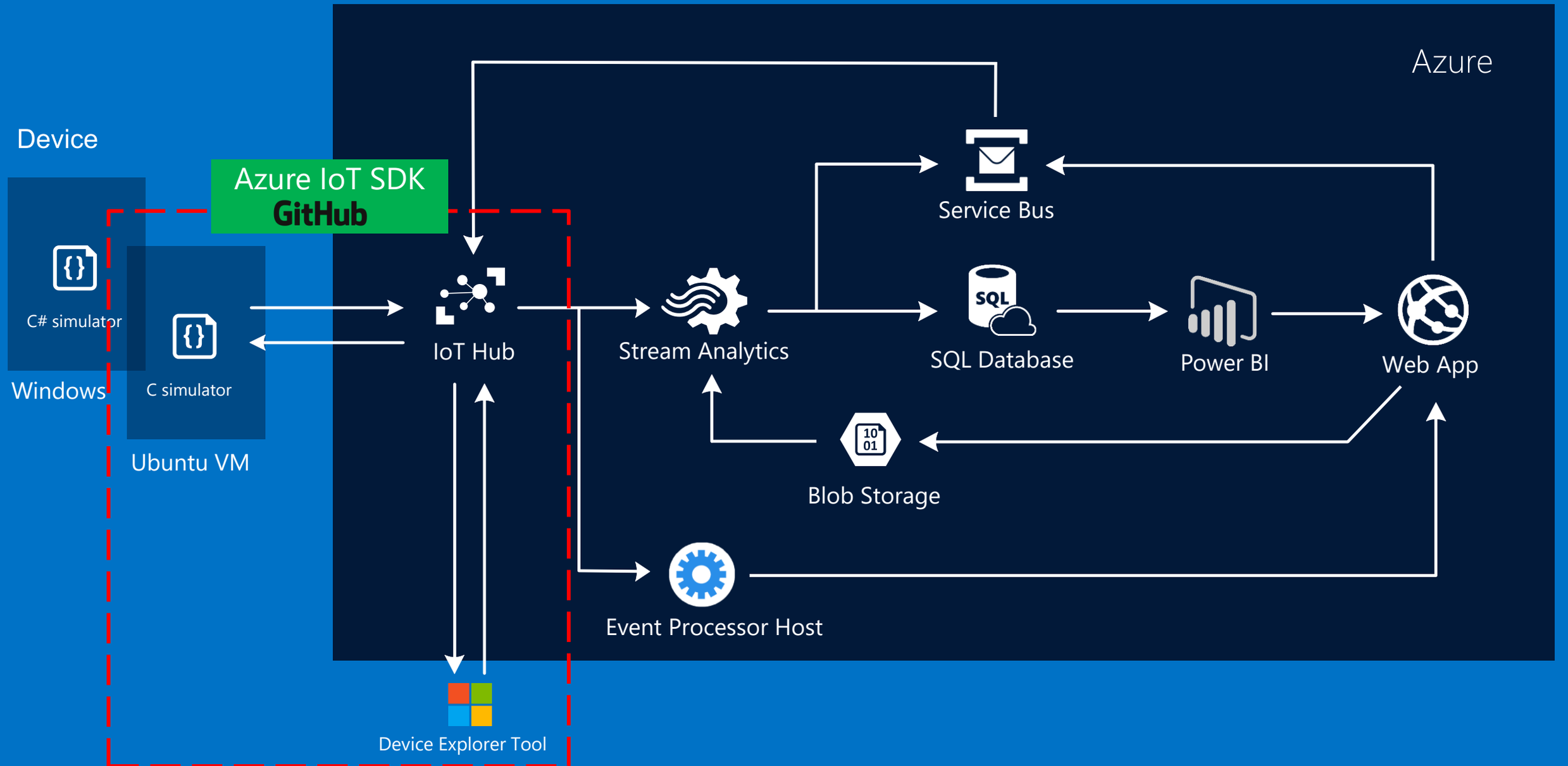
Connecting the Device to Cloud - Part I

Azure IoT Hub and SDK

HOL 2 – Connect the Device to IoT Hub



Agenda

- Considerations for Cloud connectivity
- Introduction to IoT Hub
- Connectivity Patterns
- SDK
- Management
- Development Tools

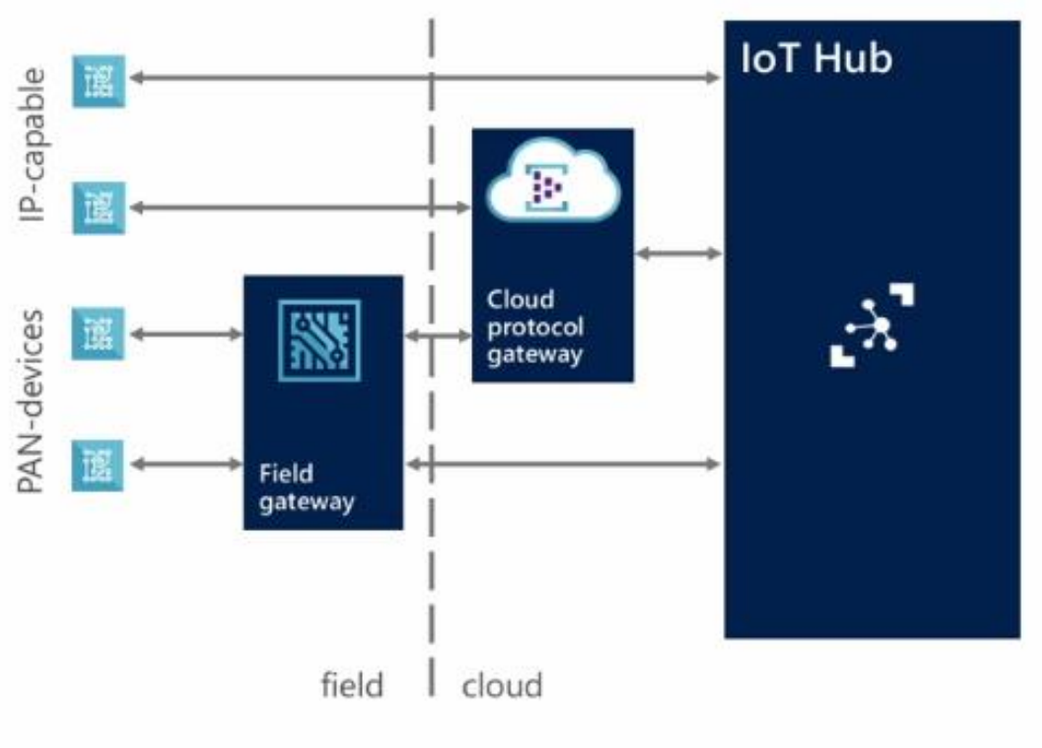
Considerations for Cloud Connectivity

Considerations - Device

Protocols: What protocols are your devices able to communicate with? It needs to be established whether it is supported by the cloud gateway or will require a custom protocol gateway. In addition, what physical networking protocol is it using? Can it support an IP based network?

Platforms: Are your devices compatible with one of the many supported platforms?

Security: Do you require more than server authentication?

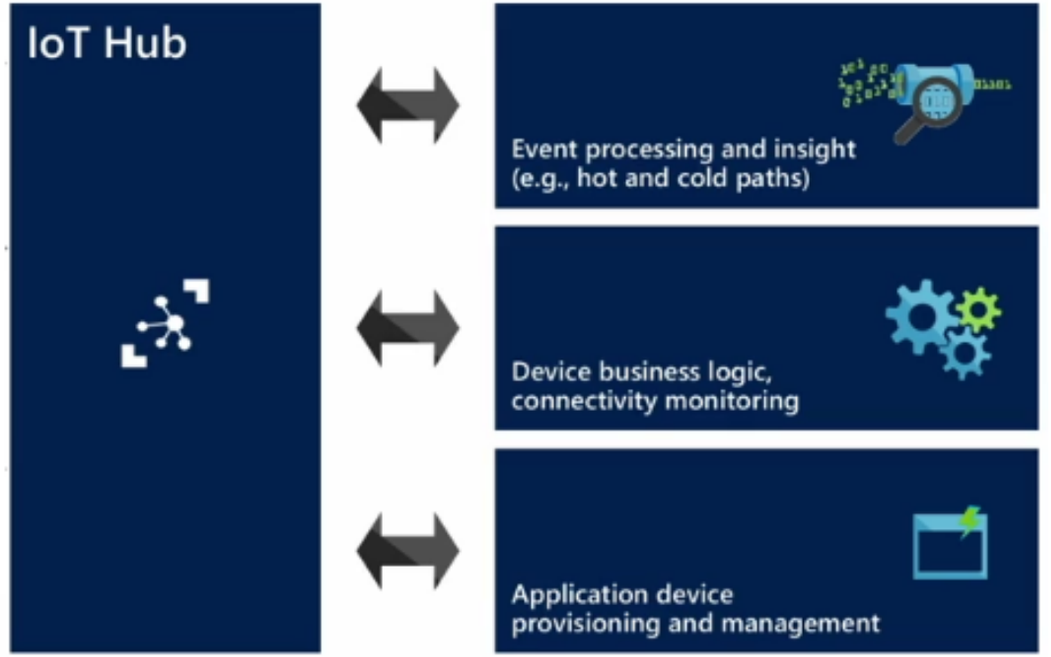


Considerations - Cloud

Telemetry: How will you be monitoring your devices? What outputs do you require for analysing data from the hub?

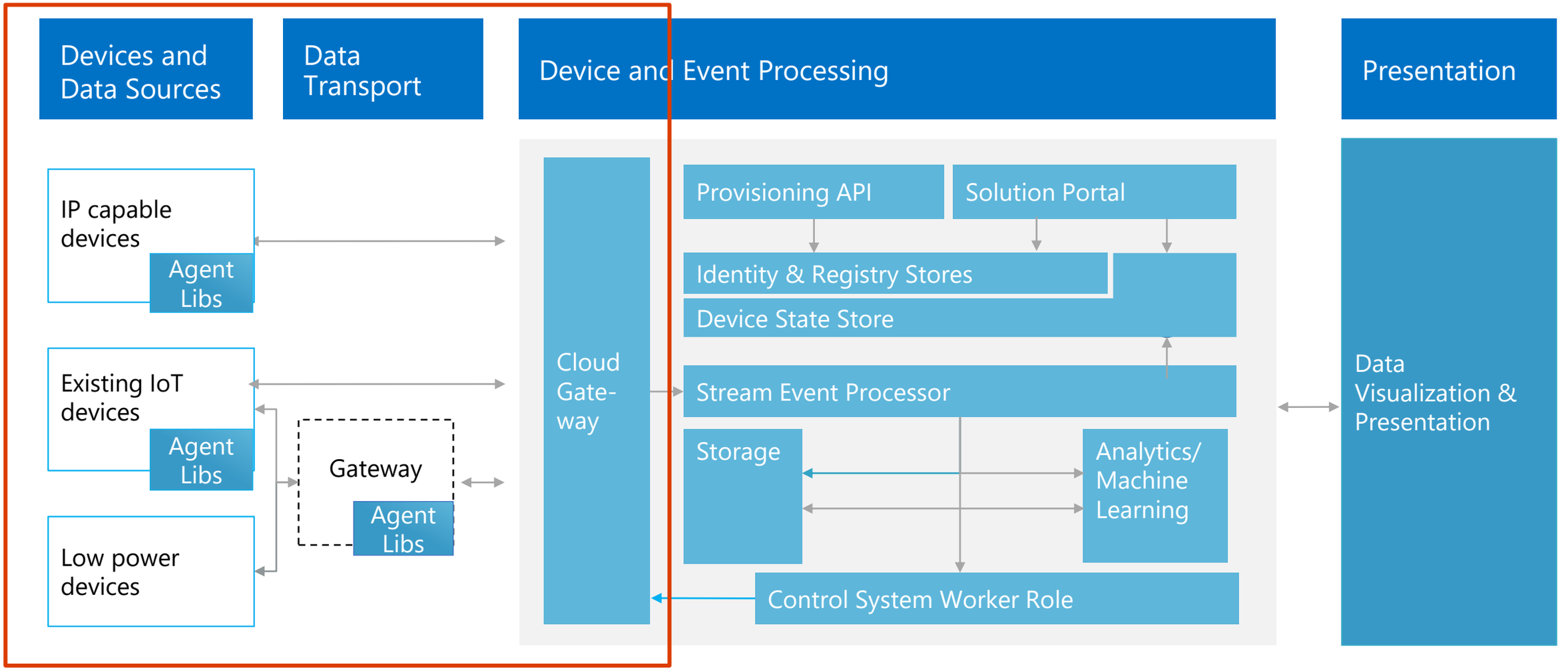
Device Business Logic: How will your devices process commands and software updates?

Management: Devices being provisioned need to be managed correctly by the hub. How will this be maintained and updated?



Introduction to Azure IoT Hub

Solution Architecture



Azure IoT Hub

- Designed for IoT
  - Connect millions of devices
- Security
  - X.509 via AMQPS/HTTPS
  - Use IoT Hub to enable secure bi-directional comms
  - IP Filtering
- Cloud-scale messaging
  - Device-to-cloud and Cloud-to-device
  - Durable messages (at least once semantics)
  - File upload support in portal
- Cloud-facing feedback
  - Delivery receipts, expired messages
  - Device communication errors
- Per-device authentication
  - Individual device identities and credentials
- Connection multiplexing
  - Single device-cloud connection for all communications (C2D, D2C)
- Multi-protocol
  - Natively supports AMQP, HTTPS, MQTT
  - Designed for extensibility to custom protocols
  - AMQP over WebSocket
- Multi-platform
  - Device SDKs available for multiple platforms (e.g. RTOS, Linux, Windows, iOS, Android)
  - Multi-platform Service SDK.
  - .NET/C/C#, Java, Node.js and Python supported.

Connectivity

Connectivity - Protocols

- **AMQP**
  - Uses binary protocol which is significantly more compact than HTTP
  - Supports server push which enables immediate push of messages. This is important if delivery latency is a concern.
  - AMQP could have problems with traversal if the network does not support non-HTTP protocols
  - Should be using AMQP wherever possible
- **HTTP/1**
  - Does not have an efficient way to implement server push, therefore, it is less suitable for field gateway scenarios
  - Polls the IoT Hub for messages which is less inefficient
  - Libraries are much smaller that AMQP, ideal for low resource devices
  - Only use if network or device configuration doesn’t support AMQP

Connectivity - Protocols

- **MQTT**
  - Like AMQP, uses binary protocol which is more compact than HTTP
  - Supports server push which enables immediate push of messages. This is important if delivery latency is a concern.
  - MQTT could have problems with traversal if the network does not support non-HTTP protocols
  - Should be using AMQP if possible; however, MQTT is widely supported by devices and software so may make more sense in some scenarios

Connectivity
Device to Cloud

Device-to-cloud messages

Interface

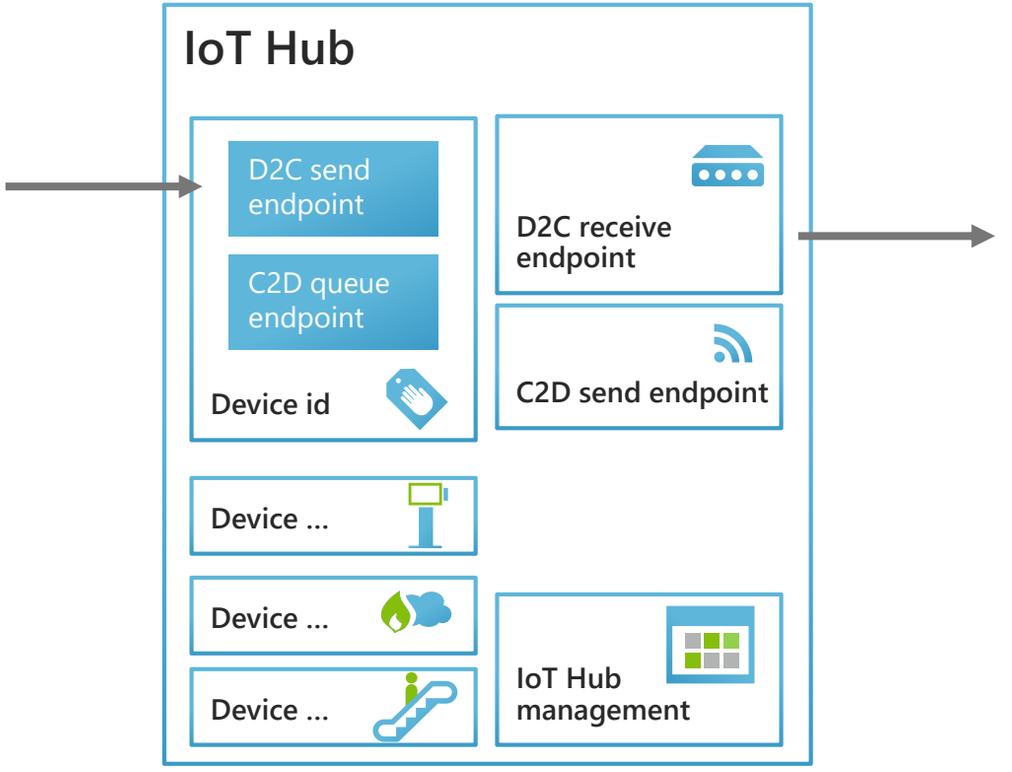
AMQP and HTTPS device-side endpoint
AMQP service-side endpoint
Device and service SDKs

Compatible with Event Hubs

Partitioned receiver, client check-pointing
Integrations with Azure Stream Analytics, Storm, …
100% compatible with Event Hubs receivers

IoT Hub services for D2C

Millions of simultaneously connected devices
Per-device authentication
Connection-multiplexing:
C2D and D2C traffic
Across multiple devices for gateway scenarios



Connectivity
Cloud to Device

Cloud-to-device messages

Interface

AMQP, MQTT and HTTPS device-side endpoint

AMQP service-side endpoint

At-least-once semantics

Durable messages

Device acknowledges receipt
(Send - Receive - Abandon OR Complete)

TTL and receipts

Per-message TTL

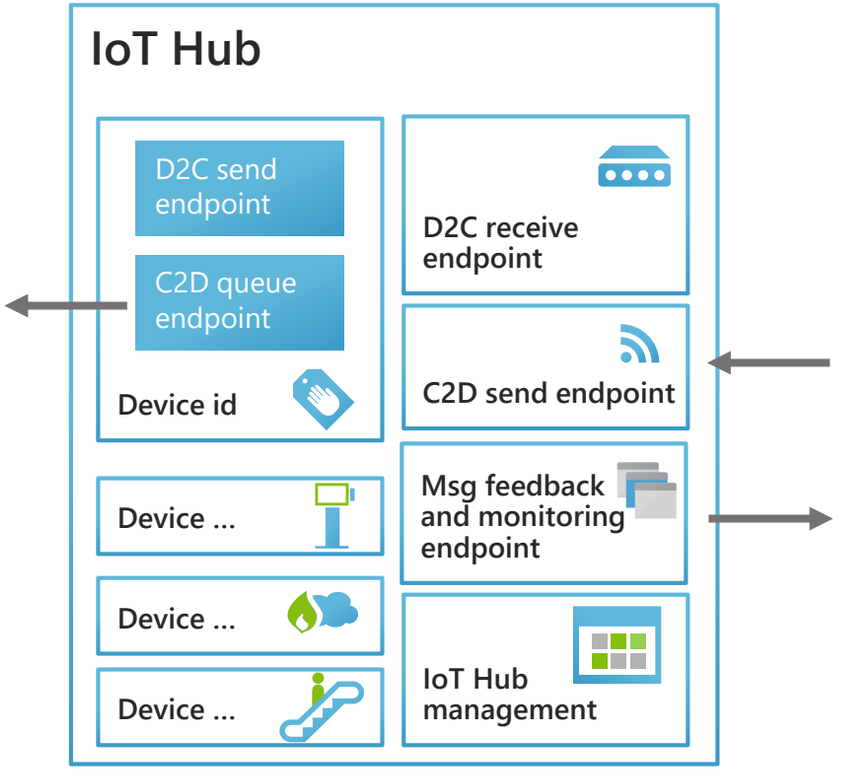
Per-message positive and negative receipts

Command lifecycle pattern

Use correlated D2C for responses

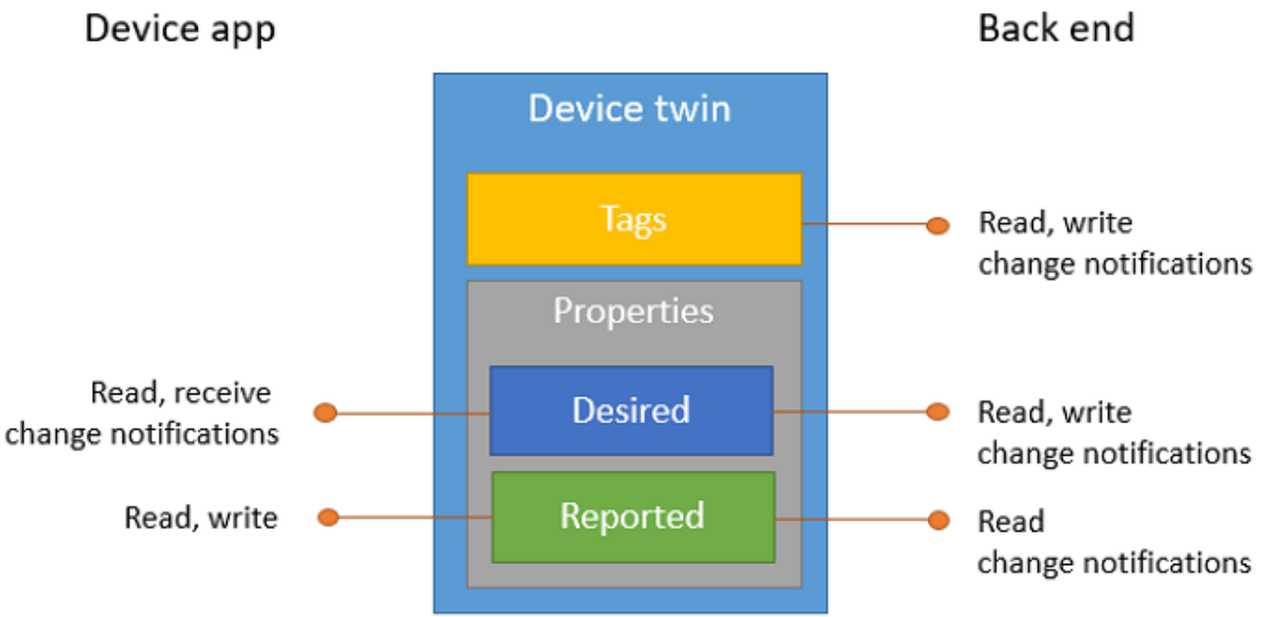
Use feedback information to retry

Store command state in command registry



Device Management

Device Twin




```
{
    "deviceId": "devA",
    "generationId": "123",
    "status": "enabled",
    "statusReason": "provisioned",
    "connectionState": "connected",
    "connectionStateUpdatedTime": "2015-02-28T16:24:48.789Z",
    "lastActivityTime": "2015-02-30T16:24:48.789Z",

    "tags": {
        "$etag": "123",
        "deploymentLocation": {
            "building": "43",
            "floor": "1"
        }
    },
    "properties": {
        "desired": {
            "telemetryConfig": {
                "sendFrequency": "5m"
            },
            "$metadata" : {...},
            "$version": 1
        },
        "reported": {
            "telemetryConfig": {
                "sendFrequency": "5m",
                "status": "success"
            }
            "batteryLevel": 55,
            "$metadata" : {...},
            "$version": 4
        }
    }
}
```

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-device-twins

File Upload

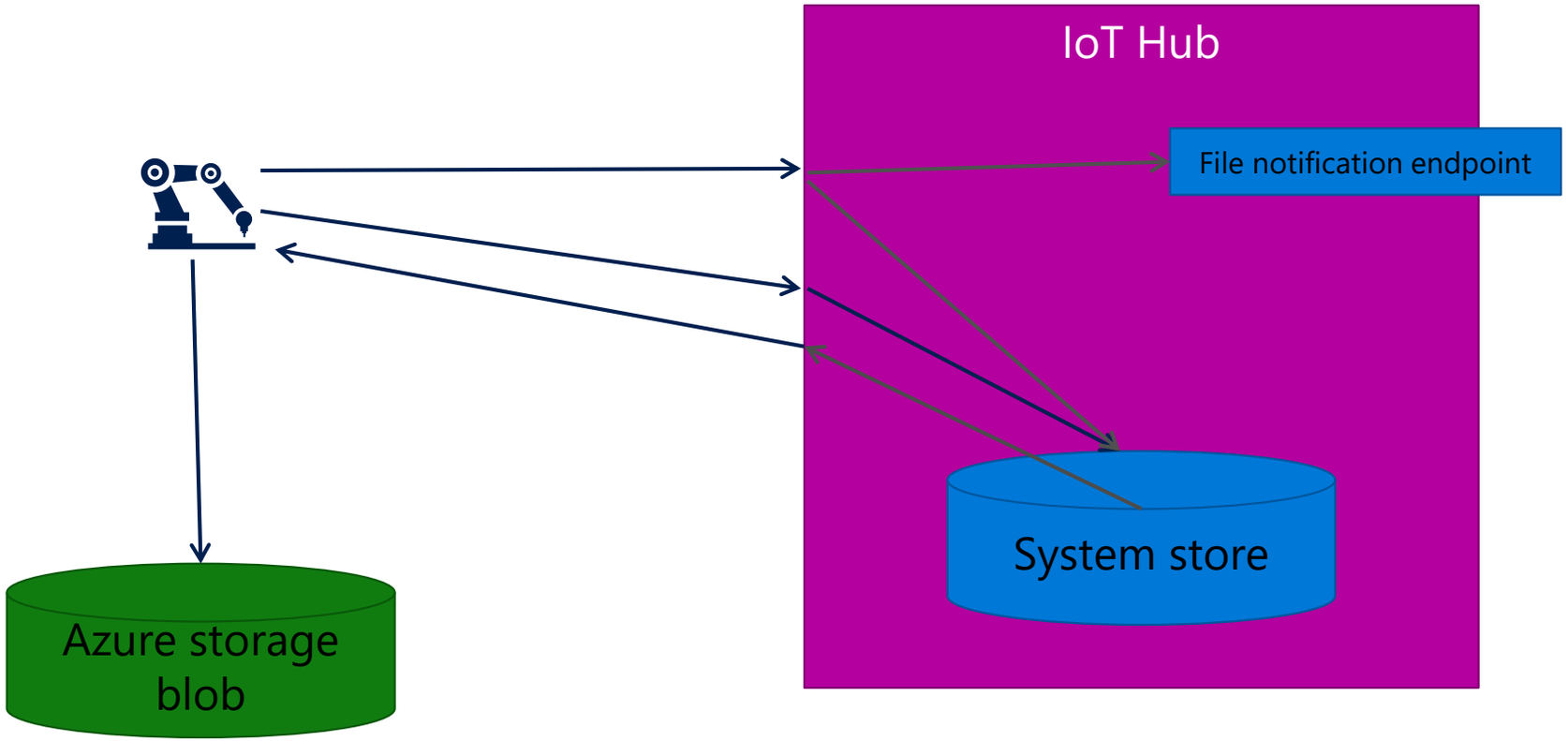
IoT Hub - File Upload into IoT Hub

- Design Goals
  - Devices can upload files of any size, limited only by Azure Storage limits
  - IoT Hub maintains storage account keys – no need to store keys on devices
  - Users can enable file upload notifications on a specific IoT Hub endpoint

IoT Hub File Upload: When to Use & Benefits vs. DIY

- When to Use
  - To upload files >256KB in size
  - To move data directly into storage for cold path processing
  - To interface with an existing data processing pipeline
- Benefits vs. DIY
  - Monitoring via operations monitoring
  - Enhanced security
  - The SDKs take care of talking with storage
  - No need to manage storage keys per device

File Upload: How it Works



SDK

SDK

- SDK and agent libraries easily accessible in GitHub
- Cross platform support – choose OS, platform and language
- Device support with IP and access control capabilities
- Connect IP and non-IP devices via gateway and field protocols
- Open source framework to accommodate development of custom agents for devices
- Simple and secure D2C and C2D connectivity for messaging, device management and command and control
- Multiple OS support - RTOS, Linux, Windows, Android, iOS etc

SDKs - Device

- **Operating Systems Supported**
  - Debian Linux (v 7.5)
  - Fedora Linux (v 20)
  - mbed OS (v 2.0)
  - Raspbian Linux (v 3.18)
  - Ubantu Linux (v 14.04)
  - Windows Desktop (7, 8, 10)
  - Windows IoT Core (v 10)
  - Windows Server (v 2012 R2)
  - Yocto Linux (v 2.1)
  - Android
- **C Libraries Supported**
  - Debian Linux (v 7.5) HTTPS, AMQP, MQTT
  - Fedora Linux (v 20) HTTPS, AMQP, MQTT
  - mbed OS (v 2.0) HTTPS, AMQP
  - Ubuntu Linux (v 14.04) HTTPS, AMQP, MQTT
  - Windows Desktop (7,8,10) HTTPS, AMQP, MQTT
  - Yocto Linux (v 2.1) HTTPS, AMQP

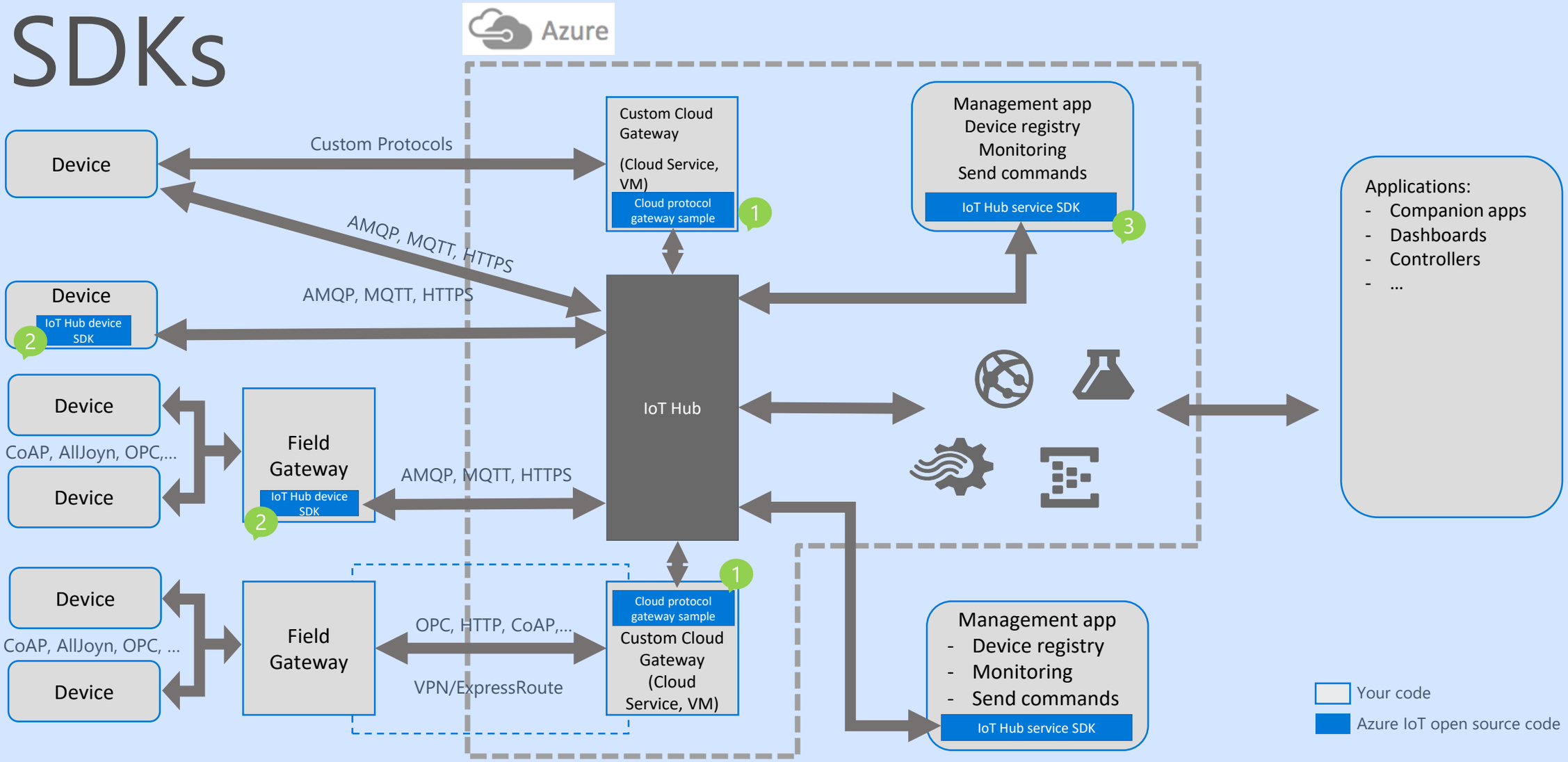
SDKs - Service

- **Node.js library**
  - Node.js (v 4.1.0) *HTTPS*
- **Java library**
  - Java (v 1.7) *HTTPS, AMQP*
  - Java (v 1.8) *HTTPS, AMQP*
- **C# libraries supported**
  - Windows Desktop (7,8,10) *HTTPS, AMQP*
  - Windows IoT Core (10) *HTTPS*
  - Managed agent code requires .NET framework 4.5

GitHub - What's available?

- Azure IoT SDKs
  - https://github.com/Azure/azure-iot-sdks

SDKs

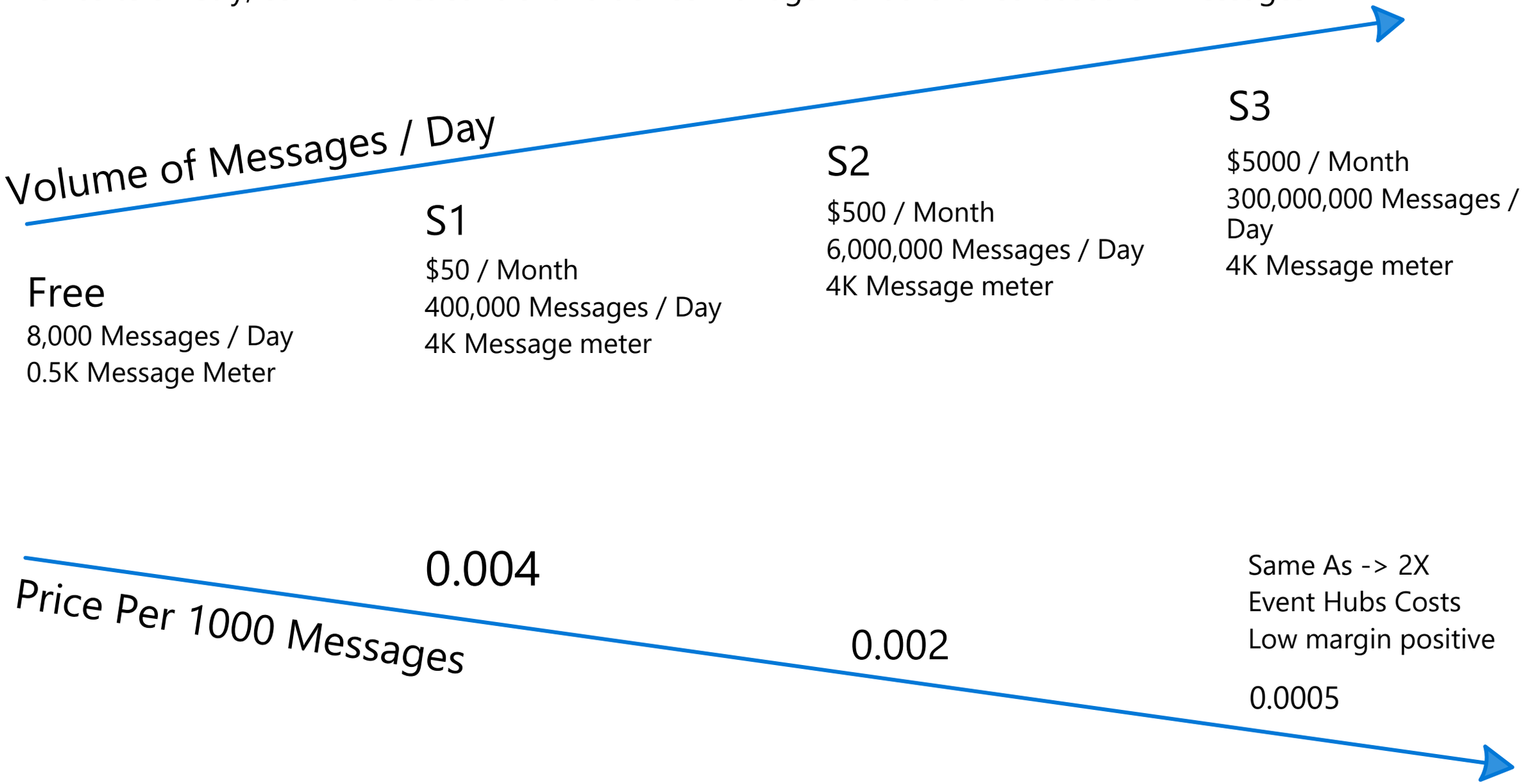


1. https://github.com/Azure/azure-iot-protocol-gateway
2. https://github.com/Azure/azure-iot-sdks - look for device SDKs, https://github.com/Azure/azure-iot-gateway-sdk
3. https://github.com/Azure/azure-iot-sdks - look for service SDKs

Management

Azure IoT Hub SKUs

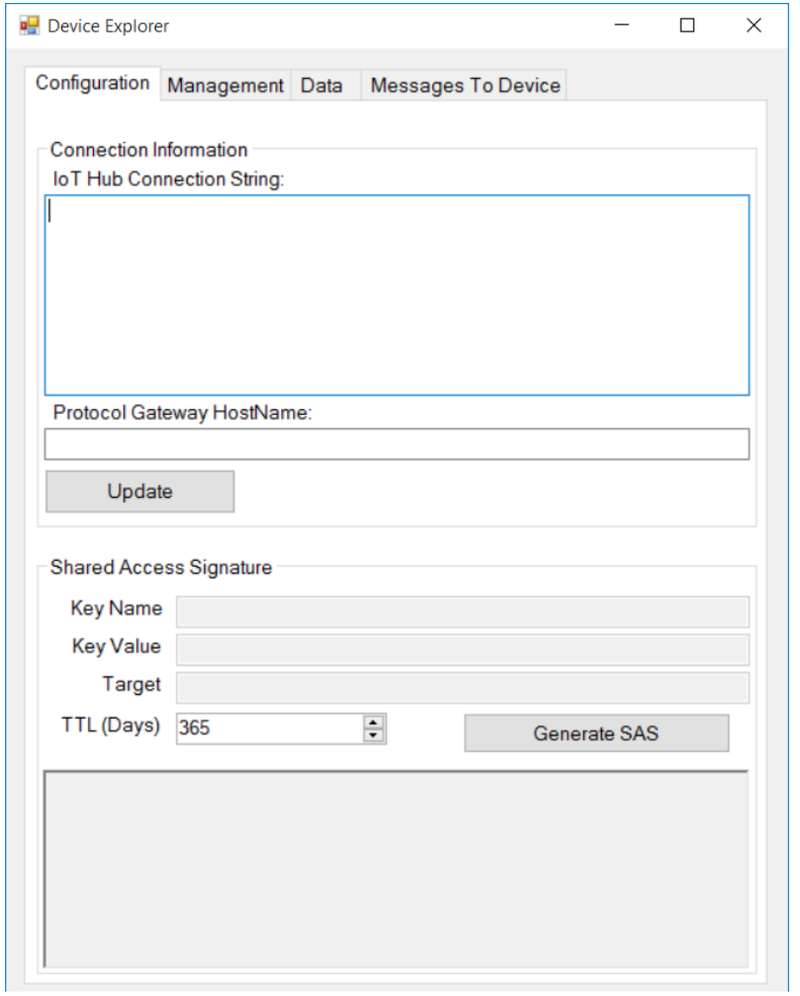
Device telemetry, command & control and device management are billed based on messages



Development

Development - Device Explorer

- Allows users to browse and amend devices in the IoT Hub
- Connection created with connection string and has ability to create SAS keys
- Create/List/Update/Delete devices
- Monitor data from the IoT Hubs event hub so that you can monitor data flowing to the devices
- Ability to send messages to the device and display an output response
- Code is in GitHub; improve and send a pull request…



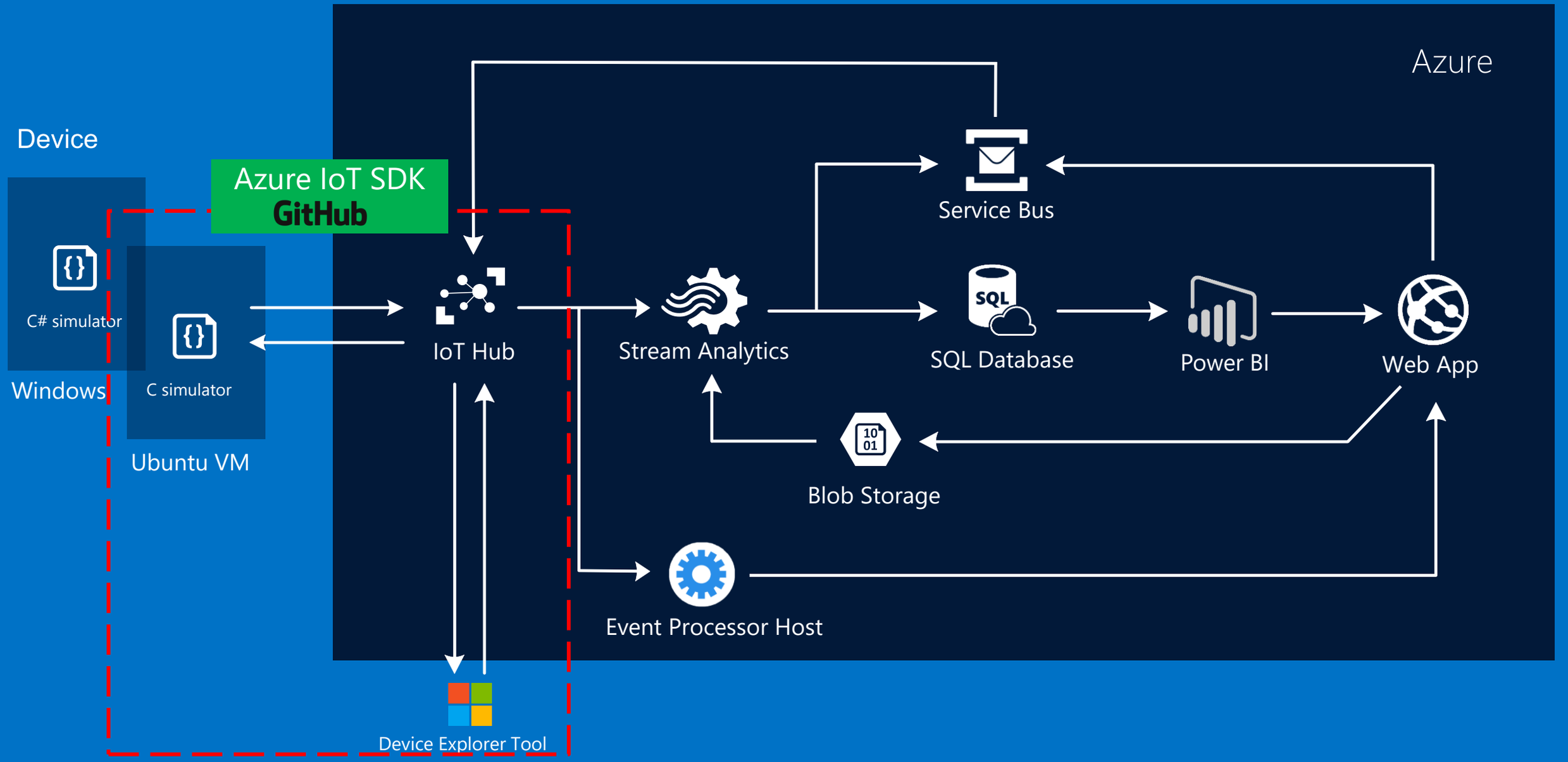
Development - iothub-explorer

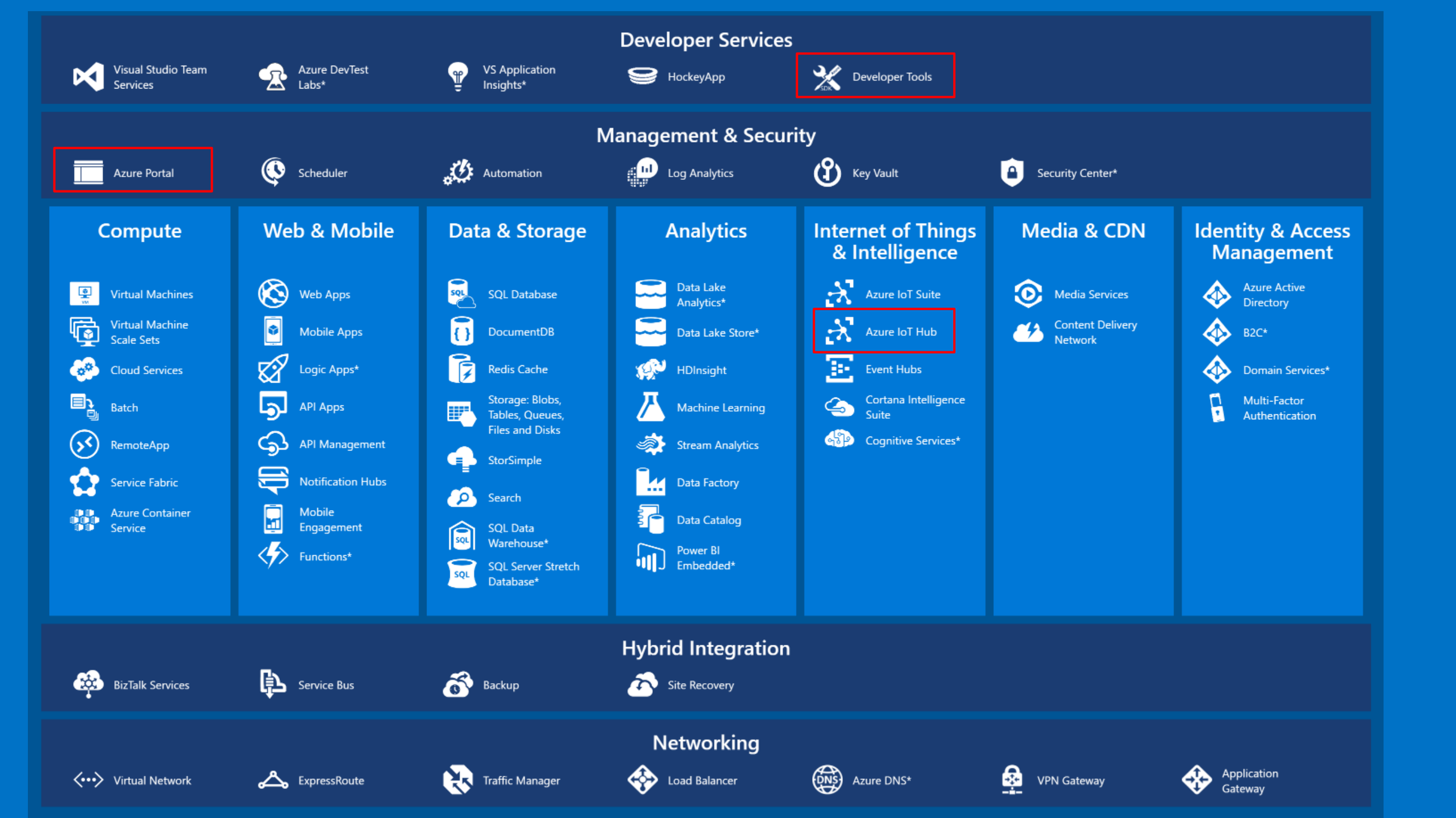
- Allows users to browse and amend devices in the IoT Hub
- Connection created with connection string and has ability to create SAS keys
- Create/List/Update/Delete devices
- Monitor data from the IoT Hubs event hub so that you can monitor data flowing to the devices
- Ability to send messages to the device and display an output response
- Code is in GitHub; improve and send a pull request…

```
$ iothub-explorer 'HostName=<my-hub>.azure-devices.net;SharedAcc

Monitoring events from device myFirstDevice
Listening on endpoint iothub-ehub-<my-endpoint>/ConsumerGroups/$
Listening on endpoint iothub-ehub-<my-endpoint>/ConsumerGroups/$
Event received:
{ deviceId: 'myFirstDevice', windSpeed: 10.92403794825077 }

Event received:
{ deviceId: 'myFirstDevice', windSpeed: 10.671534826979041 }

Event received:
{ deviceId: 'myFirstDevice', windSpeed: 13.557703581638634 }

Event received:
{ deviceId: 'myFirstDevice', windSpeed: 11.123057782649994 }
```

HOL 2 – Connect the Device to IoT Hub

HOL 2 – Connect the Device to IoT Hub




Developer Services
Visual Studio Team Services
Azure DevTest Labs*
VS Application Insights*
HockeyApp
Developer Tools
Management & Security
Azure Portal
Scheduler
Automation
Log Analytics
Key Vault
Security Center*
Compute
Virtual Machines
Virtual Machine Scale Sets
Cloud Services
Batch
RemoteApp
Service Fabric
Azure Container Service
Web & Mobile
Web Apps
Mobile Apps
Logic Apps*
API Apps
API Management
Notification Hubs
Mobile Engagement
Functions*
Data & Storage
SQL Database
DocumentDB
Redis Cache
Storage: Blobs, Tables, Queues, Files and Disks
StorSimple
Search
SQL Data Warehouse*
SQL Server Stretch Database*
Analytics
Data Lake Analytics*
Data Lake Store*
HDInsight
Machine Learning
Stream Analytics
Data Factory
Data Catalog
Power BI Embedded*
Internet of Things & Intelligence
Azure IoT Suite
Azure IoT Hub
Event Hubs
Cortana Intelligence Suite
Cognitive Services*
Media & CDN
Media Services
Content Delivery Network
Identity & Access Management
Azure Active Directory
B2C*
Domain Services*
Multi-Factor Authentication
Hybrid Integration
BizTalk Services
Service Bus
Backup
Site Recovery
Networking
Virtual Network
ExpressRoute
Traffic Manager
Load Balancer
Azure DNS*
VPN Gateway
Application Gateway

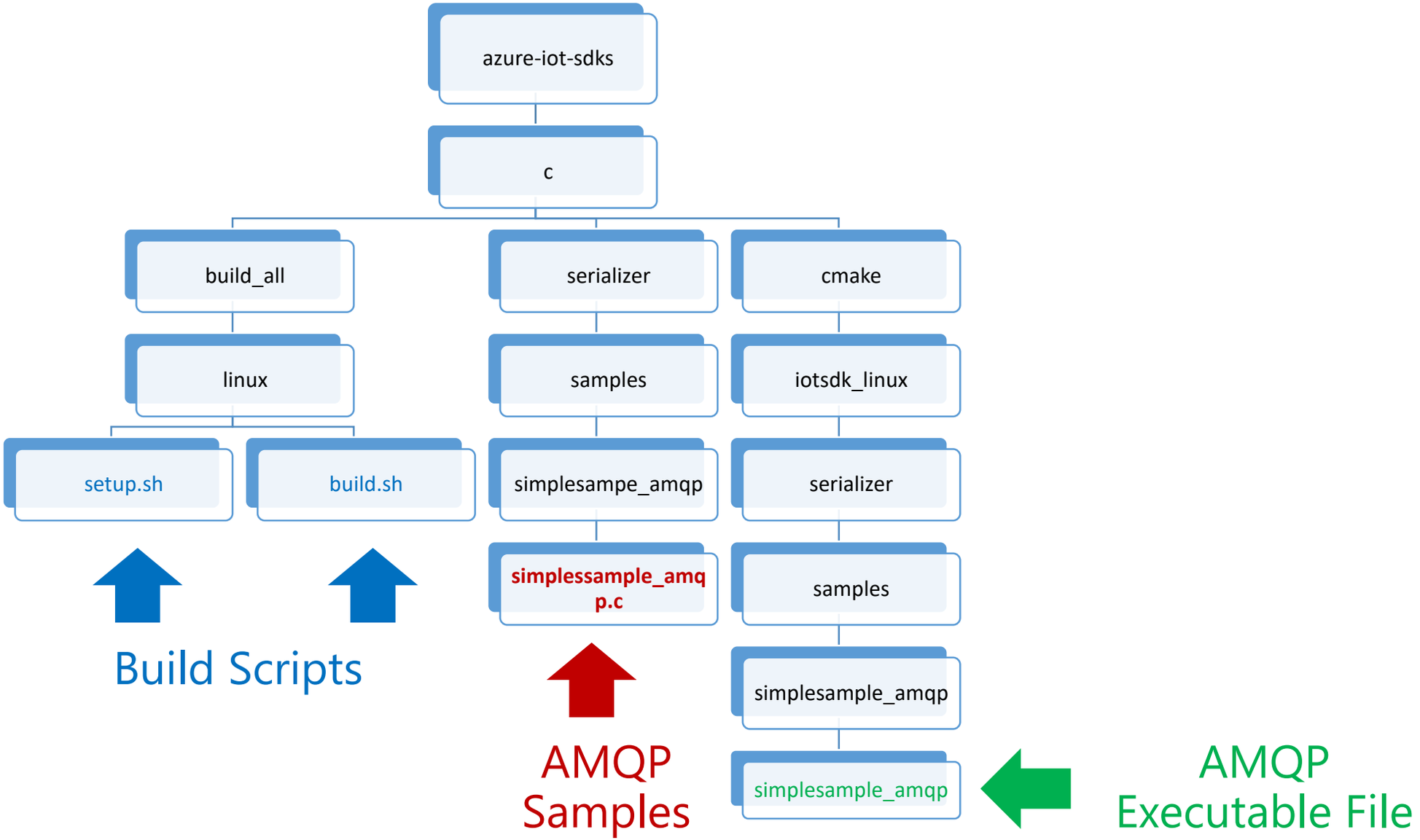
Let's Go

- Device Connecting to **IoT Hub** through Azure **IoT Device SDK**.

  (Please refer the "02-HOL-Azure IoT Hub and SDK" file)
  - Provision a new **IoT Hub** in Azure Portal
  - Setup the **build environment** of Linux
    - Setup for IoT Device SDK
    - Build the SDK and AMQP sample code
    - Execute the AMQP C sample code
  - Test the data communication between Device and Cloud in the **Device Explorer Tool**
  - Introduction to **Azure Certified for IoT Program**

BACKUP

Source Tree of Azure IoT C SDK



Setup.sh

deps=

"curl build-essential libcurl4-openssl-dev git cmake libssl-dev uuid-dev valgrind“

repo=

"https://github.com/Azure/azure-iot-sdks.git"

Build.sh

build_folder= $build_root"/cmake/iotsdk_linux“

....

Usage ()

```
{
    …
    echo " -cl, --compileoption <value>  specify a compile option to be passed to gcc"
    echo " --run-e2e-tests               run the end-to-end tests (e2e tests are skipped by default)"
    echo " --skip-unittests              skip the running of unit tests (unit tests are run by default)"
    echo " --no-amqp                     do no build AMQP transport and samples"
    echo " --no-http                   do no build HTTP transport and samples"
    echo " --no-mqtt                    do no build MQTT transport and samples"
    echo " --no-make                    do not run make after cmake"
    echo " --use-websockets               Enables the support for AMQP over WebSockets."
    echo " --toolchain-file <file>       pass cmake a toolchain file for cross compiling“
    …..
}
```