



Azure IoT – The Intelligent IoT Edge- DGT0001

Digital Transformation and
the Internet of Things

Francis Dogbey
DSA



Agenda

Azure IoT Edge

- Concepts
- Architecture
- Gateway Design Patterns
- Demos

Linux

Windows

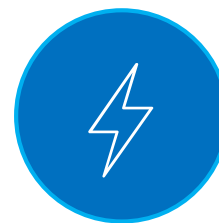
Defining IoT



Things



Insights



Action

IoT projects are still complex



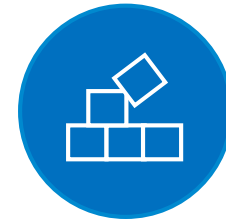
Security
is a **challenge**



Time-consuming
to get started



Incompatible with
existing infrastructure



Challenging
to scale

IOT & INTELLIGENT INFRASTRUCTURE

Remote Monitoring

Predictive Maintenance

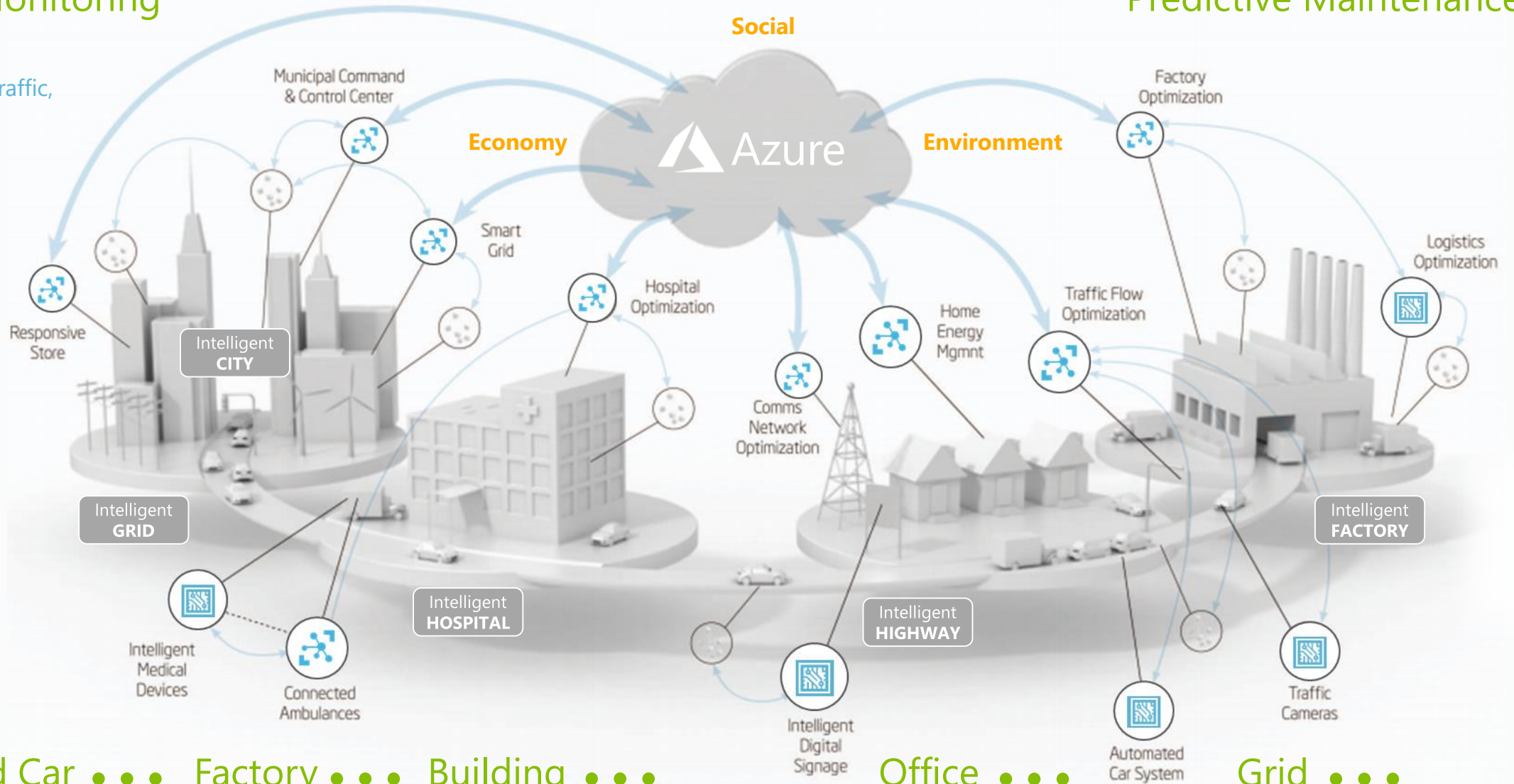
Transportation, Traffic,
Parking, Vehicles

Housing,
Buildings

Energy,
Water
& Utilities

Citizen
Engagement

Public Safety &
Security

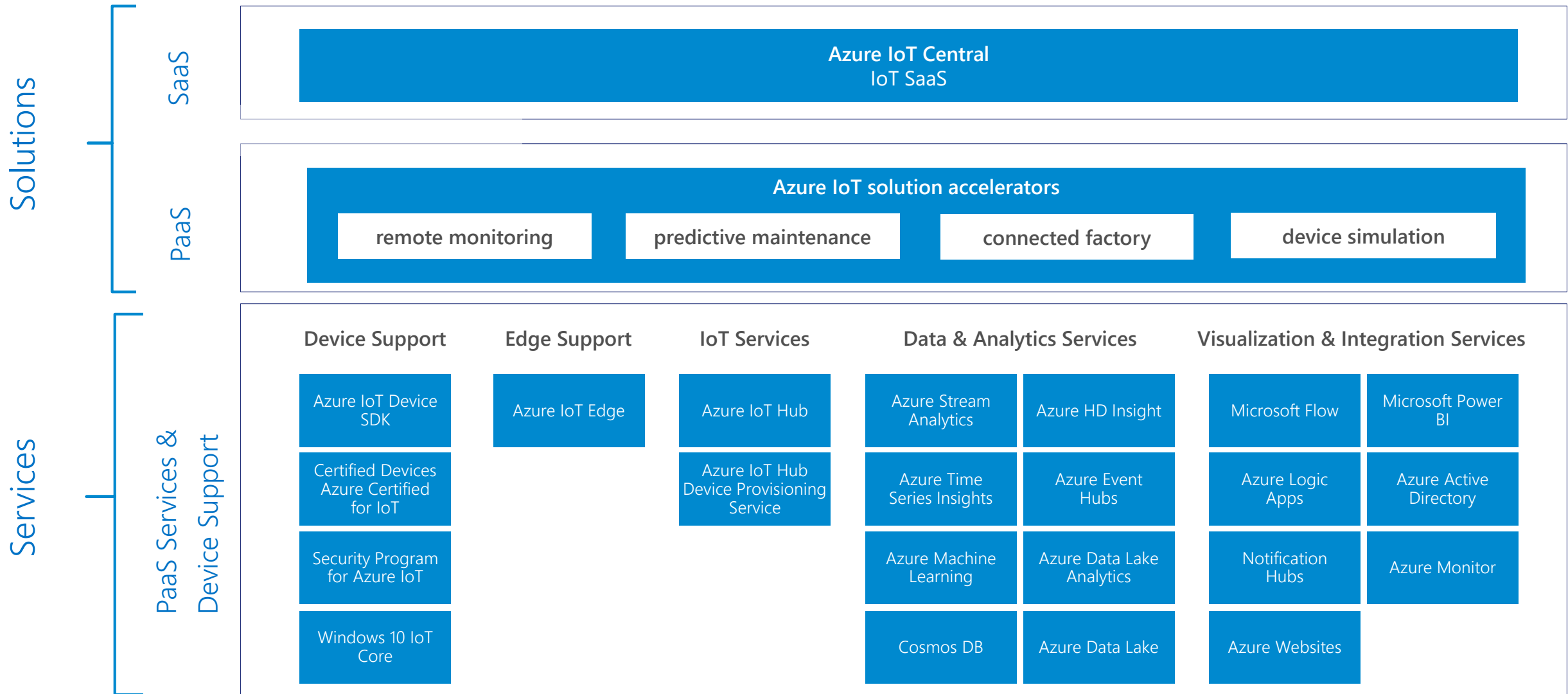


Connected Car . . . Factory . . . Building . . .

Office . . .

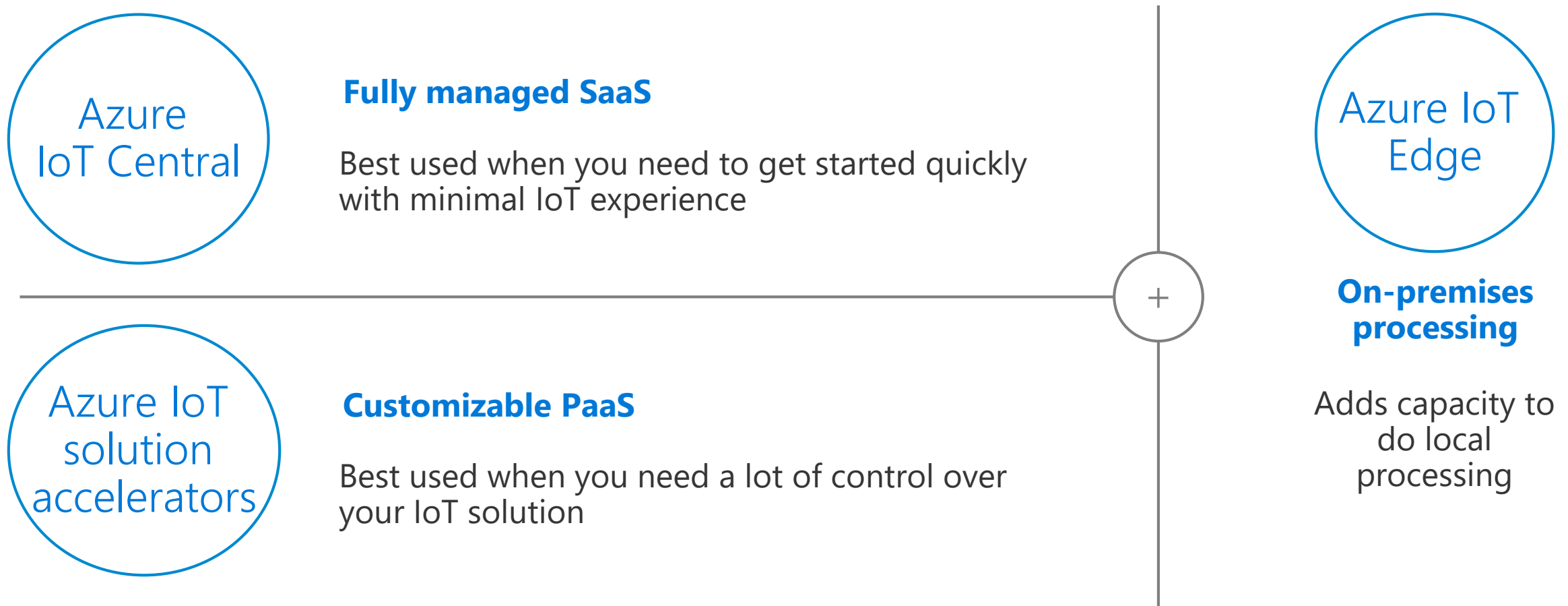
Grid . . .

Comprehensive Set of Capabilities for IoT Solutions



Azure IoT Hero Portfolio

Microsoft's vision is to democratize IoT by allowing everyone to access the benefits of IoT and provide the foundation for digital transformation



Azure IoT solution accelerators



End-to-end implementation



Completely customizable



Open-source microservices based architecture



Device connectivity and management



Dashboards, visualization and insights



Workflow automation and integration



Command and control



Preconfigured solutions



Remote Monitoring



Connected Factory



Predictive Maintenance



DPS

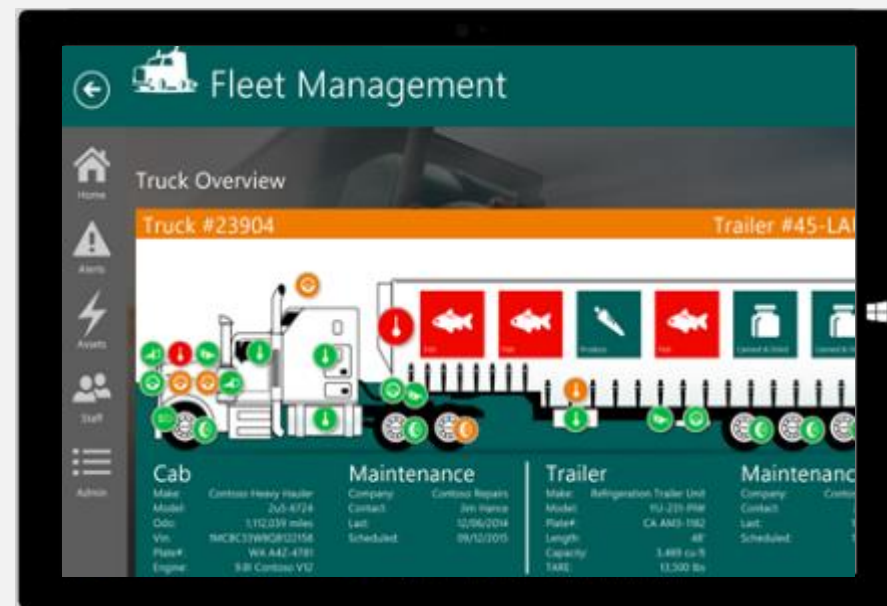
IoT Central Accelerate time to value

Start quickly for
common IoT scenarios



- Get started in minutes
- Modify existing rules and alerts
- Add your devices and begin tailor to your needs

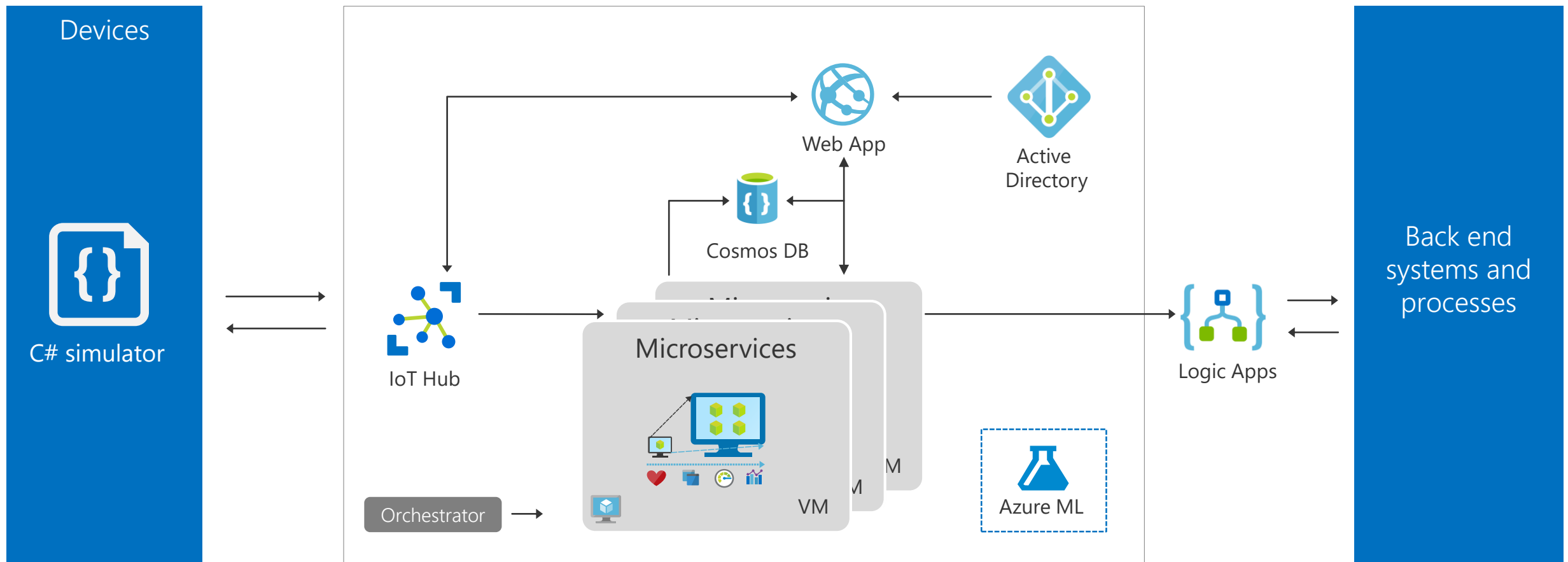
Finish with your IoT
application



- Fine-tuned to specific assets and processes
- Highly visual for your real-time operational data
- Integrate with back-end systems

Components of a preconfigured solution

remote monitoring | predictive maintenance | connected factory | device simulation





Azure IoT Edge



Move cloud and custom workloads to the edge, securely



Seamless deployment of AI and advanced analytics



Configure, update and monitor from the cloud



Compatible with popular operating systems

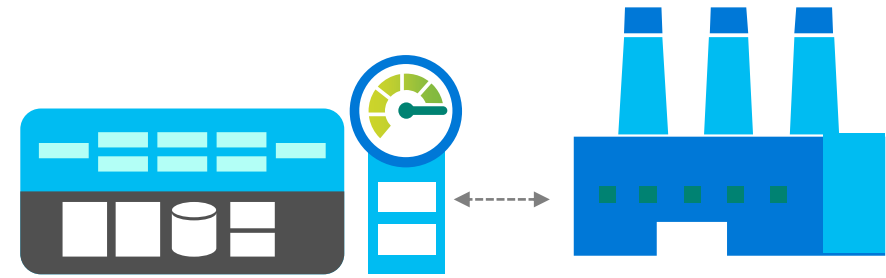
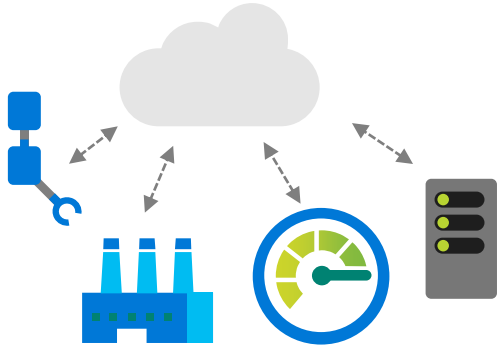


Code symmetry between cloud and edge for easy development and testing



Secure solution from chipset to cloud

IoT in the Cloud and on the Edge



IoT in the Cloud

- Remote monitoring and management
- Merging remote data from multiple IoT devices
- Infinite compute and storage to train machine learning and other advanced AI tools

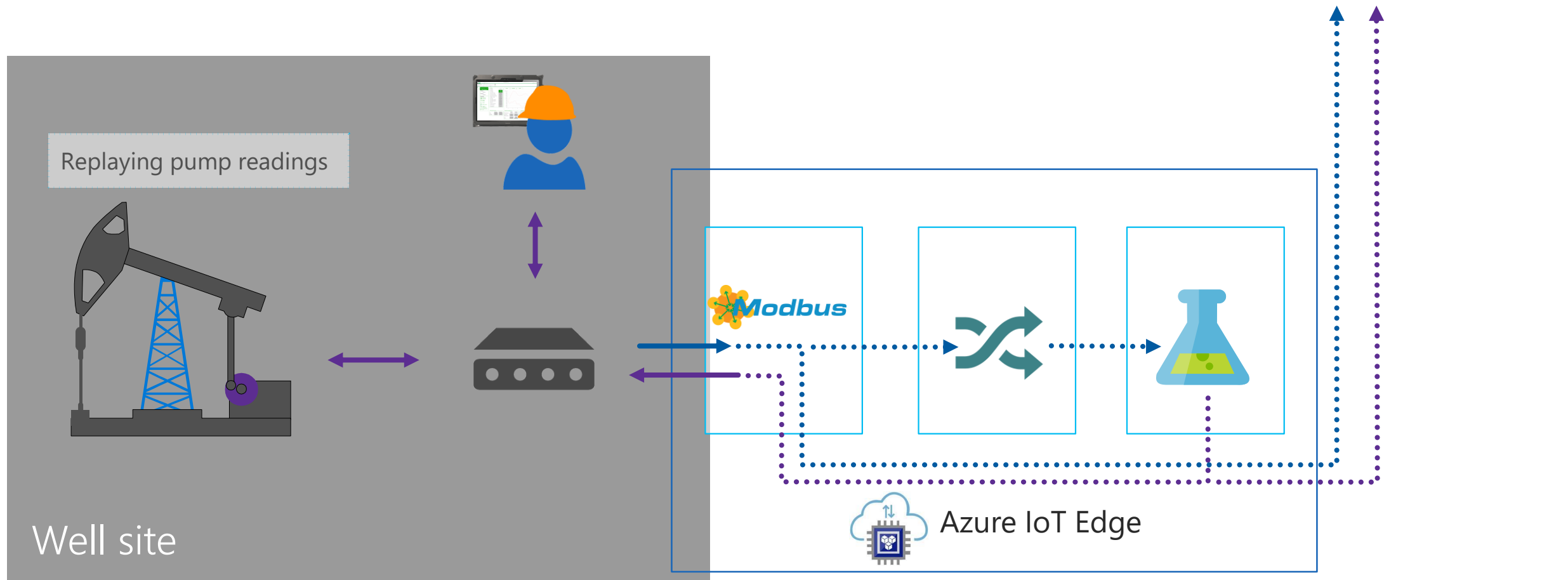
IoT on the Edge

- Low latency tight control loops require near real-time response
- Protocol translation & data normalization
- Privacy of data and protection of IP

Symmetry

Schneider – Machine Learning on the Edge

Model On IoT Edge Example



Design

Design principles

Secure

Provides a secure connection to the Azure IoT Edge, update software/firmware/configuration remotely, collect state and telemetry and monitor security of the device

Cloud managed

Enables rich management of Azure IoT Edge from Azure provide a complete solution instead of just an SDK

Cross-platform

Enables Azure IoT Edge to target the most popular edge operating systems, such as Windows, Linux, ARM, x86 Commodity HW

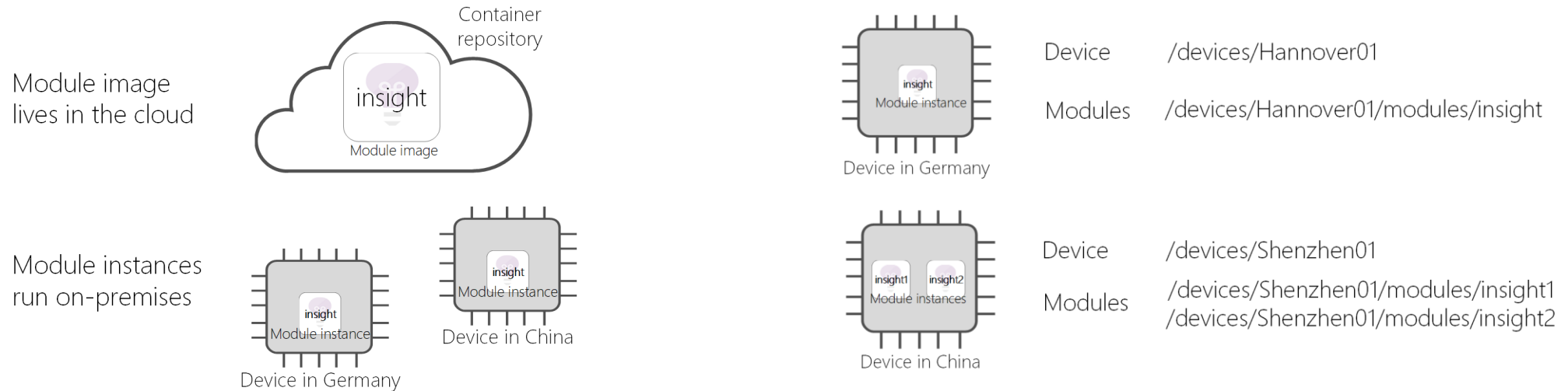
Portable

Enables Dev/Test of edge workloads in the cloud with later deployment to the edge as part of a continuous integration / continuous deployment pipeline

Extensible

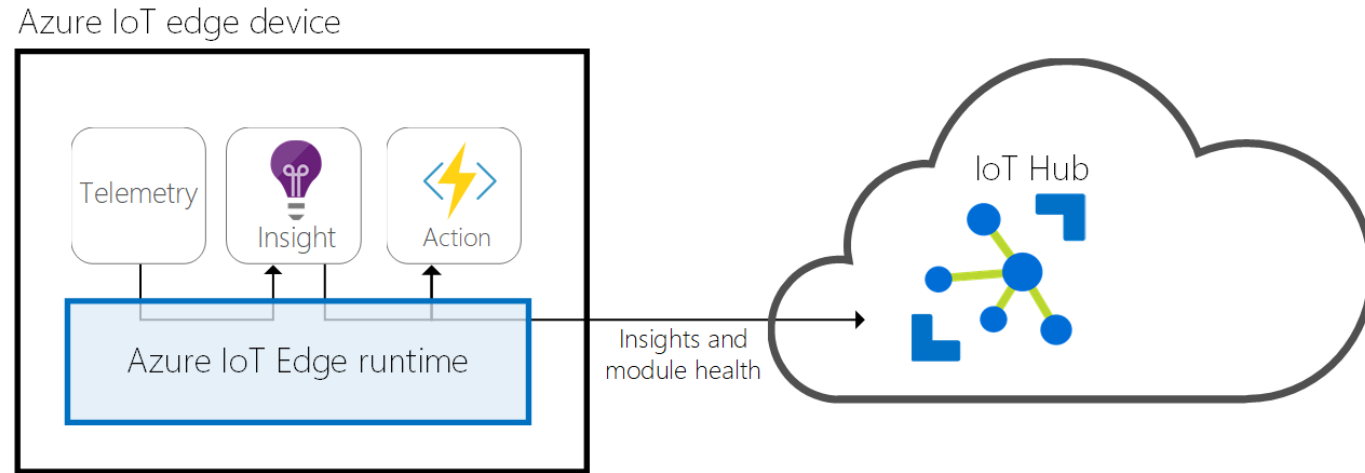
Enables seamless deployment of advanced capabilities such as AI from Microsoft, and any third party, today and tomorrow

Concept – Module



- A **module image** is a package containing the software that defines a module.
- A **module instance** is the specific unit of computation running the module image on an IoT Edge device. The module instance is started by the IoT Edge runtime.
- A **module identity** is a piece of information (including security credentials) stored in IoT Hub, that is associated to each module instance.
- A **module twin** is a JSON document stored in IoT Hub, that contains state information for a module instance, including metadata, configurations, and conditions.

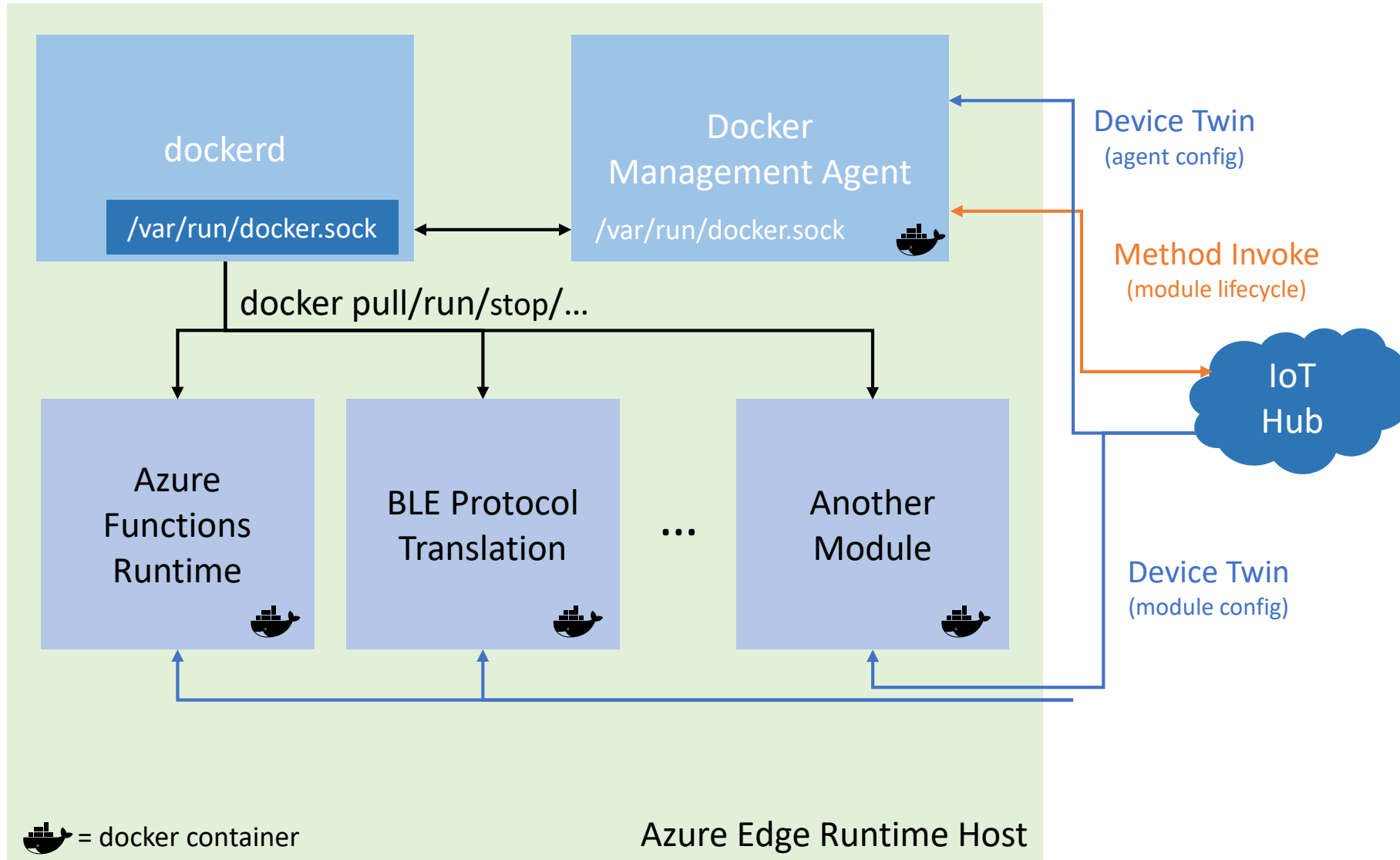
Concept – Azure IoT Edge Runtime



- Installs and updates workloads on the device.
- Maintains Azure IoT Edge security standards on the device.
- Ensures that IoT Edge modules are always running.
- Reports module health to the cloud for remote monitoring.
- Facilitates communication between downstream leaf devices and the IoT Edge device.
- Facilitates communication between modules on the IoT Edge device.
- Facilitates communication between the IoT Edge device and the cloud

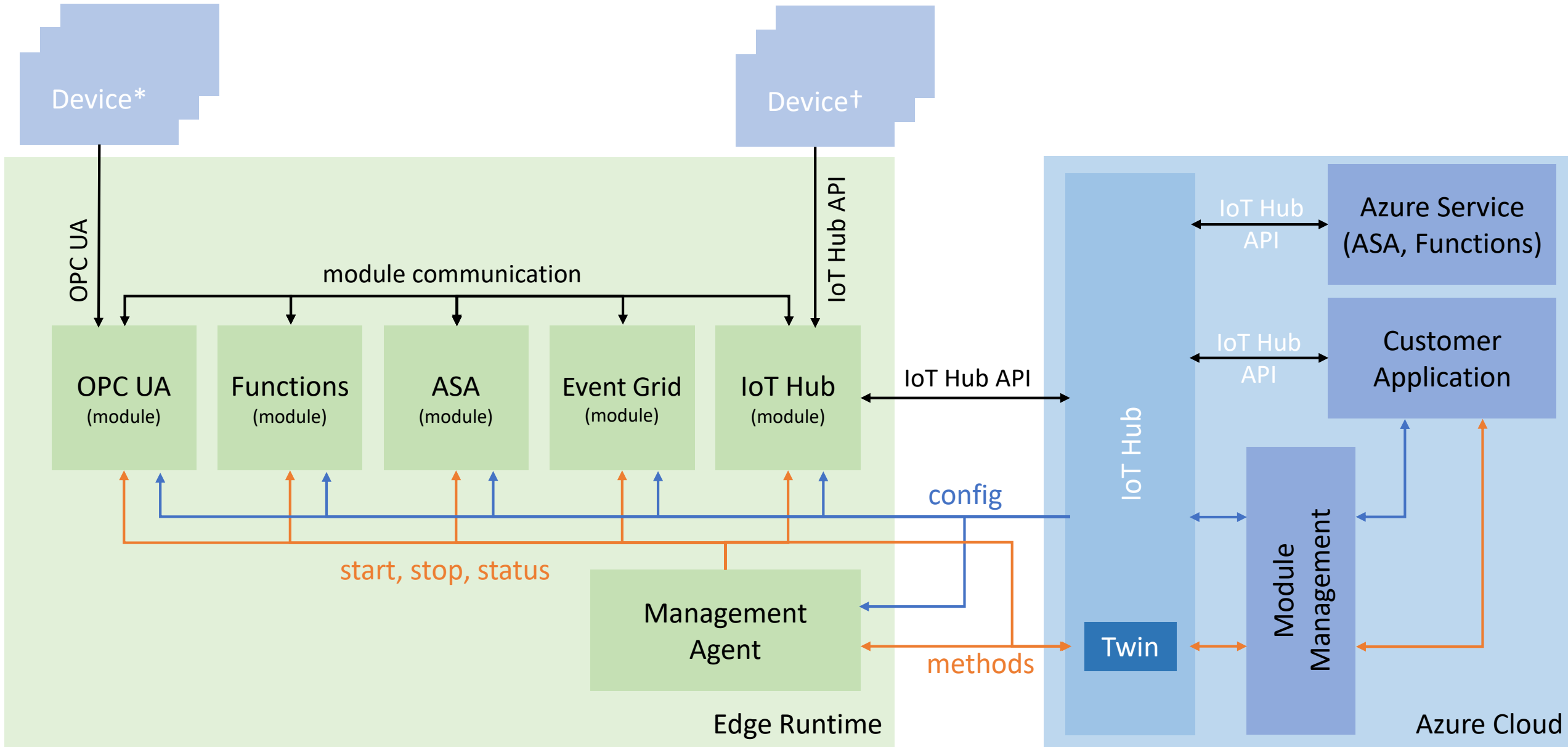
Edge Runtime

Using Docker Containers



Note: containers allow modules to be developed in language/framework of choice

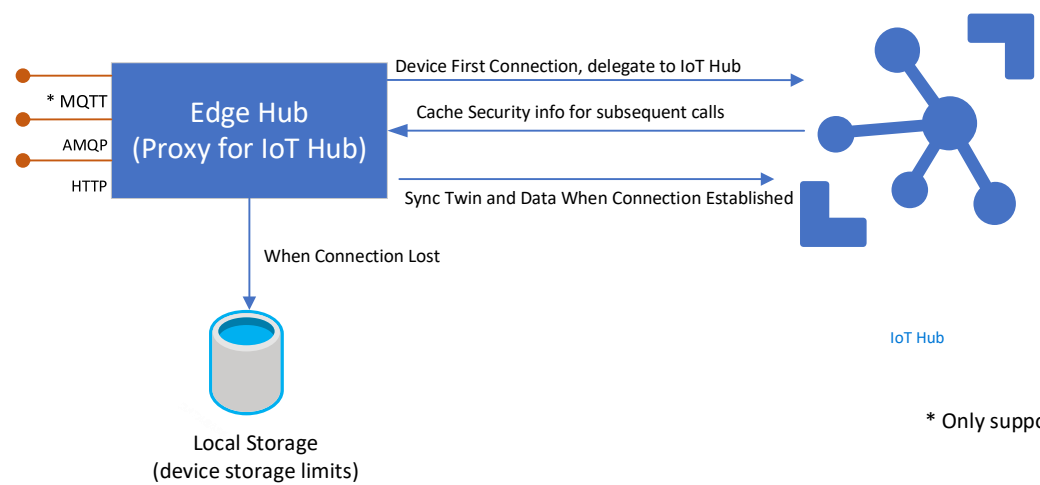
- docker command
- config (device twin)
- control (methods)



* Devices requiring module for protocol translation
† Devices capable of using IoT Hub SDK

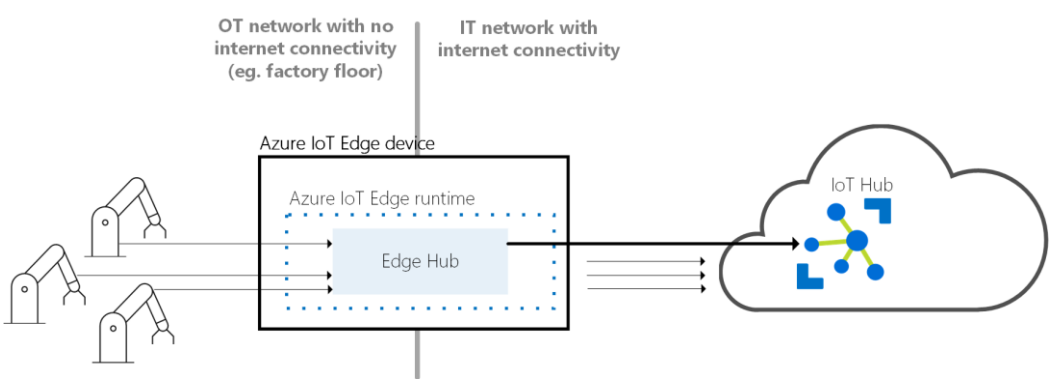
Azure IoT Edge Runtime Modules – Edge Hub

Edge Hub

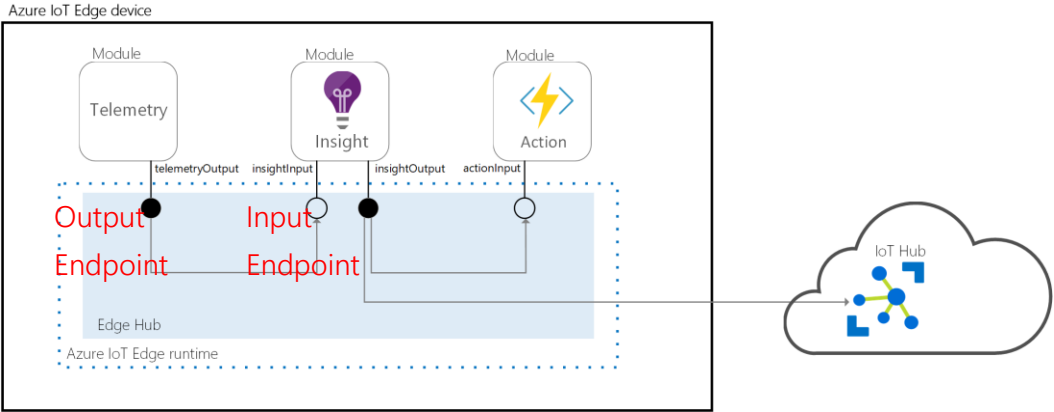


* Only support in public preview

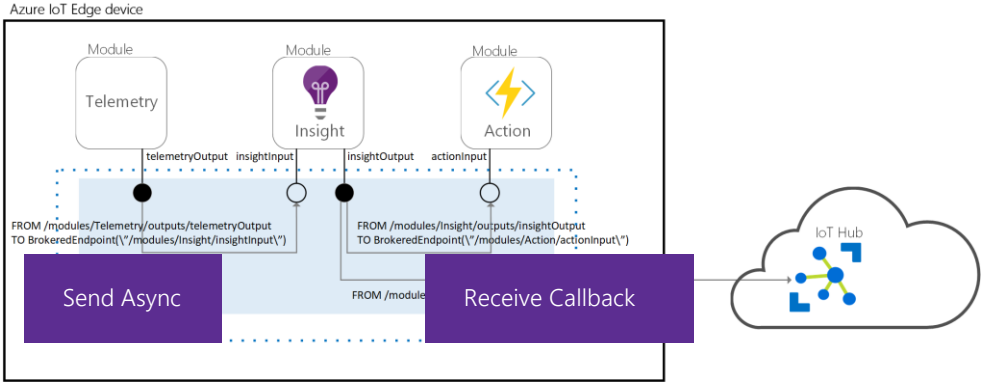
Supports Multiplexing



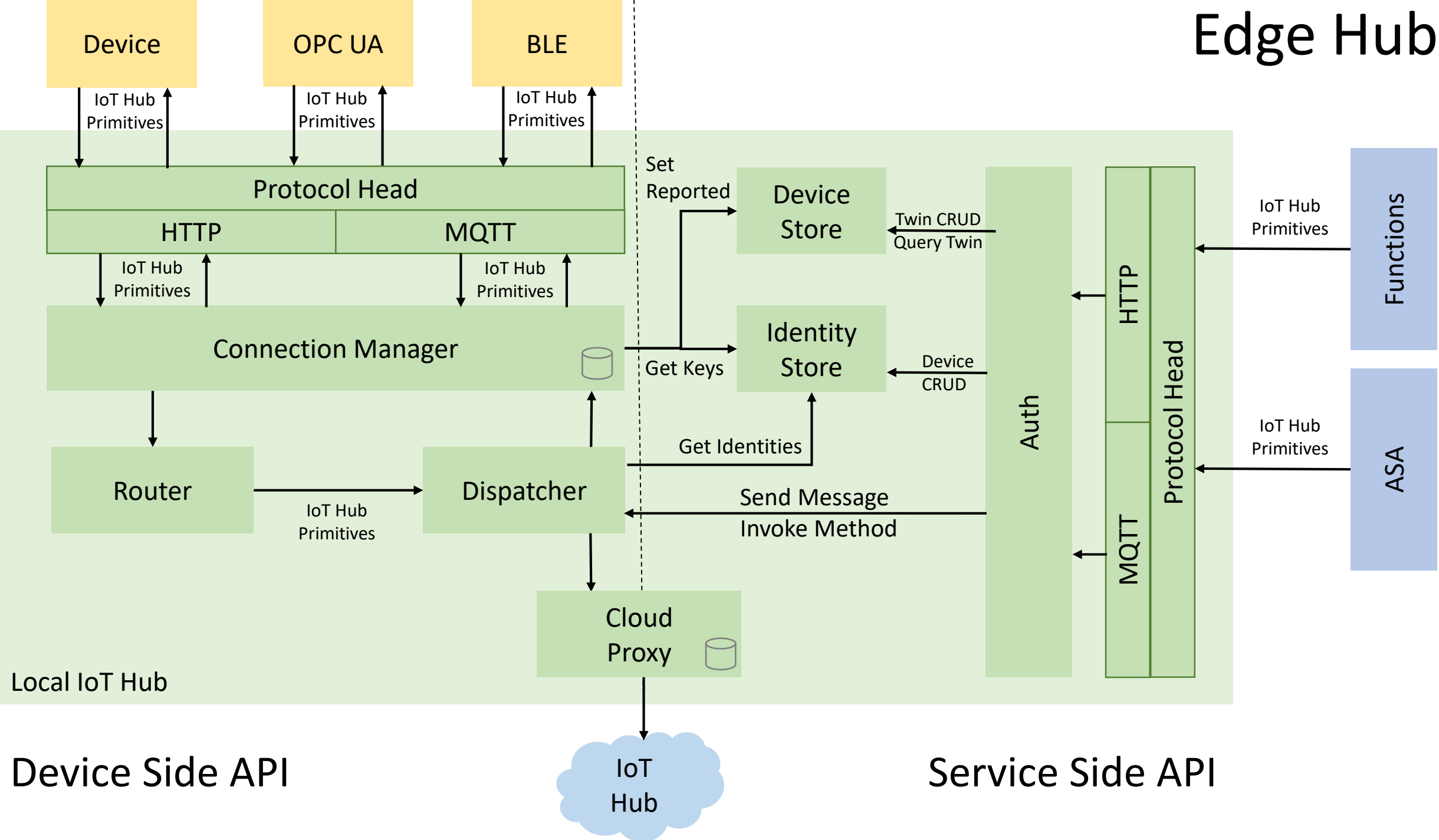
Module Communication



Send/Receive Data

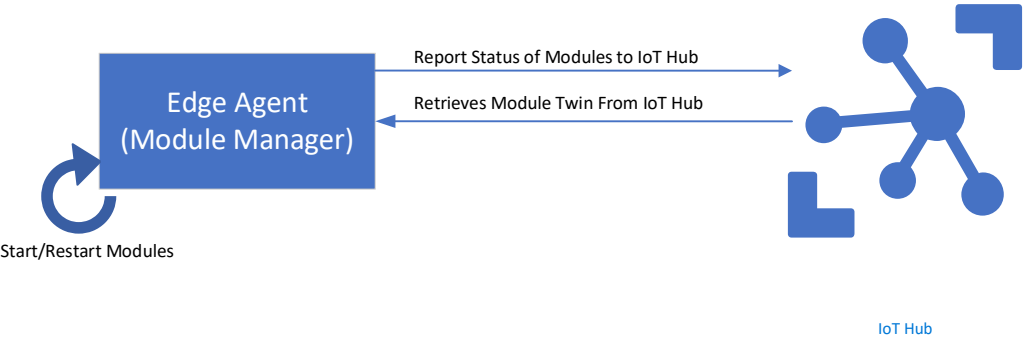


Edge Hub



Azure IoT Edge Runtime Modules – Edge Agent

Edge Agent



Module Dictionary

Image Settings	Create Options (Sent to Docker)
Status	Restart Policy

Supported Statuses

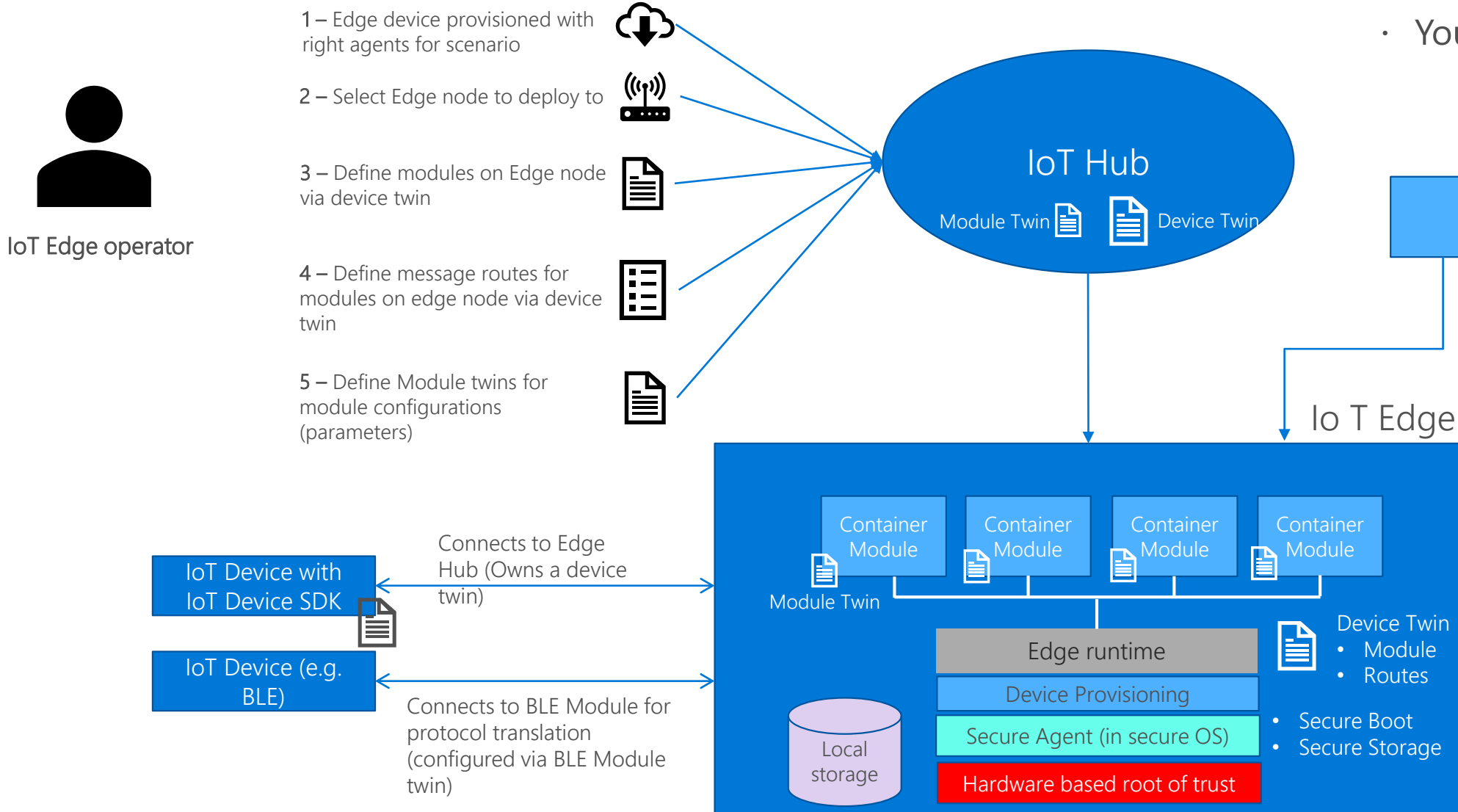
Downloading	Running
Unhealthy	Failed
Stopped	

Restart Policy

Never	On Failure
Unhealthy	Always

IoT Edge in action

- Container based workloads
- Azure Functions
- Azure Stream Analytics
- Azure AI/Machine Learning
- Your own code

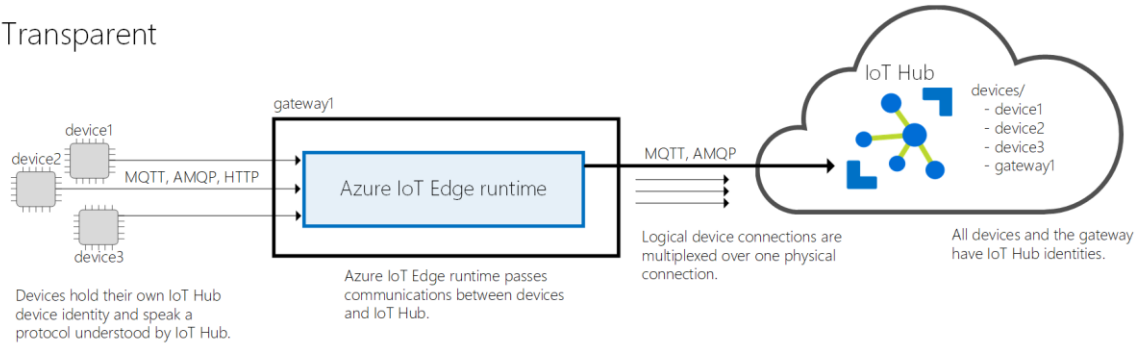


Container Modules

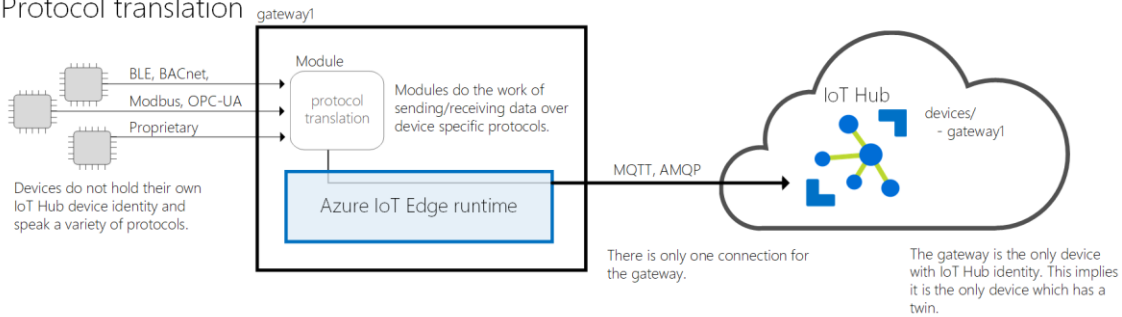
IoT Edge

Azure IoT Edge as a Gateway (Patterns)

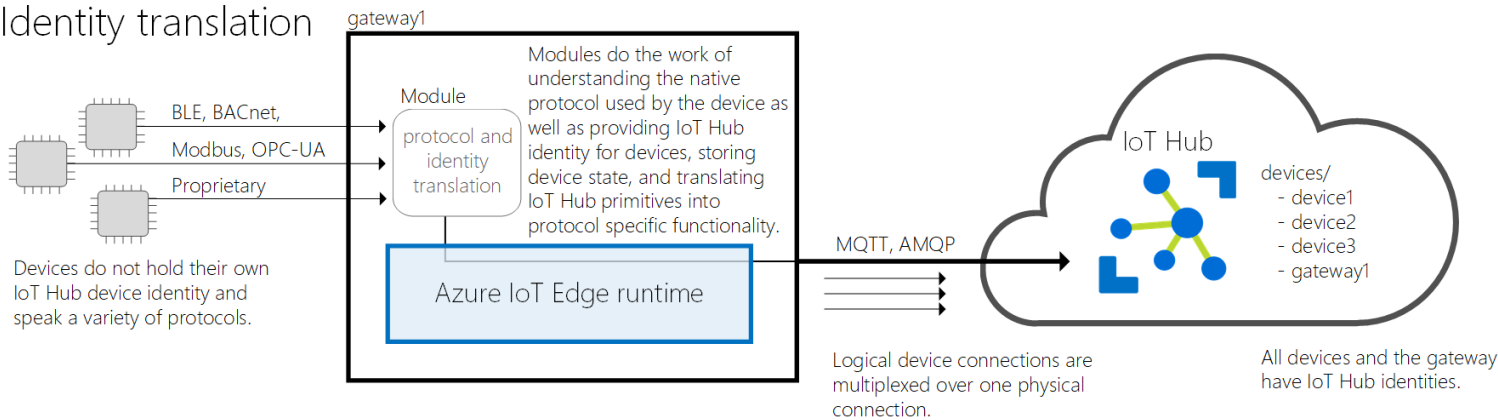
Transparent



Protocol translation



Identity translation



Azure IoT Edge Gateway Use Cases

Edge Analytics

(ML & AI Services)

Downstream Device Isolation

(Shield Downstream Devices from Internet)

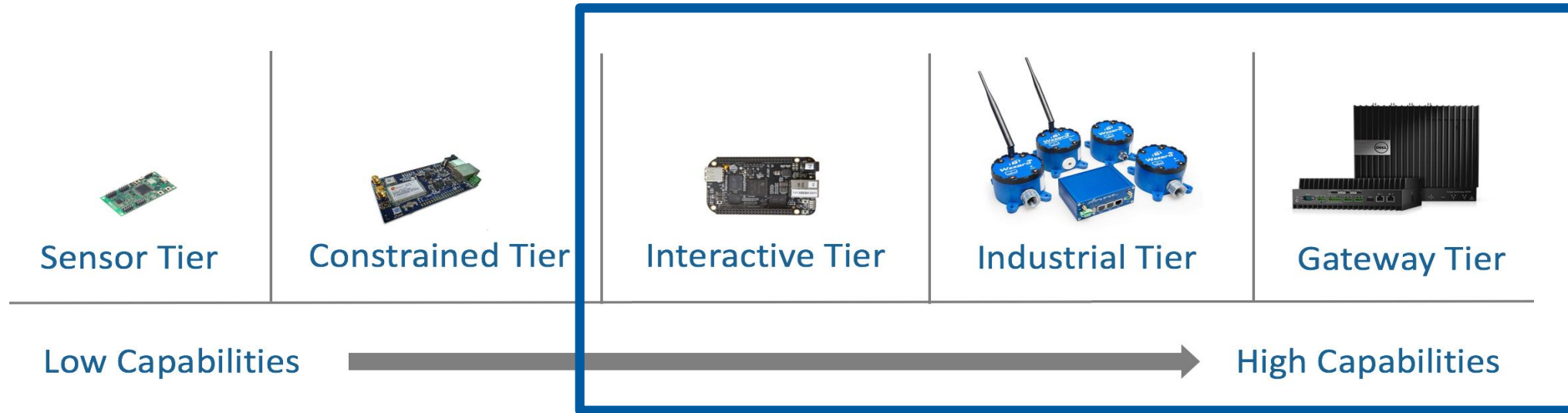
Connection Multiplexing

(All Devices use same underlying connection)

Traffic Smoothing
(Back-off in case of IoT Hub throttling and use local storage)

Limited Offline Support
(Store messages and twins locally when IoT Hub connectivity lost)

Hardware for Azure IoT Edge



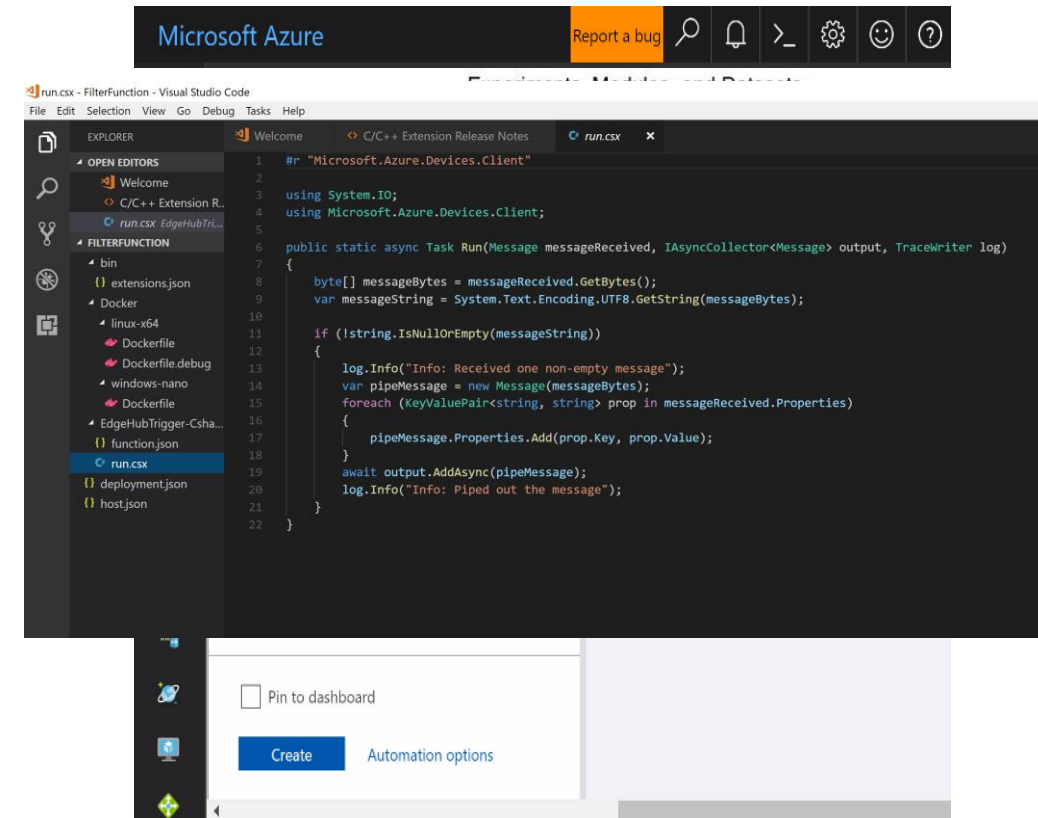
Linux and Windows supported on x64 and ARM (support for containers required)

Hardware sizing dependent on workloads

Internal tests on devices as small as Raspberry Pi 3

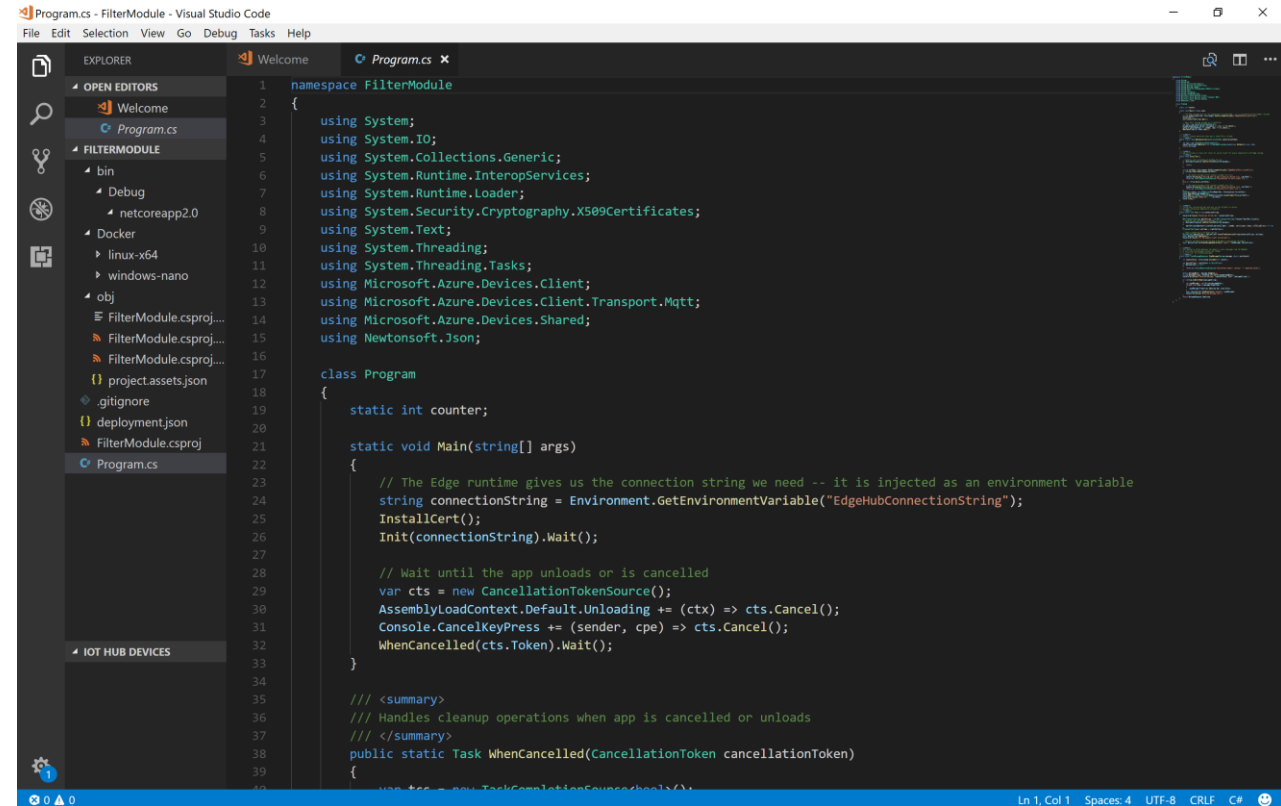
Azure IoT Edge – Package services in containers

- Azure Stream Analytics – In line experience in the ASA web portal
- Azure Functions – Use VSCode to develop Function for your scenario and package as a container
- AI and Azure Machine Learning – Package AI and ML model as a module in a container after training using Azure ML Python SDK, ML Studio, Custom Vision AI. Deploy packaged ML modules to the IoT Edge



Azure IoT Edge – Developer experience

- Azure IoT SDK for developing modules, which provide:
 - Protocol and messaging support
 - Security for module (identification and authentication)
 - Module twin support
- Develop and debug in your favorite language (C# released, C, Python, Java and Node.JS coming soon)
- Build container with your code and host in a container repo (e.g. Docker Hub or Azure Container Registry)

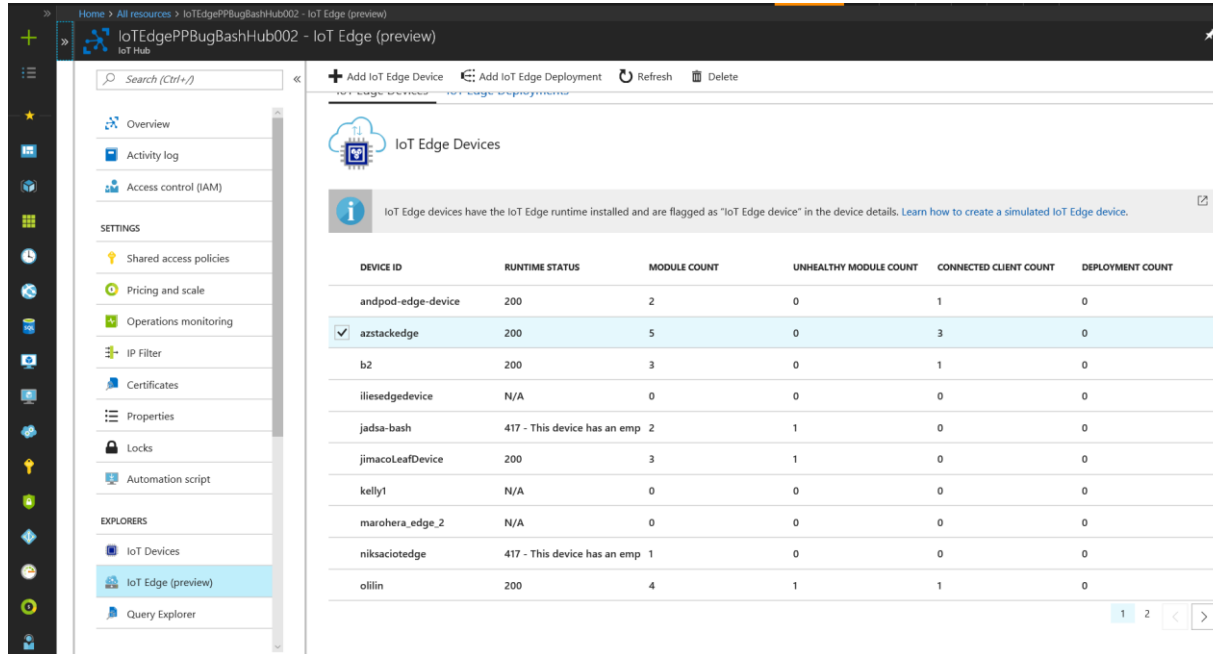


The screenshot shows the Visual Studio Code interface with a C# file named `Program.cs` open. The Explorer sidebar on the left shows a project structure for a module named `FilterModule`, including folders for `bin`, `Debug`, `netcoreapp2.0`, `Docker`, `linux-x64`, `windows-nano`, `obj`, and several `FilterModule.csproj` files. The main editor displays the following C# code:

```
1 namespace FilterModule
2 {
3     using System;
4     using System.IO;
5     using System.Collections.Generic;
6     using System.Runtime.InteropServices;
7     using System.Runtime.Loader;
8     using System.Security.Cryptography.X509Certificates;
9     using System.Text;
10    using System.Threading;
11    using System.Threading.Tasks;
12    using Microsoft.Azure.Devices.Client;
13    using Microsoft.Azure.Devices.Client.Transport.Mqtt;
14    using Microsoft.Azure.Devices.Shared;
15    using Newtonsoft.Json;
16
17    class Program
18    {
19        static int counter;
20
21        static void Main(string[] args)
22        {
23            // The Edge runtime gives us the connection string we need -- it is injected as an environment variable
24            string connectionString = Environment.GetEnvironmentVariable("EdgeHubConnectionString");
25            InstallCert();
26            Init(connectionString).Wait();
27
28            // Wait until the app unloads or is cancelled
29            var cts = new CancellationTokenSource();
30            AssemblyLoadContext.Default.Unloading += (ctx) => cts.Cancel();
31            Console.CancelKeyPress += (sender, cpe) => cts.Cancel();
32            WhenCancelled(cts.Token).Wait();
33        }
34
35        /// <summary>
36        /// Handles cleanup operations when app is cancelled or unloads
37        /// </summary>
38        public static Task WhenCancelled(CancellationToken cancellationToken)
39        {
40            var tcs = new TaskCompletionSource<bool>();
```

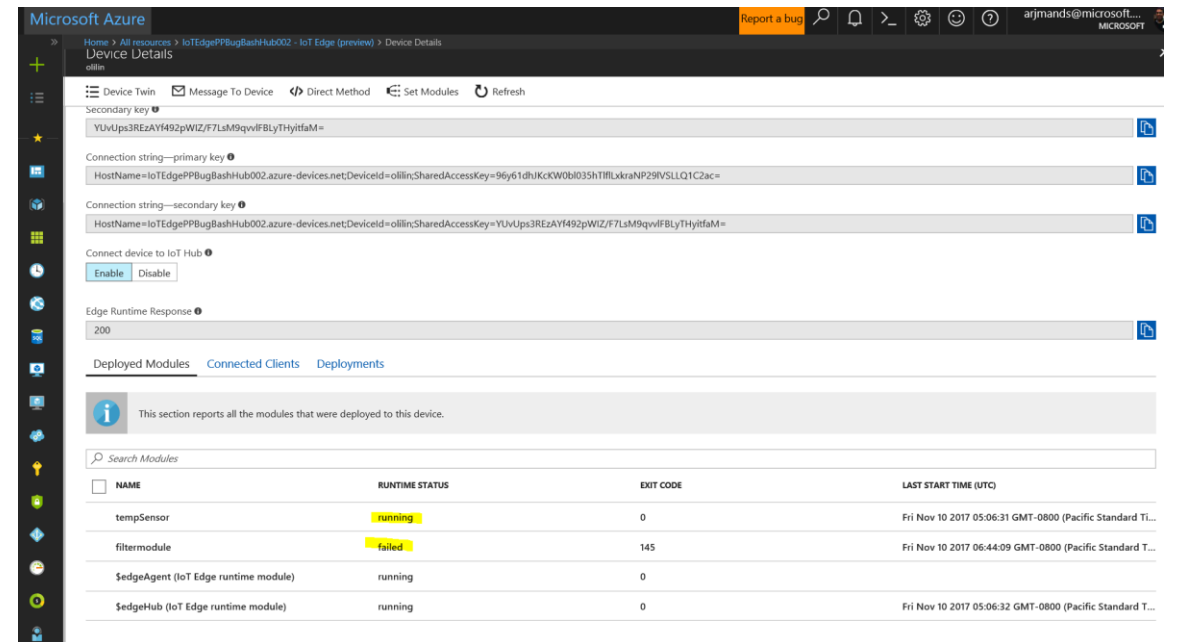
The status bar at the bottom indicates the current position is Line 1, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the C# language.

Azure IoT Edge - Deployment and management



The screenshot shows the Azure IoT Edge (preview) portal. The left sidebar contains navigation options: Overview, Activity log, Access control (IAM), SETTINGS (Shared access policies, Pricing and scale, Operations monitoring, IP Filter, Certificates, Properties, Locks, Automation script), and EXPLORERS (IoT Devices, IoT Edge (preview), Query Explorer). The main content area is titled "IoT Edge Devices" and includes an information box stating: "IoT Edge devices have the IoT Edge runtime installed and are flagged as 'IoT Edge device' in the device details. [Learn how to create a simulated IoT Edge device.](#)" Below this is a table listing IoT Edge devices.

DEVICE ID	RUNTIME STATUS	MODULE COUNT	UNHEALTHY MODULE COUNT	CONNECTED CLIENT COUNT	DEPLOYMENT COUNT
andpod-edge-device	200	2	0	1	0
<input checked="" type="checkbox"/> azstackedge	200	5	0	3	0
b2	200	3	0	1	0
ilisedgedevice	N/A	0	0	0	0
jadsa-bash	417 - This device has an emp	2	1	0	0
jimacoLeafDevice	200	3	1	0	0
kelly1	N/A	0	0	0	0
marohera_edge_2	N/A	0	0	0	0
niksaciotedge	417 - This device has an emp	1	0	0	0
ollin	200	4	1	1	0

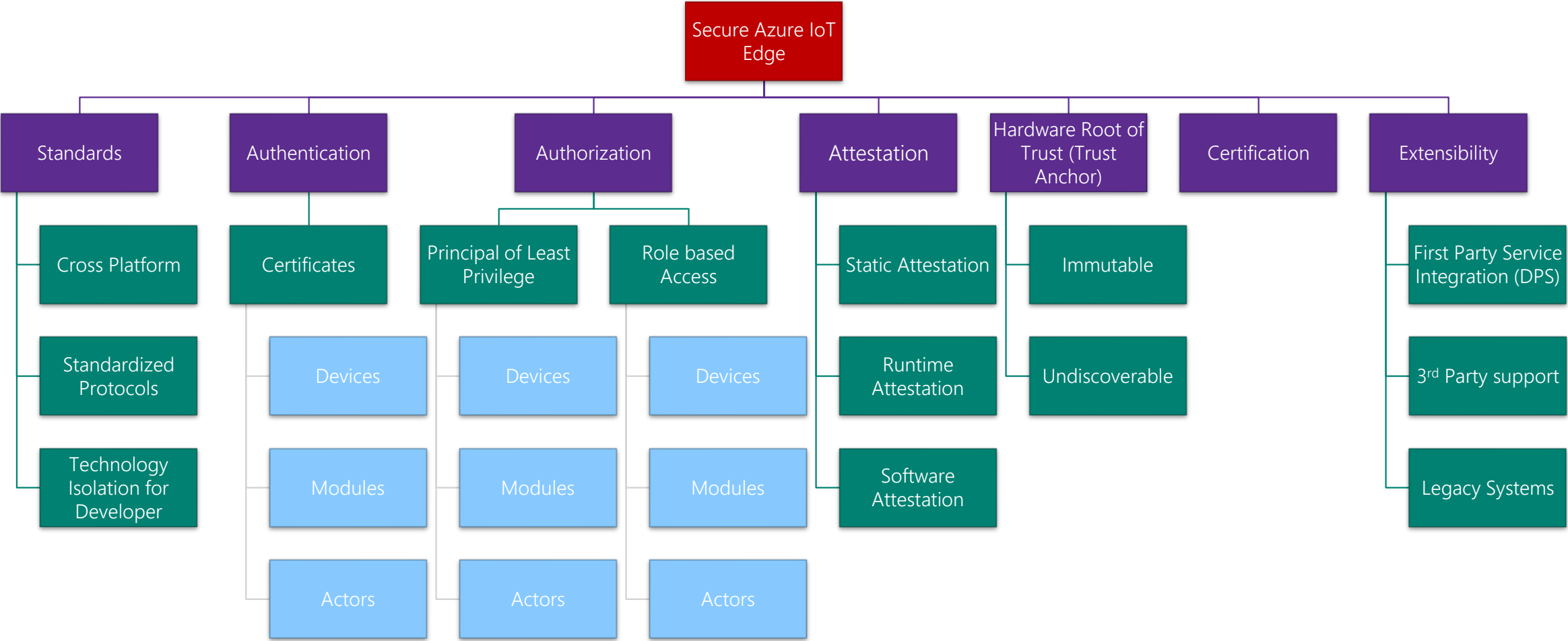


The screenshot shows the "Device Details" page for the "ollin" device. The top navigation bar includes "Device Twin", "Message To Device", "Direct Method", "Set Modules", and "Refresh". The page displays the secondary key, connection strings (primary and secondary), and the edge runtime response (200). Below this, there are tabs for "Deployed Modules", "Connected Clients", and "Deployments". The "Deployed Modules" tab is active, showing a table of modules deployed to the device.

NAME	RUNTIME STATUS	EXIT CODE	LAST START TIME (UTC)
tempSensor	running	0	Fri Nov 10 2017 05:06:31 GMT-0800 (Pacific Standard T...
filtermodule	failed	145	Fri Nov 10 2017 06:44:09 GMT-0800 (Pacific Standard T...
\$edgeAgent (IoT Edge runtime module)	running	0	
\$edgeHub (IoT Edge runtime module)	running	0	Fri Nov 10 2017 05:06:32 GMT-0800 (Pacific Standard T...

Portal demo

Securing Azure IoT Edge



Enable edge intelligence with Azure IoT Edge



Extending the power of the cloud
to the edge

Learn more: aka.ms/azure-iot-edge

Get started: aka.ms/azure-iot-edge-get-started

Demo

Azure IoT Edge

- Deploy *Azure services* to IoT Edge devices
- *Manage* IoT Edge and downstream devices
- Deploy your *own code in language of your choice*
- Do all of this *securely*, in a *scalable fashion* from the Azure IoT Hub

