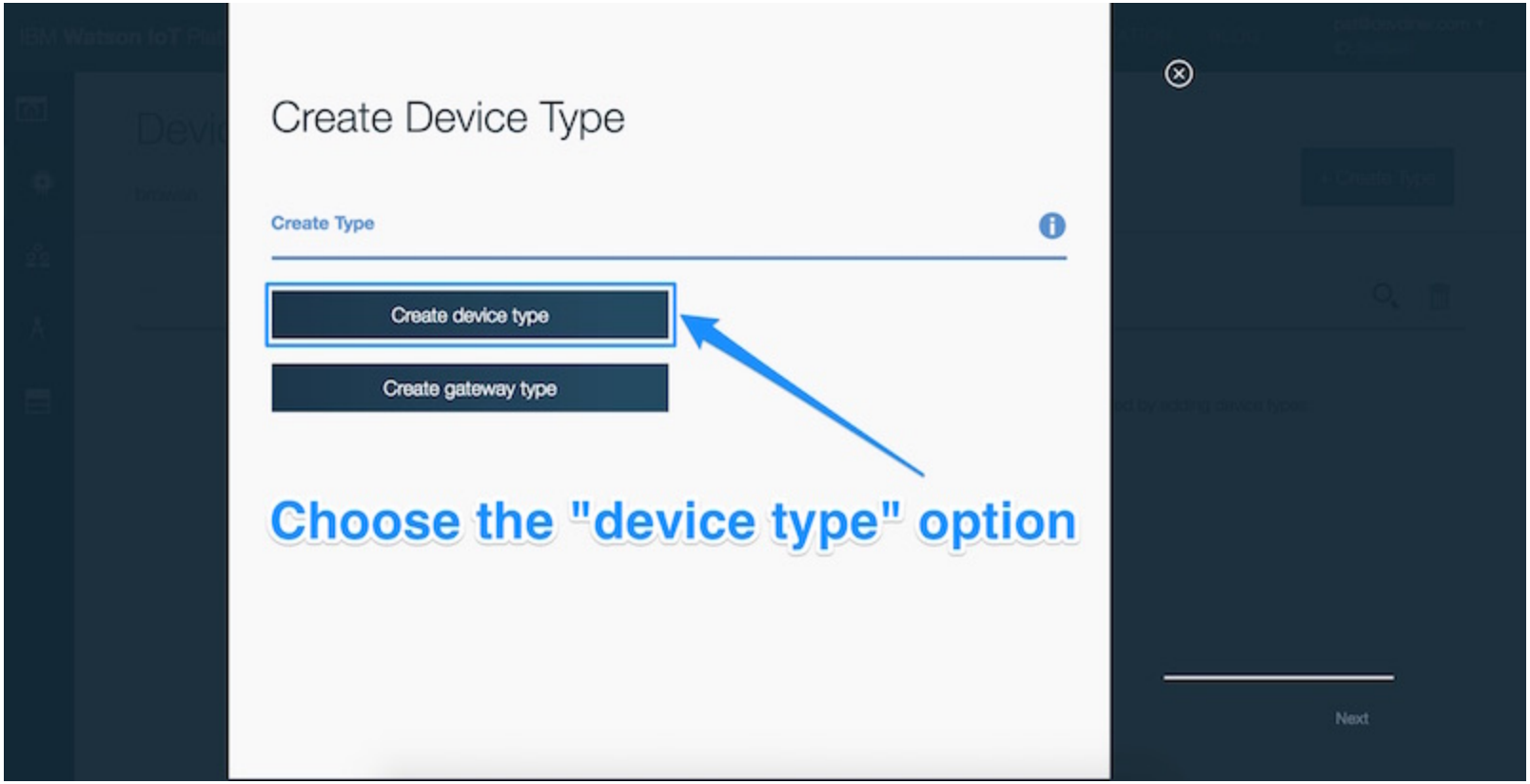
Connecting Raspberry Pi to IBM Bluemix



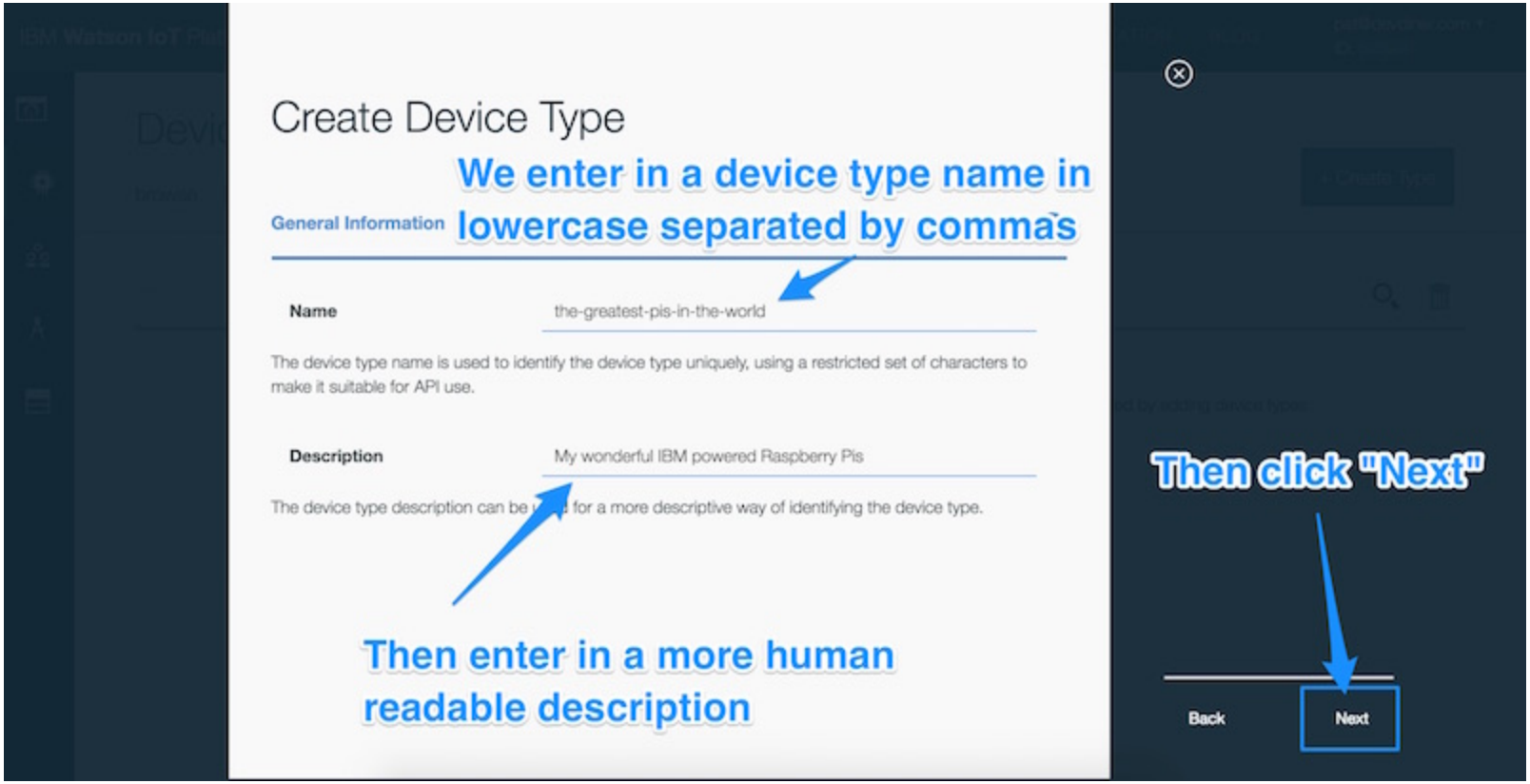
Connecting Raspberry Pi to IBM Bluemix



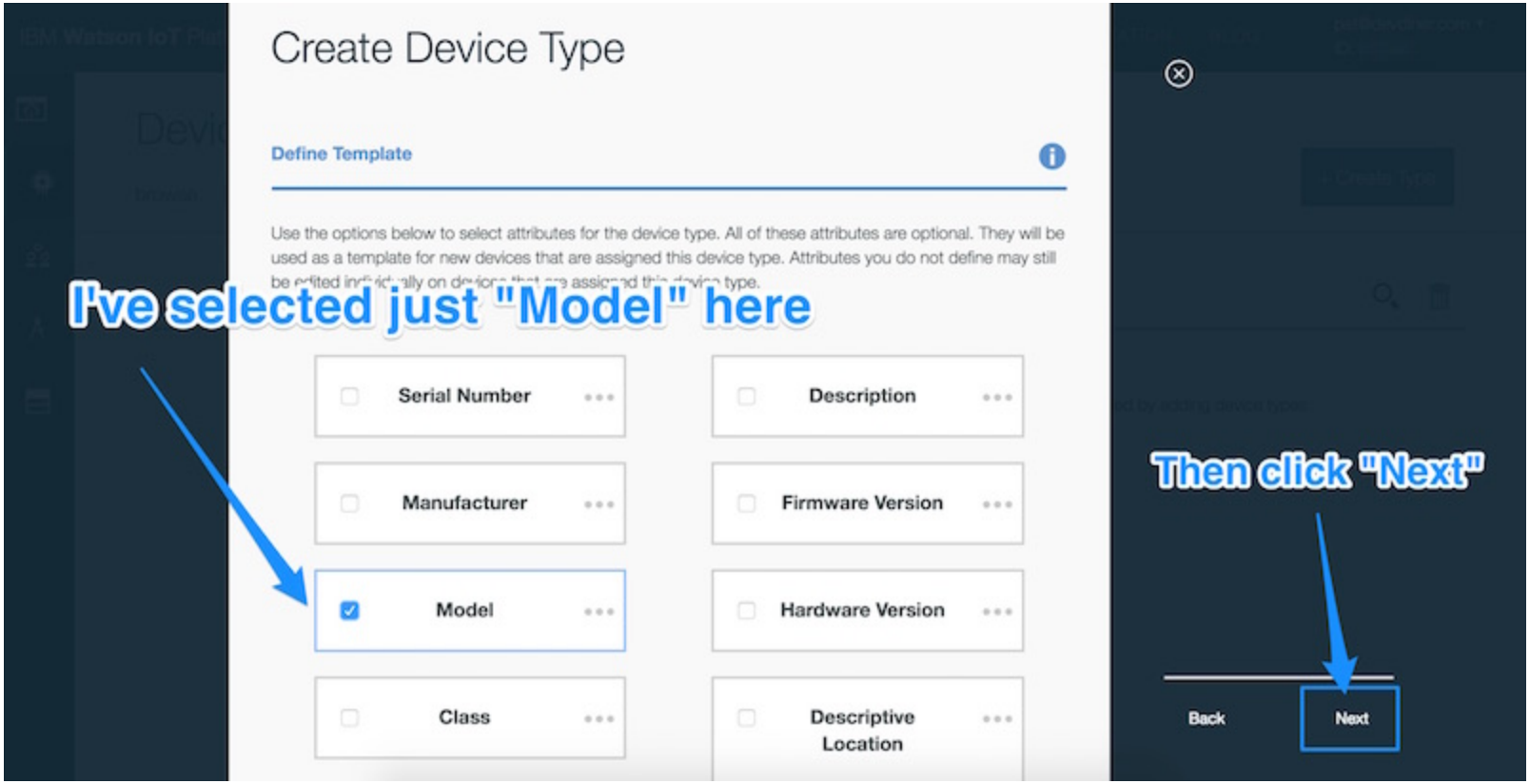
Connecting Raspberry Pi to IBM Bluemix



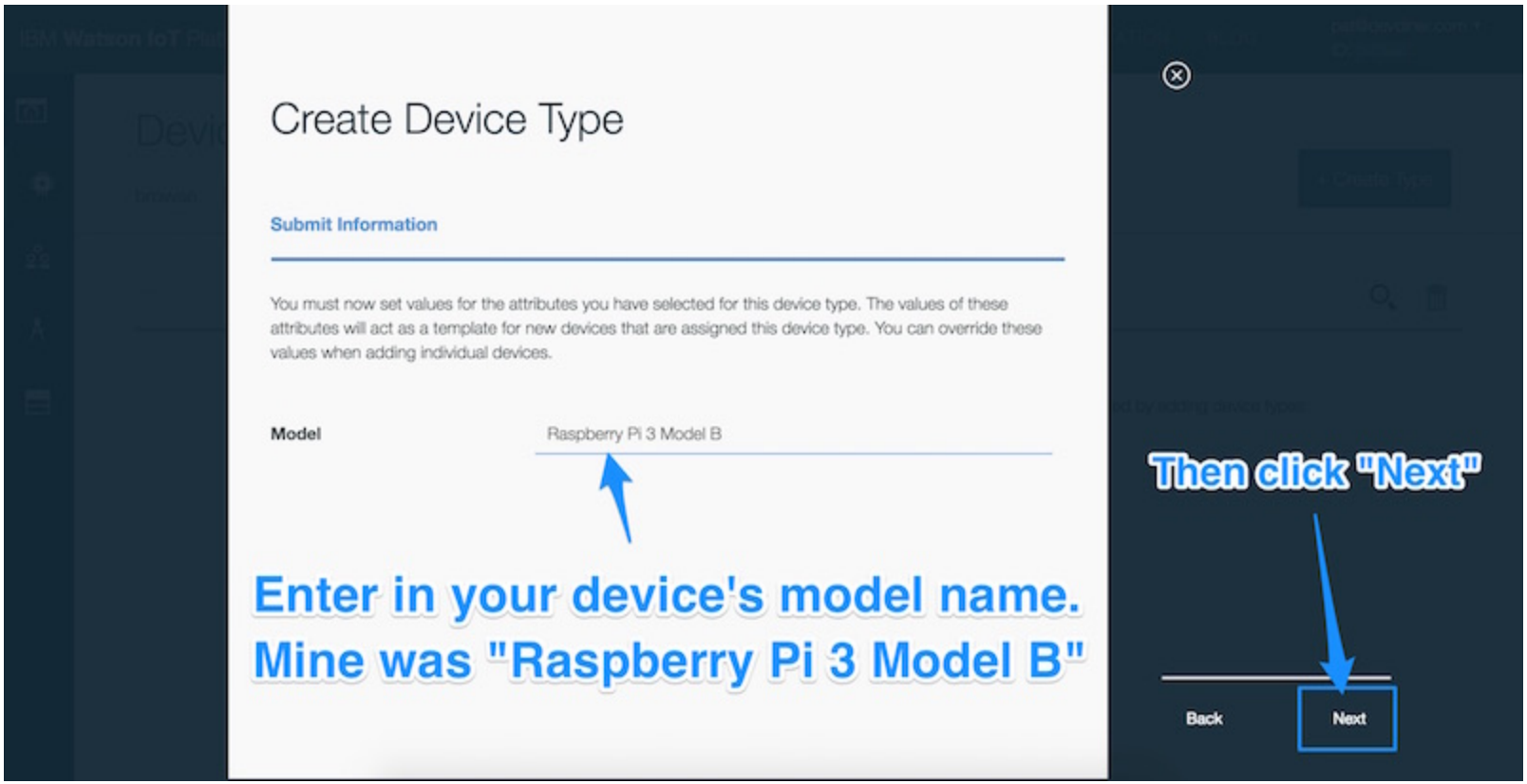
Connecting Raspberry Pi to IBM Bluemix



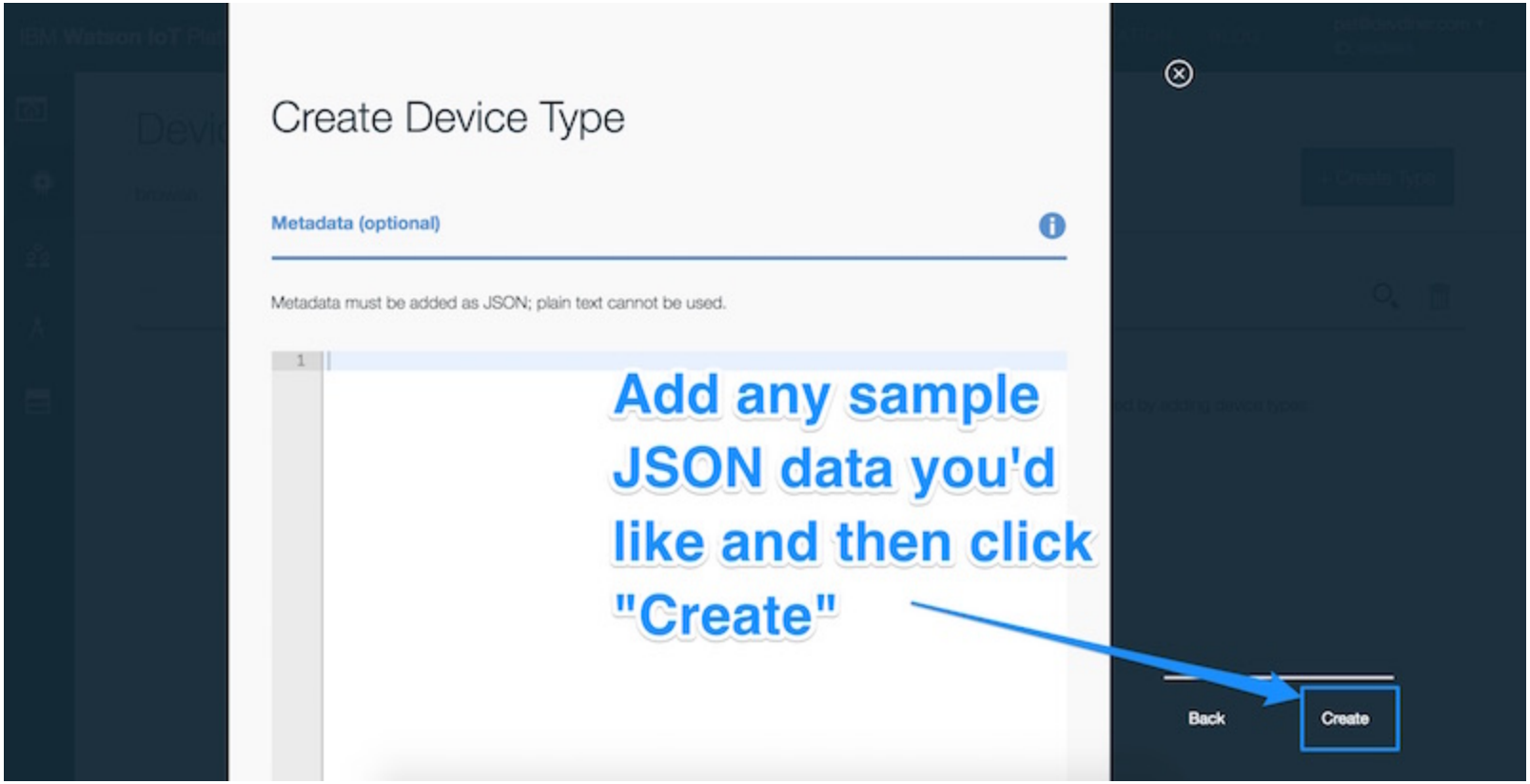
Connecting Raspberry Pi to IBM Bluemix



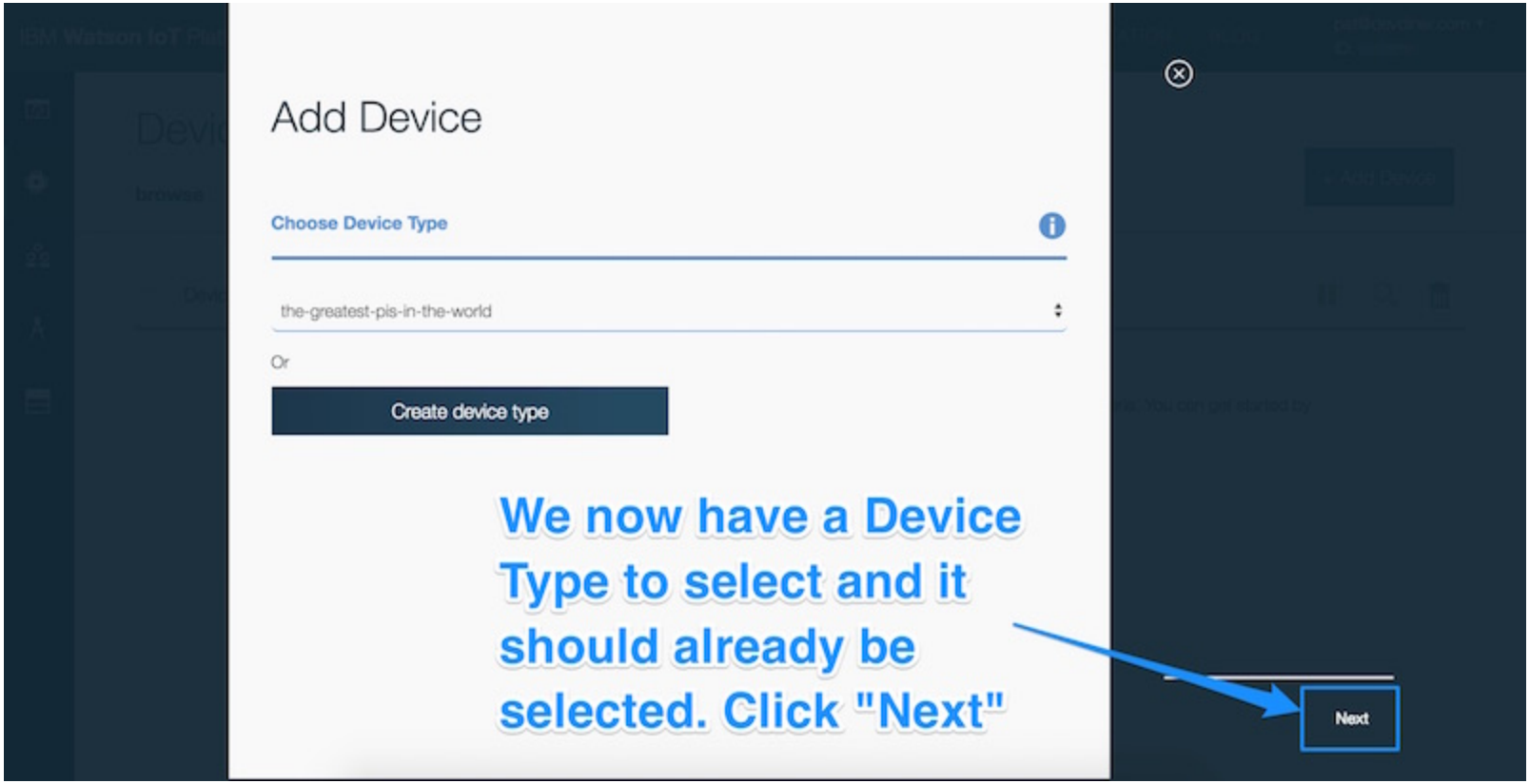
Connecting Raspberry Pi to IBM Bluemix



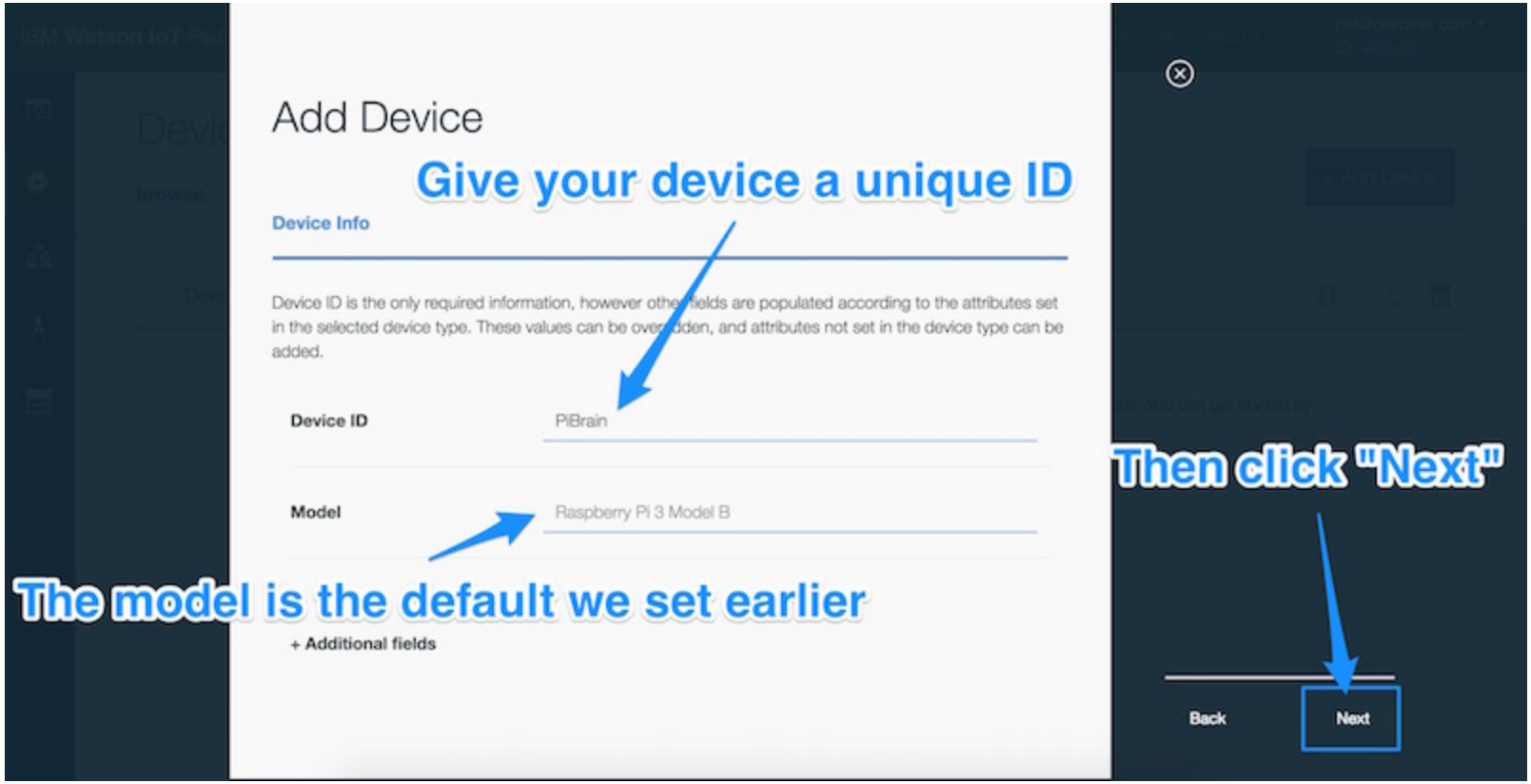
Connecting Raspberry Pi to IBM Bluemix



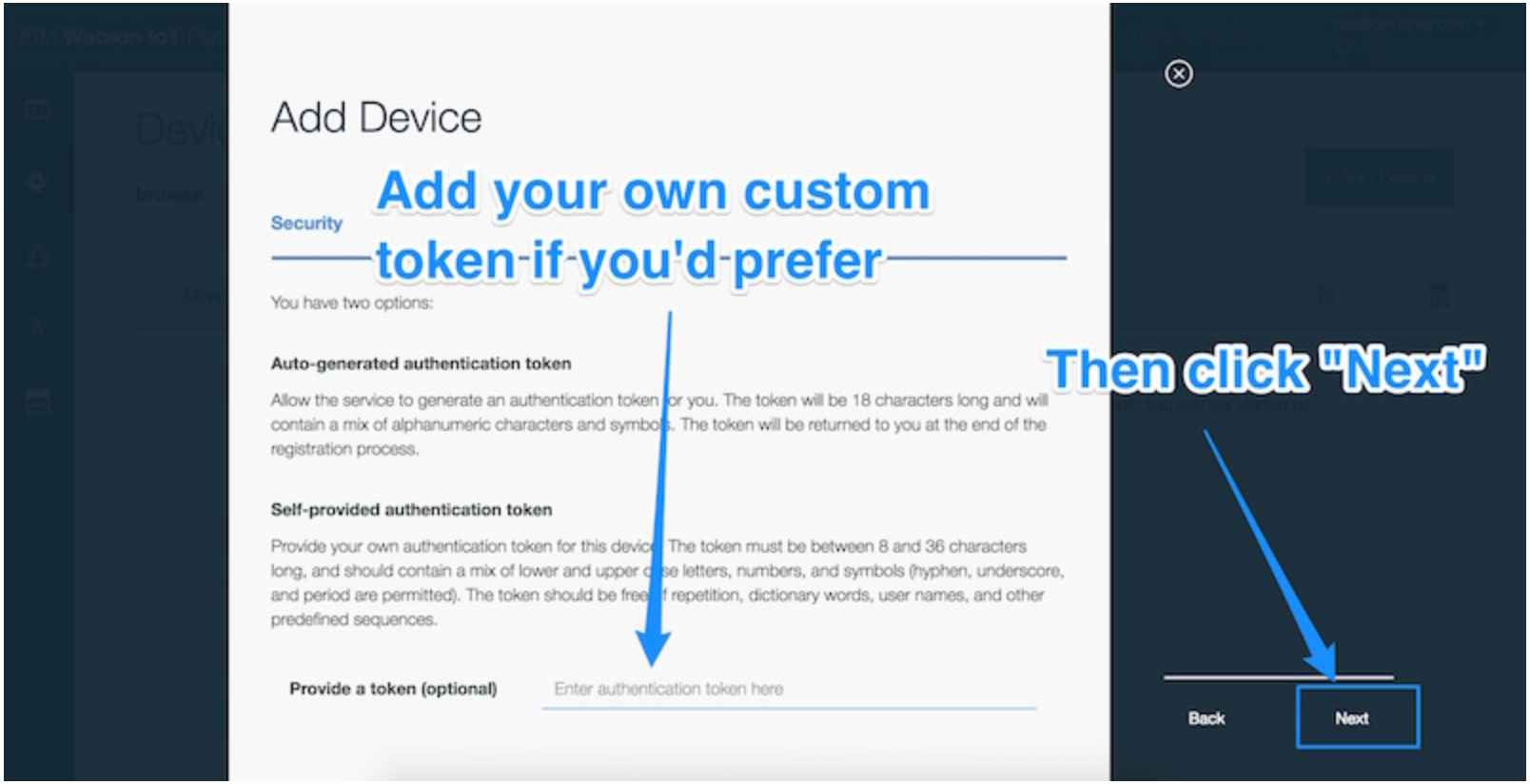
Connecting Raspberry Pi to IBM Bluemix



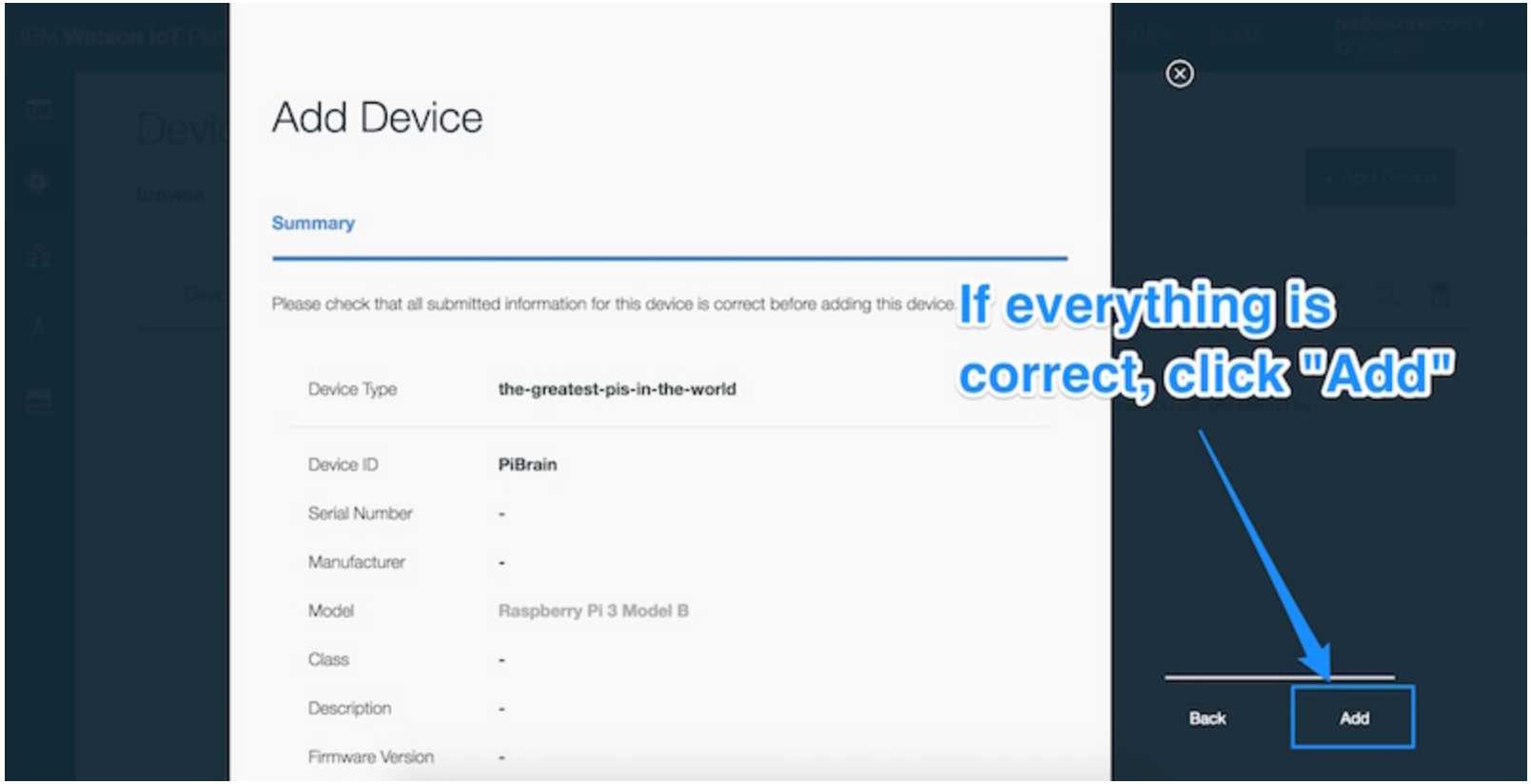
Connecting Raspberry Pi to IBM Bluemix



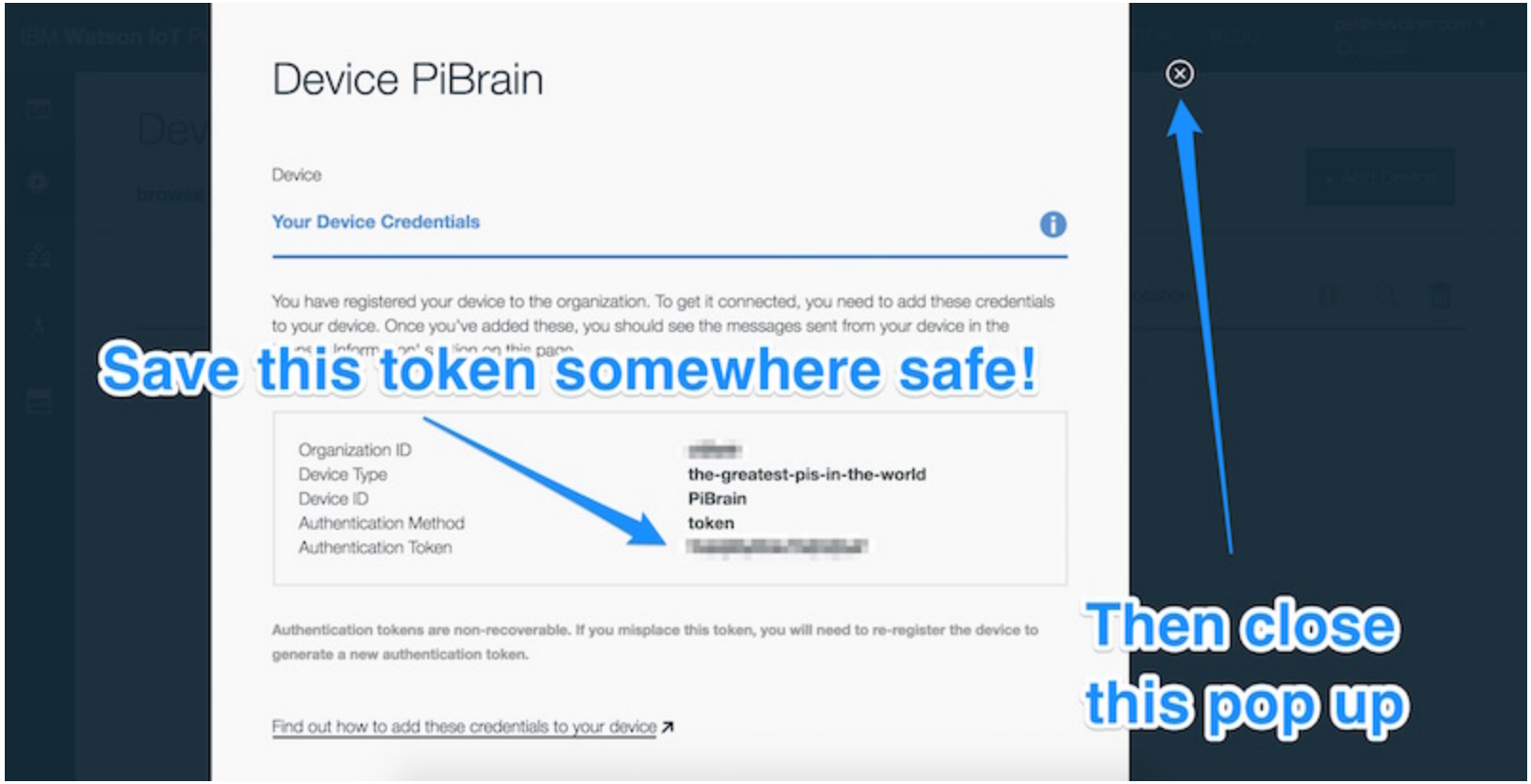
Connecting Raspberry Pi to IBM Bluemix



Connecting Raspberry Pi to IBM Bluemix



Connecting Raspberry Pi to IBM Bluemix



Fog Computing

What is Fog Computing

- **Fog computing** is a term created by Cisco that refers to extending cloud **computing** to the edge of an enterprise's network. Also known as **Edge Computing** or Fogging

Fog Computing

Why the Edge?

IoT in the Cloud

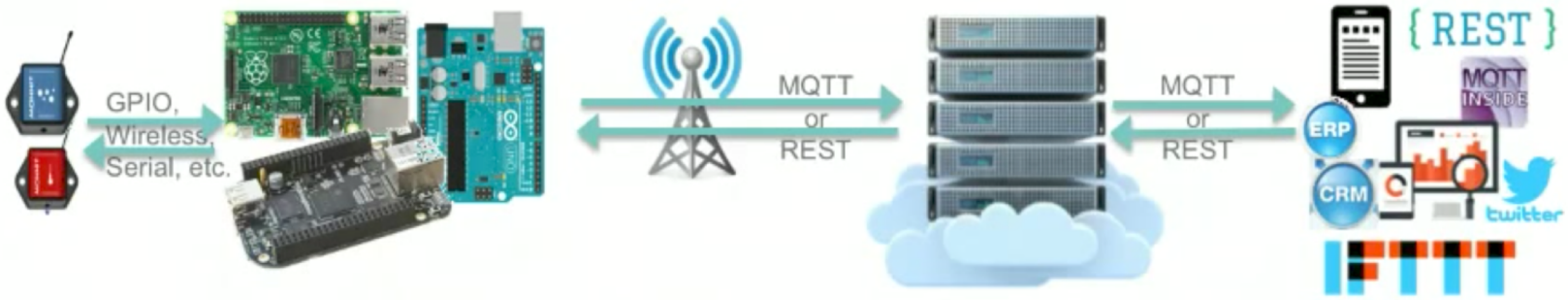
- Remote monitoring and control
- Merging remote from across multiple IoT devices
- Near infinite compute and storage to train machine and other advanced artificial intelligence tools

IoT on the Edge

- Low latency tight control loops require near time-time response
- Public Internet inherently unpredictable
- Privacy of data and protection of IP

Why the Edge?

Fragmentation



Sensor Protocols	Controller	Network	Standards	Comm. protocol / Device Mgmt
• ZigBee • Z-Wave • CANBus • MODBus • Bluetooth/BLE • DECT • 6LoWPAN • LoRA	• OSes • Compilers • Programming languages • Processors	• Cellular • Wi-Fi • Gigabit Eth • Proprietary network (ISM band) • Safety spectrum (Non-ISM band)	• oneM2M • Thread • AllSeen • OPC / OPCUA	• MQTT • LWM2M • CoAP • AllSeen

Fog Computing

IoT Edge Duties

- Hardware and Field Abstraction
  - Sensor Connectivity
  - I/O Access
- Manage Network and Connectivity
  - Wireless, Firewall, VPN
  - Online/Offline mode
- Manage Applications
  - Remote Device Management
  - Remote Start/Stop, Install/Uninstall
- Manage IoT Connectivity
  - e.g, MQTT/CoAP Connections
  - Data Buffering and Retries
  - Provisioning, Credentials and Certificates

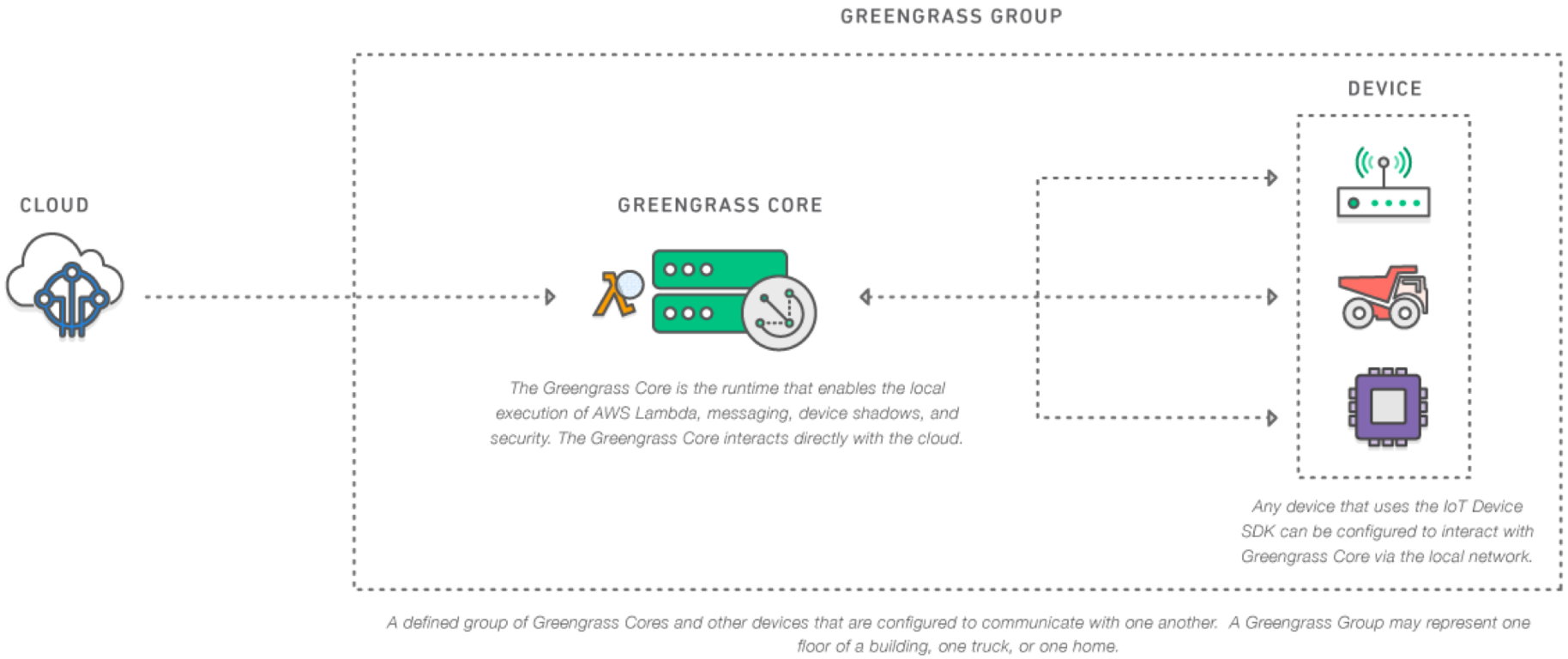
What is Greengrass?

- An extension of AWS IoT, including Lamba, Broker, same security model, critical shadow functions
- Greengrass Core: The runtime responsible for Lambda execution, messaging, device shadows, security, and for interacting directly with the cloud
- Greengrass Core Group:
- System Requirements: Min Single-Core 1GHz, Min 128MB RAM, x86 and ARM, Linux(Ubuntu or Amazon)

- GreenGrass SDK, in C++ at this point

Why Greengrass?

- Low latency processing
- Process your data on the Edge before sending to Cloud
- Run completely offline



AWS Greengrass

Components

- Greengrass Core: Greengrass Core is the runtime that enables the local execution of AWS Lambda, messaging, device shadows and security. Core interacts directly with the Cloud.

Components

Local Lambda

- Lambdas are event-driven compute functions. With Greengrass, you can write Lambda functions in the cloud and deploy them locally

❑ Command and control

❑ Offline operation

❑ Data filtering & aggregation

❑ Iterative learning

- Greengrass runs Lambdas written in Python 2.7
- Invoke Lambda functions with messaging and shadow updates

Components

Shadow

- JSON documents that represent state of your devices and Lambdas
- Sync to the cloud or keep them local

Messaging

- Local MQTT Pub/Sub messaging
- Define subscriptions between publishers and subscribers
- Apply MQTT topic filters

Security

- Mutual auth, both locally and also with the cloud
- Certificate on your devices can be associated to SigV4 credentials in the cloud

Use Case – Gas Turbine Filter Maintenance

- Each turbine consists of multiple devices and sensors
- Each turbine has a single Greengrass core
- Lambda functions have enough logic to perform edge machine learning
- Models and Lambda changes can be centrally updated
- Lambda filters data and uses its edge machine learning capabilities to determine when to request maintenance/replacement of filters
- Only required data is sent to AWS IoT

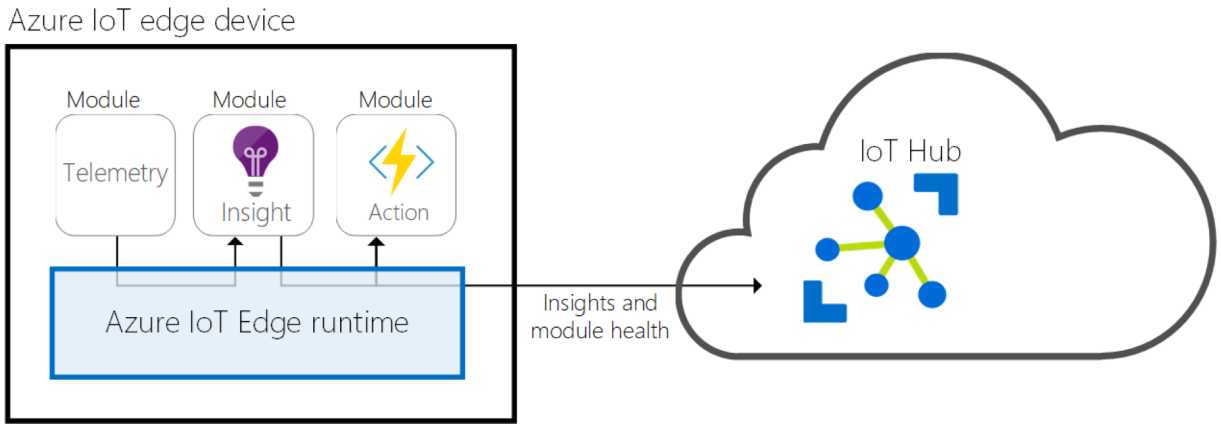
Azure IoT Edge

Github Page: `https://github.com/Azure/iot-edge`

Azure IoT Edge

Edge Runtime provides fundamental services:

- Manage connectivity and security for devices otherwise isolated from Internet
- Multiplexing capabilities
- Store and forward(Offline)

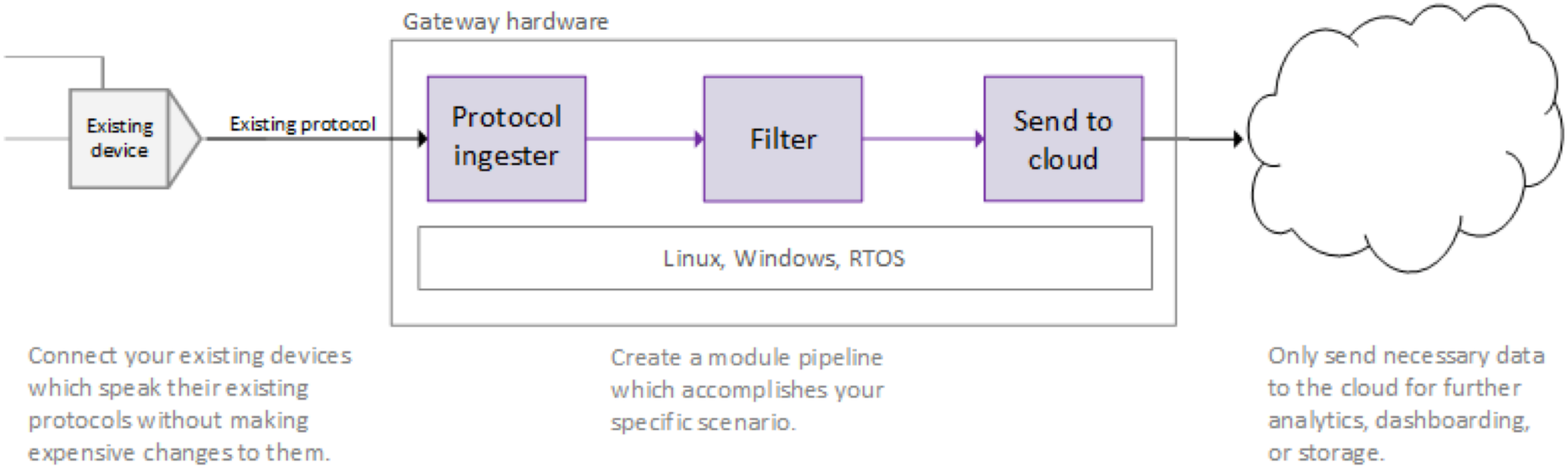


Microsoft Azure IoT Edge

Azure IoT Edge

Edge Runtime manages modules:

- Modules add capabilities to the runtime
- Each Module performs an action
- Chain of modules can be thought of as a data processing pipeline, solving an end to end scenario



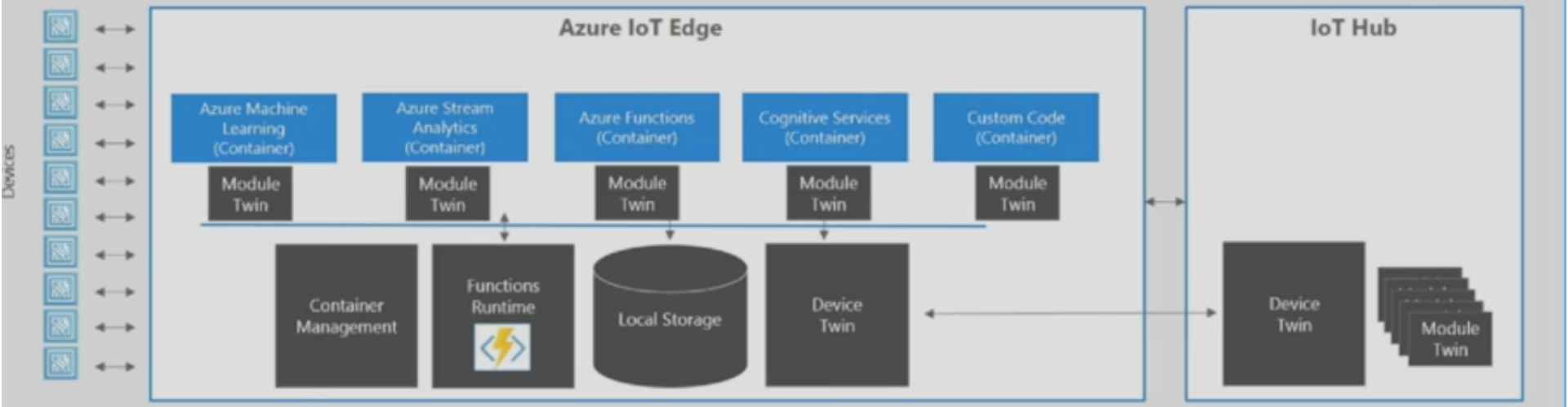
- Modules are Docker containers.
- You can create custom modules in the language you choose.

Microsoft Azure IoT Edge

Azure IoT Edge

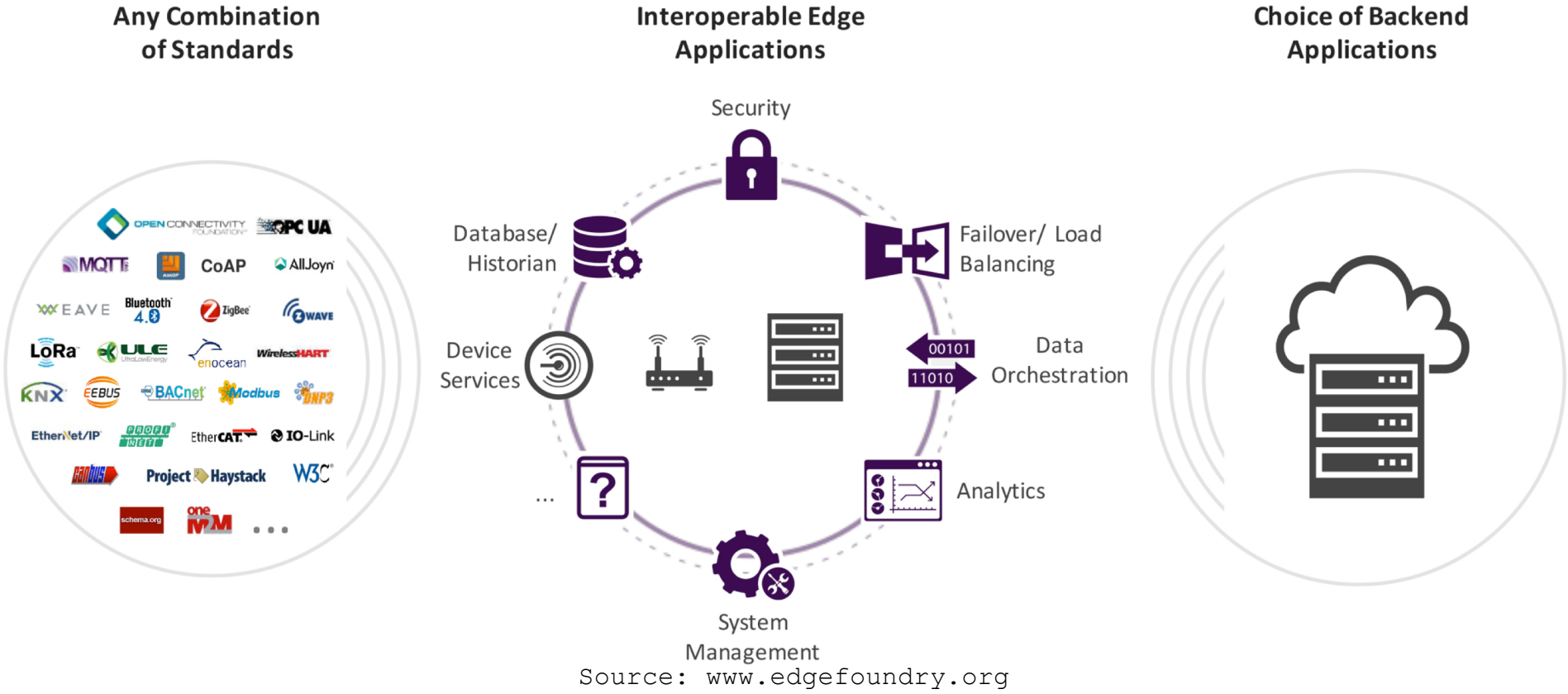
- Container based modules
- Azure Functions
- Azure Stream Analytics
- Azure Machine Learning
- Cognitive Services

- Offline / Synchronized Device Twins
- Local Storage
- Cloud Management & Deployment
- High Availability / Fault Tolerance
- Cloud Dev/Test Support



EdgeX Foundry

Overview



Source: www.edgefoundry.org

- EdgeX Foundry™ is a vendor-neutral open source project building a common open framework for IoT edge computing.
- An interoperable framework hosted within a full hardware- and OS-agnostic reference software platform, to enable an ecosystem of plug-and-play components.

EdgeX Foundry

Components

- MongoDB: EdgeX Foundry uses MongoDB as the persistence mechanism for sensor data as well as metadata about the devices/sensors that are connected.
- Java: The open sourced microservices of EdgeX Foundry are written in Java 8.
- Eclipse: EdgeX Foundry micro services were created in Eclipse using Java Maven projects.

- https://www.edgexfoundry.org/

Eclipse Kura

Open Source