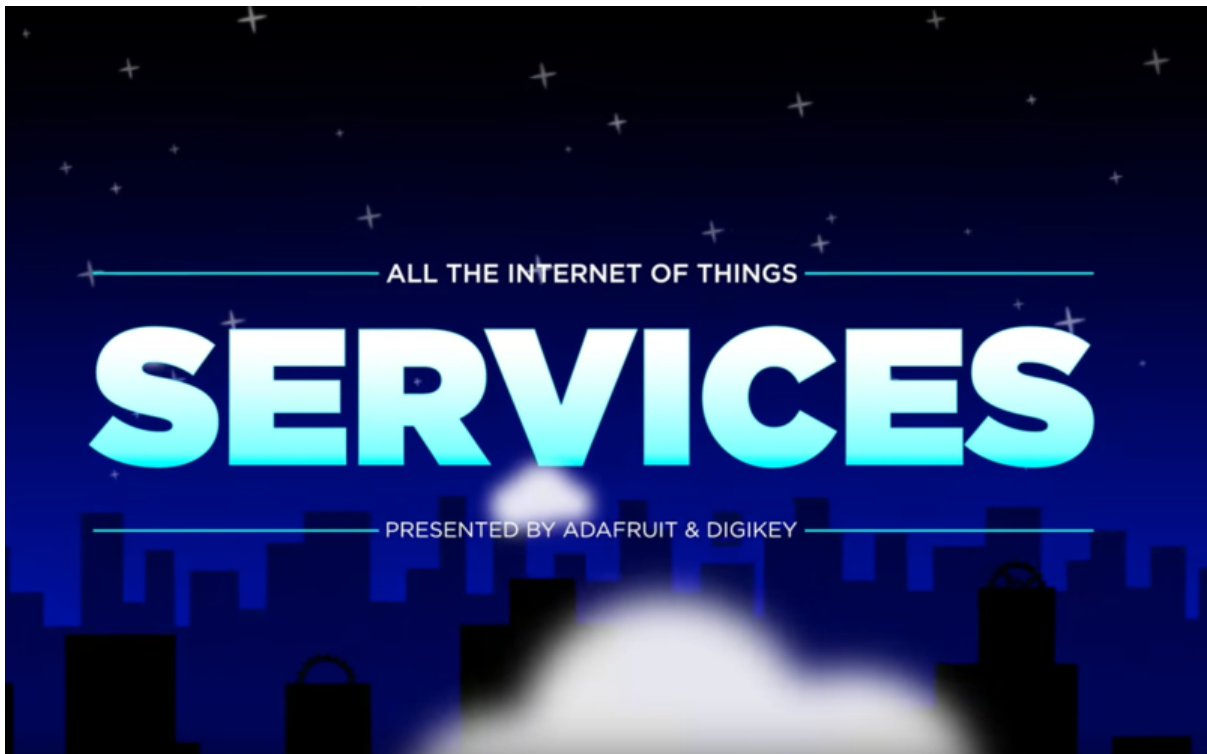




All the Internet of Things - Episode Three: Services

Created by Biniam Tekola



<https://learn.adafruit.com/all-the-internet-of-things-episode-three-services>

Last updated on 2024-06-03 02:44:11 PM EDT

Table of Contents

Introduction	3
<ul style="list-style-type: none">• Hello!	
Services for Things	3
<ul style="list-style-type: none">• Which service is right for you?	
Storing and Processing Data	4
<ul style="list-style-type: none">• We're going to need a bigger hard drive• Servers to the rescue	
Communication with Other Devices	5
<ul style="list-style-type: none">• Communication is Key	
Talking to the Internet	7
<ul style="list-style-type: none">• You can't spell IoT without "Internet"	
Configuration and Updates	8
<ul style="list-style-type: none">• Upgrade your update!	
Dealing with Humans	9
<ul style="list-style-type: none">• Me, myself, and UI	
Service Providers	10
<ul style="list-style-type: none">• At your Services	
Analytics and Data Visualization	11
<ul style="list-style-type: none">• Initial State• Plotly	
Prototype-Friendly IoT Infrastructure	12
<ul style="list-style-type: none">• PubNub• Carriots	
End to End Solutions	14
<ul style="list-style-type: none">• End-to-End and back again• Particle.io• Electric Imp	
Large Scale Infrastructure as a Service	15
<ul style="list-style-type: none">• Trade-offs to consider:	
Conclusion	16

Introduction

Hello!

Welcome to Episode 3 of Adafruit and Digi-key's All the Internet of Things series - a six part program which covers to everything you need to know about IoT. In the first two episodes, we looked at [Transports \(https://adafru.it/FXS\)](https://adafru.it/FXS), the physical and wireless mechanisms used to transfer data between things, and [Protocols \(https://adafru.it/FXT\)](https://adafru.it/FXT), the communication standards which enable devices at each end of a transport to "speak the same language" and understand what is being communicated.

Now that we have our things connected and talking, it's time to make them work together to do something useful, and that's what this episode, **Services**, is all about.

Services for Things

Which service is right for you?

Like the human internet, there are services available for almost anything you might want your devices to do. There are free services, paid services, enterprise-level, and DIY maker-friendly services available. Some services perform only one specific function, some allow you to design your own custom sub-services, and everything in between. As you might imagine, there are also innumerable useful services that don't even exist yet, just waiting to be conceived.

The choice you make at this stage depends on what you've learned in the previous two videos. If the service you've got your eyes on does not support the [MQTT \(https://adafru.it/f29\)](https://adafru.it/f29) protocol, it can have a big impact on your data and battery usage. Likewise, if you picked a transport like SigFox, you'll need to make sure your service can hook into the SigFox gateway network.

One helpful way to investigate all of these services for things is to list all the processes your "things" need assistance in performing. Most of these processes can be broken down into a handful of categories: storing, retrieving and processing sensor data, coordinating communication with other IoT devices, talking to non IoT stuff on the Internet, receiving configuration changes, and dealing with humans. Knowing what you need will help you choose which service may be right for you!

Storing and Processing Data

Just about every IoT service provides stable long-term storage for the data your things are producing. Why is that?

Take this [simple IoT weather station \(https://adafru.it/FXU\)](https://adafru.it/FXU) as an example:

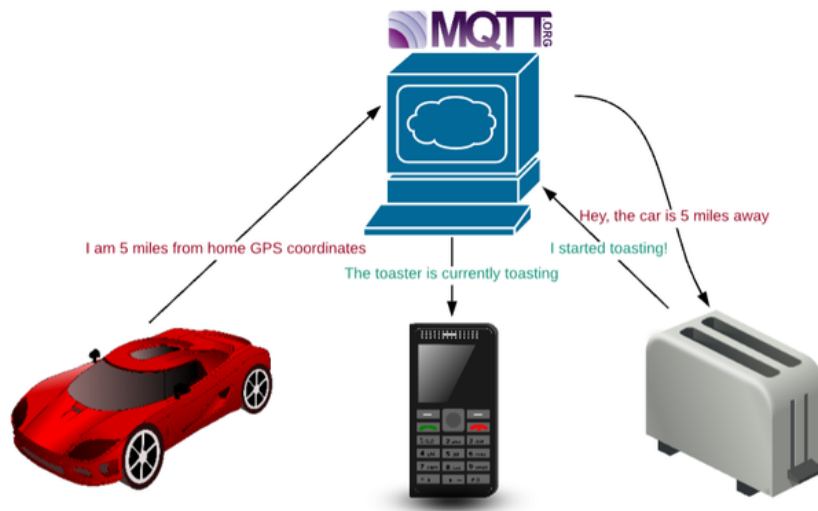


Say this is measuring the temperature once per second and storing that as a single byte of data, It will produce over 3 kilobytes of data every hour. Over a year, that will grow to over 30 megabytes!

Sure, 30MB doesn't sound like much from the perspective of a laptop or workstation, but for small power efficient embedded hardware, such as one of Adafruit [ESP32 \(http://adafru.it/3405080980\)](http://adafru.it/3405080980) or SAMD-based [Feather devices \(https://adafru.it/BC6\)](https://adafru.it/BC6), which measure storage space in kilobytes, 30MB is a very big deal.

We're going to need a bigger hard drive

If this weather station is also producing data for humidity, barometric pressure, and wind speed, it's easy to see that we're going to need to store this data off-device. Even if you have a high powered single board Linux computer, with 8 Gigs of storage, you're one bit away from a corrupted filesystem, with all the data lost. If you imagine having tens, hundreds, or thousands of these devices, all producing weather data from different geographic locations, it would be very convenient to have all of this data automatically collected together in one, backed-up, place.



Servers to the rescue

Don't forget, your devices are connected to the internet! So there's no need to try and store all the data on the device itself. Instead, your Thing will use services to store the data they produce at the very moment they produce it. Now that service is responsible for storing this raw data, typically in a time-stamped database. They'll also provide a way to access the data, either in the form of a user interface, and/or as an API which other Things or apps can use.

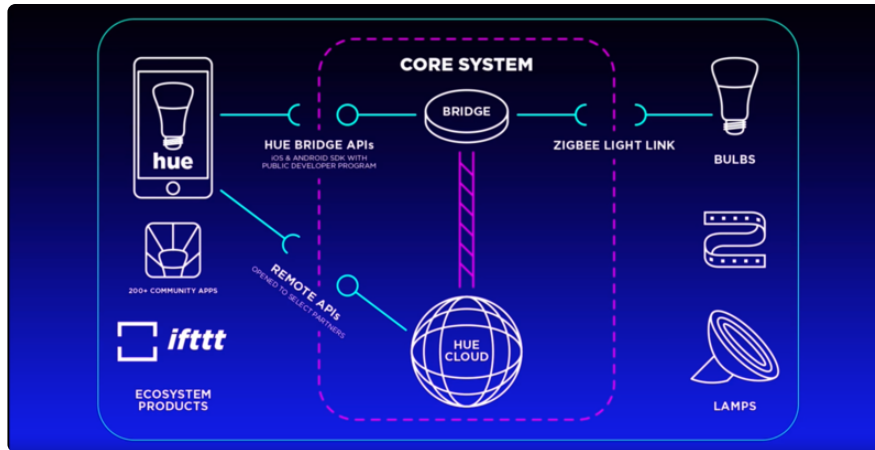
Communication with Other Devices

Communication is Key

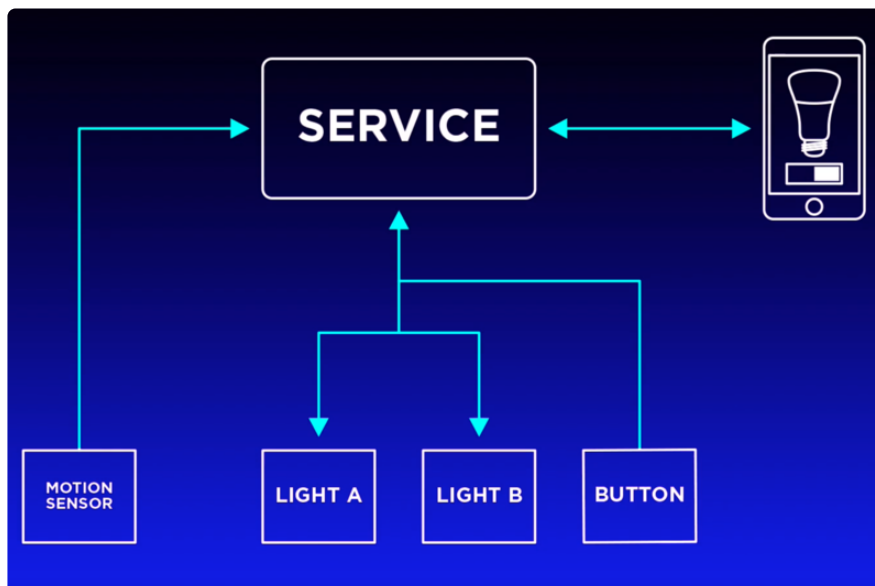
For data-creating objects, we're set. But there's plenty of devices we've demonstrated that don't just manage sensor data. Instead, they're used for autonomous or remote control, like an IoT lamp. Another common process that services provide is a mechanism for multiple devices in an IoT application to communicate and receive events from each other.

If you happen to have a setup where all devices are in the same location, using the same transport, on the same physical network, speaking the same protocol, and within range of each other, it's possible that they can communicate directly without a service. An example of this would be how Philips Hue lights and switch controls communicate with a local bridge device over Zigbee.

But that's an uncommon situation these days, and can be very constricting. For example, if you wanted to turn off all the lights in your home, using an application on your phone or computer, you'd need to install a ZigBee modem. And once you get out of that 30-meter ZigBee range, say because you're at work, or out of town, you'd be completely out of luck.



In the case of the Hue lights, this is solved with an internet-facing IoT service. The Hue bridge chats with all those lights and switches over ZigBee, but will also connect to an online message-passing service at meethue.com. That services forwards events and messages between the bridge in your living room and the phone app. The phone app and light system can independently connect to that messaging service, and the service will broker communication between them, even though the devices cannot connect directly.



Another common scenario is when multiple devices need the ability to publish and subscribe to the same events. Rather than having each device monitor the activity of all the other things, a service makes it possible for each device to use a single,

efficient messaging channel which produces and consumes only the events that are pertinent to the particular device.

Talking to the Internet

You can't spell IoT without "Internet"

This series is called the Internet of Things, so having internet access is obviously going to be one of the common service-desires! As we've shown in the Protocols video, there's thousands of API's available, from the common news/stocks/weather and beyond. Many of the available IoT services support this sort of functionality by allowing you to create "event listeners" - specific conditions that run code on the host service to issue a REST call, often referred to as a "webhook," to a remote server. You can use this feature to easily integrate your service with almost any REST API on the Internet.

For example, perhaps you want your weather station to publish an update to your web site or post a status message to Twitter. In your service's administration console, you would configure a custom listener to watch for a specific event to occur, such as a temperature update event. The exact logic of when the event should fire is customized using a plugin or some scripting language provided by the service. For example, you might want the service to trigger an update on your website with one REST call, and then conditionally publish a message through Twitter's API with another REST call, but only if the temperature is above a certain threshold.

```
when get_temperature() {
  # Always post to my site on every reading
  REST_POST(www.mysite.com/publish?temp)

  if (temp < 10) or (temp > 40) {
    # Wow its really hot or cold!
    # Lets post to twitter

    REST_POST(api.twitter.com/post?tweet=
              "The temperature is "+temp)
  }
}
```

You can even use this functionality to connect multiple IoT services together. You might choose one service for its robust data storage and event processing features, and then use a webhook to connect your core service platform to another service that you prefer for analytics. Event listeners, custom functions, and webhooks provide a surprisingly versatile way to build custom business logic on top of your application.

And, for the most popular APIs on the Internet, you'll often have a ready-to-go plugin provided.

Configuration and Updates

Upgrade your update!

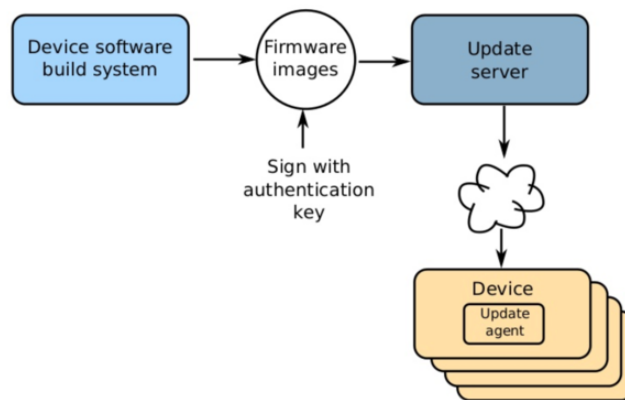
Once your project involves multiple devices, the ability to manage configuration settings, credentials, and firmware updates at the device level becomes your greatest challenge. Unsurprisingly, many services provide an administration panel as well as a REST API for configuring and maintaining your Things.

On smaller projects, you might use the administration features to set everything up manually, registering each device in the admin panel, and then copying a device ID or private key to each device. Compare this to a commercial product, where you'd want users to be able to unbox their device and register it from your website. For this, you'd need to produce and deliver access credentials to the device in a more automated way, and then publish these access credentials to the service via its REST device configuration interface.

Once your devices are registered and connected, some services also provide a channel for remotely pushing configuration data to your devices. Usually, this is just a built-in way to push a developer-defined blob of data to the device, and it's up to the developer to sort out the specific details of what the device does with it. It could be something as simple as updating configuration metadata, or as complicated as pushing over-the-air firmware updates. This kind of service-activity should not be taken lightly. Updating devices with pushed data is a massive security and operations risk. If your update procedure is hackable, your sensor network could be turned into a distributed botnet. If you make a mistake in deployment, every one of your devices turns into a brick.

This is an example of how over-the-air updates might happen:

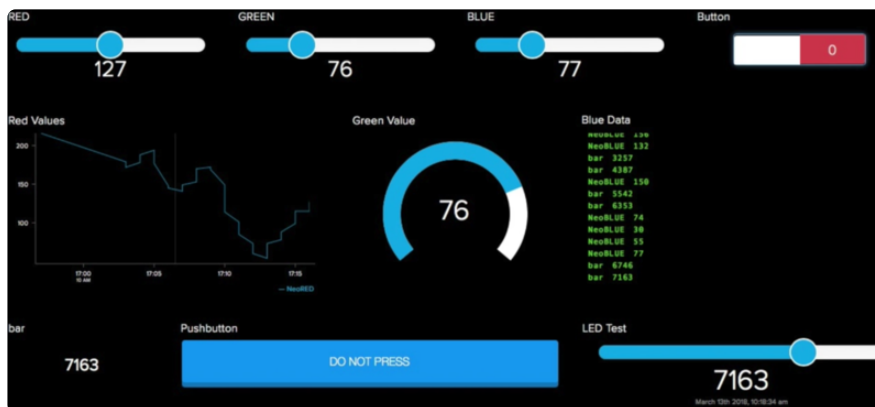
OTA update components



Dealing with Humans

Me, myself, and UI

Ultimately, almost every Internet of Things project will, at some level, need to provide a way for the things to deal with humans. User interface considerations can range from a mobile app that allows a user to turn lights on and off and adjust the volume on a music player, to output features such as status monitors, analytics, and data visualization dashboards.



Most providers with a publish/subscribe service for Things will also provide a friendly website and REST or MQTT API's for you to see and use the data stored. Your end-user applications can be built on top of those APIs to receive device status, manage deployments, maybe even publish messages directly into the event stream. If there's a mobile app, look for providers that offer native notification support for iOS and Android. Native notifications can give your IoT applications a way to tell the user that something is happening, even while your app is in the background or not running.

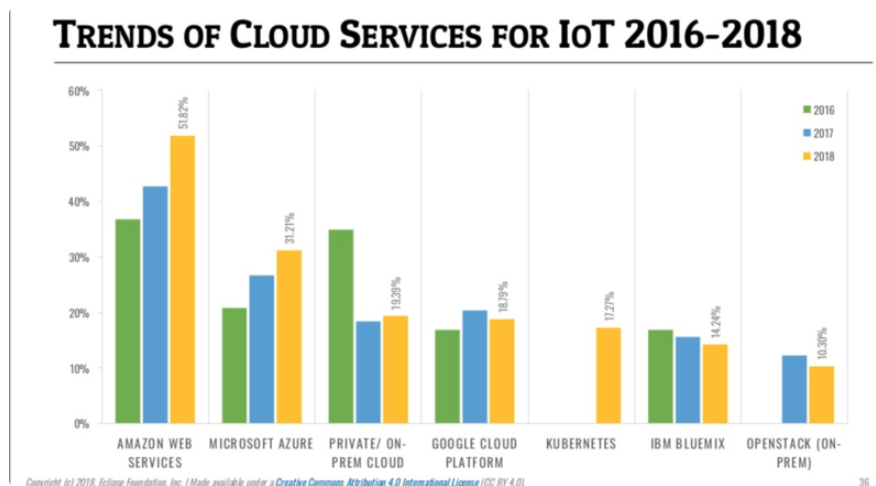
There are also many services which are focused completely on analytics and data visualization. You can find everything from turn-key analytics dashboards, to advanced charting software development kits (SDK) that let you build custom interfaces for your favorite platforms and languages. These can be great for industrial and architectural IoT applications where there's data-overload.

Most importantly, you, as an engineer, may be really great at RF layout - but not so great at UI design. If you can leverage the service's dashboard, that's one less thing you have to code up before launching a product.

Service Providers

At your Services

There are many, many services available that have been designed for storing, connecting, configuring and visualizing your IoT projects. From giant corporations like Amazon, Google, and Microsoft, to smaller, IoT-focused companies like PubNub, Initial State and Adafruit, you're likely to find that many of the services you need have already been built, allowing you to focus more on your project rather than designing, building and maintaining a complete service infrastructure. Even if you want to host your own data, there are open source IoT platforms that allow you to self host your entire service infrastructure, much like privately hosting your own database or web servers.



Like transports and protocols, choosing the right service providers will depend a lot on your project requirements. What's "best" can depend on whether you're looking to create a personal project, quickly iterate on a prototype, manage an industrial deployment, or support the needs of a popular consumer product.

Most of the services currently available have been designed with one or more of these needs in mind, and can be roughly divided into the following categories:

- analytics and data visualization
- prototype-friendly infrastructure
- hardware-specific end to end solutions
- and large scale infrastructure

Even if you end up changing your mind, there's a good chance another service out there will support your transport and protocol, so you don't have to start all over.

Let's take a look at some examples of each.

Analytics and Data Visualization

Collecting and examining time-stamped data is a really common use case. Maybe you're tracking soil moisture on your farm, or temperature data from each room in a building. You'll want to keep a historical record, look for trends, and trigger other actions when appropriate. These services can be used to build a beautiful dashboard for your hobby project, an interface for monitoring the status of industrial system, or even an analytics back end for testing and debugging a consumer product in the field.

We'll look at two examples: Initial State and Plotly.

Initial State

Initial State provides analytics and custom dashboards for time-series data. You can post event data to the service using REST either directly from your device, or as an output from another service such as PubNub or Carriots (more on these later). You could also use this service to import comma separated values (CSV) log data that you captured from a device you're debugging or prototyping.



Plotly

Plotly can be used to create interactive charts and custom dashboards. Most of their products are focused on data science and visualization, but they also have an REST API and an HTTP-based streaming endpoint which you can use to publish device data. The streaming endpoint allows your devices to send a series of events to the service using a single long-term connection to the server. They provide libraries and example projects for Arduino and Raspberry Pi, which might be just what you need for that project you've been thinking about building.



Prototype-Friendly IoT Infrastructure

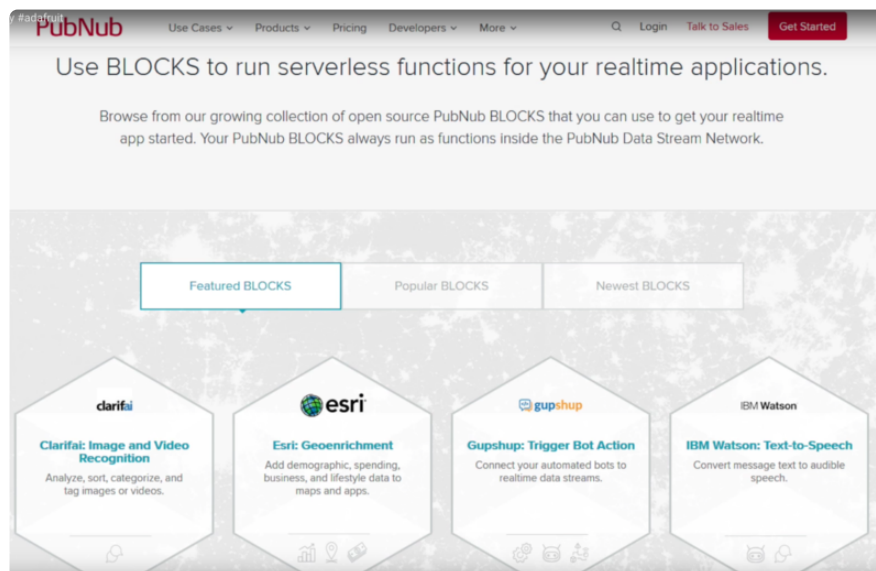
If you are doing anything that requires device communication, event messaging, or mobile integration, there are a number of services that provide cloud services, device libraries, and mobile SDKs to make that happen. We give some of these the “prototype-friendly” designation, but not because they are only applicable for hobby or smaller projects. These services put particular effort into making it really straightforward to get something working, are likely to support many of the devices

used in hobby or prototype work, such as the Raspberry Pi or ESP32-based hardware, and might even have demos and howtos for solving common IoT use cases.

Some examples in this category are PubNub and Carriots.

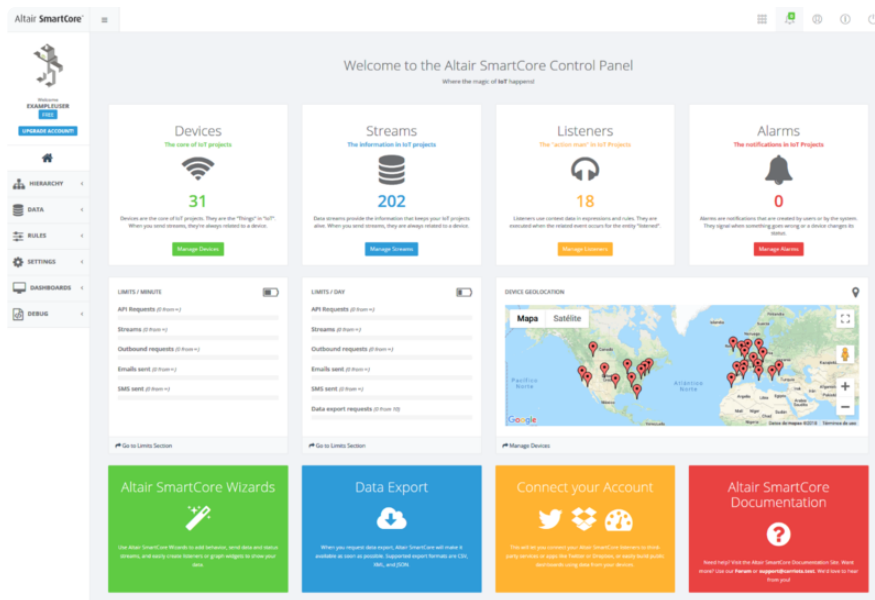
PubNub

At the core of [PubNub \(https://adafru.it/FXW\)](https://adafru.it/FXW) is an MQTT-based Publish/Subscribe service. In addition to allowing events to be streamed to and from devices, the PubNub backend also allows you to create custom functions that can be executed when specific events occur. This could be used to modify or route events in-flight, or even integrate with a REST API from another service, firing off a chunk of data whenever one of your devices publishes an event. Finally, to make it easier to get your IoT project communicating with mobile and web applications, they also provide SDKs for iOS, Android, Python and Javascript (among others).



Carriots

[Carriots \(https://adafru.it/FXX\)](https://adafru.it/FXX) provides an MQTT-based Publish/Subscribe infrastructure for data collection and routing, as well as REST APIs for device status updates, administration and provisioning. Event listeners can be configured to trigger actions when specific event conditions occur, allowing you to integrate with other services. The administration interface also makes it easy to integrate directly with dashboard services like Initial State.



End to End Solutions

End-to-End and back again

There are also cloud services that have been created by some manufacturers of popular embedded devices. With these, you get guaranteed-working hardware with wireless certifications all done and ready to go. They provide the data storage and event messaging services and APIs that you would expect from other cloud IoT services, and you can also expect a high level of integration between their supported hardware, APIs, device libraries, and documentation. In particular, one headache that is taken care of for you is secure and reliable product provisioning and firmware deployment. These two things are very hard to DIY right, so it's best to leave it to the experts.

Two examples are Electric Imp, who create a number of "impModule" IoT-focused plug-in boards, and Particle.io, makers of the popular Photon, Electron and Particle Mesh microcontroller boards. These services could provide an easy ramp up if you're already a fan of their hardware. They both provide small-scale quantities for hobbyists with the hope that your success will lead to enterprise-level hardware purchases.

Particle.io

Particle.io focuses on supporting a prototype to rapid production business need. In addition to supporting their WiFi-only Photon, Cellular-only Electron and next-gen Mesh (which has BLE and WiFi or Cellular), they also have first class support for running the Particle firmware on Raspberry Pi. The API provides webhooks, and a

REST interface, and socket-based publish/subscribe messaging interface. Finally, Particle also provides a desktop and web-based IDE for working with their devices, libraries, and cloud APIs.

Electric Imp

Electric Imp seems to lean a bit more toward business to business and industrial solutions only. Their support is a bit more specific to their own hardware, and devices are programmed with a Javascript-like language called Squirrel. Their cloud APIs are also mostly REST-based, with the addition of an HTTP stream API that lets a device push multiple events over a single, long-held connection in addition to a long-polling HTTP event API that's used to receive push updates.

Large Scale Infrastructure as a Service

The final category is the IoT services that live within large scale cloud infrastructures, such as Amazon AWS IoT, Google Cloud IoT, and Microsoft Azure IoT Suite. These services all provide complete MQTT and REST APIs, and have the benefit of being tightly integrated with the storage and compute products that are at the core of each company's respective cloud offering.

For developers that regularly work with other cloud products provided by these platforms, their IoT cloud APIs might be a familiar and obvious fit, with benefits that include sophisticated deployment tools and robust security models.

On the other hand, a first introduction to these services can be a bit intimidating, as you soon find yourself generating X.509 certificates to authenticate your devices and navigating the nomenclature and white papers that come with the territory.

Trade-offs to consider:

- Ease of getting started -vs- applicability for larger deployment and customization
- Off the shelf solutions or build your own
- Quality of documentation, examples, howtos, and community support
- Platform longevity, vendor reputation & business model
- End-to-end platform or mix and match multiple services
- Open source, data ownership
- Dependency lock-in and migration options

- Costs to entry and scaling
-

Conclusion

Ok, so that was our episode on services. Now there are dozens if not hundreds of services out there so we just touched upon a couple of the more popular ones. Once you've picked a service you want to use and you also have your transport and protocols set up, you're going to want to think about security.

We will be covering more about security after we cover one more service that is very close to our hearts here: Adafruit's own [Adafruit IO \(https://adafru.it/FXY\)](https://adafru.it/FXY)!