Azure Digital Twins - how to deal with more complex IoT ecosystem - from device to analytics

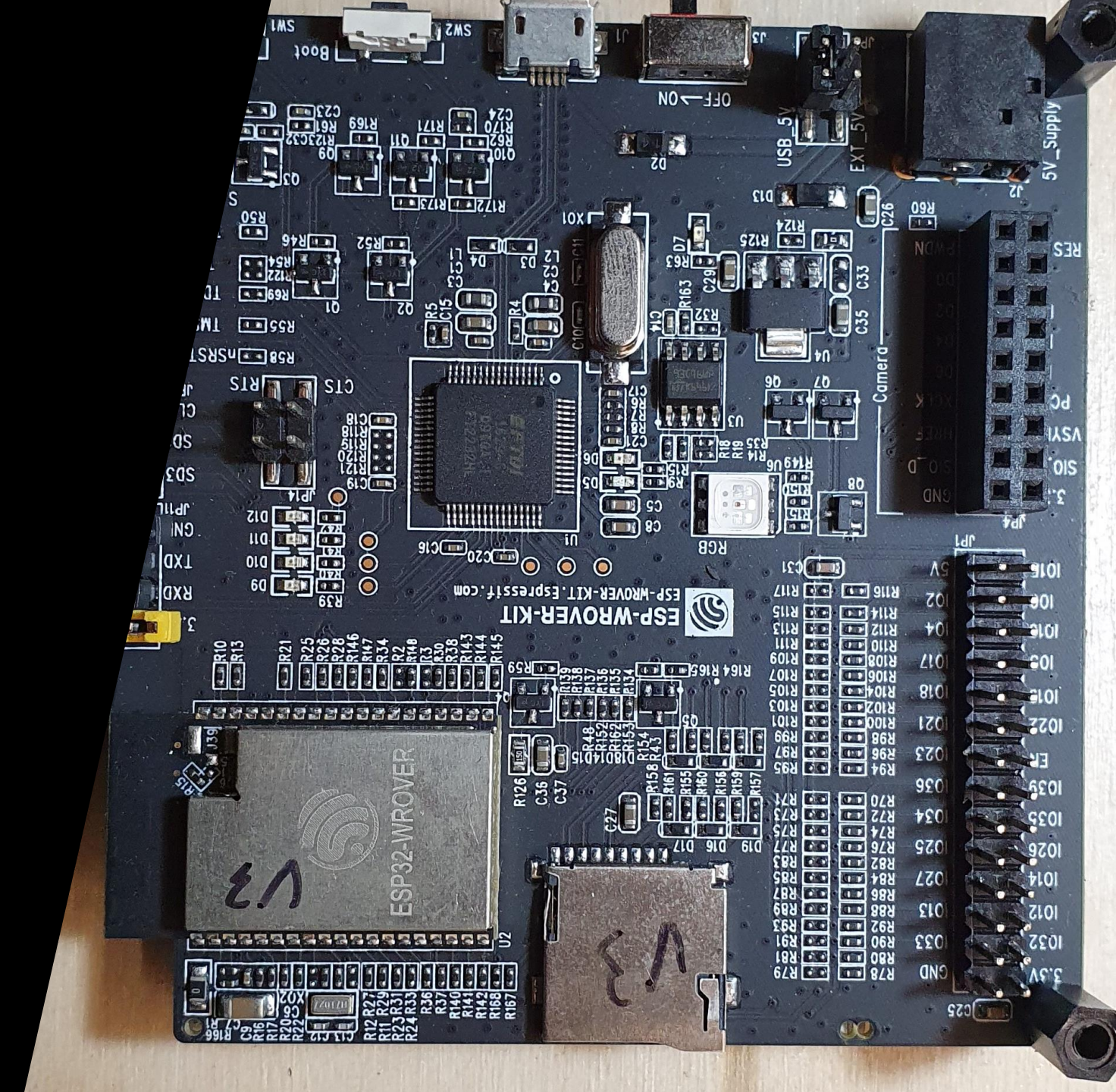
2.5 h MAX!!!!!!!

Tomasz Kopacz,
tkopacz@microsoft.com

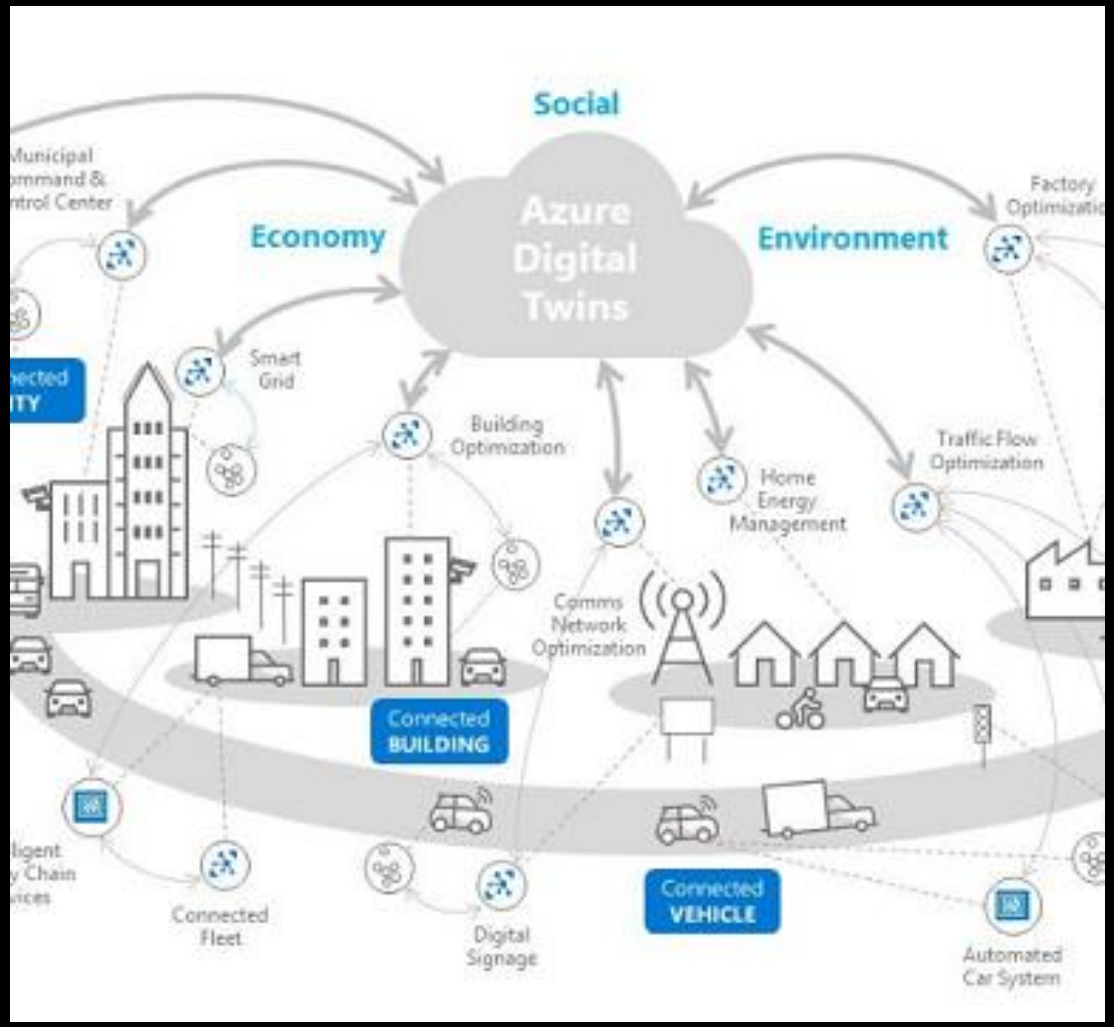
Plan

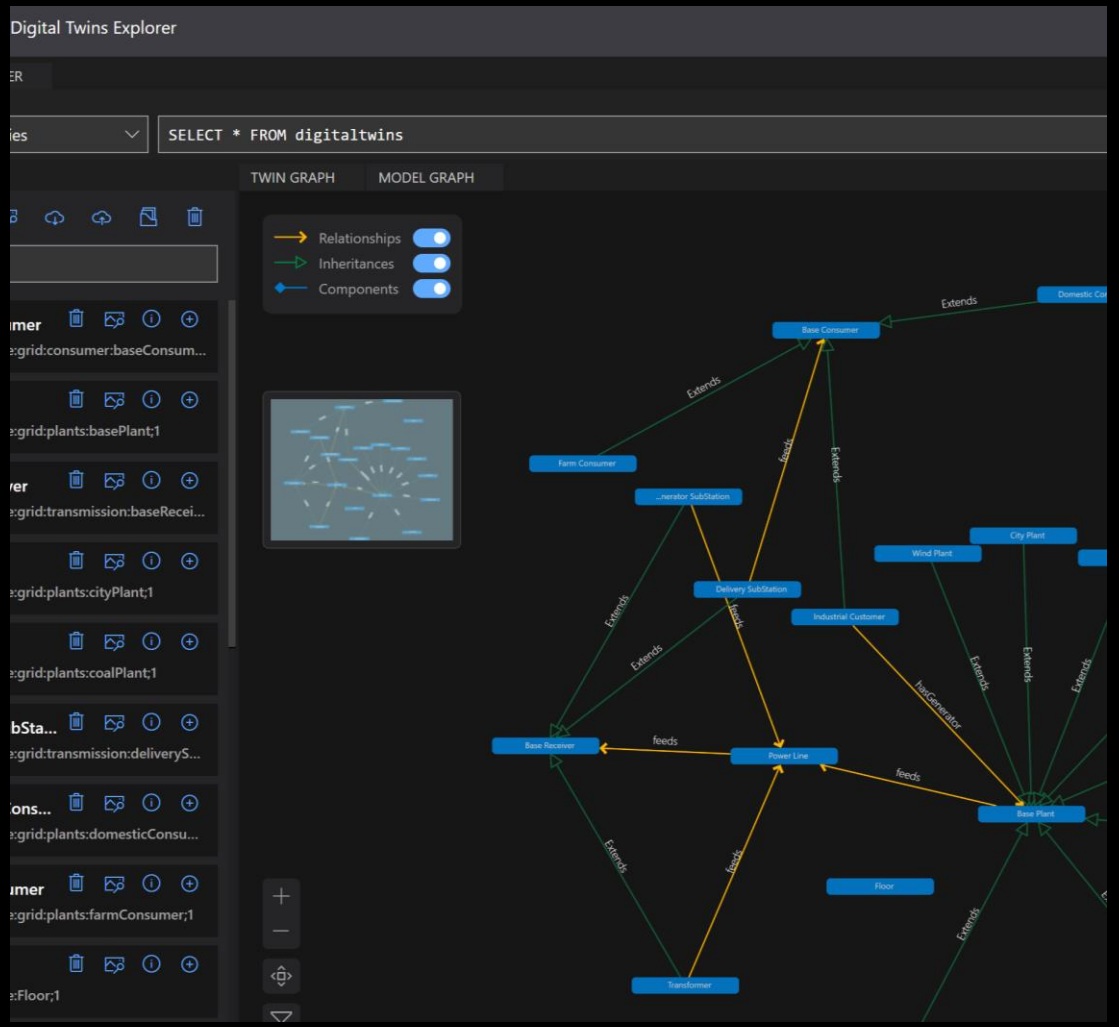
- IoT Plug and Play
- Concept of Digital Twins
- Implementation in Azure Digital Twins
- Usage of Azure Digital Twins and Azure Services
- And – how to work with DATA and IoT and ADT

Context – not only “small devices”

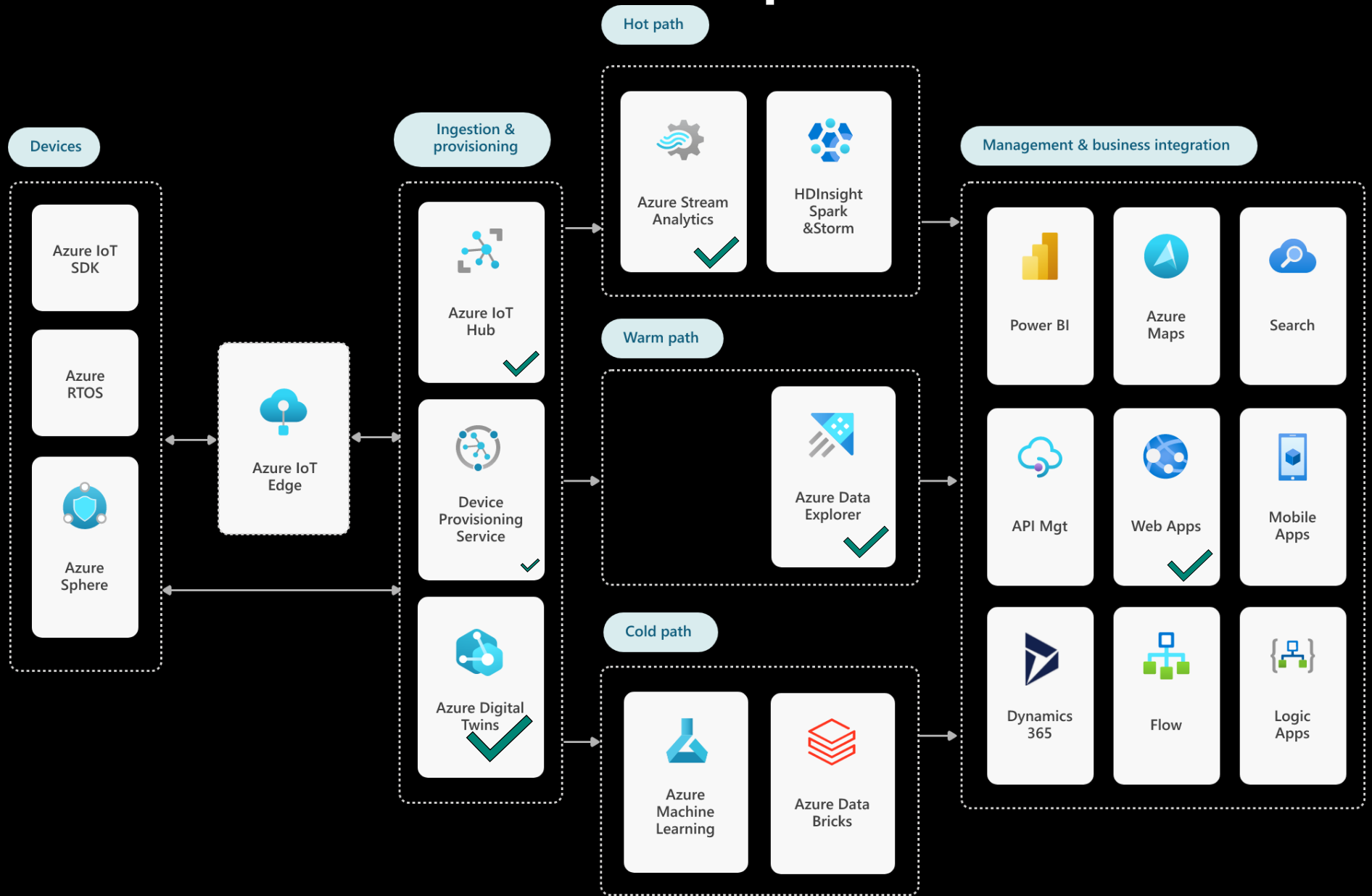


Rather:


Social
Economy
Azure Digital Twins
Environment
Smart Grid
Building Optimization
Home Energy Management
Traffic Flow Optimization
Comms Network Optimization
Connected BUILDING
Connected VEHICLE
Connected Fleet
Digital Signage
Automated Car System


Digital Twins Explorer
SELECT * FROM digitaltwins
TWIN GRAPH
MODEL GRAPH
Relationships
Inheritances
Components
Base Consumer
Farm Consumer
Delivery SubStation
Industrial Customer
Wind Plant
City Plant
Base Receiver
Power Line
Base Plant
Transformer
Floor
Extends
feeds

General, Microsoft IoT Components

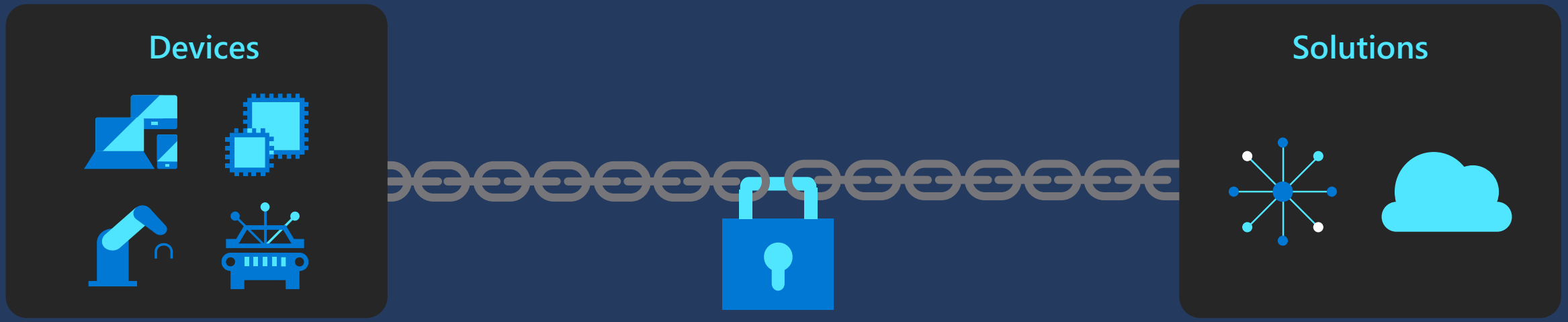


IoT PnP – to define standards for devices

```
{
    "Temperature": 23
}
or ?
{
    "Temp": 23
}
or ?
{
    "temperaturka": 23
}
or ?
{
    "data":
    [
        {
        "timestamp": "2021-01-01 01:01:01.123",
        "typeOfMeasurement": "Temperature",
        "value": 23
        }
    ]
}
```

IoT Today

Tight coupling between software on device and IoT solution in the cloud



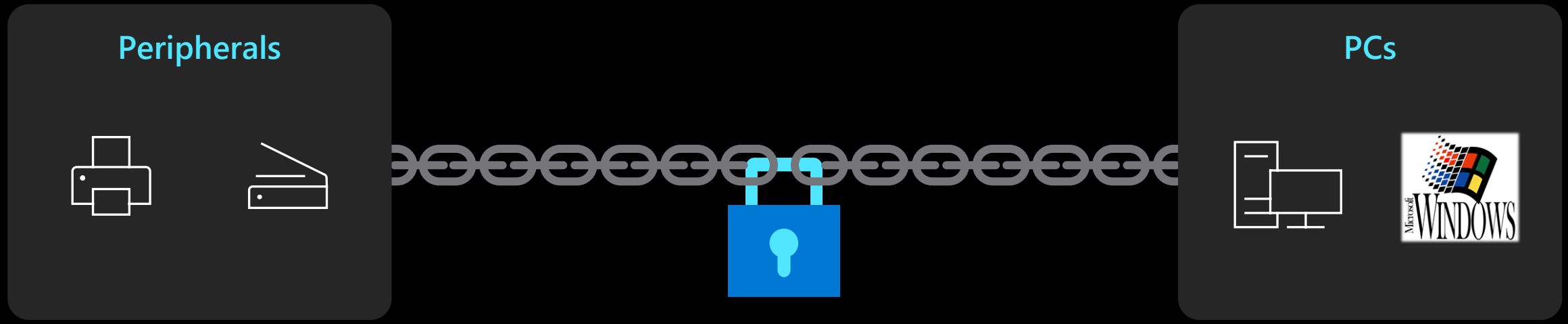
Because of: telemetry format, how to interpret data, how to send commands etc.

Demo

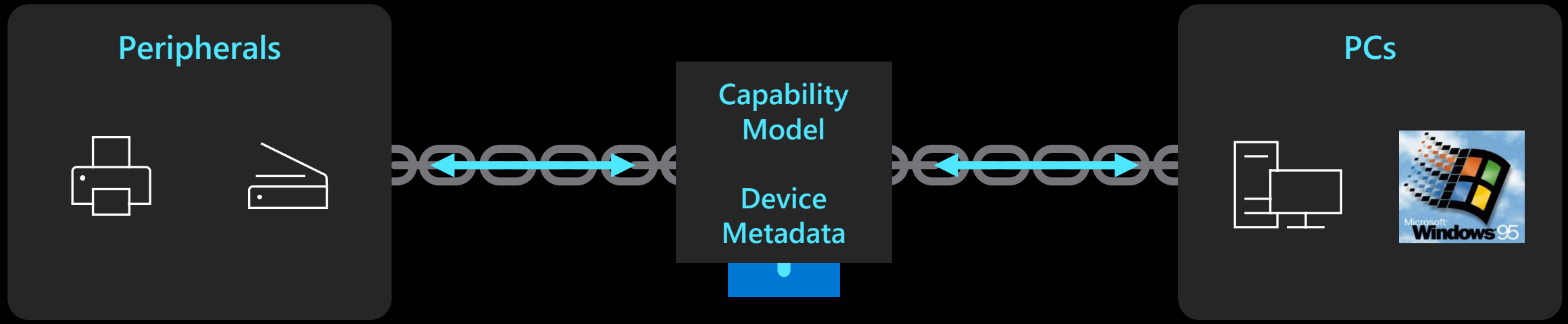
Challenges with “model-less” devices

(and – recap – how to “debug” telemetry!)

We had a similar challenge in the past



That was solved in Windows with Plug and Play

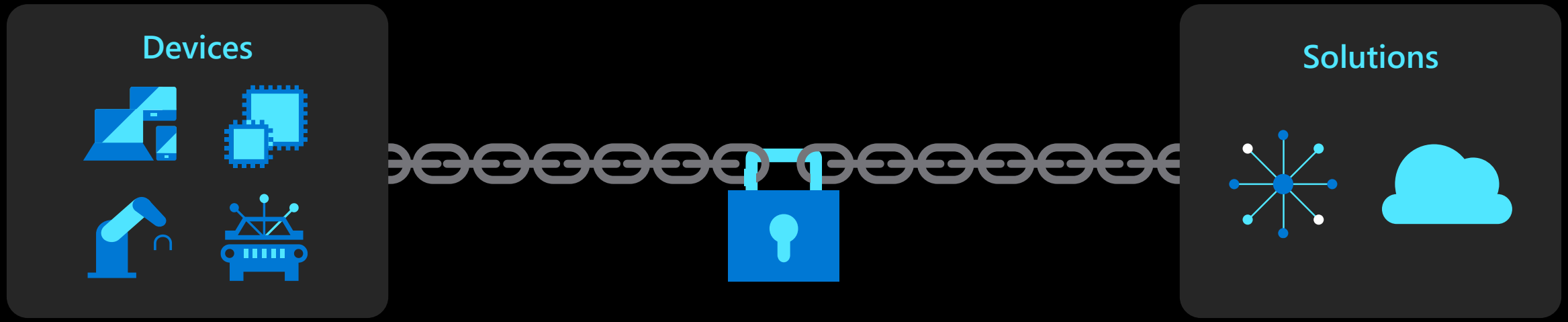


Devices published their **capability models** and adhered to them
Windows used the capability model to know how to **interact** with them

https://web.archive.org/web/20040616023120/http://www.microsoft.com/whdc/system/pnppwr/pnp/default.mspx
1999 - local docs

Introducing IoT Plug and Play

Simplifies device interactions in IoT solutions with an open modeling language



Devices self-describe capabilities based on open Digital Twins Definition Language.

Solutions can *automatically* adapt to devices

All **without custom code**

Describe interaction model in DTDL Open modeling language

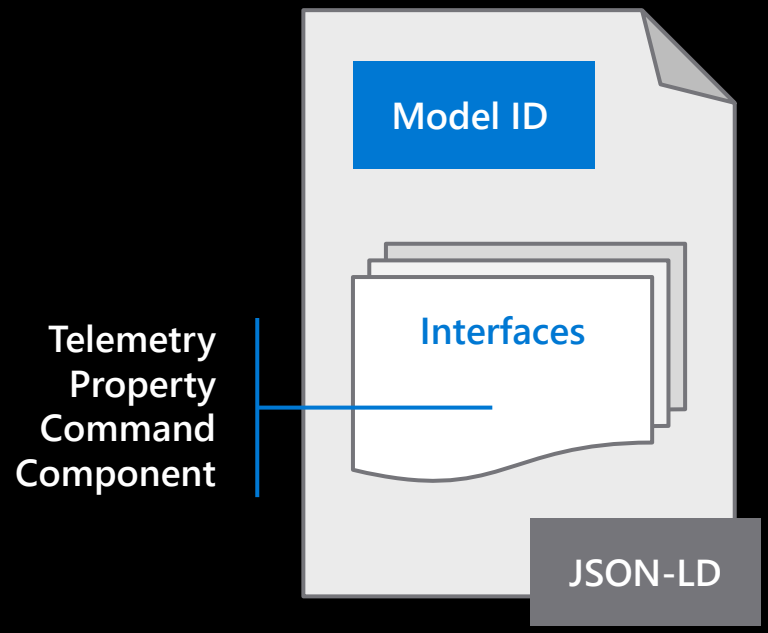
Describe device capability and interaction model

Each device model has a unique ID: **Digital Twin Model ID (DTMI).**

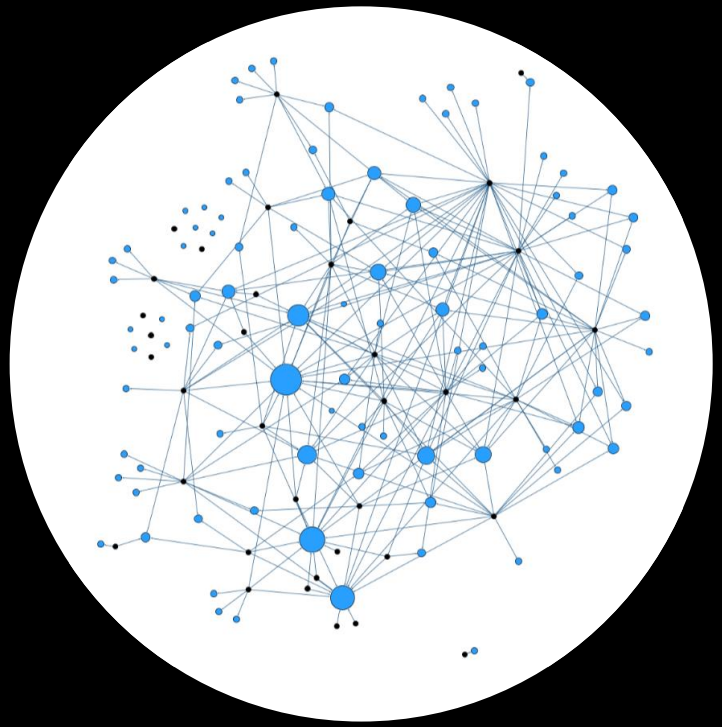
Each device model consists of a set of **interfaces.**

Interfaces describe attributes of the device:
telemetry, property, command
Single or multiple **components.**

The solution can query and parse the Device model to understand the interaction model.



Azure Digital Twins alignment



Digital Twin enables the creation of knowledge graphs based on digital models of physical environment.

Allows creation of relationships among twins.

Enriches twins by adding metadata and attributes.

IoT Plug and Play is fully aligned with Azure Digital Twins.

The IoT Plug and Play device model becomes one of the twins in the graph.

Allows addition of (plug) IoT device into the graph.

Allows interaction with (play) IoT device and data from it in the graph.

IoT Plug and Play device model authoring

IoT Plug and Play device model = Digital Twin model

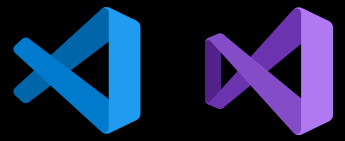
- A schema that describes device **capabilities** and **attributes**
- Written in **DTDL v2** based on **JSON-LD**
- **DTDL v2** for schemas, semantics

  https://aka.ms/dtdl

Tools for device model authoring

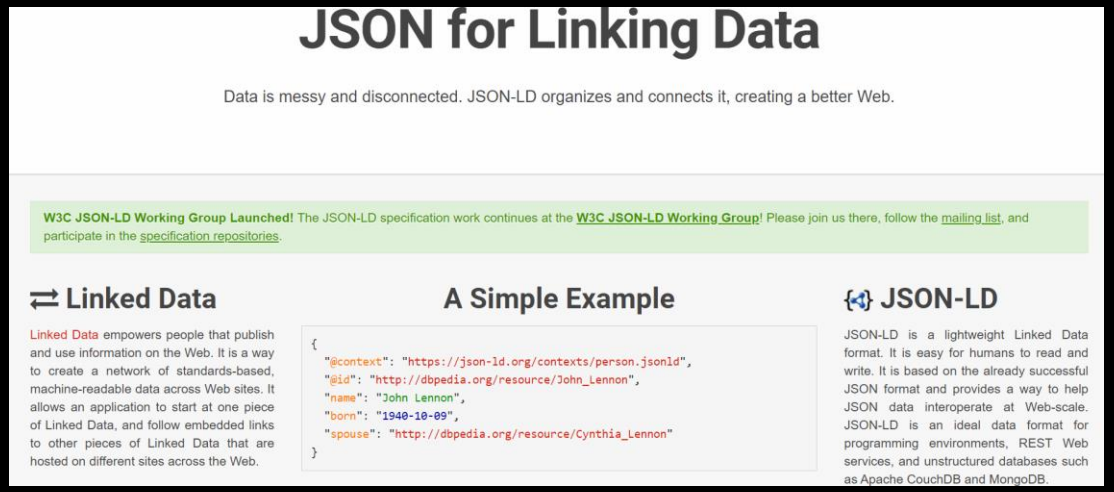
- **Visual Studio / Visual Studio Code**
  - Extensions for model authoring
- **DTDL Parser Library**
  - For syntax check

```
{
    "@context": "dtmi:dtdl:context;2",
    "@id": "dtmi:com:example:Thermostat;1",
    "@type": "Interface",
    "displayName": "Thermostat",
    "description": "IoT Plug and Play Device Model for Thermostat",
    "contents": [
      {
        "@type": [
          "Telemetry",
          "Temperature"
        ],
        "name": "temperature",
        "displayName" : "Temperature",
        "description" : "Temperature in degrees Celsius.",
        "schema": "double",
        "unit": "degreeCelsius"
      },
```



This new industry standard is a collaborative effort between Microsoft and the Digital Twin Consortium.

Base: JSON-LLD




```
{
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:com:example:Thermostat;1",
  "@type": "Interface",
```

https://json-ld.org/

Linking “data”

Used in Device Twins, Azure Twins.

Also: in general RDF (Semantic Web Resource Description Framework) and many other standards

IoT Plug and Play device model example

Temperature sensor

Telemetry payload

{“temp” : “12.34”}

Without device model

“temp is 12.34”

With device model

“Temperature is 12.34 degrees in Celsius”

```
{
  "@context":"dtmi:dtdl:context;2",                    DTDL v2
  "@id":"dtmi:com:mycompany:Thermostat;1",             Model ID
  "@type":"Interface",                                 Interface type
  "displayName":"IoT Plug and Play Device",
  "description":"Device Model Example",
  "contents": [
    {
      "@type":["Telemetry",                            Telemetry type
      "Temperature"],                                  Semantic temperature
      "name":"temp",                                   Name of data
      "displayName":"Temperature Data",
      "description":"Temperature in degrees Celsius.",
      "schema":"double",                               Schema
      "unit":"degreeCelsius"                           Unit
    },
```

DTDL

DTDL

Microsoft | 7,414 installs | ★★★★★ (1) | Free

This extension provides syntax highlighting to read and edit JSON documents using the Digital Twins Definition Language

Install Trouble Installing?

Overview | Version History | Q & A | Rating & Review

DTDL Editor for Visual Studio Code

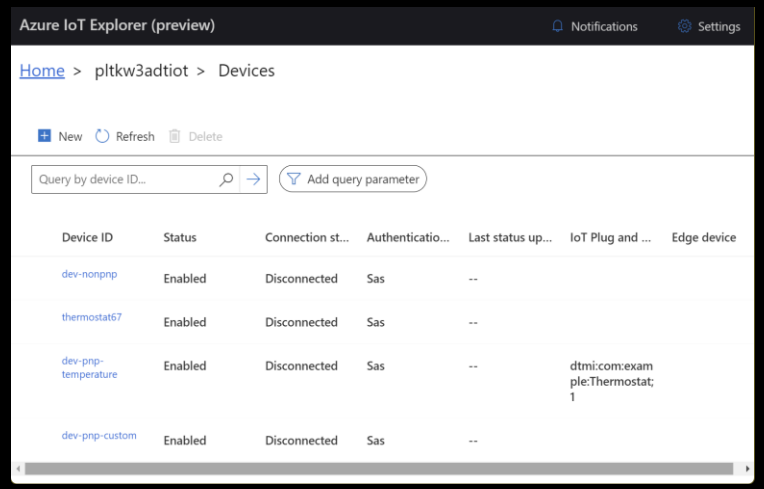
Overview

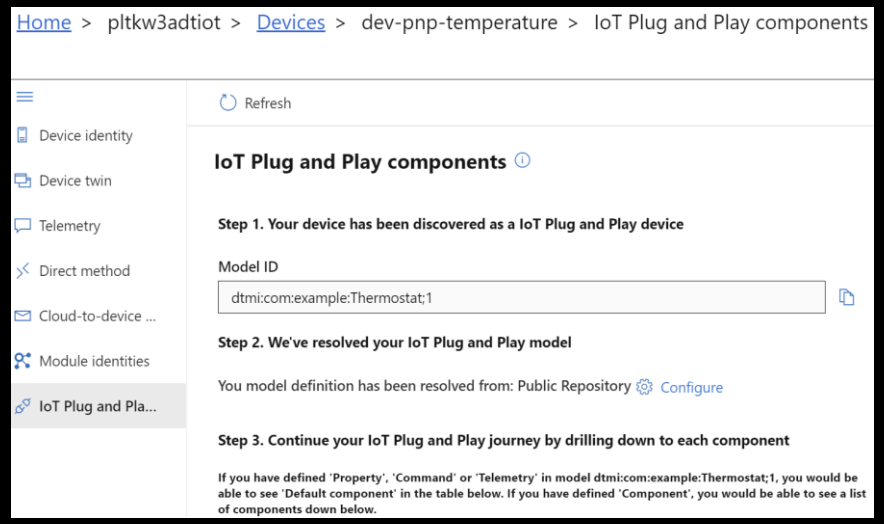
The Digital Twin Definition Language (DTDL) is a language for describing models for Plug and Play devices, device digital twins, and logical digital twins. Broadly, modeling enables IoT solutions to provision, use, and configure digital twins of all kinds from multiple sources in a single solution. Using DTDL to describe any digital twin's abilities enables the IoT platform and IoT solutions to leverage the semantics of each digital twin.

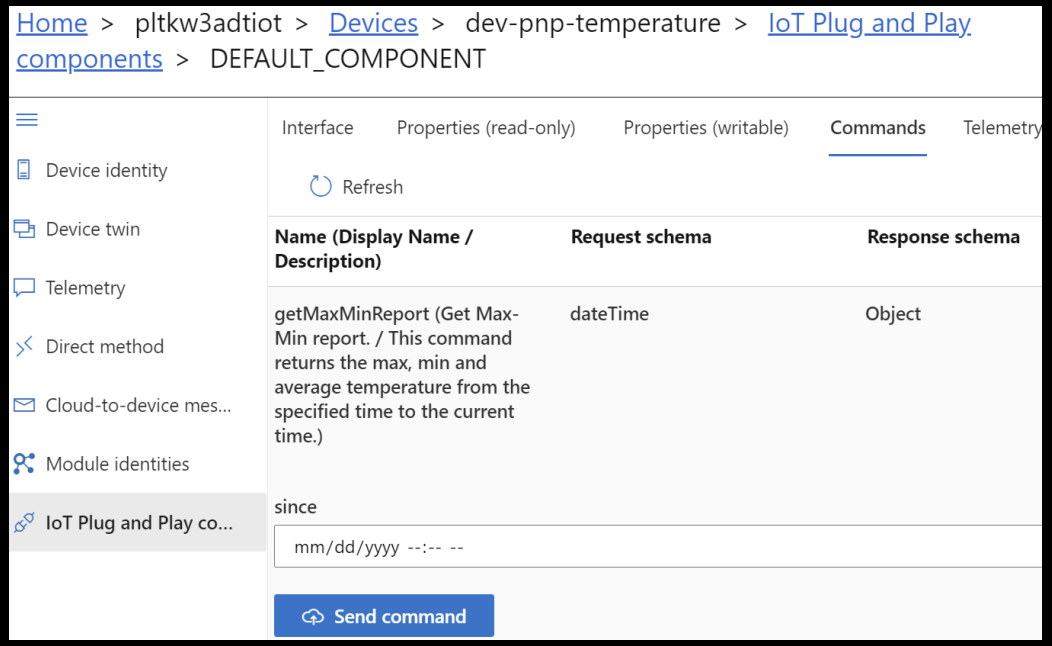
With the DTDL extension for Visual Studio Code , you can read and write documents using DTDL more efficiently

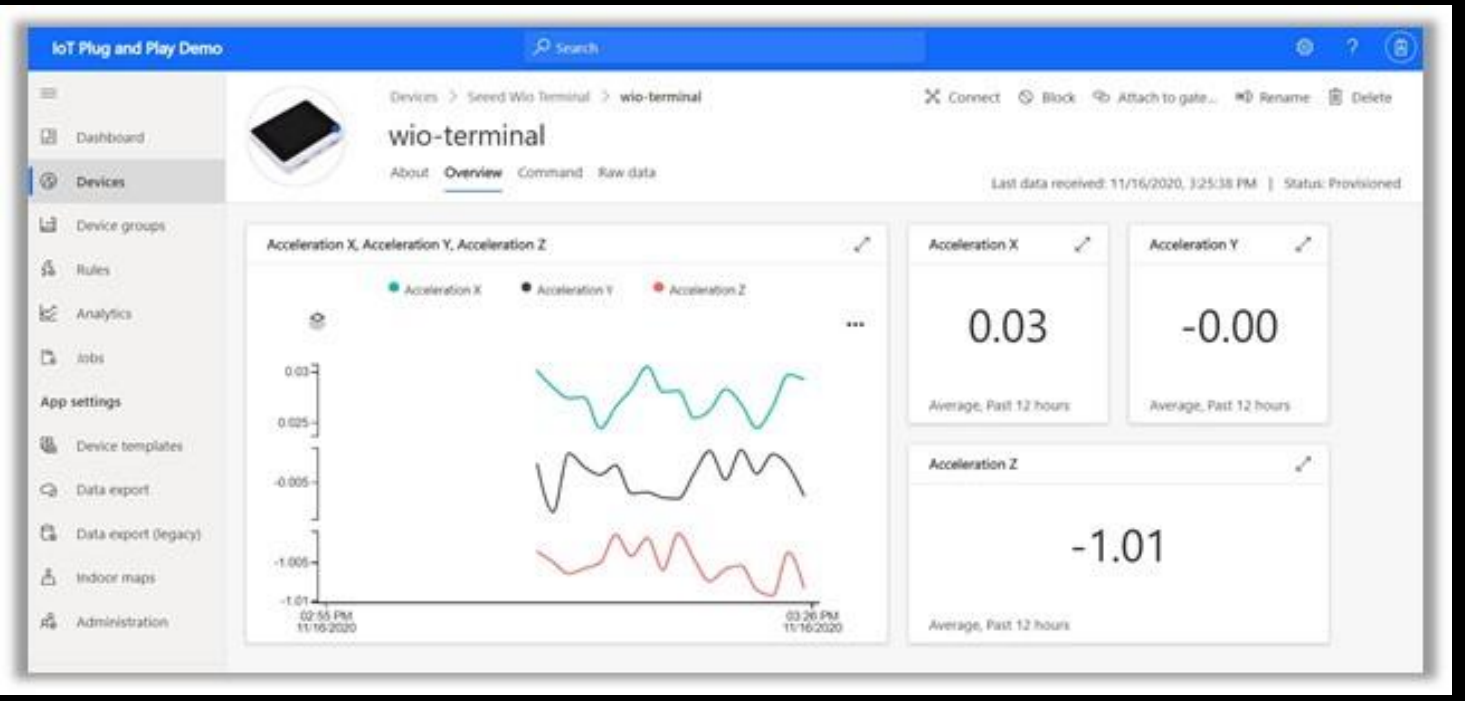
Demo – ok – browsing models ;)

Usage – „UI"

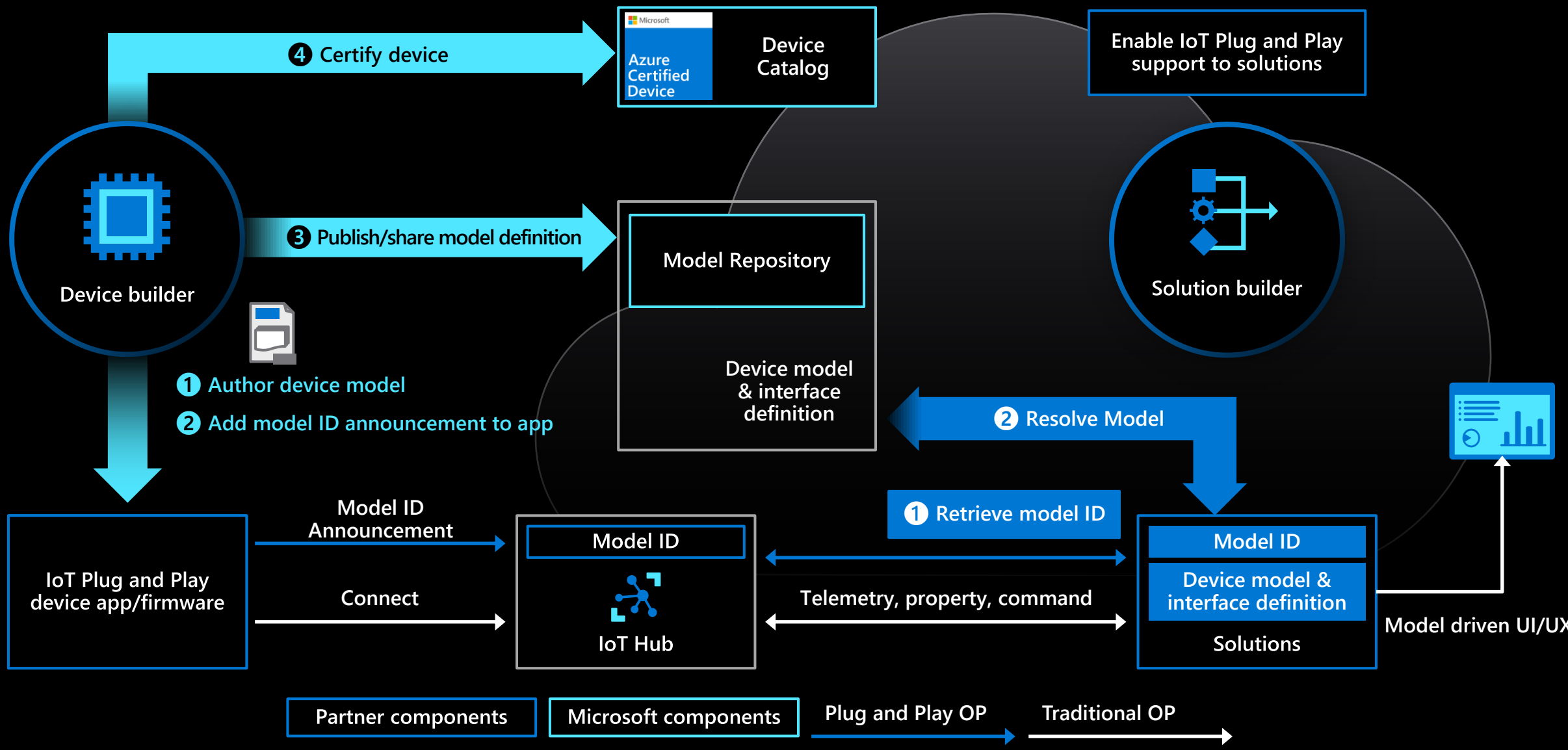

Azure IoT Explorer (preview)
Notifications
Settings
Home > pltkw3adtiot > Devices
New
Refresh
Delete
Query by device ID...
Add query parameter
Device ID
Status
Connection st...
Authenticatio...
Last status up...
IoT Plug and ...
Edge device
dev-nonpnp
Enabled
Disconnected
Sas
--
thermostat67
Enabled
Disconnected
Sas
--
dev-pnp-temperature
Enabled
Disconnected
Sas
--
dtmi:com:example:Thermostat;1
dev-pnp-custom
Enabled
Disconnected
Sas
--


Home > pltkw3adtiot > Devices > dev-pnp-temperature > IoT Plug and Play components
Refresh
Device identity
Device twin
Telemetry
Direct method
Cloud-to-device ...
Module identities
IoT Plug and Pla...
IoT Plug and Play components
Step 1. Your device has been discovered as a IoT Plug and Play device
Model ID
dtmi:com:example:Thermostat;1
Step 2. We've resolved your IoT Plug and Play model
You model definition has been resolved from: Public Repository
Configure
Step 3. Continue your IoT Plug and Play journey by drilling down to each component
If you have defined 'Property', 'Command' or 'Telemetry' in model dtmi:com:example:Thermostat;1, you would be able to see 'Default component' in the table below. If you have defined 'Component', you would be able to see a list of components down below.


Home > pltkw3adtiot > Devices > dev-pnp-temperature > IoT Plug and Play components > DEFAULT_COMPONENT
Device identity
Device twin
Telemetry
Direct method
Cloud-to-device mes...
Module identities
IoT Plug and Play co...
Interface
Properties (read-only)
Properties (writable)
Commands
Telemetry
Refresh
Name (Display Name / Description)
Request schema
Response schema
getMaxMinReport (Get Max-Min report. / This command returns the max, min and average temperature from the specified time to the current time.)
dateTime
Object
since
mm/dd/yyyy --:-- --
Send command


IoT Plug and Play Demo
Search
Dashboard
Devices
Device groups
Rules
Analytics
Jobs
App settings
Device templates
Data export
Data export (legacy)
Indoor maps
Administration
Devices > Seeed Wio Terminal > wio-terminal
wio-terminal
About
Overview
Command
Raw data
Connect
Block
Attach to gate...
Rename
Delete
Last data received: 11/16/2020, 3:25:38 PM | Status: Provisioned
Acceleration X, Acceleration Y, Acceleration Z
Acceleration X
Acceleration Y
Acceleration Z
0.03
-0.00
-1.01
Average, Past 12 hours

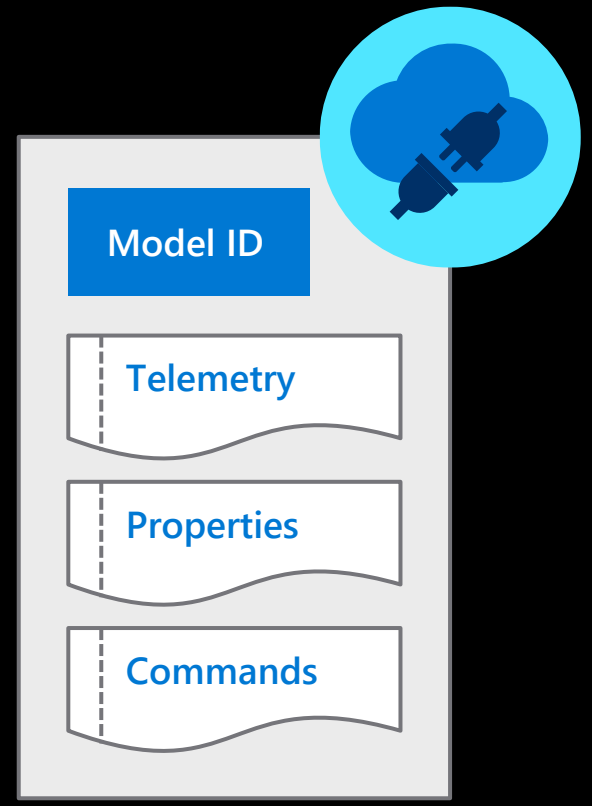
IoT Plug and Play process overview



IoT Plug and Play device app / firmware development

Implement IoT Plug and Play convention based on the device model

- Model ID announcement: minimum required
- Telemetry – Data from device to cloud
- Writable properties – Settings from cloud to device
- Read-only properties – Reported by device
- Commands – Invoking method on device from cloud
- Follows IoT Plug and Play convention



IoT Plug and Play conventions

IoT Plug and Play conventions

- Devices must follow the conventions when exchanging messages with IoT Hub.
- This ensures consistency in the device model and metadata in the messages.
- Learn more about IoT Plug and Play conventions.
- (open it!)


```
"reported": {
  "thermostat1": {
  "__t": "c",
  "targetTemperature": {
      "value": 23,
      "ac": 200,
      "av": 3,
      "ad": "complete“
    }
  }
}
```

Azure IoT device model repository (DMR) & Model Resolution

Solution needs to access to the model definition file (JSON file)

- Through Azure IoT Model Repository
- Through your **own repository** such as GitHub or file share

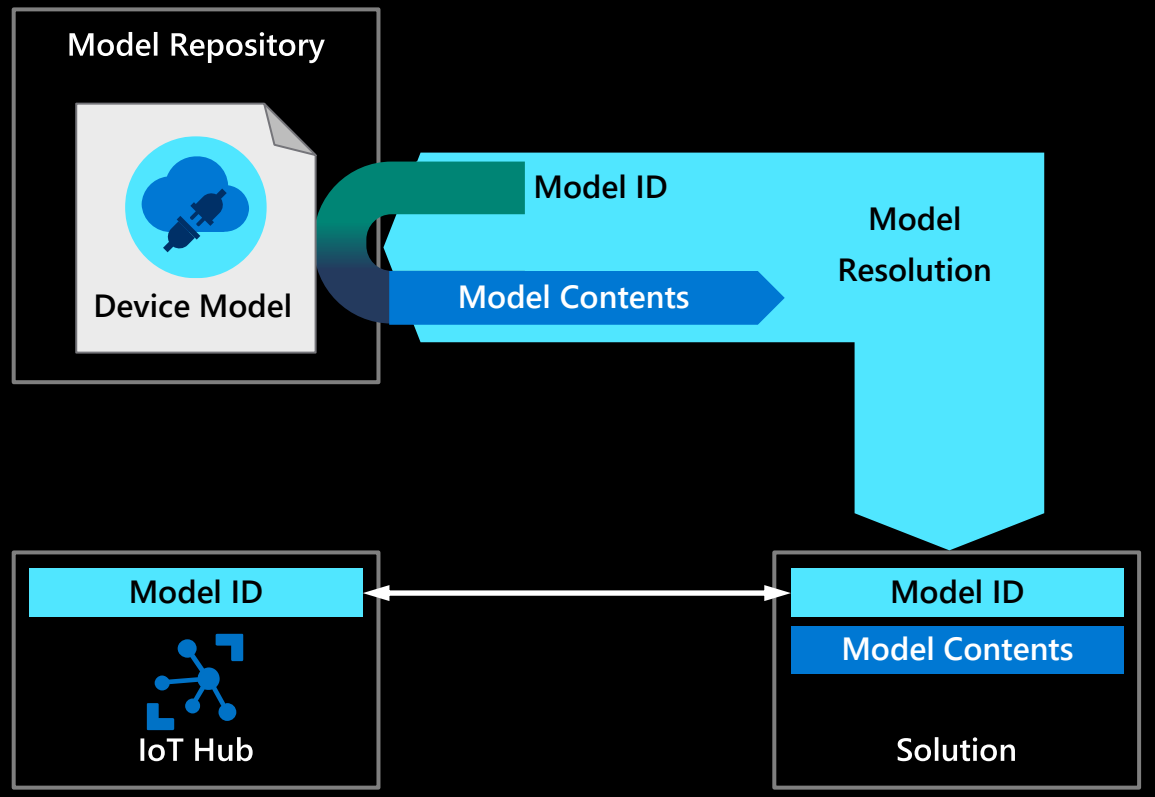
Models are immutable

- Approved models in public model repository are immutable

Access through Azure IoT model repository

- Microsoft-hosted public model repository
- Custom model repository
- Supports HTTP API and local file access
- Model Repo Client SDK

Z:\AzFY21-PaaS03\16-IoT-Hub\2022IotPnP\iot-plugandplay-models\dtmi



Demo

IoT vs IoT PnP (C#)
How to work with writable properties/desired state/commands
PnP – playing with Model from code (Device Twins queries)

Custom model in folder

JS, C/C++ IoT Hub SDK and PnP

JS

```
const client =Client.fromConnectionString(deviceConnectionString, Protocol);
  try {
    // Add the modelId here
    await client.setOptions(modelIdObject);
    await client.open();
```

C/C++

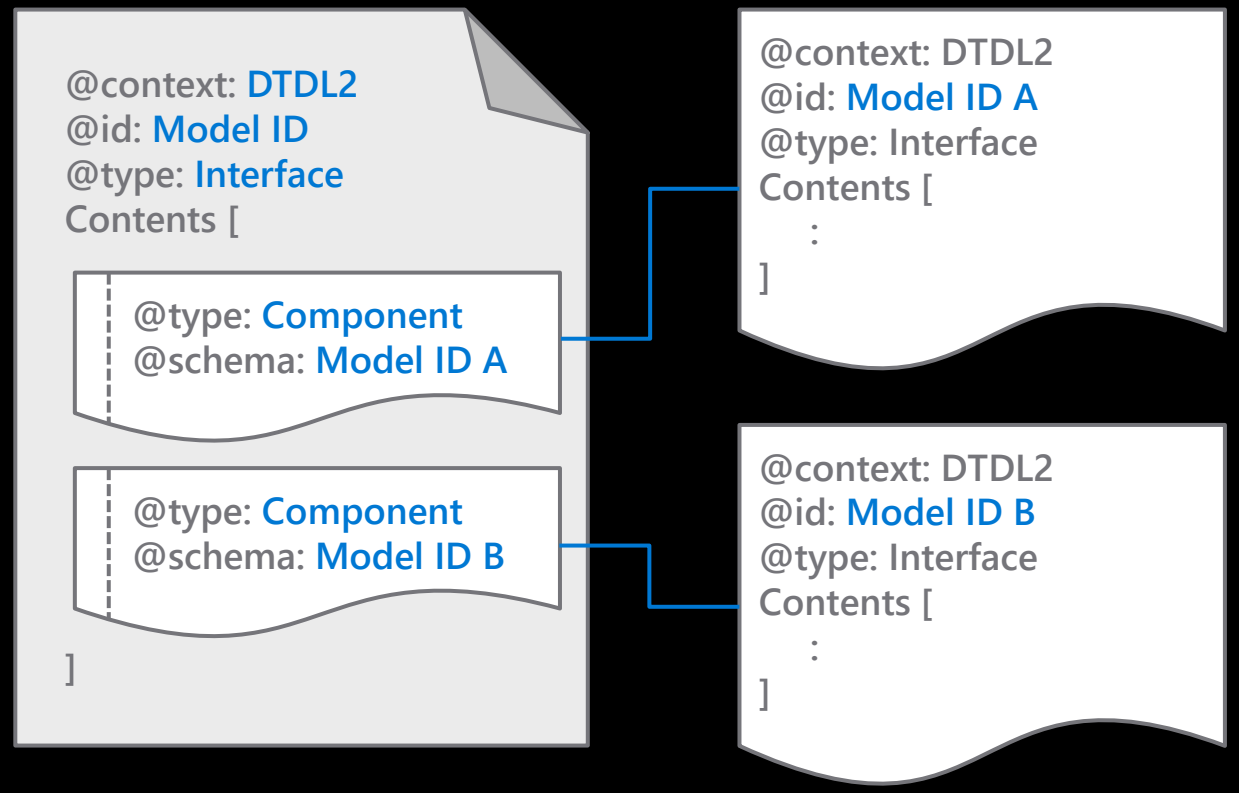
```
deviceHandle = IoTHubDeviceClient_LL_CreateFromConnectionString(cnn, MQTT_Protocol));
// Sets the name of ModelId for this PnP device.
// This *MUST* be set before the client is connected to IoTHub.  We do not automatically connect when the
// handle is created, but will implicitly connect to subscribe for device method and device twin callbacks below.
iothubResult = IoTHubDeviceClient_LL_SetOption(deviceHandle, OPTION_MODEL_ID, modelId));
...
iothubResult = IoTHubDeviceClient_LL_SetDeviceMethodCallback(deviceHandle, callback, NULL)...
iothubResult = IoTHubDeviceClient_LL_SetDeviceTwinCallback(deviceHandle, callback, (void*)deviceHand)
...
```

What if we have many sensors? Like – humidity and temperature and “switch” – in single device

Componentizing IoT Plug and Play device model

Enables creation of a device model interface as an assembly of other interfaces

Default device model is a **single interface (or component-less)** model



(Gateway patterns)

Transparent

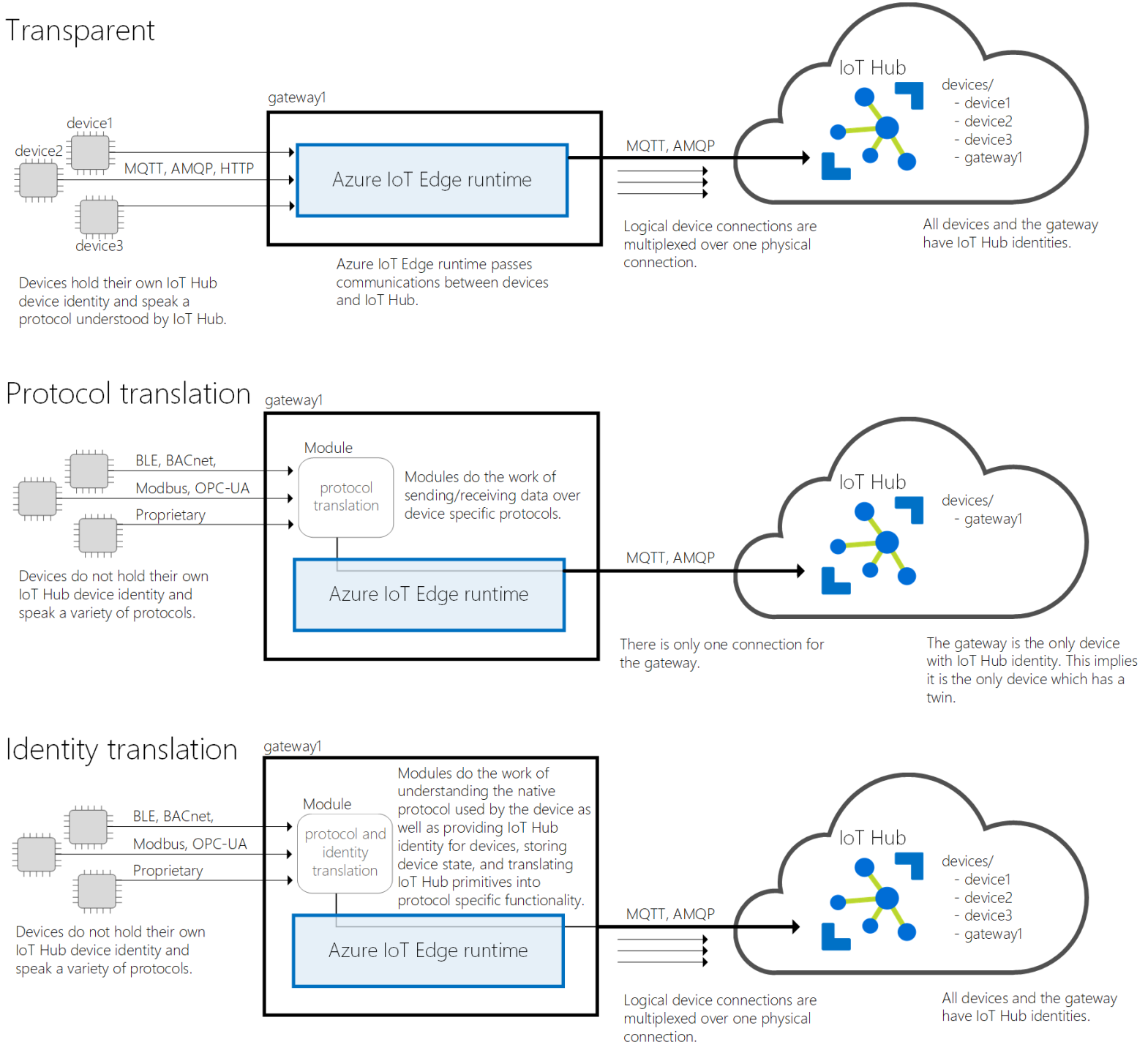
IoT Hub-aware devices connect as leaf devices

Protocol translation

Module talks to leaf devices and acts as a single device in IoT Hub

Identity translation

Module talks to leaf devices and impersonates/manages identities of leaf devices as unique devices in IoT Hub

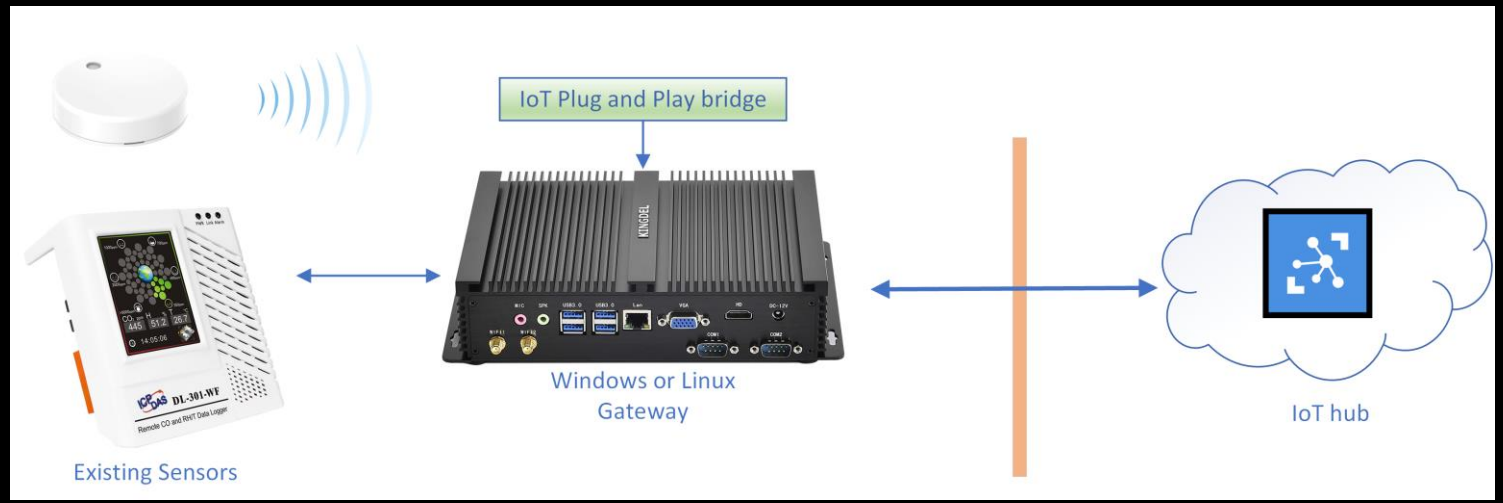


IoT Plug and Play bridge

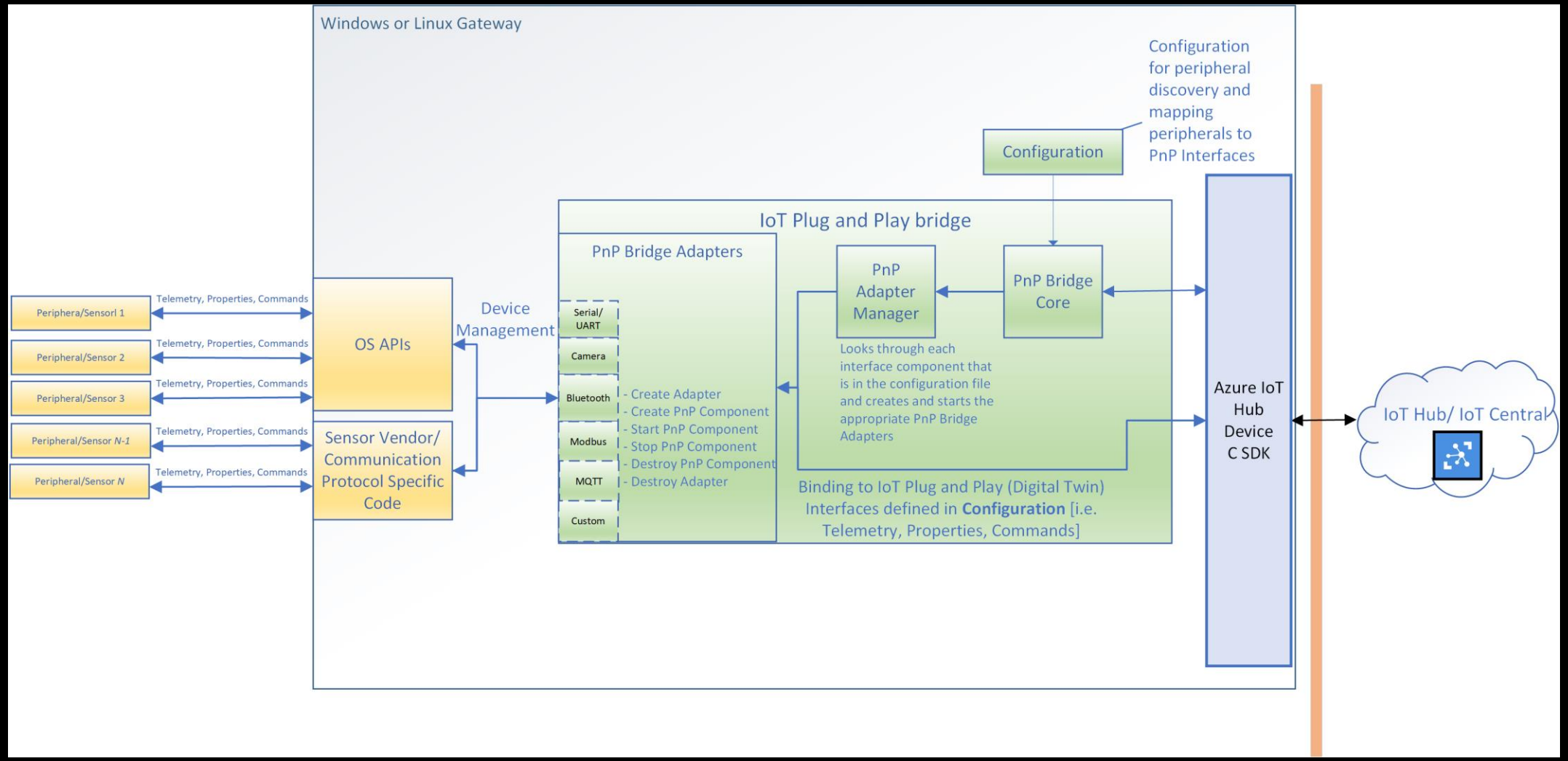
An open-source application for connecting existing devices attached to Windows or Linux gateway as IoT Plug and Play devices

IoT Plug and Play bridge enable:

- Convert supported sensors connected to a Windows / Linux gateway into IoT Plug and Play devices w/o writing any code
- Open-source framework that can be extended to support proprietary protocols
- Leveraging of existing OS capabilities w/o having to write to both a device and cloud SDK



IoT Plug and Play bridge architecture



Capabilities

Peripheral	Windows	Linux
Bluetooth sensor adapter connects detected Bluetooth Low Energy (BLE) enabled sensors.	Yes	No
Camera adapter connects cameras on a Windows 10 device.	Yes	No
Modbus adapter connects sensors on a Modbus device.	Yes	Yes
MQTT adapter connects devices that use an MQTT broker.	Yes	Yes
SerialPnP adapter connects devices that communicate over a serial connection.	Yes	Yes
Windows USB peripherals uses a list of adapter-supported device interface classes to connect devices that have a specific hardware ID.	Yes	Not Applicable

Open Source

https://github.com/Azure/iot-plug-and-play-bridge

Extensibility: https://github.com/Azure/iot-plug-and-play-bridge/blob/master/pnpbridge/docs/author_adapter.md

Set of structures and pointers to methods (telemetry etc)

Also: serialpnp - Serial PnP protocol – Arduino, STM

Can be run as

- Use pnpbridge_bin.exe / pnpbridge_bin to run the bridge native application
- Use pnpbridgesvc.exe to run the bridge as a service on an IoT device or gateway running windows
- Use pnpbridge_module to run the bridge as a module on an IoT edge runtime running linux.

Demo

Windows, laptop and IoT PnP Gateway

DPS (and PnP) – to manage MANY devices

Zero-touch provisioning to a single IoT solution without hardcoding IoT Hub connection information at the factory (initial setup)

Load-balancing devices across multiple hubs

Connecting devices to their owner's IoT solution based on sales transaction data (multitenancy)

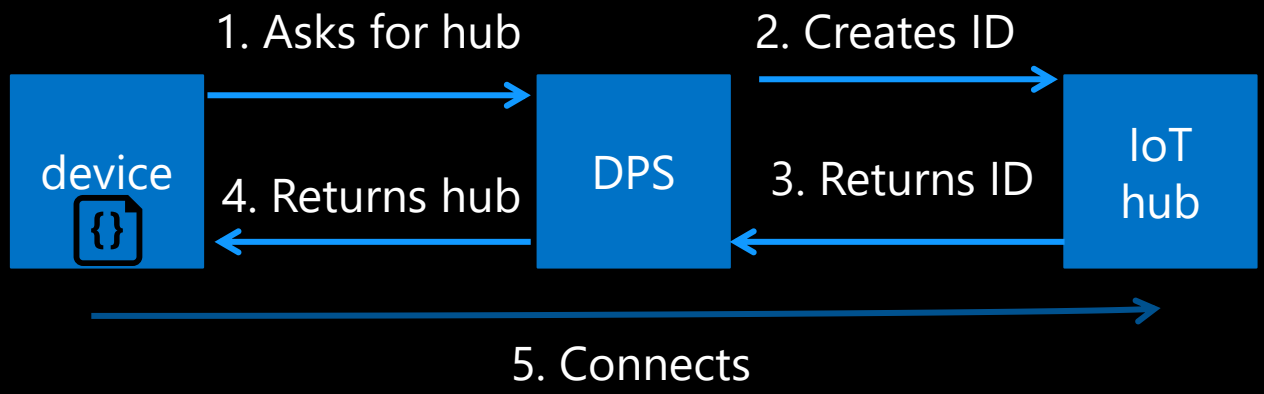
Connecting devices to a particular IoT solution depending on use-case (solution isolation)

Connecting a device to the IoT hub with the lowest latency (geo-sharding)

Reprovisioning based on a change in the device

Rolling the keys used by the device to connect to IoT Hub (when not using X.509 certificates to connect)

Zero touch provisioning

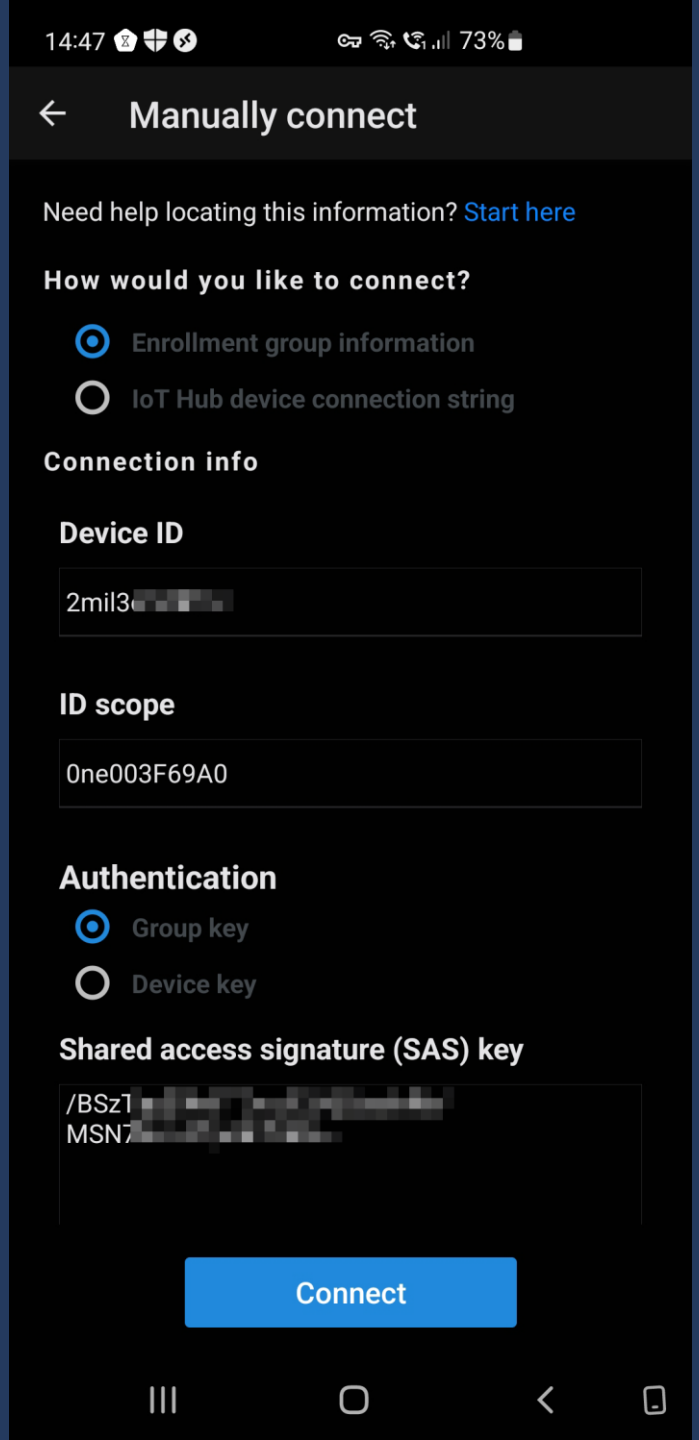


Connect to IoT Hub through DPS

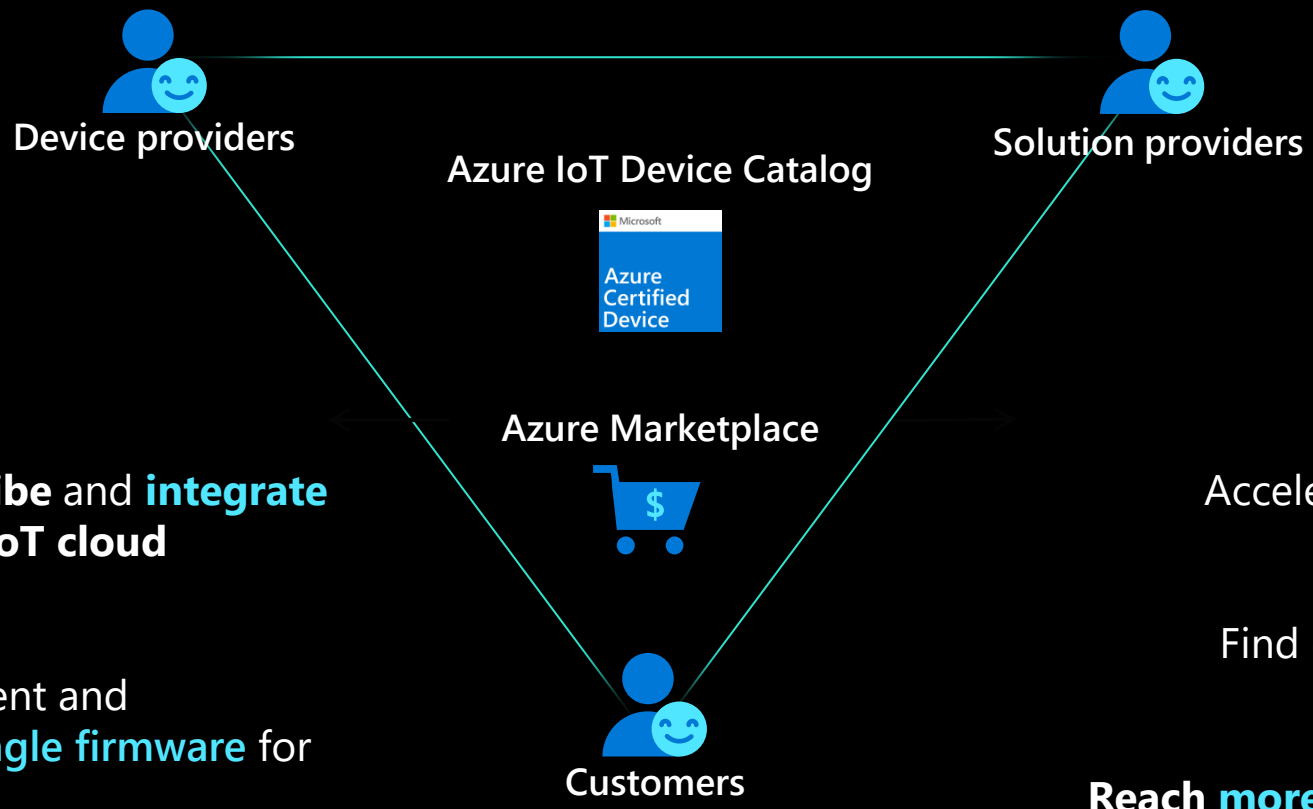
```
SecurityProvider symmetricKeyProvider = new
  SecurityProviderSymmetricKey(parameters.DeviceId, parameters.DeviceSymmetricKey, null);
ProvisioningTransportHandler mqttTransportHandler =
  new ProvisioningTransportHandlerMqtt();
ProvisioningDeviceClient pdc = ProvisioningDeviceClient.Create(parameters.DpsEndpoint,
  parameters.DpsIdScope,
  symmetricKeyProvider, mqttTransportHandler);

var pnpPayload = new ProvisioningRegistrationAdditionalData
    { JsonData = $"{{ \"modelId\": \"{ModelId}\" }}" };
DeviceRegistrationResult dpsRegistrationResult = await
    pdc.RegisterAsync(pnpPayload, cancellationToken);
var authMethod = new DeviceAuthenticationWithRegistrySymmetricKey(
    dpsRegistrationResult.DeviceId, parameters.DeviceSymmetricKey);
var options = new ClientOptions { ModelId = ModelId };

DeviceClient deviceClient = DeviceClient.Create(dpsRegistrationResult.AssignedHub,
                                                authMethod, TransportType.Mqtt, options);
///We have deviceClient based on IoT Hub assigned by DPS based on enrollment rules
```

Demo

PnP (Phone) and DPS (Azure)



Building the IoT Plug and Play ecosystem – 2021 and beyond



Device providers

Build devices that **self-describe** and **integrate seamlessly** with **any Azure IoT cloud solution.**

Reduces firmware development and maintenance effort with **a single firmware** for all solutions.

Increase visibility to solution developers via the **Azure Certified Device Catalog.**

Solution providers

Accelerate solutions **without writing embedded code.**

Find devices with the **capabilities to fit your solution needs.**

Reach more customers by promoting your solutions in the **Azure Marketplace.**

Azure Certified Device Program overview

The Azure Certified Device Program currently offers three device certifications

Certification	Promise	Requirements	Targeted Devices
 Azure Certified Device	Works with IoT Hub	• Device to Cloud telemetry • Device Provisioning with DPS	• Microcontroller • IoT Devices • IoT Edge Devices • Edge Appliances • Edge Stack
 Edge Managed	IoT Edge RT works	• Device Provisioning with DPS • IoT Edge RT running on Moby container subsystem • Moby & Edge Security Manager preinstalled	• IoT Edge Devices • Edge Appliances • Edge Stack

 IoT Plug and Play – New!	Simplify IoT solution	• IoT Plug and Play device model compliance
		• Including Azure Certified Device requirements
		  • D2C telemetry
		  • Device Provisioning with DPS
			• Microcontroller
			• IoT Devices
			• IoT Edge Devices

IoT Plug and Play certification requirements

Technical requirements

- Device model(s) describing the device and peripherals.
- Support Device to Cloud message (D2C) with the IoT Plug and Play convention.
- Support Device Provisioning Service (DPS) with one or more device authentication types:
  - Symmetric Key
  - X.509
  - Trusted Platform Module (TPM)
- Model ID announcement
  - During device provisioning through DPS
  - During the MQTT connection
- Publish device model to Model Repo provided by Microsoft

Optional features

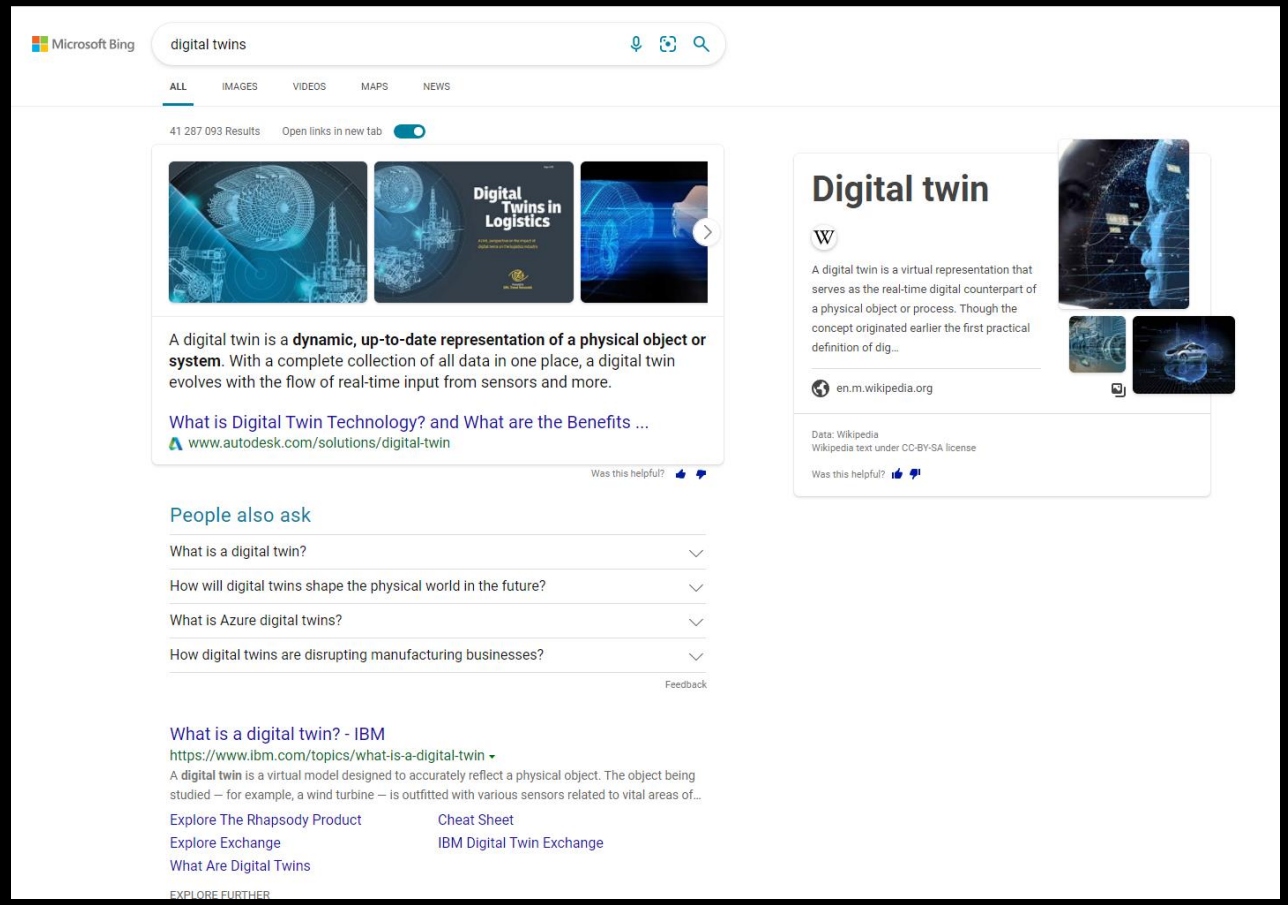
- Cloud to Device (C2D) message
- Device method (or direct method)
- Device properties, writable and non-writable

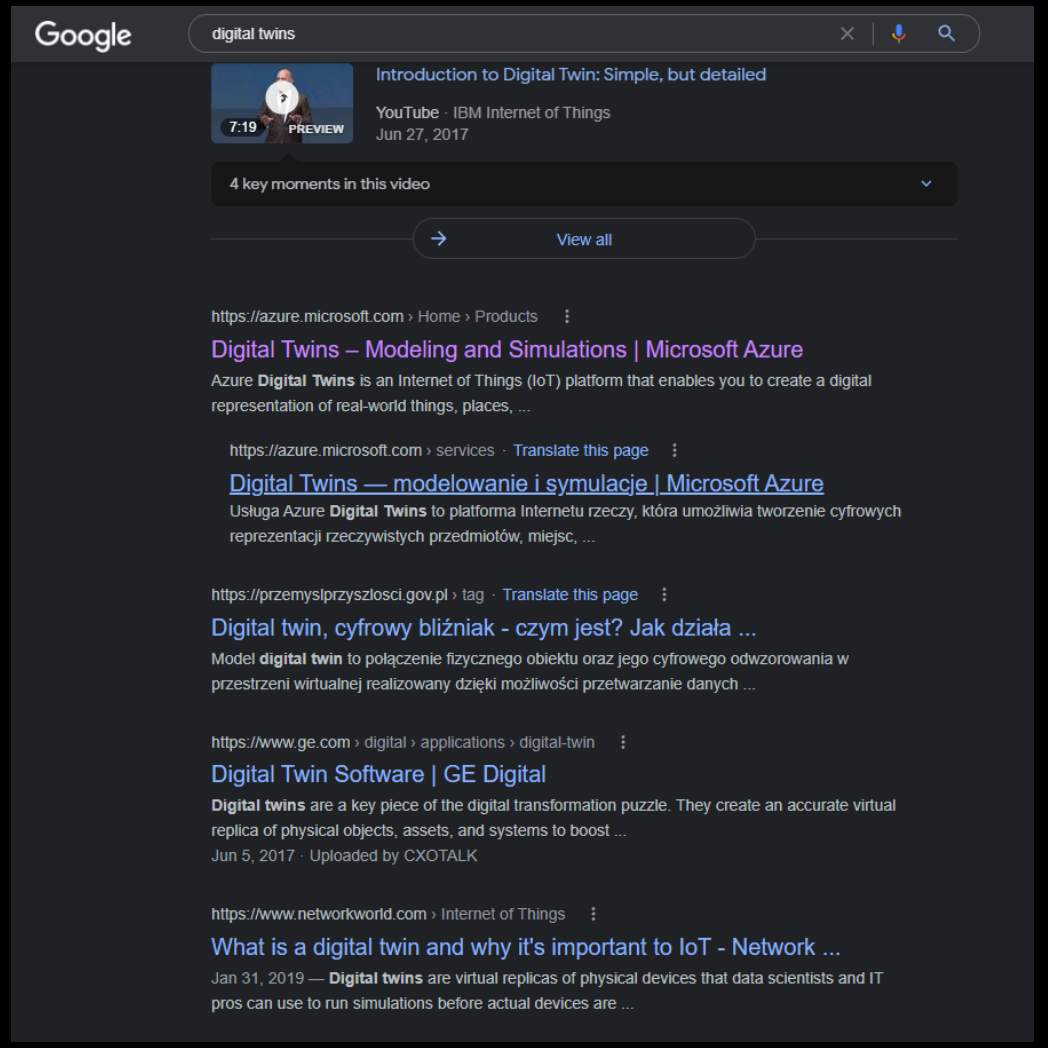
Yesss…. It was all for SINGLE device

What can we do to work effectively in more “complex” environments?

Intro to Digital Twins and Azure Digital Twins

Digital twin?

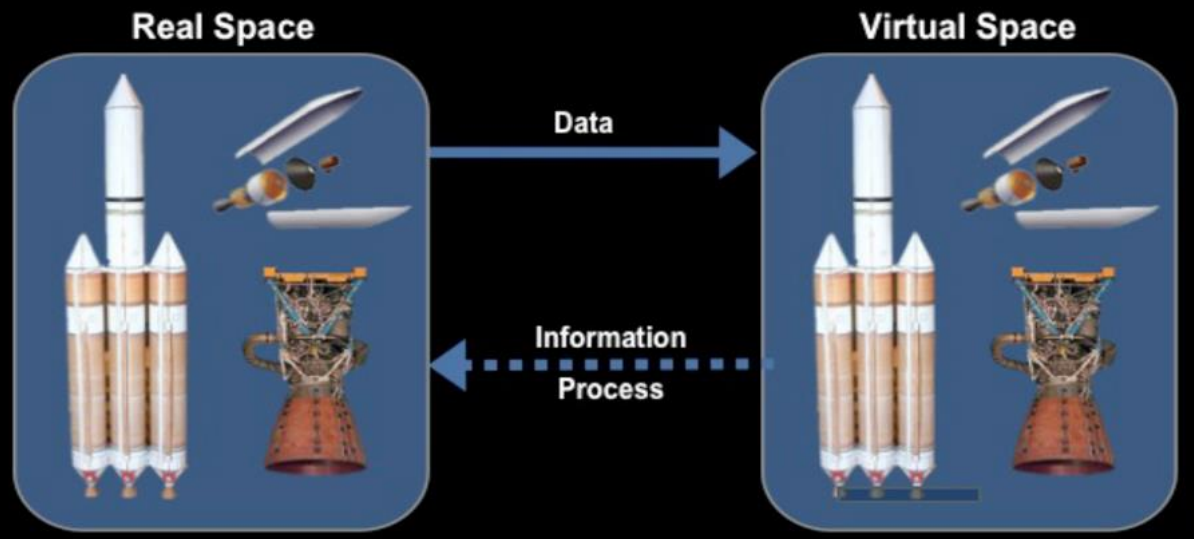




Wikipedia definition + comments

A digital twin is a **virtual representation that serves as the real-time digital counterpart of a physical object or process**. [...] the first practical definition of digital twin originated from NASA in an attempt to improve physical model simulation of spacecraft in 2010

[...] Digital twins were anticipated by David Gelernter's 1991 book Mirror Worlds.[...] Grieves proposed the digital twin as the conceptual model underlying product lifecycle management (PLM).



By Wilmjakob - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=93687174

Towards Connected Environments

As connected solutions continue to evolve, businesses are looking to connect entire environments

Connected Environments

Manufacturing

Plants

Factories

Industrial Sites

Utilities

Retail

Stores

Warehouses

Distribution Centers

Malls

Micro-fulfillment

Real Estate

Building

Office

Entertainment

Restaurants

Hotels

Enterprise Campus

Cities & Government

Public Spaces

Arenas & Stadiums

Transportation Hubs

Subway & Train Stations

Ports

Utilities

Water Treatment Facilities

Healthcare

Hospitals

Labs

Pharmacies

Clinics

Retirement Facilities

Agriculture

Farms

Water Supply

Power

Storage

Processing

Education

Schools

Campuses

Virtual Environments

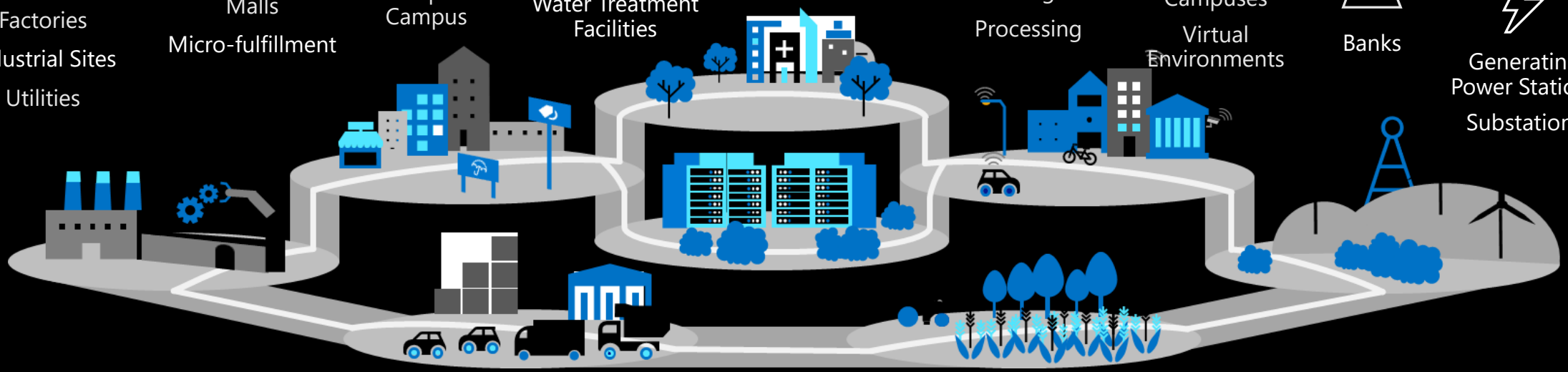
Finance

Banks

Energy

Generating Power Stations

Substations



Azure Digital Twins

IoT solutions that model the real world

Client Apps

Manage models and the digital twins graph

IoT Hub

Workflow integration
(e.g. Logic Apps)

Business systems/services
integration via REST APIs

REST API

INPUT

Digital Twins
Definition
Language

AZURE DIGITAL
TWINS GRAPH

Azure Digital Twins

LIVE EXECUTION ENVIRONMENT

External Compute

Handles business logic and data processing

Workflow integration
(e.g. Logic Apps)

Cold Storage

ADX

Analytics

OUTPUT

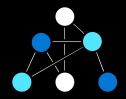
Demo

Design ADT - DTDL, tools, validator etc
https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v2/dtdlv2.md

https://github.com/Azure/opendigitaltwins-tools
(converter from Web Ontology Language (OWL) and many others!)

Azure Digital Twins

Model any environment, connect sensors and business systems to the model.
Control the present, track the past and predict the future



Open Modeling Language

Live Execution Environment

Input from IoT & Business Systems

Output to ADX, Storage & Analytics

Digital Twin Partnerships

- Create custom domain models using “Digital Twins Definition Language” (DTDL)
- ADT Models describe twins in terms of
  - Telemetry
  - Properties
  - Relationships
  - Components
- Models define semantic relationships to connect twins into a knowledge graph
- Models can specialize other twins using inheritance
- Digital Twins Definition Language is aligned with
  - IoT Plug and Play
  - ADX Model
- https://github.com/Azure/opendigitaltwins-dtdl

```
{
    "@id": “dtmi:example:Station;1",
    "@type": "Interface",
    "extends": “dtmi:example:Room;1",
    "contents": [
        {
            "@type": "Property",
            "name": “isOccupied",
            "schema": "boolean“
        },
        {
            "@type": “Property",
            "name": “hasAVSystem",
            "schema": “boolean“
        },
        {
            "@type": "Property",
            "name": “capacity",
            "schema": “integer“
        }
    ],
    "@context": "dtmi:dtdl:context;2"
}
```

Important to remember!

Property

- Data fields that represent the state of an entity.

Telemetry

- Measurements or events, based on sensor readings. Stream of data – NOT STORED in ADT

Component

- Defines an assembly of other interfaces, that make up a model.

Relationship

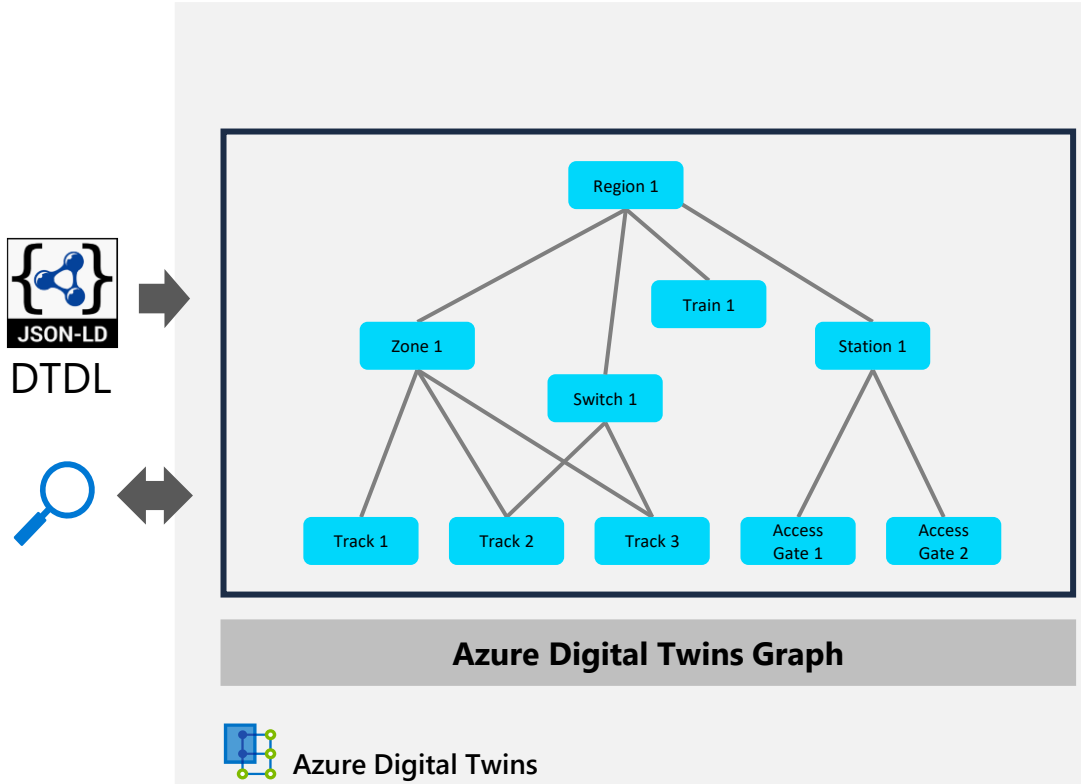
- Represents how your model interacts with other models. Defines the graph of related entities.

Azure Digital Twins

Model any environment, connect sensors and business systems to the model.
Control the present, track the past and predict the future

Open Modeling Language | **Live Execution Environment** | Input from IoT & Business Systems | Output to ADX, Storage & Analytics | Digital Twin Partnerships

- Create a live execution environment from the DTDL models in Azure Digital Twins
- Twin instances and relationships form a live graph representation of the environment
- Use a rich event system to drive business logic and data processing. Use external compute such as Azure Functions
- Extract insights from the live execution environment with a powerful query API
- Query using rich search conditions, including property values, relationships, relationship properties, type information and more

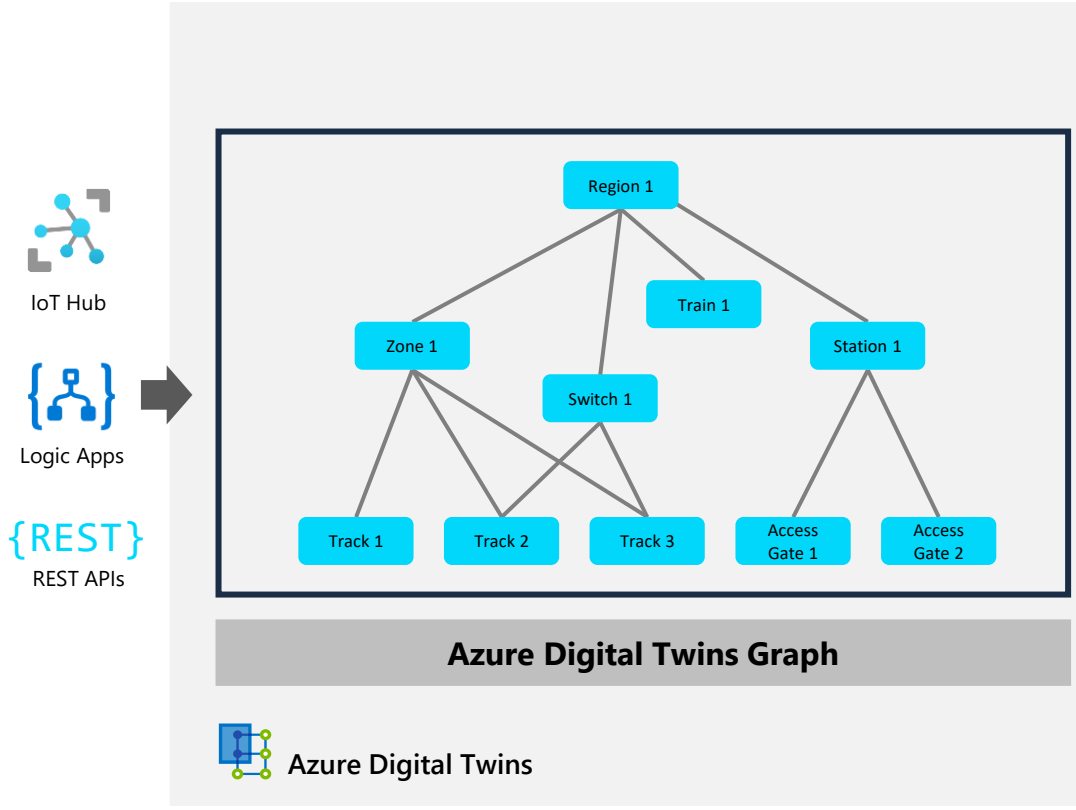


Azure Digital Twins

Model any environment, connect sensors and business systems to the model.
Control the present, track the past and predict the future

Open Modeling Language | Live Execution Environment | **Input from IoT & Business Systems** | Output to ADX, Storage & Analytics | Digital Twin Partnerships

- Use IoT Hub to connect to IoT and IoT Edge devices to keep the live execution environment up to date
- Use a new or an existing IoT Hub (IoT Hub is no longer internal to Azure Digital Twins)
- Drive Azure Digital Twins from other data sources using REST APIs or create a Logic Apps connector

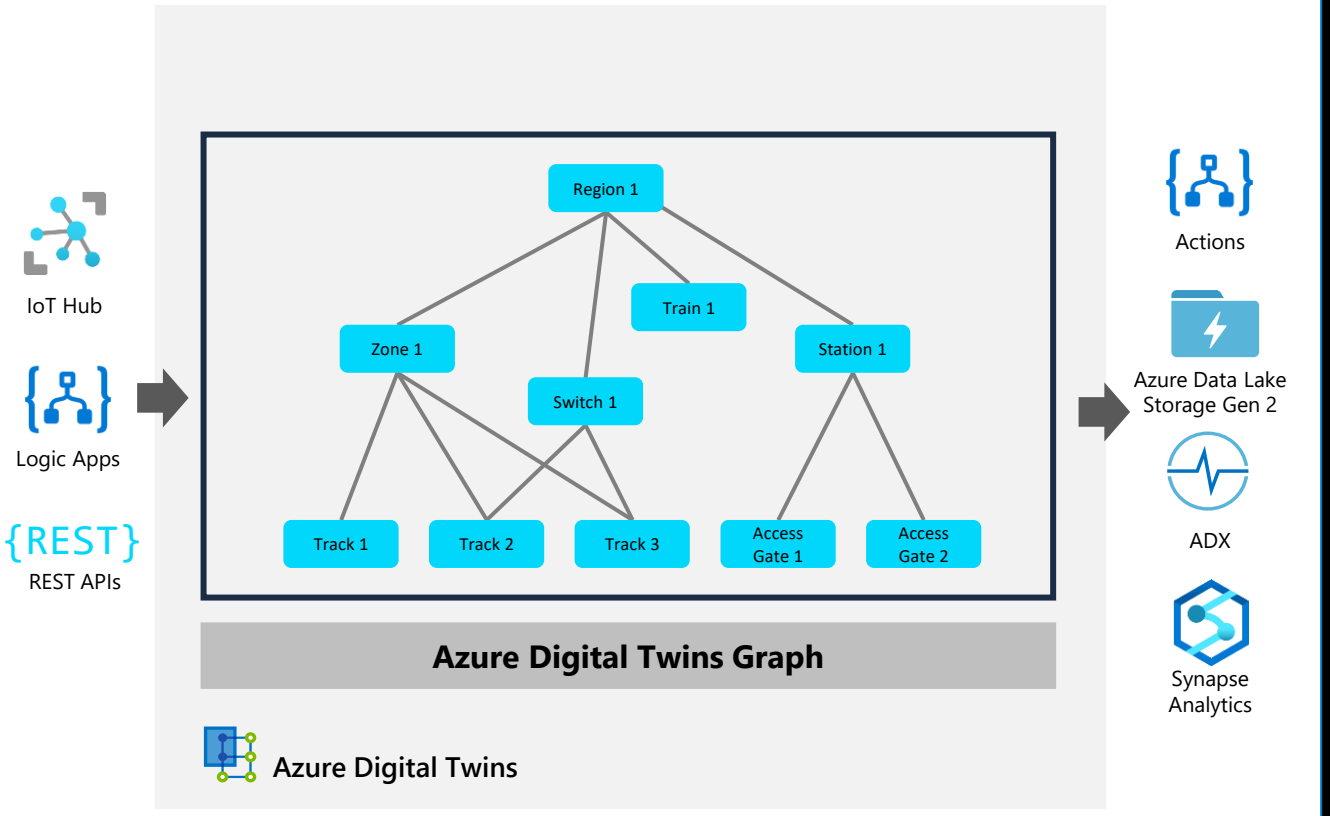


Azure Digital Twins

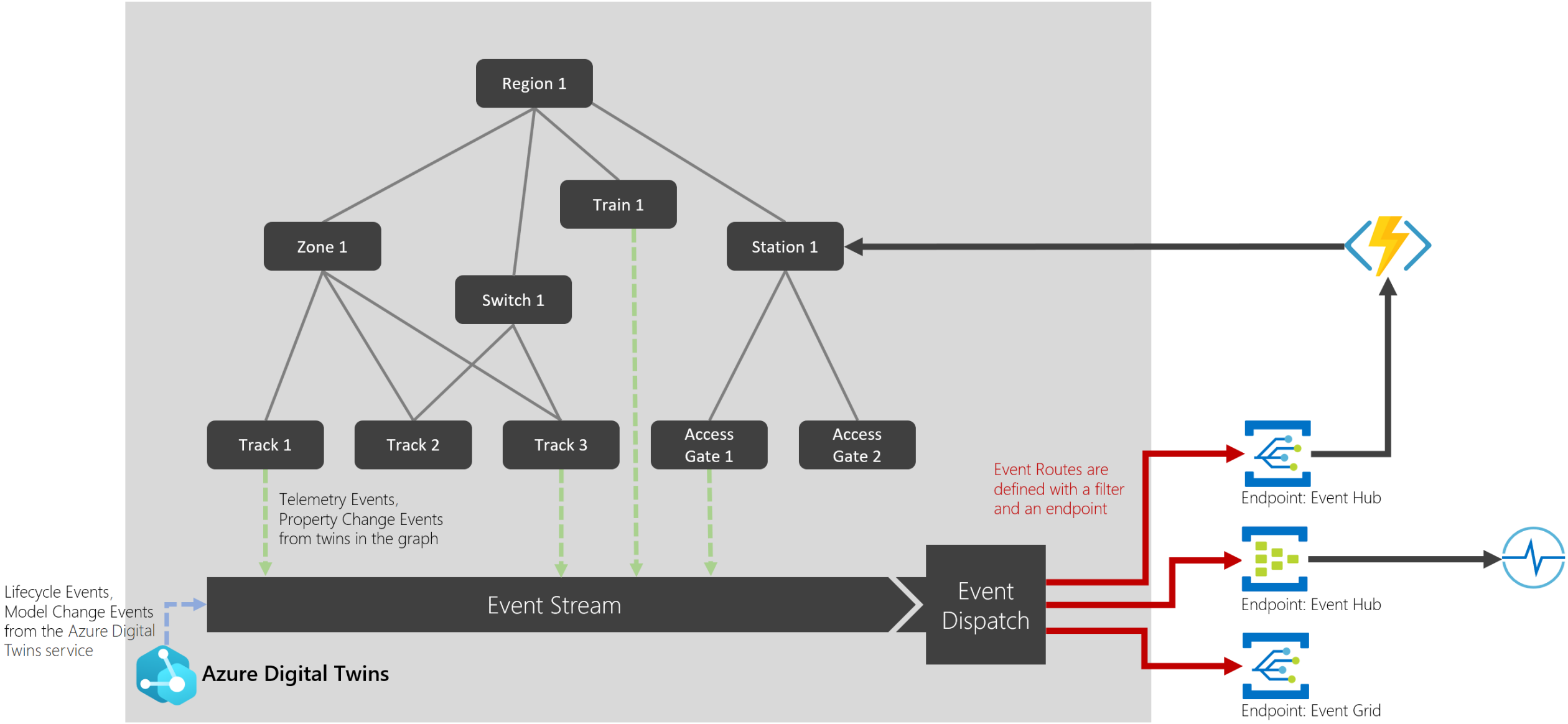
Model any environment, connect sensors and business systems to the model.
Control the present, track the past and predict the future

Open Modeling Language | **Live Execution Environment** | **Input from IoT & Business Systems** | **Output to ADX, Storage & Analytics** | **Digital Twin Partnerships**

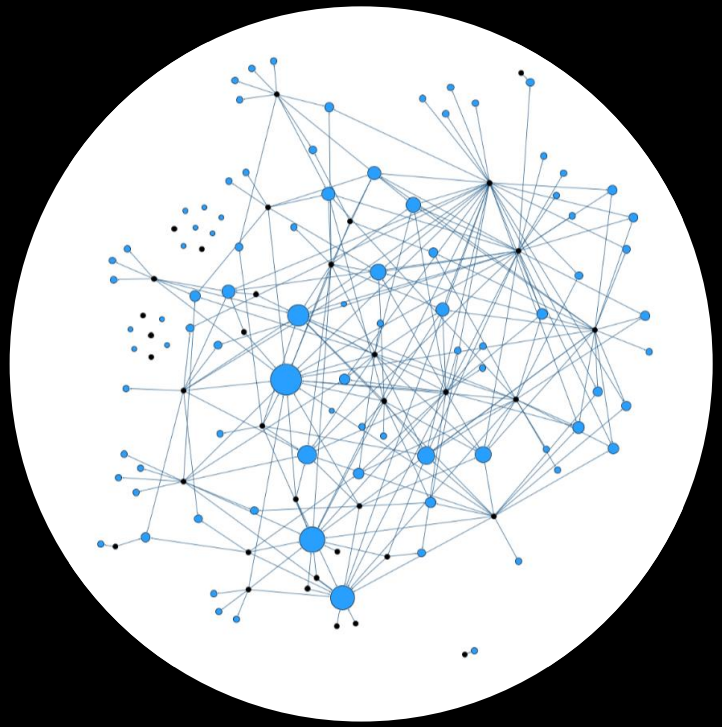
- Use event routes to send data to downstream services via Event Hub, Event Grid or Service Bus
- Store data in Azure Data Lake Storage Gen 2, analyze data with Azure Synapse and other Microsoft data tools for analytics, integrate workflow with Logic Apps
- Connect Azure Digital Twins to ADX to track time series history of each node
- Aligned Time Series Model in Azure Time Series Insights mastered from Azure Digital Twins



Generic flow in complex IoT Environment



Reminder: Azure Digital Twins alignment



Digital Twin enables the creation of knowledge graphs based on digital models of physical environment.

Allows creation of relationships among twins.

Enriches twins by adding metadata and attributes.

IoT Plug and Play is fully aligned with Azure Digital Twins.

The IoT Plug and Play device model becomes one of the twins in the graph.

Allows addition of (plug) IoT device into the graph.

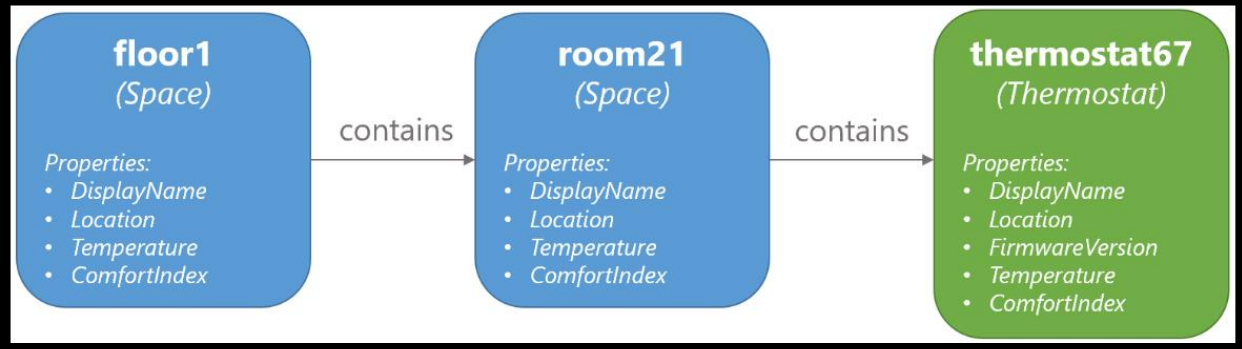
Allows interaction with (play) IoT device and data from it in the graph.

Demo

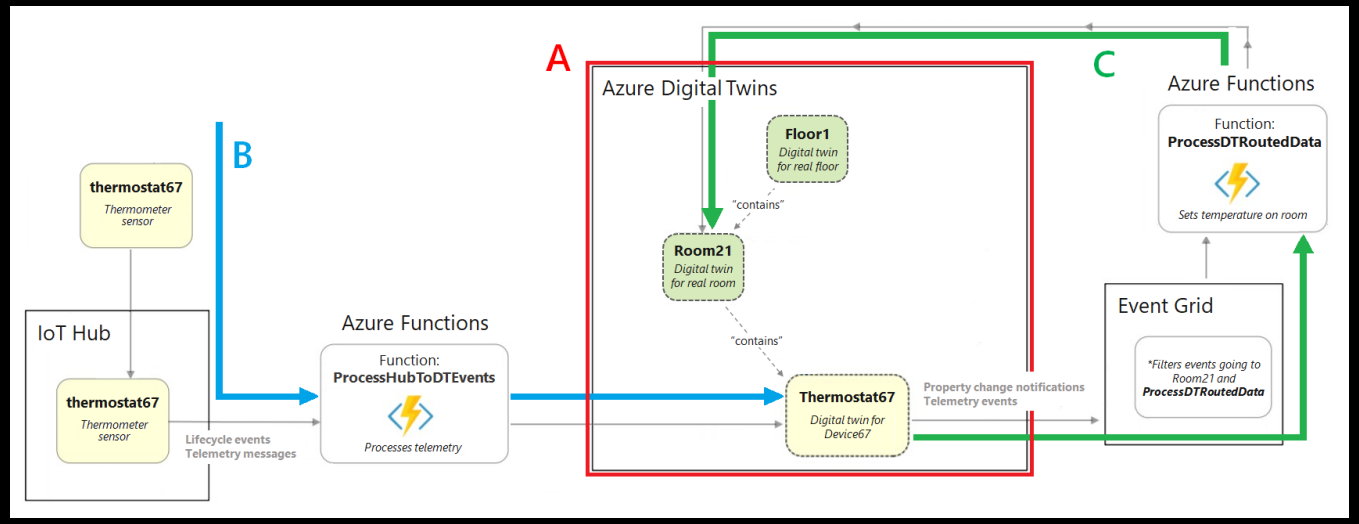
Azure Digital Twins – ready to use „environment”, fast “demo”

Reference diagrams

Twins definition

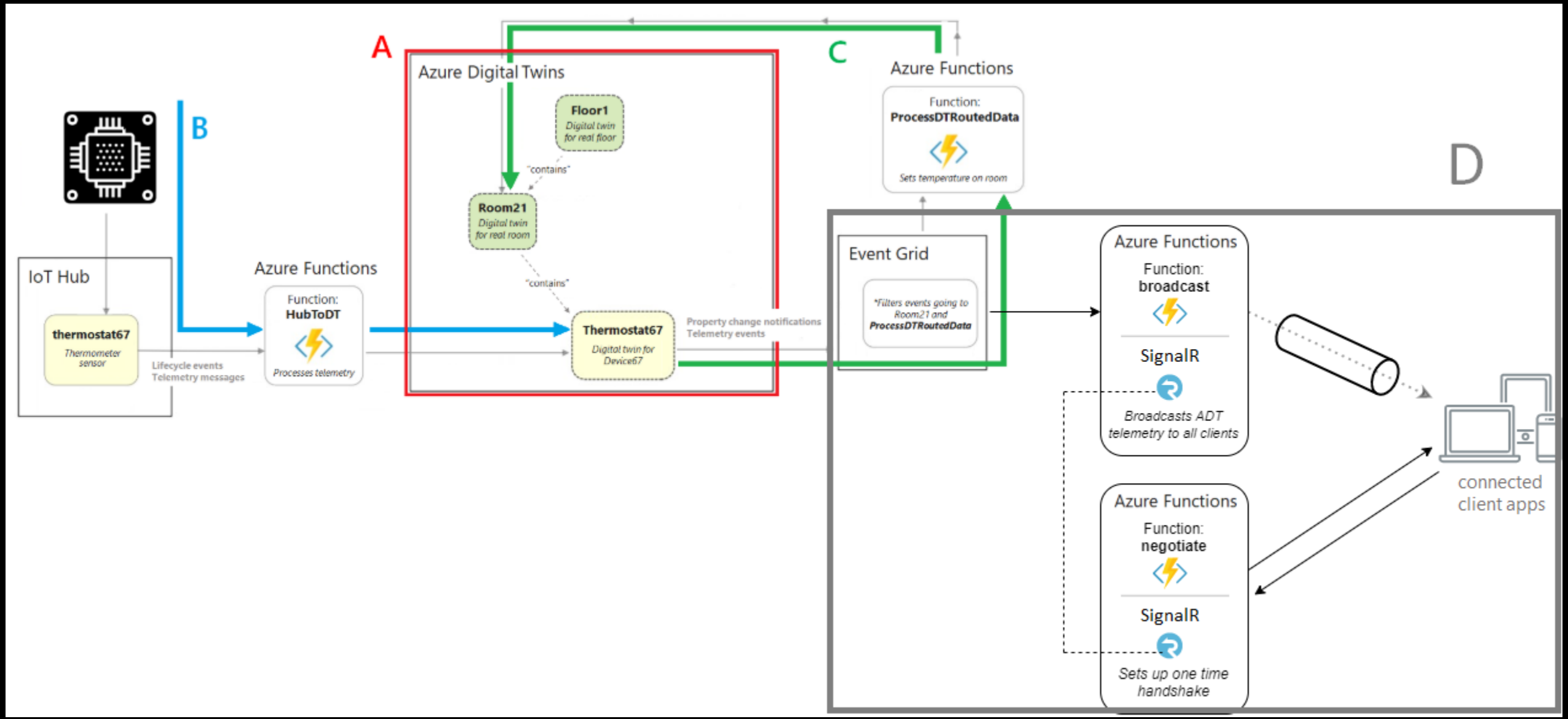


Flows



A – setup ADT
B – flow from IoT to DT (real life)
C – update ADT based on internal relations (between twins)

(And real-time client update)



Important

Device Twin (IoT Hub)

JSON documents that store device state information, including metadata, configurations, and conditions

- Virtual “view” for SINGLE PHYSICAL device
- Another explanation: Mapping physical state to the virtual object

Scenarios:

- Query object (devices) which have certain values. WITHOUT communicating with physical device
- Like: Last “firmware” version to get devices that need to be updated

Azure Digital Twin

Model of connected device**s**, ecosystem.

- DTDL Ontology: components, properties, …
- Unified view of the current and desired state of the property.
- Synchronization with Real World
- (May contain data from many devices twin)

Scenarios:

- A tool for **observing** and **analyzing** the large, complex, connected IoT environment
- A **modeling and simulation** tool to check “what if”

Ontology (in information science, not philosophy !)

Ontology (information science)

From Wikipedia, the free encyclopedia

In computer science and information science, an **ontology** encompasses a representation, formal naming, and definition of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains of discourse. More simply, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject.

Every academic discipline or field creates ontologies to limit complexity and organize data into information and knowledge. Each uses ontological assumptions to frame explicit theories, research and applications. New ontologies may improve problem solving within that domain. Translating research papers within every field is a problem made easier when experts from different countries maintain a controlled vocabulary of jargon between each of their languages.[1]

For instance, the definition and ontology of economics is a primary concern in Marxist economics,[2] but also in other subfields of economics.[3] An example of economics relying on information science occurs in cases where a simulation or model is intended to enable economic decisions, such as determining what capital assets are at risk and by how much (see risk management).

What ontologies in both information science and philosophy have in common is the attempt to represent entities, ideas and events, with all their interdependent properties and relations, according to a system of categories. In both fields, there is considerable work on problems of ontology engineering (e.g., Quine and Kripke in philosophy, Sowa and Guarino in computer science),[4] and debates concerning to what extent normative ontology is possible (e.g., foundationalism and coherentism in philosophy, BFO and Cyc in artificial intelligence).

Information science

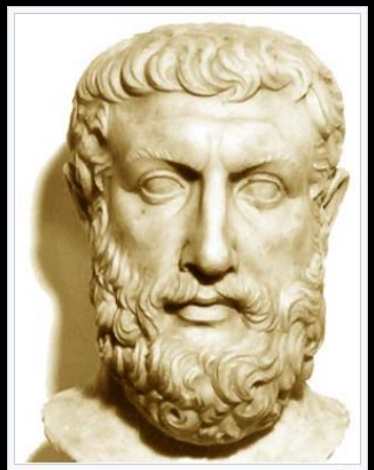
General aspects

Access · Architecture · Behavior · Management · Retrieval · Seeking · Society · Knowledge organization · **Ontology** · Philosophy · Science and technology studies · Taxonomy

Related fields and sub-fields

Bibliometrics · Categorization · Censorship · Classification · Computer data storage · Cultural studies · Data modeling · Informatics · Information technology · Intellectual freedom · Intellectual property · Library and information science · Memory · Preservation · Privacy · Quantum information science

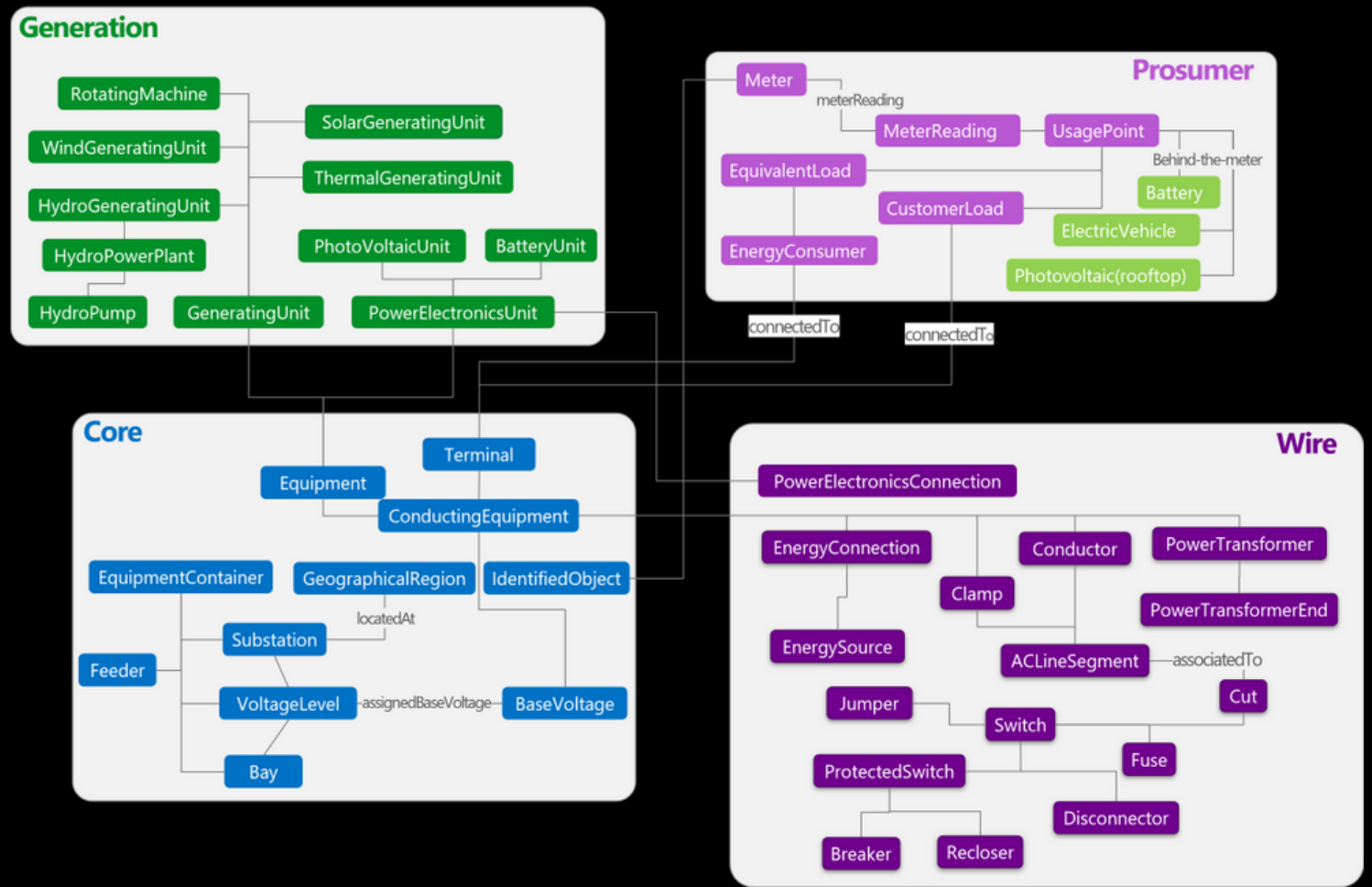
V · T · E



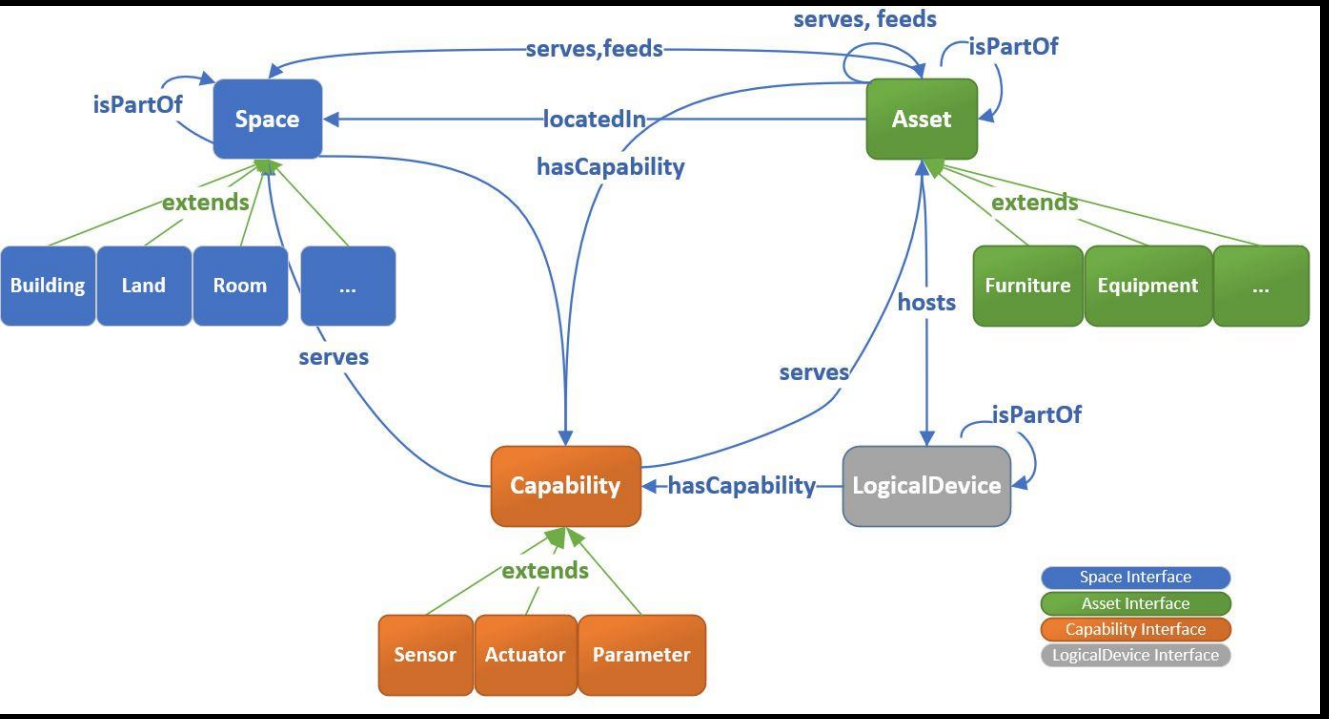
Parmenides was among the first to propose an ontological characterization of the fundamental nature of reality.

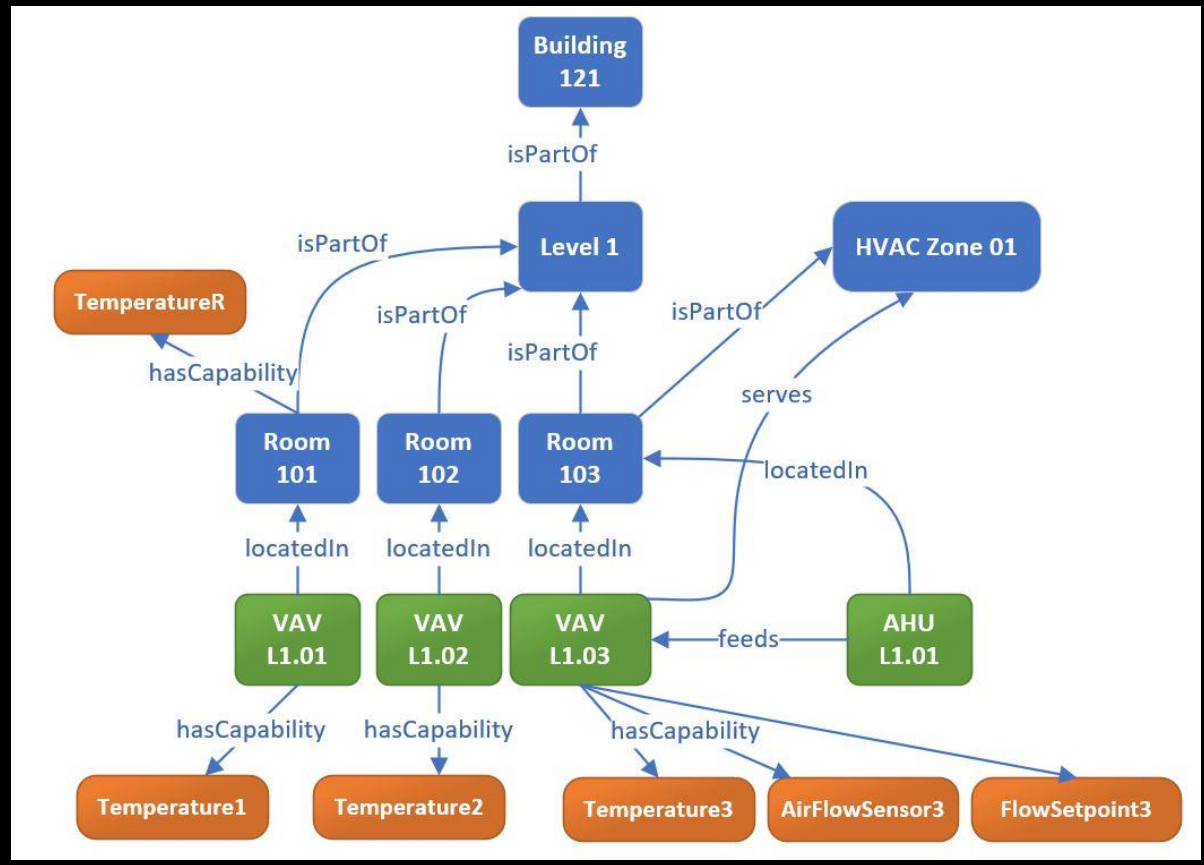
ADT Ontology for Energy Grid

https://github.com/Azure/opendigitaltwins-energygrid/

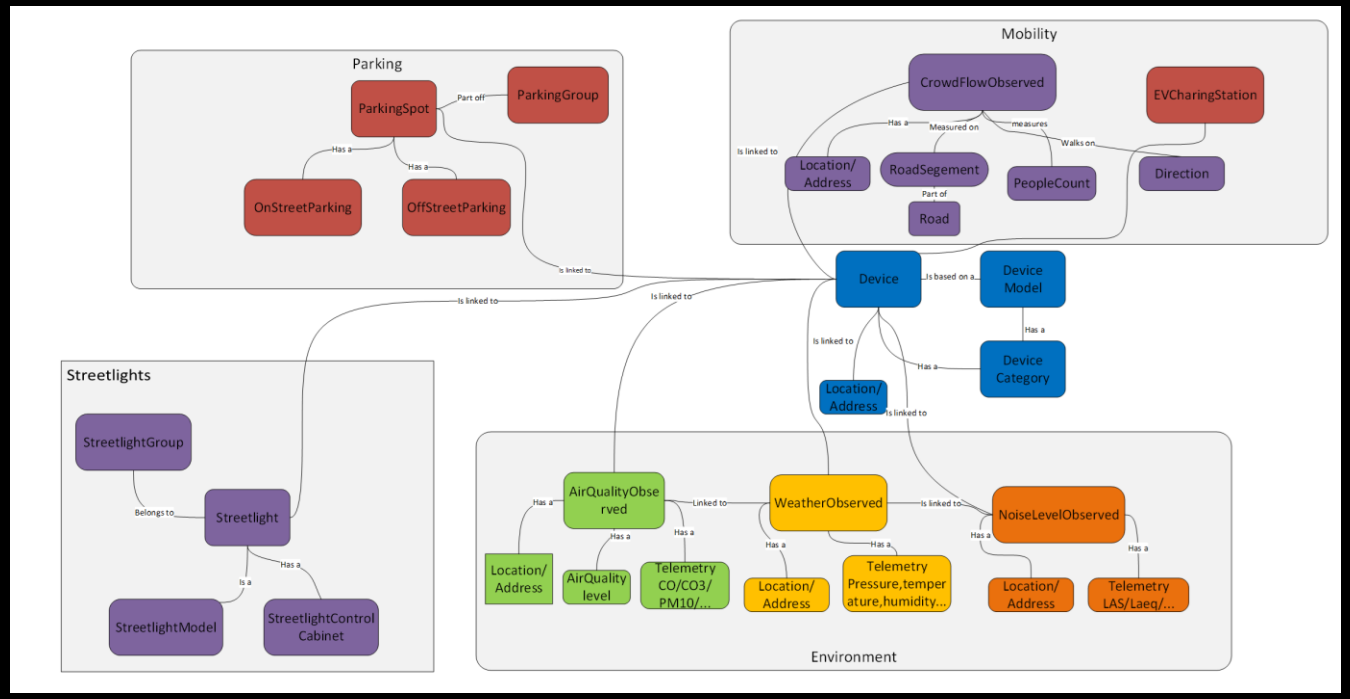


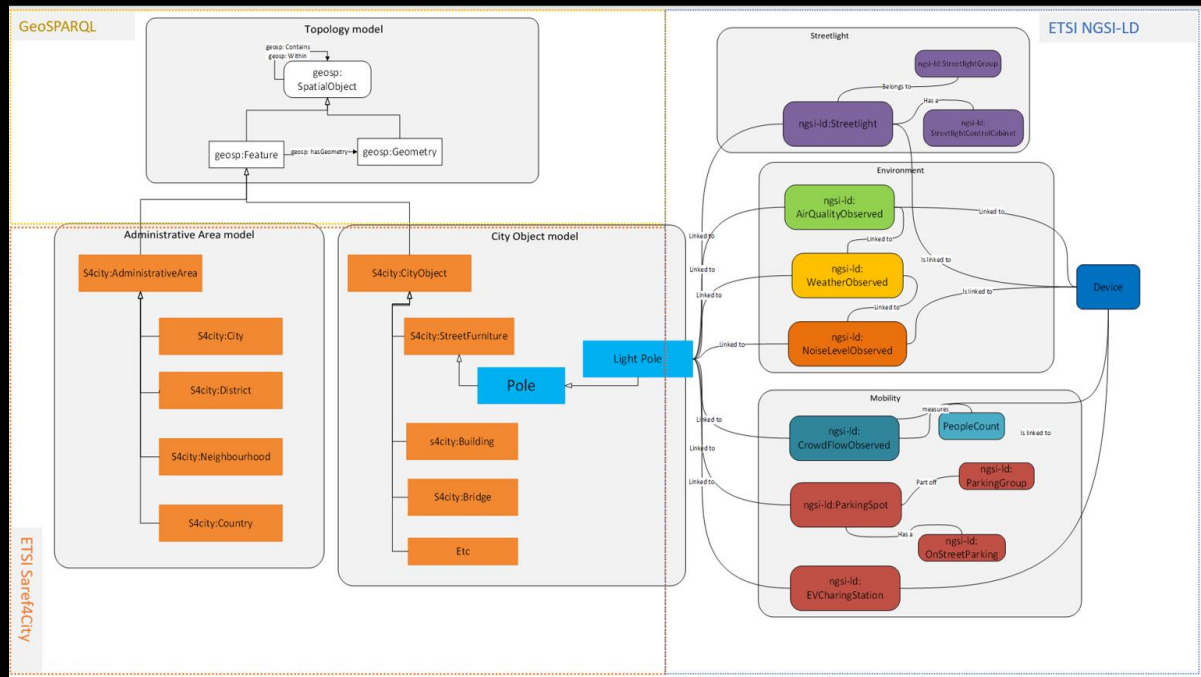
Smart Building



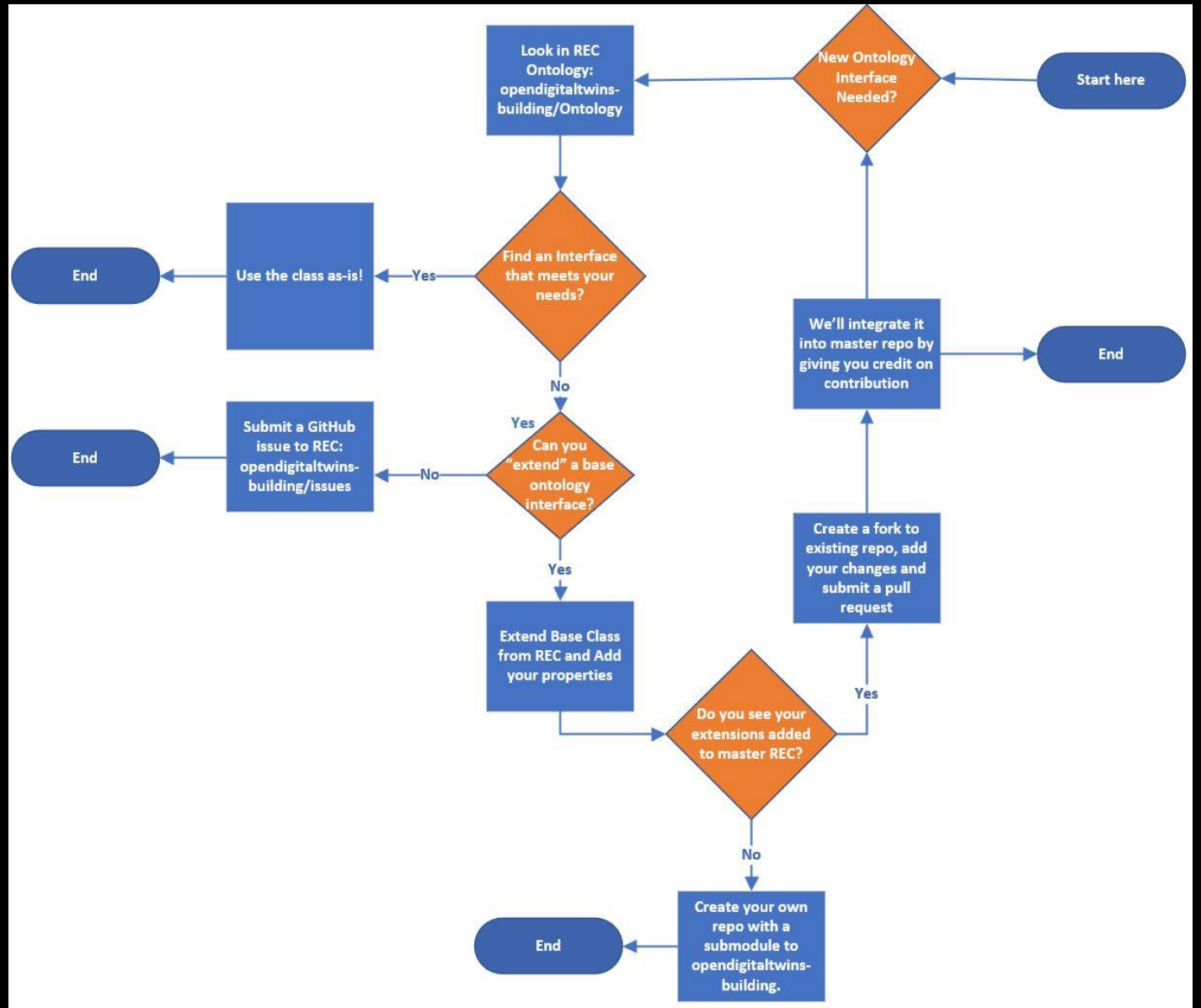


SmartCities



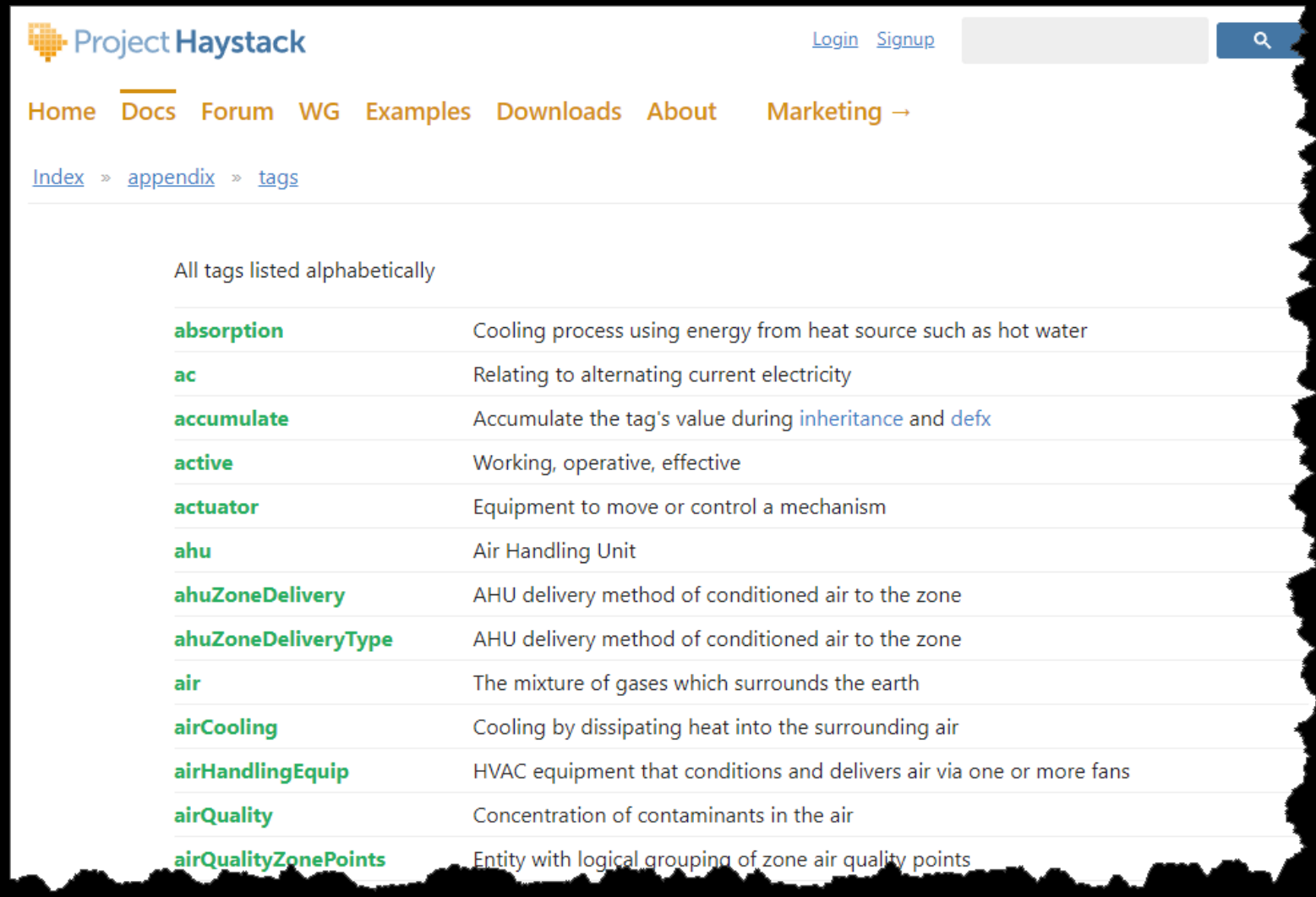


Extending Ontologies



Also: Tags for model – Haystack.org

You can use the concept of tags to further identify and categorize your digital twins. In particular, users may want to replicate tags from existing systems, such as **Haystack**



https://docs.microsoft.com/en-us/azure/digital-twins/how-to-use-tags

ADT REST API

Control Plane

- Management (names, digital twins instance etc)
- Set up Endpoints (Service Bus / Event Hub / Event Grid) + private endpoints

Data Plane

- Event Routes (which endpoints should be connected)
- Digital Twins Model and Twins Management (components, telemetry, relationship (graph) updates)
- Query for Twins (with relations)

Queries - examples

```
SELECT  * FROM DIGITALTWINS T WHERE T.firmw = '1.1' AND T.$dtId in ['123', '456'] AND T.Temperature = 70

SELECT * FROM DIGITALTWINS WHERE IS_DEFINED(Location) AND IS_DEFINED(tags.red) AND IS_NUMBER(T.Temp)
SELECT * FROM DIGITALTWINS DT WHERE IS_OF_MODEL(DT, 'dtmi:example:thing;1', exact)

SELECT target FROM DIGITALTWINS source JOIN target RELATED source.feeds WHERE source.$dtId = ‘src-twin'
SELECT source FROM DIGITALTWINS source JOIN target RELATED source.feeds WHERE target.$dtId = ‘trgt-twin'
SELECT T, SBT, R FROM DIGITALTWINS T JOIN SBT RELATED T.servicedBy R WHERE T.$dtId = 'ABC' AND
R.reportedCondition = 'clean'

SELECT COUNT() FROM DIGITALTWINS Room... AND Room.$dtId IN ['room1', 'room2']
SELECT Consumer.name AS consumerName, Edge.prop1 AS first, Edge.prop2 AS second, Factory.area AS
factoryArea FROM DIGITALTWINS Factory JOIN Consumer RELATED Factory.customer Edge WHERE Factory.$dtId = 'A'
SELECT Room FROM DIGITALTWINS Floor JOIN Room RELATED Floor.contains WHERE Floor.$dtId IN
['floor1','floor2'] AND Room. Temperature > 72 AND IS_OF_MODEL(Room, 'dtmi:com:contoso:Room;1')
SELECT device FROM DIGITALTWINS space JOIN device RELATED space.has WHERE space.$dtid = 'Room 123' AND
device.$metadata.model = 'dtmi:contoso:com:DigitalTwins:MxChip:3' AND has.role = 'Operator'
SELECT Room FROM DIGITALTWINS Room JOIN Thermostat RELATED Room.Contains WHERE Thermostat.$dtId = 'id1'
SELECT Room FROM DIGITALTWINS Floor JOIN Room RELATED Floor.Contains WHERE Floor.$dtId = 'floor11' AND
IS_OF_MODEL(Room, 'dtmi:contoso:com:DigitalTwins:Room;1')
```

ADT Pricing structure

Query payment “model”

- Azure Digital Twins Query Unit (QU) abstracts away the system resources like CPU, IOPS, and memory
- Allowing you to track usage in Query Units instead.
- Affected by:
  - The complexity of the query
  - The size of the result set (so a query returning 10 results will consume more QUs than a query of similar complexity that returns just one result)

(Concept similar to Cosmos DB pricing model)

	Price
Message	**$1** per million messages
Operation	**$2.50** per million operations
Query unit	**$0.50** per million query units

Demo

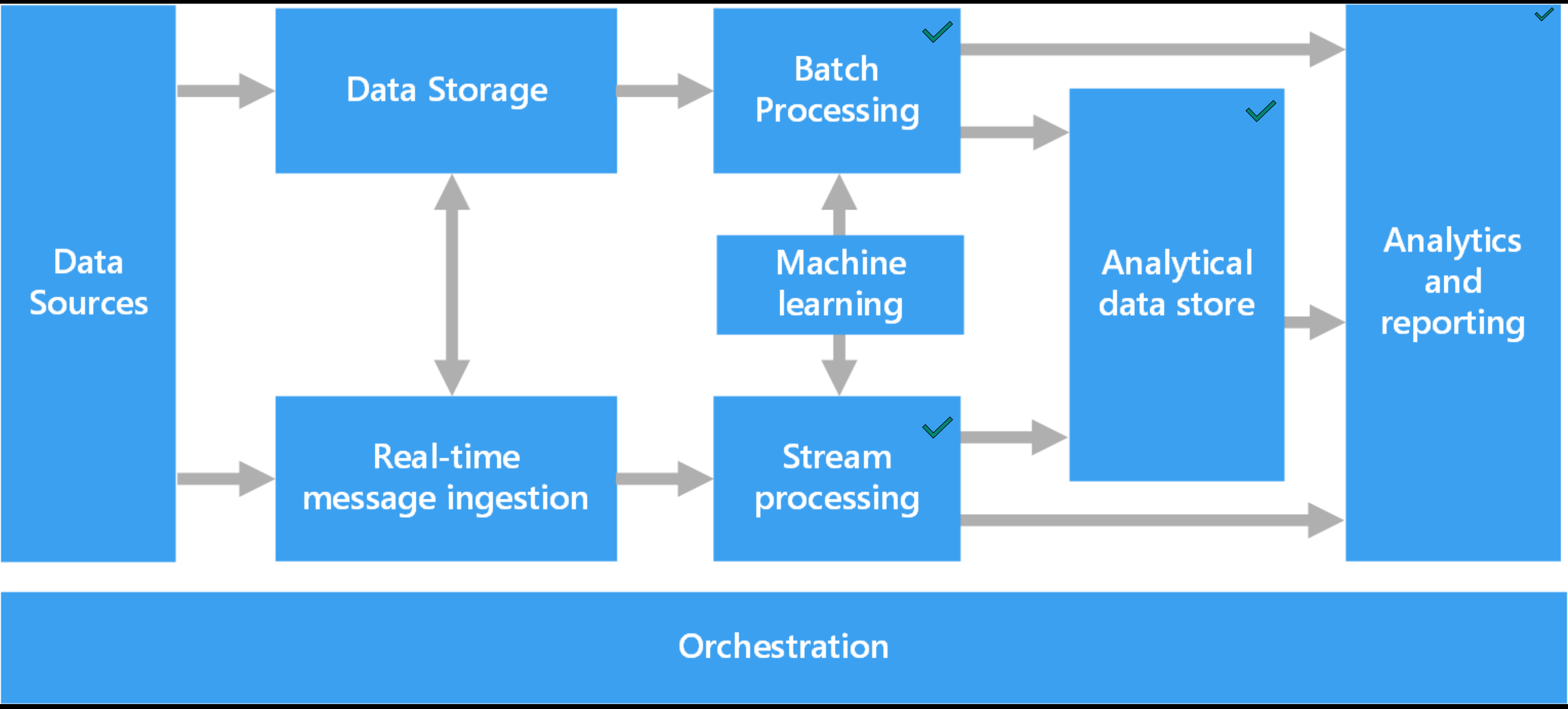
Simple ADT .NET Client

Crash course through selected Azure Services

- Azure Function: Logic and Data processing
- Logic Apps
- Event Grid / Service Bus
- SignalR
- Azure Maps (exterior and INTERIOR – custom maps!)
- (and Event Hub)

ADT and DATA

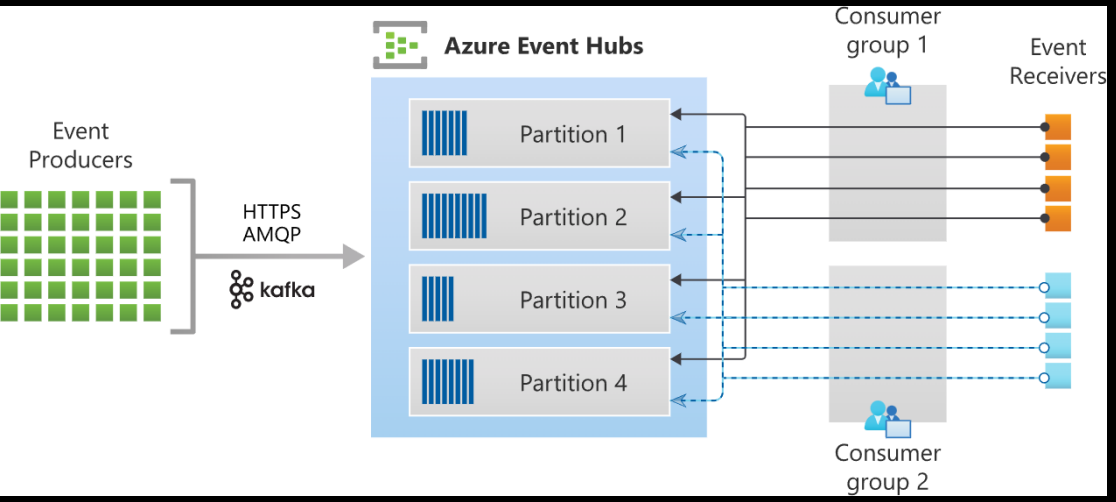
Generic Lambda Architecture

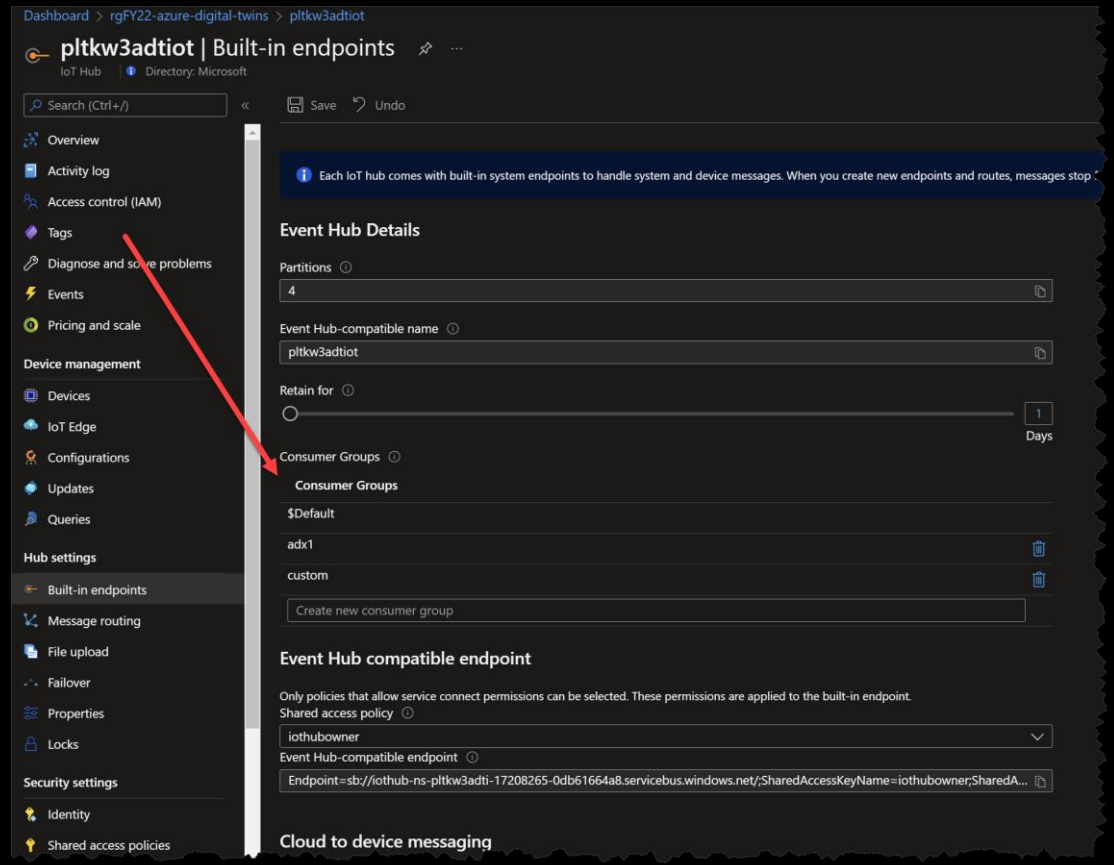


Crash course through selected DATA Azure Services

- Custom code, “reader” - discussion
- Azure Blob / Azure Data Lake v2
- Stream Insight
- Azure Data Explorer and Synapse
- (Everything in VM)

How to get telemetry in SCALE

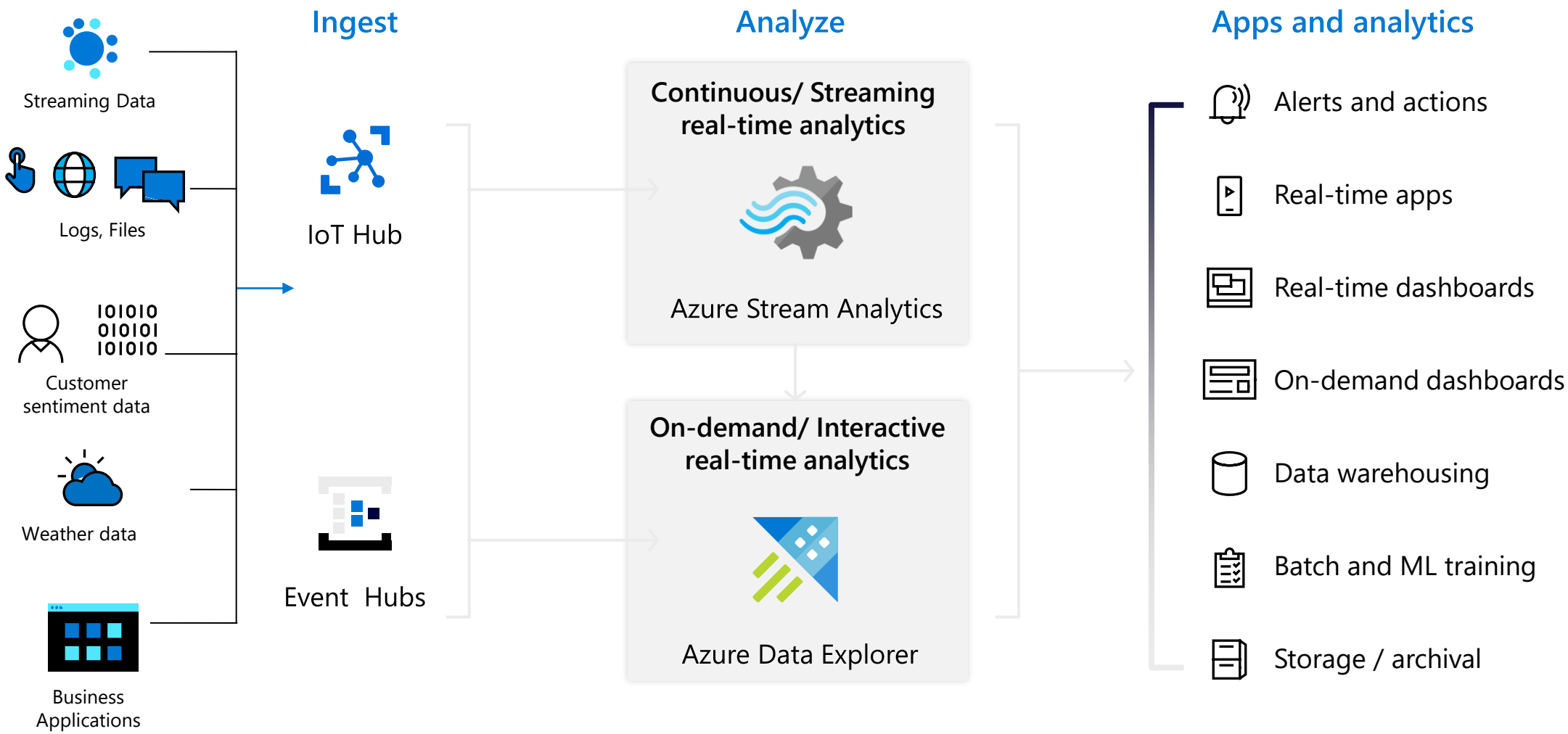




Therefore: Consumer Group per “reader” (or type of processing)

And – IoT Hub is using Event Hub

Azure powered pattern for real-time analytics

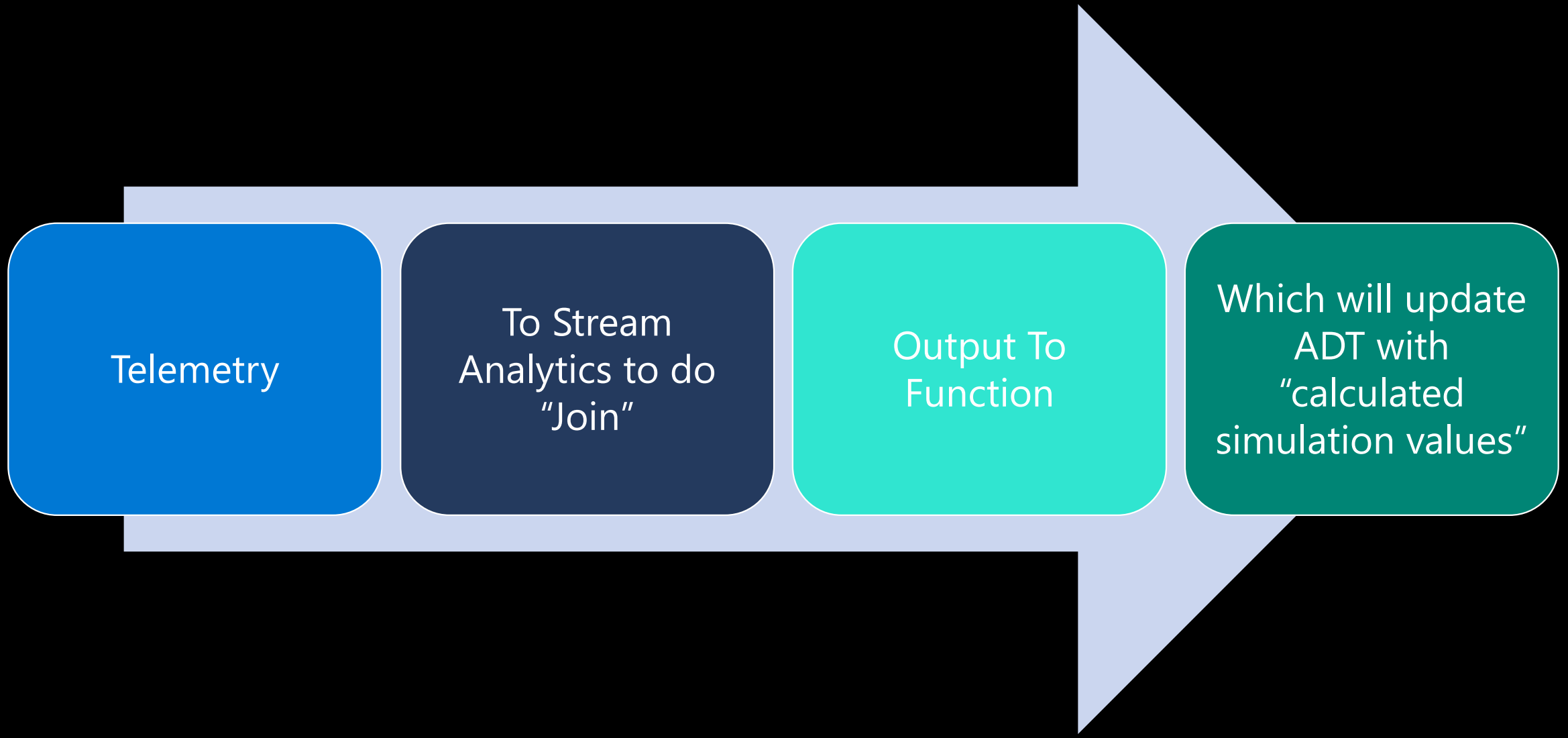


Demo

Stream Analytics (for copying & demo &)
https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-stream-analytics-query-patterns

Local VS Code Emulator

Demo Flow (start with simple API)



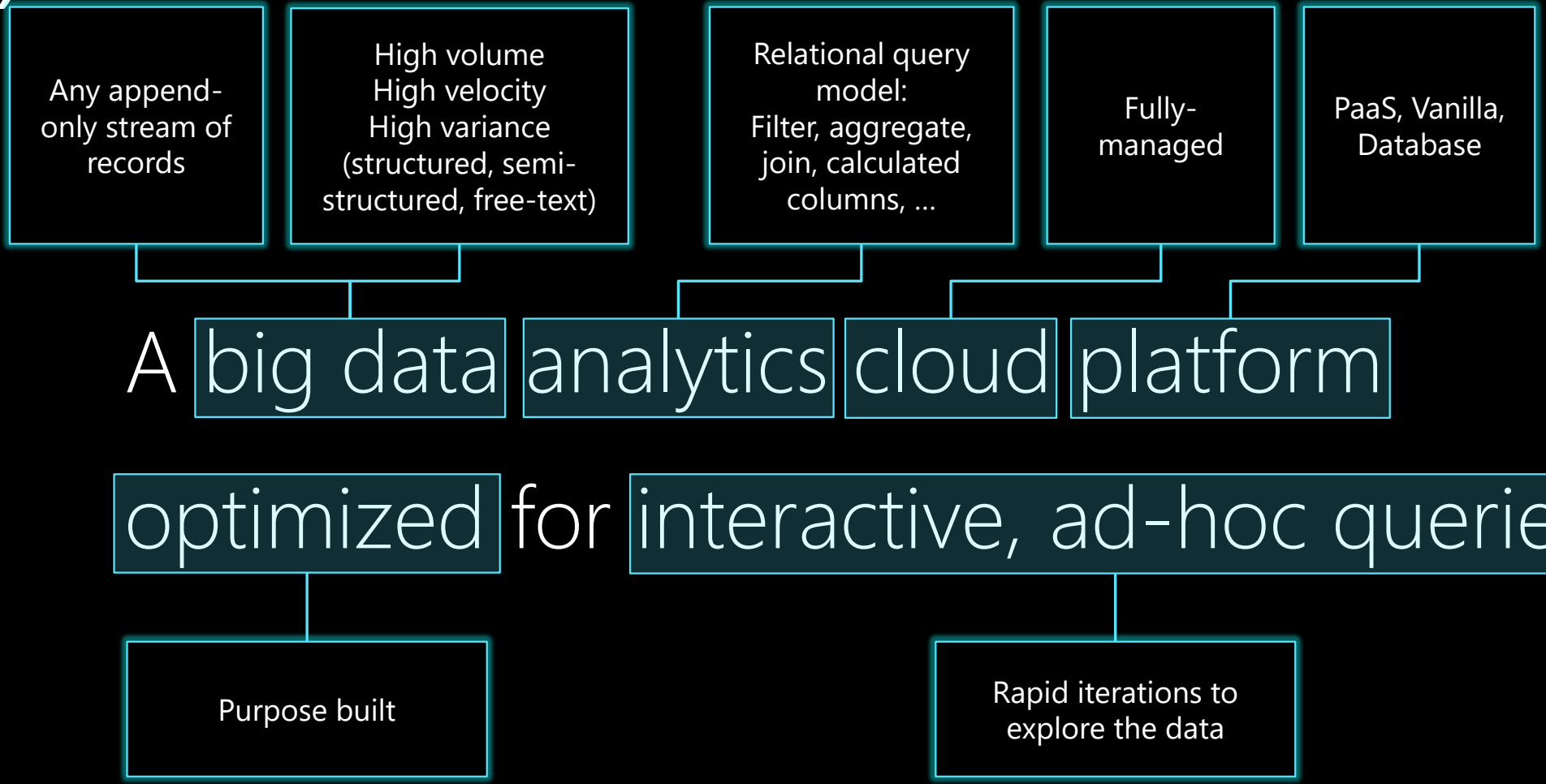
Main Samples:
(https://github.com/Azure/azure-sdk-for-net/tree/main/sdk/digitaltwins)
https://docs.microsoft.com/en-us/azure/digital-twins/tutorial-end-to-end
https://docs.microsoft.com/en-us/azure/digital-twins/how-to-integrate-azure-signalr
https://docs.microsoft.com/en-us/azure/digital-twins/how-to-send-twin-to-twin-events

Demo

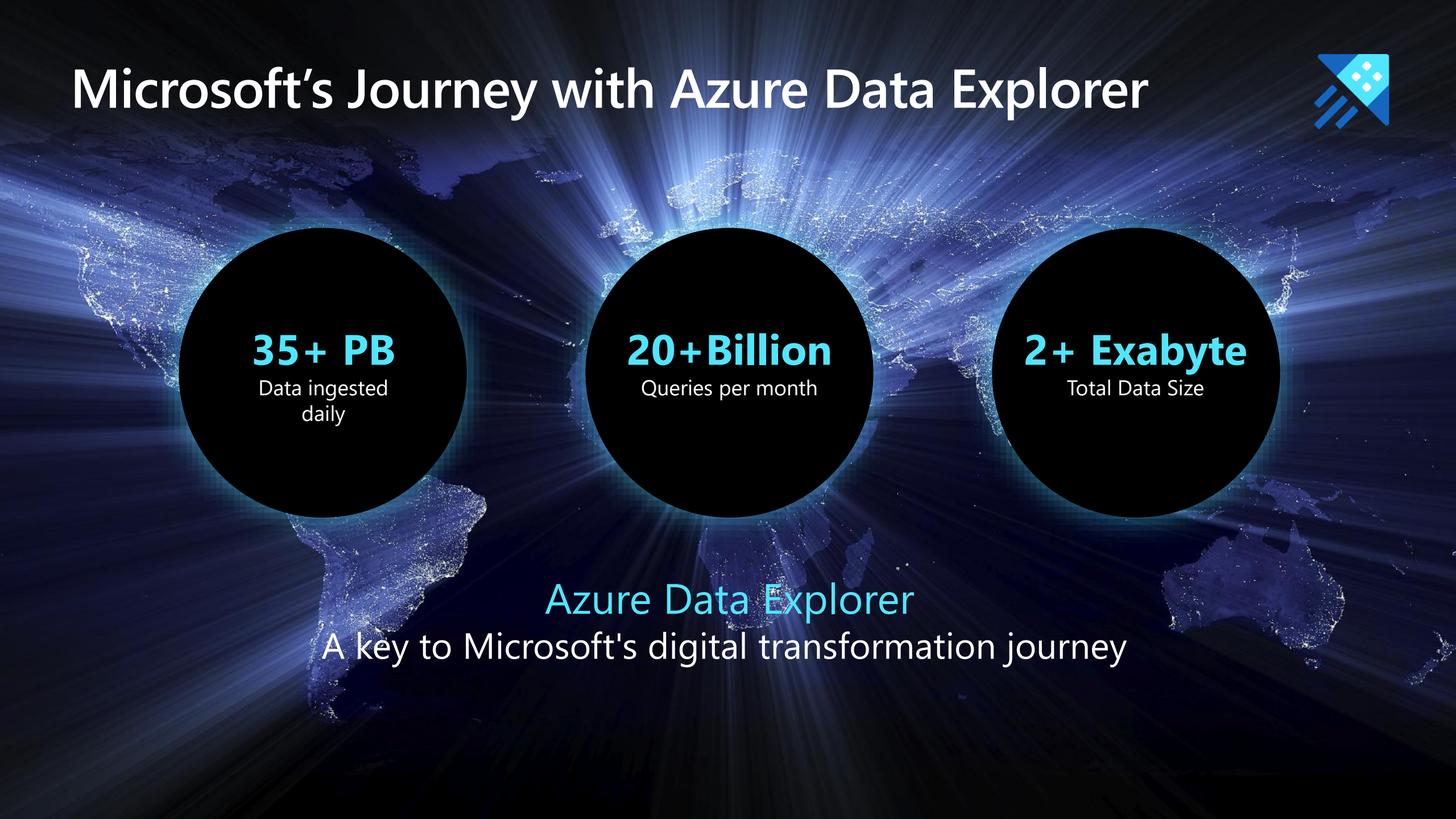
Not so simple ADT use cases

Also: Official examples from SDK:
D:\TS_EXP2022\azure-sdk-for-net\sdk\digitaltwins\Azure.DigitalTwins.Core\Azure.DigitalTwins.Core.sln
(align tags and .net version)

Introducing Azure Data Explorer (ADX or Synapse ADX)!



Microsoft's Journey with Azure Data Explorer



Azure Data Explorer

A key to Microsoft's digital transformation journey

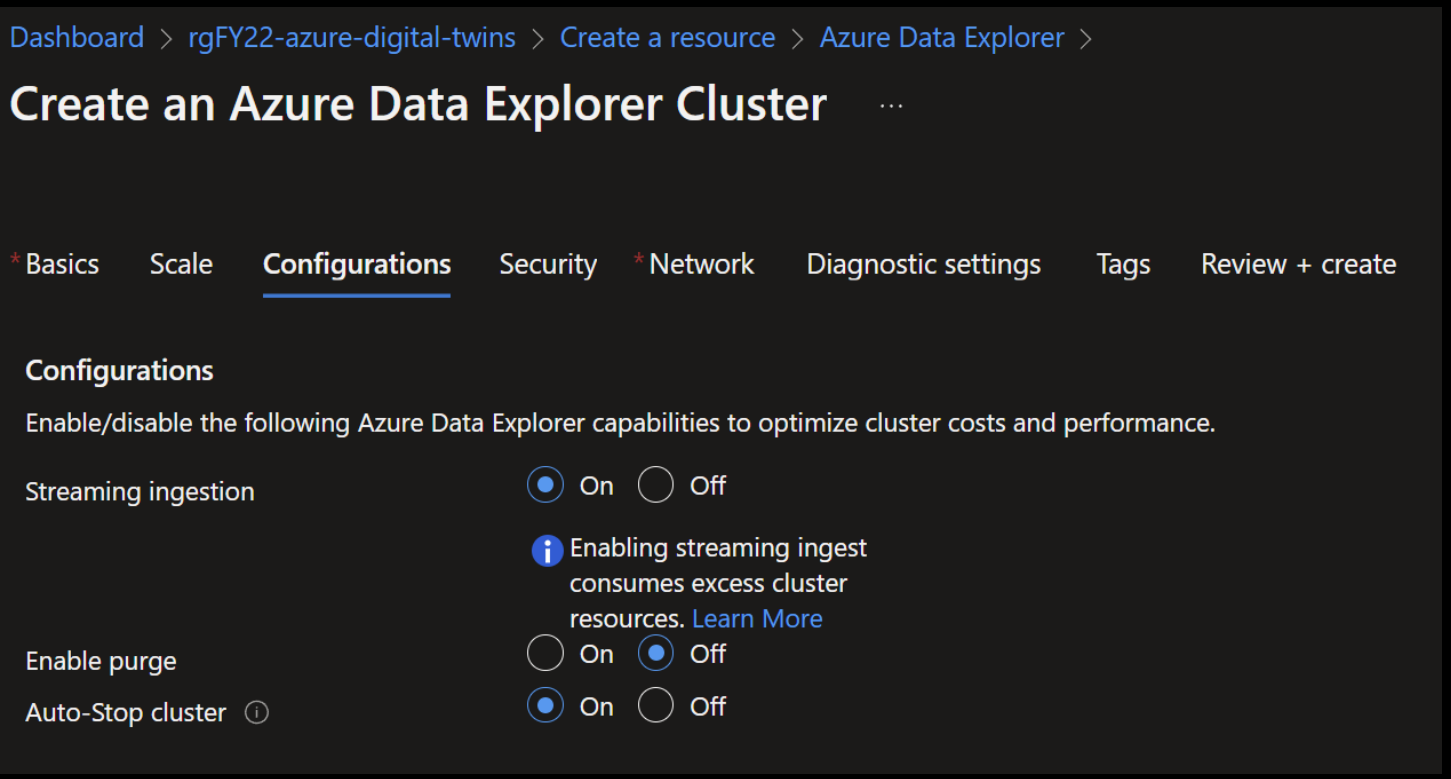
Roles

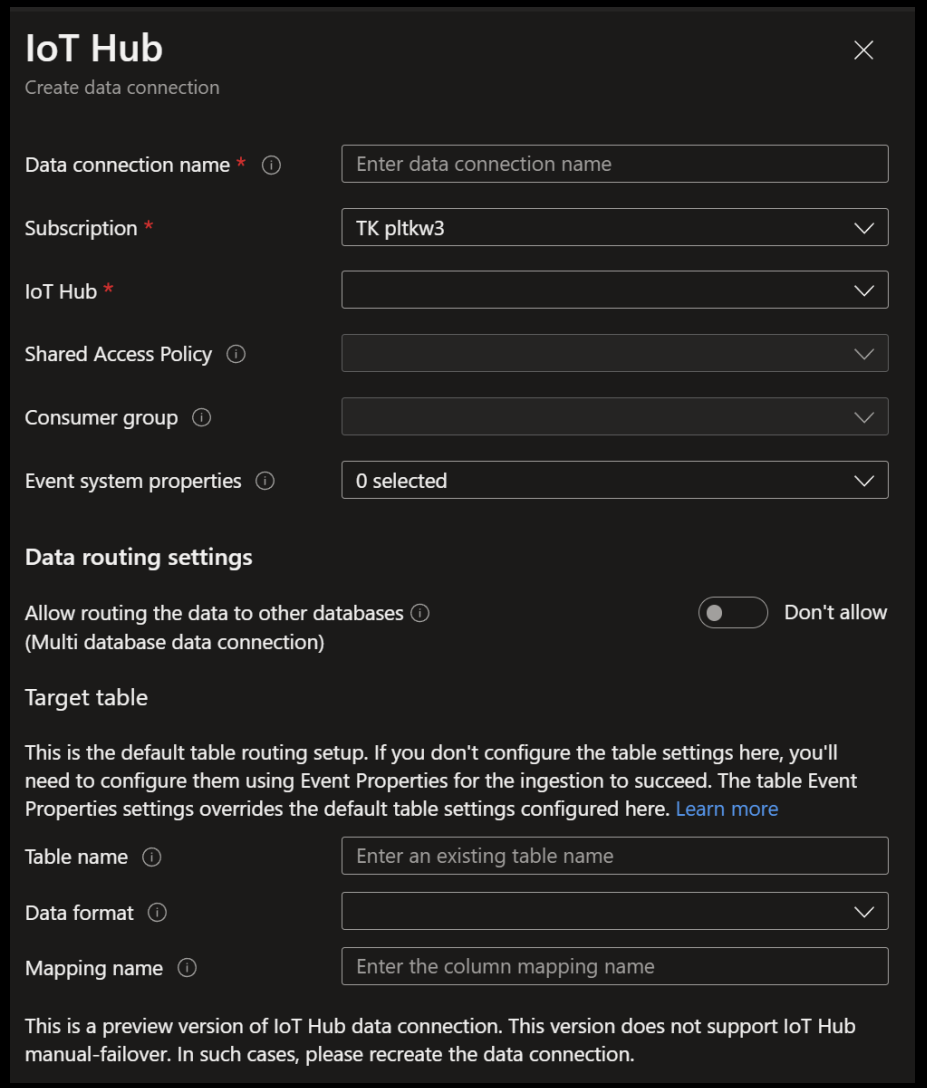
Telemetry: IoT Hub

Model: Azure Digital Twins & PnP

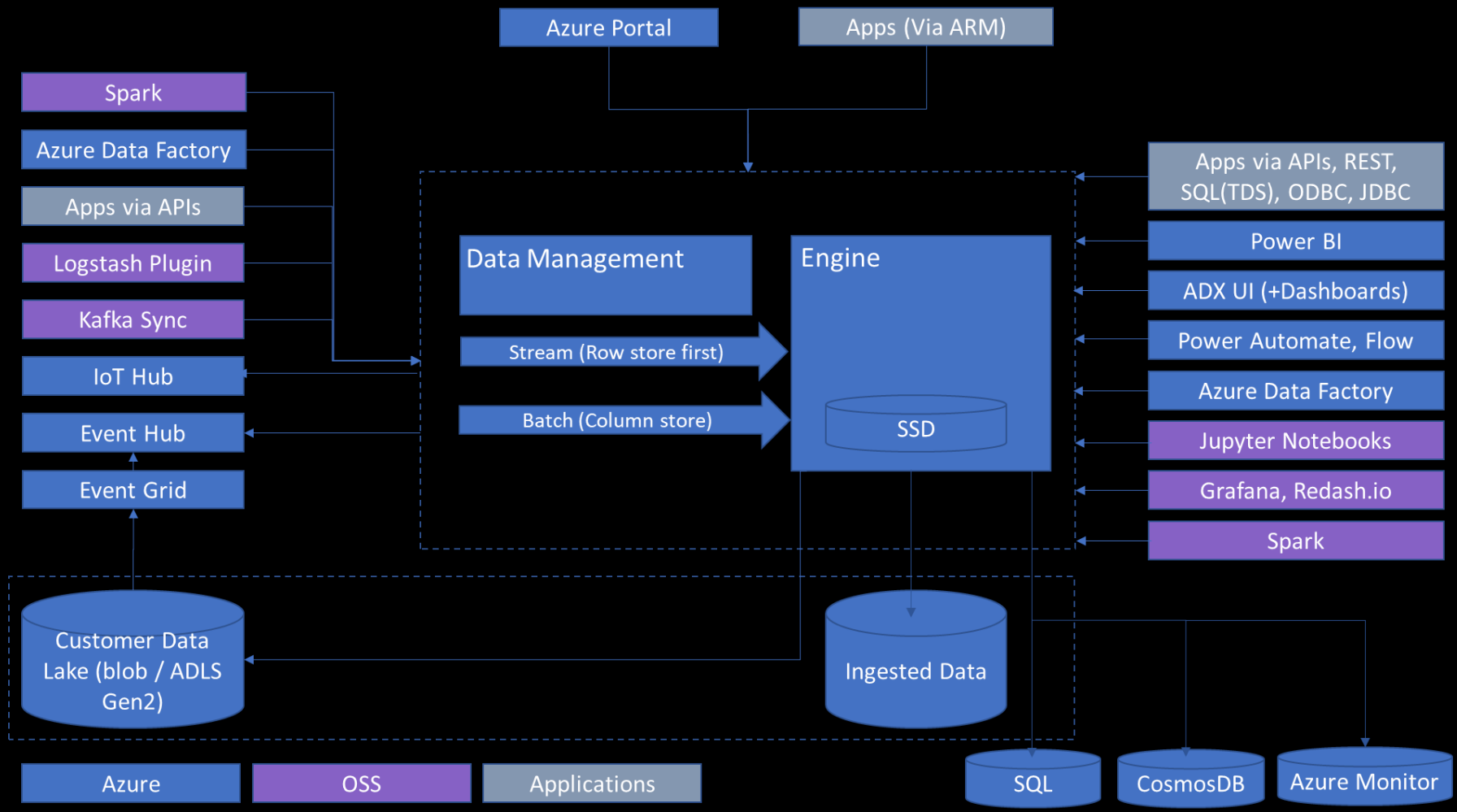
Ingest, Links, query, scale: ADX

Create a cluster with Streaming (to get telemetry)





Oneslider - Azure Data Explorer



Comprehensive Strength

- Metrics and time-series data
- Text search and text analytics
- Multi-dimensional/relational analysis

Analytics Query language

- Simple and powerful
- Publicly available
- Data Exploration
- Rich relational query language
- Full text Search
- ML Extensibility

High performance over large data sets

- Scale out in hardware
- Scale out across geos
- Granular resource utilization Control
- Cross geo queries

Data Ingestion and Management

- Low Latency ingestion
- Schema management
- Compression and indexing
- Retention
- Hot/cold resource allocation

Demo – ADX (and how to ingest from IoT)

https://dataexplorer.azure.com/clusters/fy22adxiot.westeurope/databases/tk01?tenant=72f988bf-86f1-41af-91ab-2d7cd011db47&login_hint=tkopacz%40microsoft.com

https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/

Summary

Summary

IoT Hub PnP

We need standards & metadata per “type” of sensor, telemetry, commands, …

Azure Digital Twins

We need graph for complex IoT Ecosystem

The rest of Azure

We need flexible ways to do model processing/simulation/tracking/analysis/…

Thank you,
ANY questions?

Tomasz Kopacz
tkopacz@microsoft.com