Azure IOT Pi Day

3/14/19

Speaker 1:
Speaker 2:

https://github.com/Azure/IoT-Pi-Day - Please do Laptop Setup NOW

Twitter handle: #microsoftpiday

Agenda

What is IoT?

Azure Components Overview

Lab 1 - Getting Data from a Pi

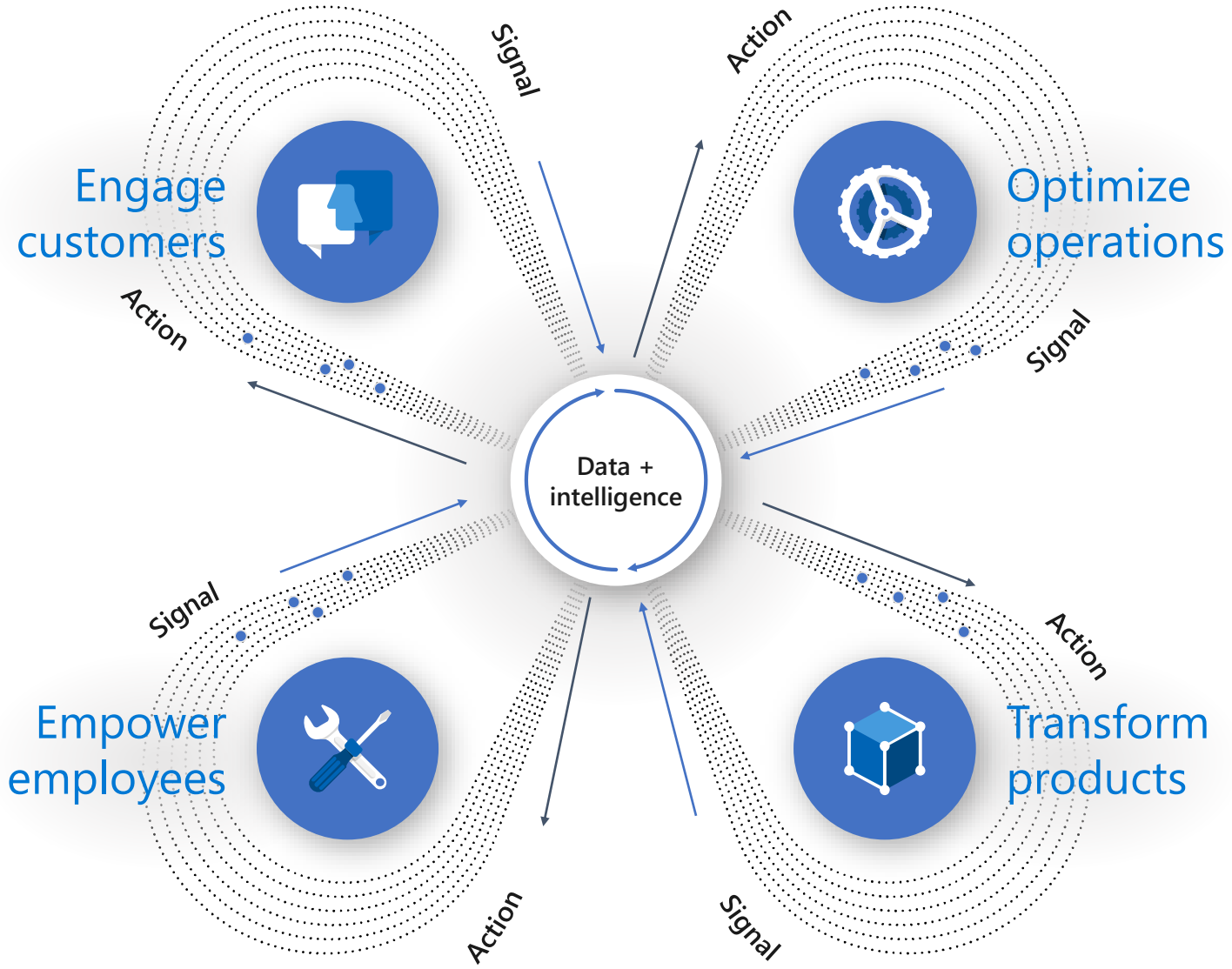
Real-Life Scenario Overview

Lab 2 – End-to-End Solution

Wrap Up and More

Why Azure IOT?

https://www.youtube.com/watch?v=1jh1qJu9_Zk
https://azure.microsoft.com/en-us/resources/videos/azure-iot-customer-stories/



80B

Connected “things” by 2025 generating 180ZB of data

$130B

New monetization avenues due to IoT-related services

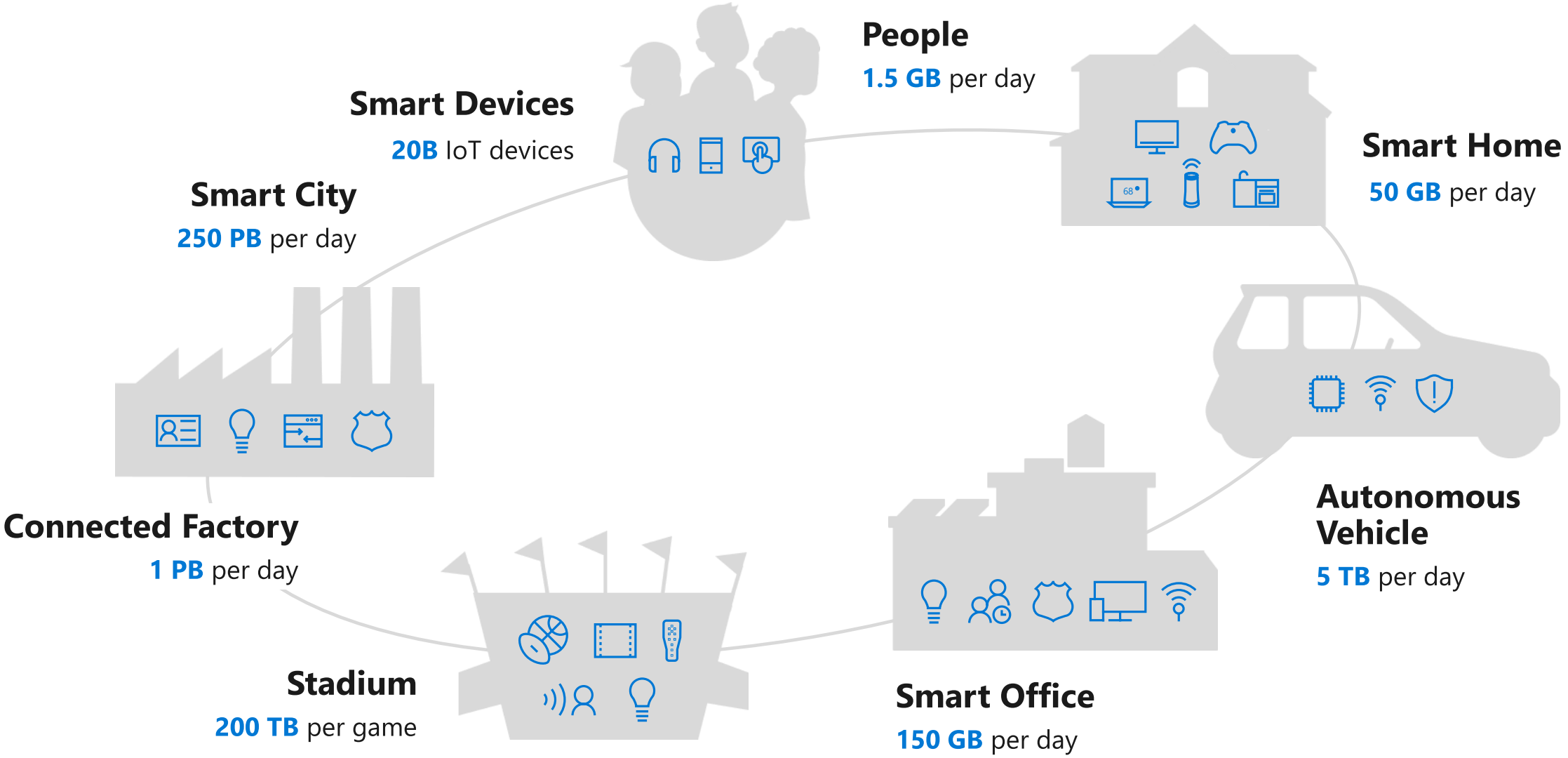
80%

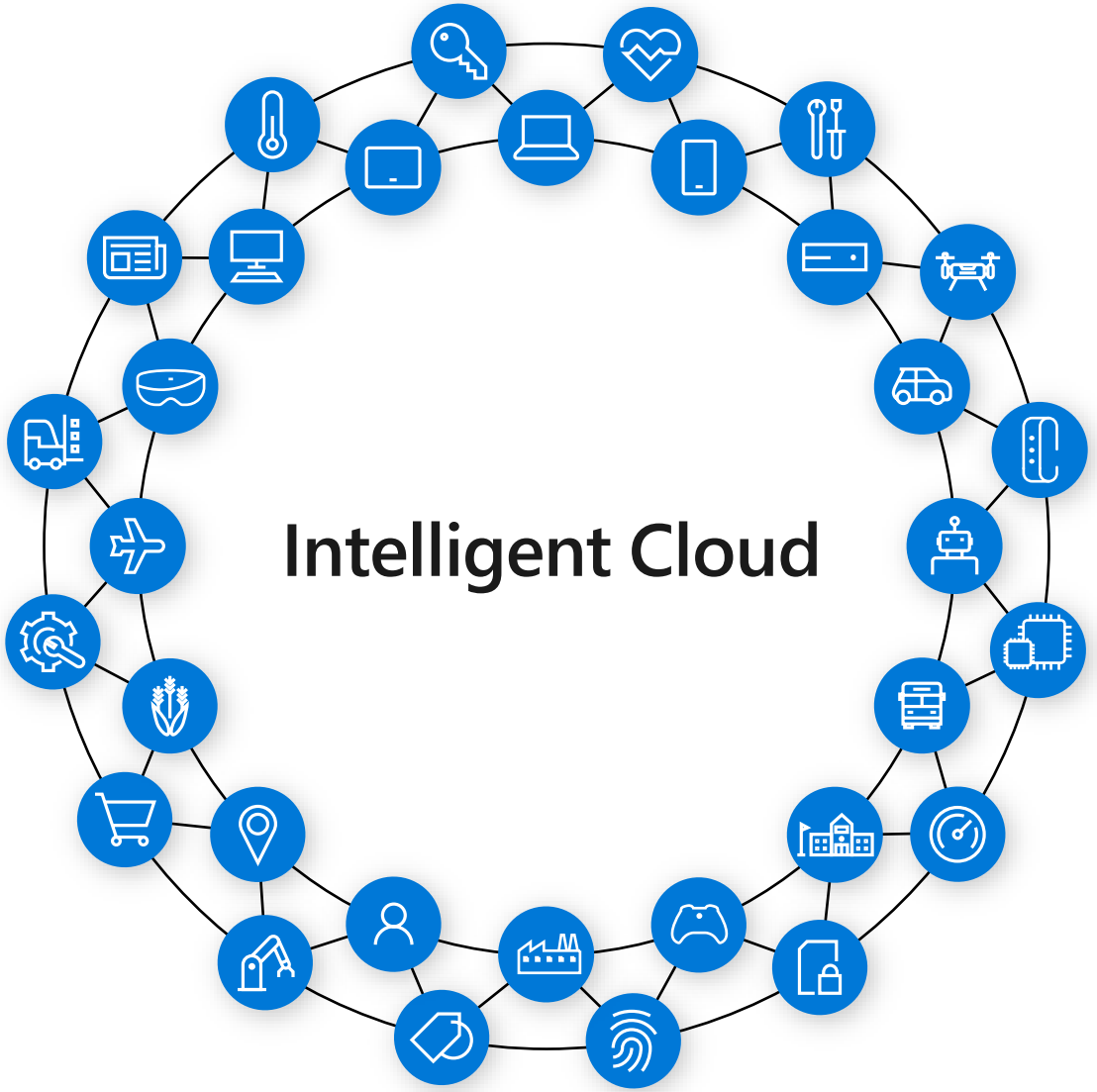
Companies that increased revenue as a result of IoT implementation

$100M

Average increase in operating income (avg. 8%) among the most digitally transformed enterprises

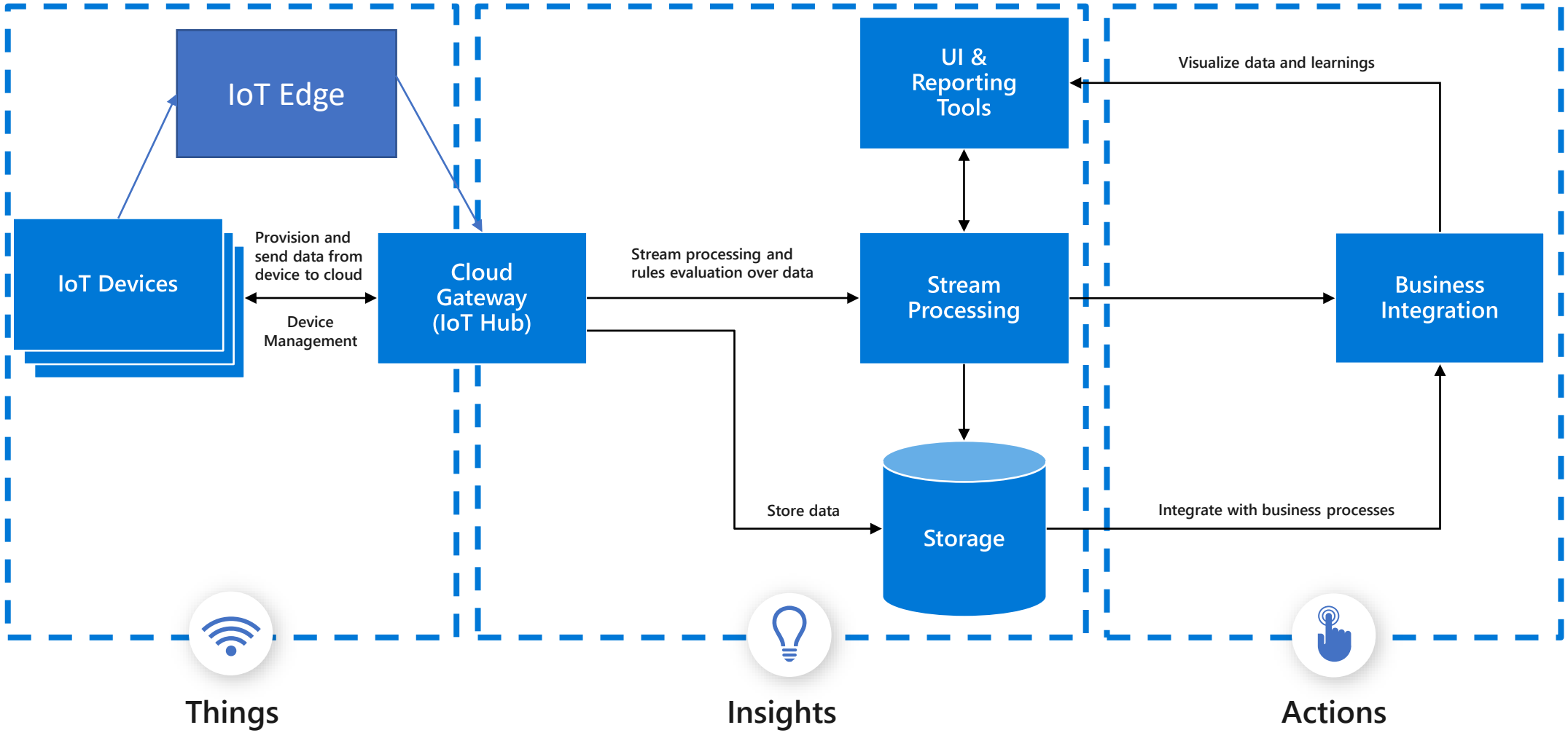
By 2020…


People
1.5 GB per day
Smart Devices
20B IoT devices
Smart City
250 PB per day
Smart Home
50 GB per day
Connected Factory
1 PB per day
Autonomous Vehicle
5 TB per day
Stadium
200 TB per game
Smart Office
150 GB per day

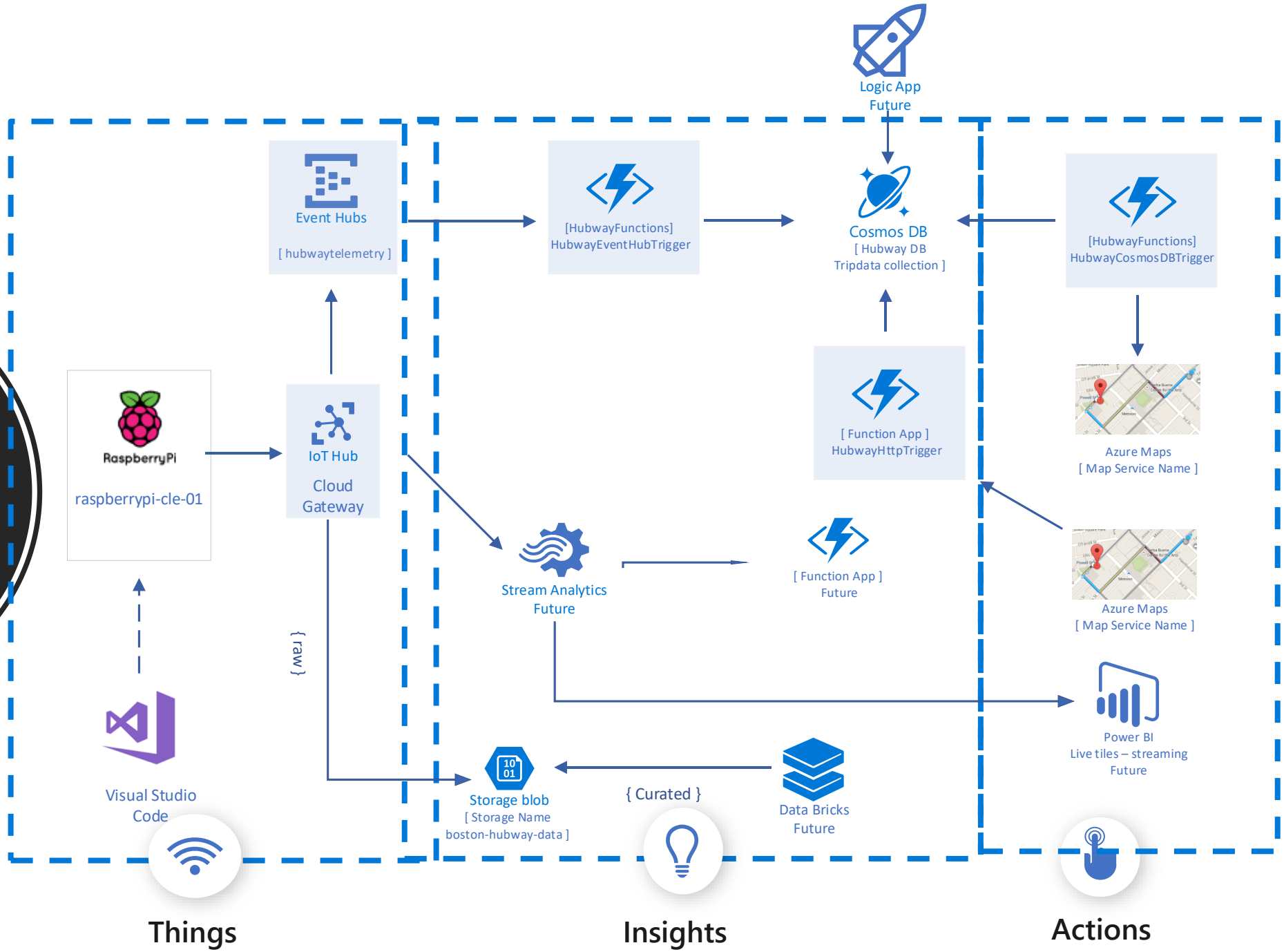

Intelligent Cloud
Intelligent Edge

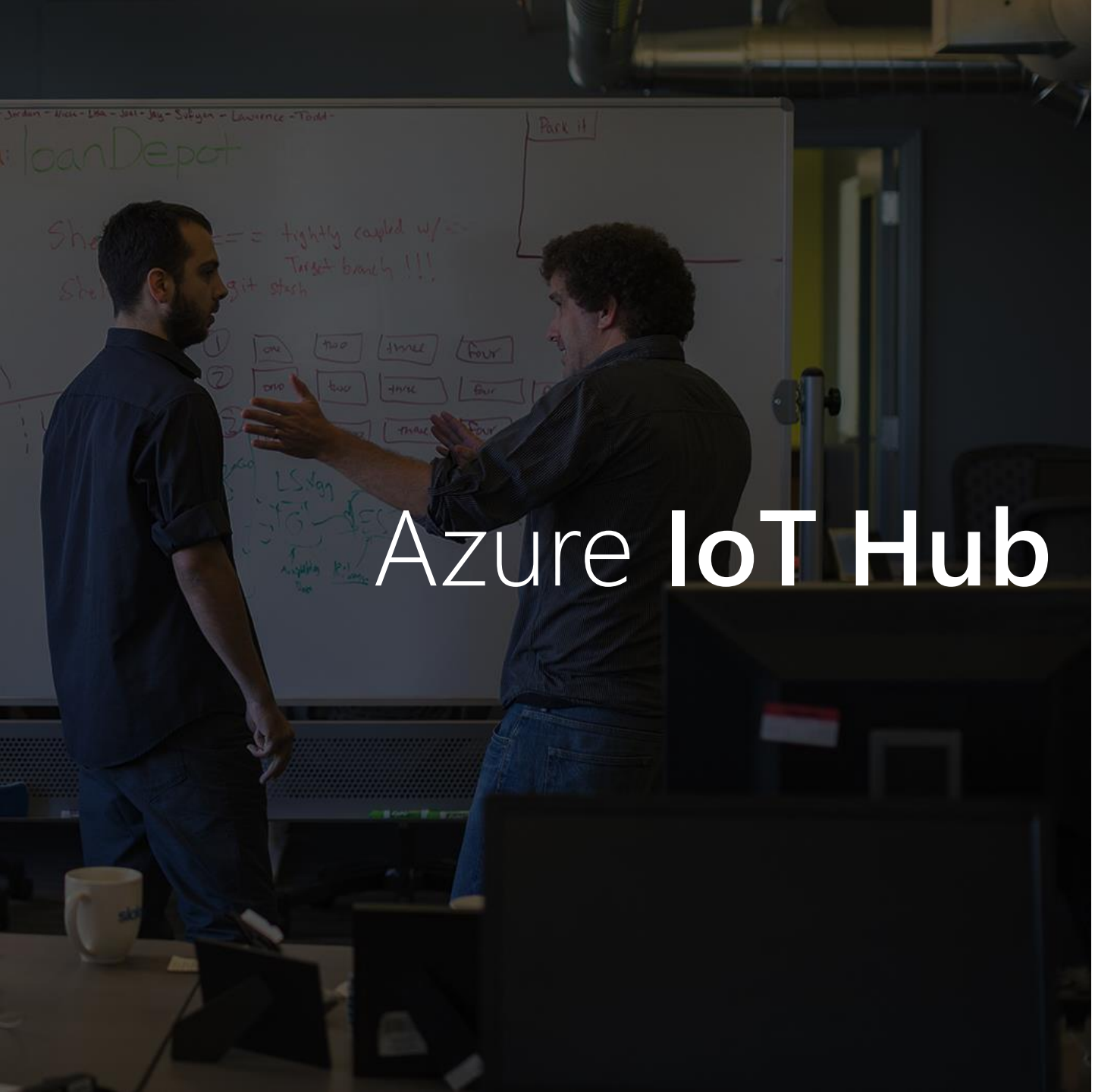
Azure IoT reference architecture

Core Subsystems



PiDay Reference Architecture





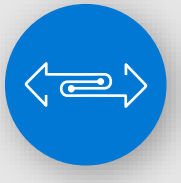
Azure **IoT Hub**

Things

- Establish bi-directional communication with billions of IoT devices
- Enhance security with per device authentication
- Provision devices at scale w/ IoT Hub Device Provisioning Service
- Manage devices at scale with device management
- Multi-language and open source SDKs

Azure IoT Hub

Things



Bi-directional communication

Millions of Devices
Multi-language, open source SDKs
HTTPS/AMQPS/MQTTS
Send Telemetry
Receive Commands
Device Management
Device Twins
Queries & Jobs

Enterprise scale & integration

Billions of messages
Scale up and down
Declarative Message Routes
File Upload
WebSockets & Multiplexing
Azure Monitor
Azure Resource Health
Configuration Management

End-to-end security

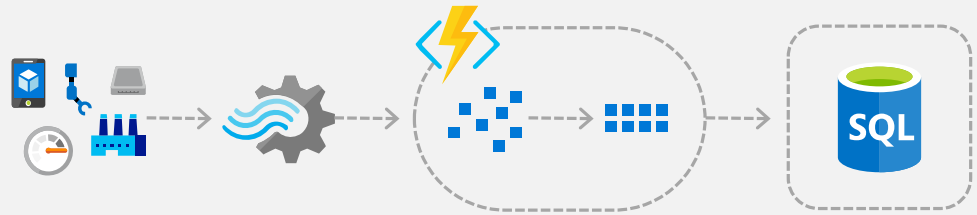
Per Device Certificates
Per Device Enable/Disable
TLS Security
X.509 Support
IP Whitelisting/Blacklisting
Shared Access Polices
Firmware/Software Updates

Scenarios for Serverless

Anything that needs to respond to events

Things

Real-time stream processing

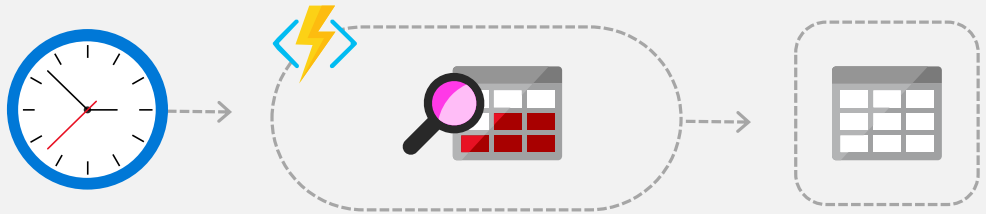


Millions of devices feed into Stream Analytics

Transform to structured data

Store data in SQL DB

Timer-based processing



Every 15 minutes

Find and clean invalid data

Clean table

Backends (Mobile/IoT/Web)

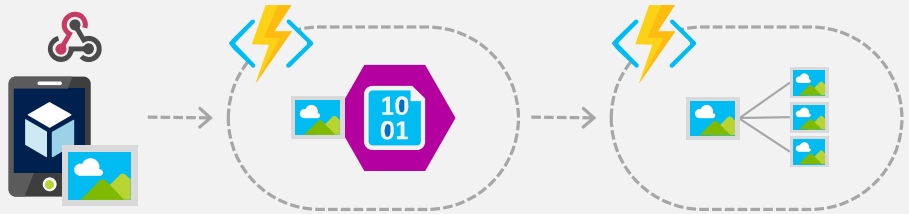
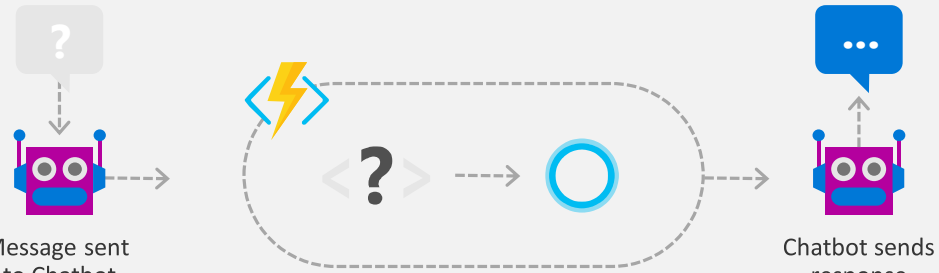


Photo taken and WebHook called

Stores in blob storage

Produces scaled images

Real-time bot messaging



Message sent to Chatbot

Cortana Analytics answers questions

Chatbot sends response

Azure Event Hub

Things

What is it?	When to use it?
Azure Event Hub is a **Publish-Subscribe** service that ingests millions of events per second and allows you to stream them into multiple applications. ➢ **A hyper-scale telemetry ingestion service** ➢ **Collects, Transforms and Stores events from remote devices** ➢ **The "front door" for an event pipeline** ➢ **Azure platform as a service**	Use Event Hubs when you need to process and analyze massive amounts of data produced by connected devices, sensors, and applications. ✓ **Highly scalable (millions per second)** ✓ **Integrates with real-time analytics** ✓ **Publish-Subscribe capabilities** ✓ **Low latency at massive scale** ✓ **Process real-time and batch on the same stream**

ToC

Functions

Develop locally using best of class developer tools

Boost productivity through triggers and bindings

Choose from a variety of programming languages

Integrate with existing DevOps processes

Things

Code

Azure Functions

Events & Data

Cosmos DB

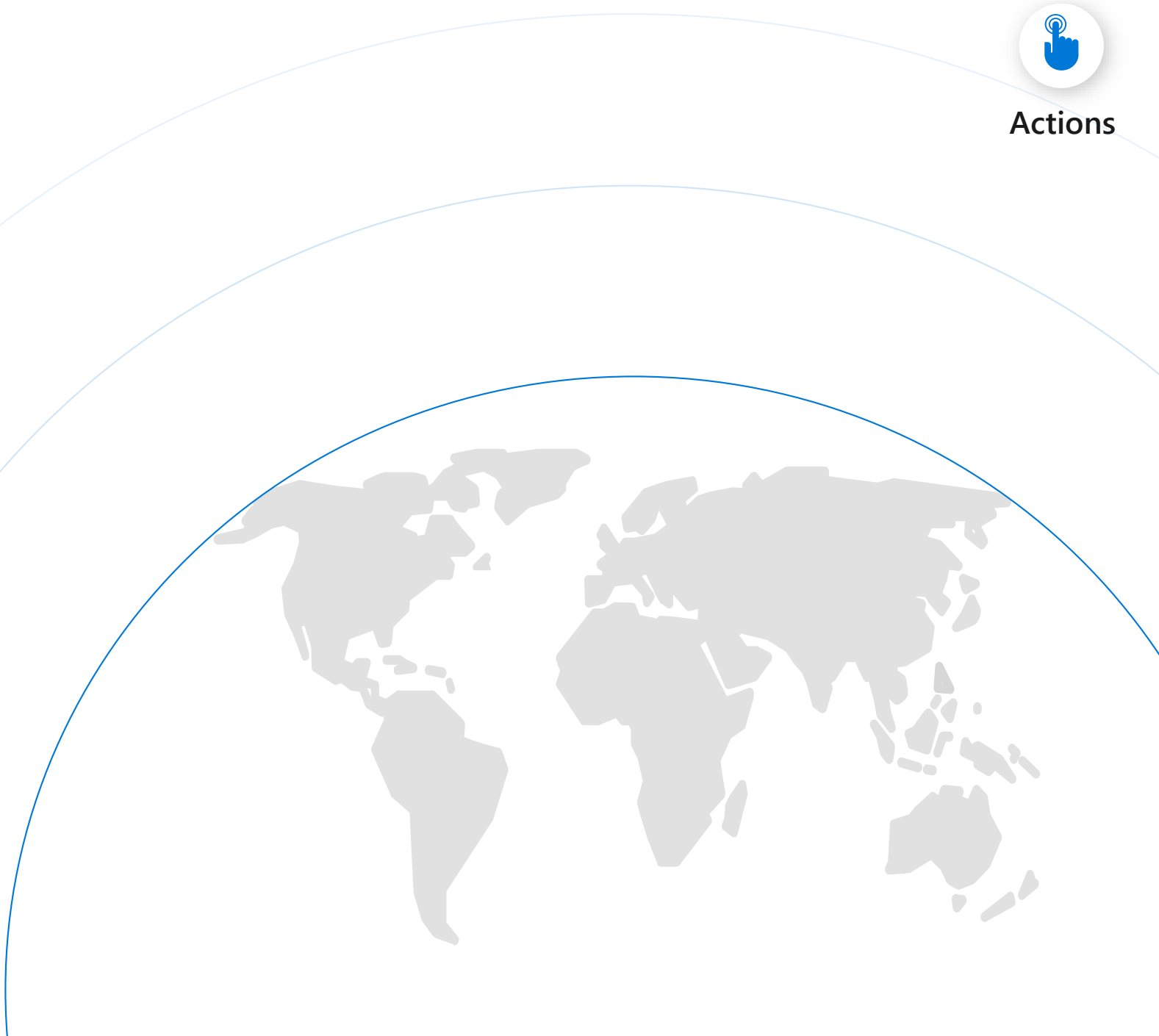
Turnkey global distribution

Elastic scale out of storage & throughput

Guaranteed low latency at the 99th percentile

Five well-defined consistency models

Comprehensive SLAs



Cosmos DB

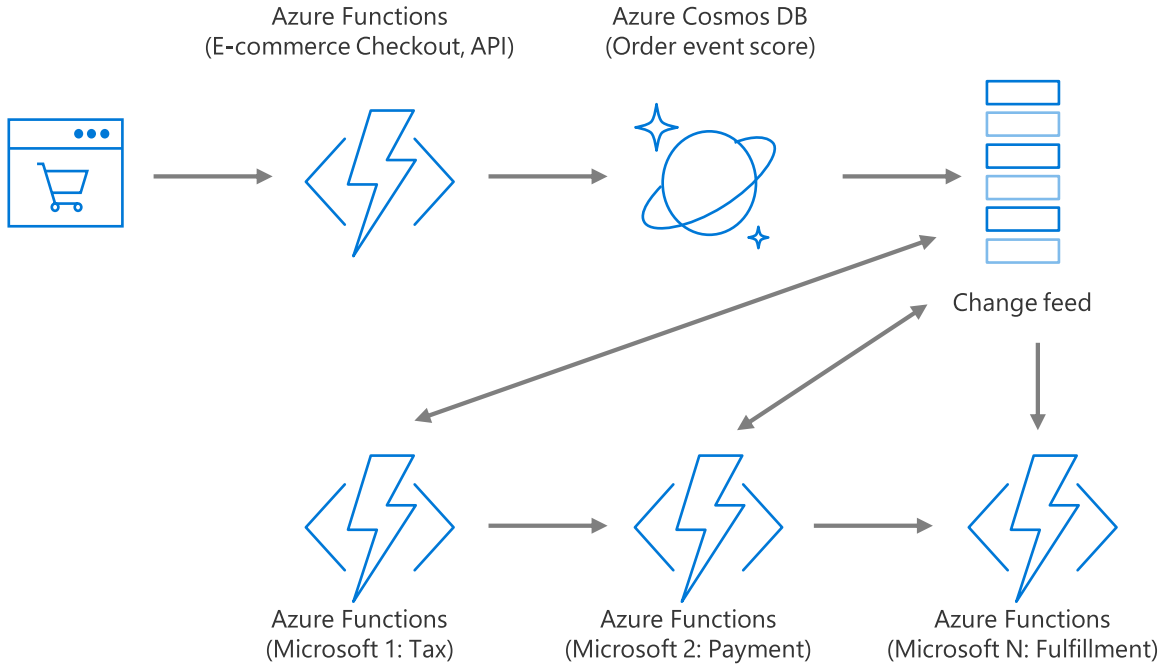
Seamless handling of any data output or volume

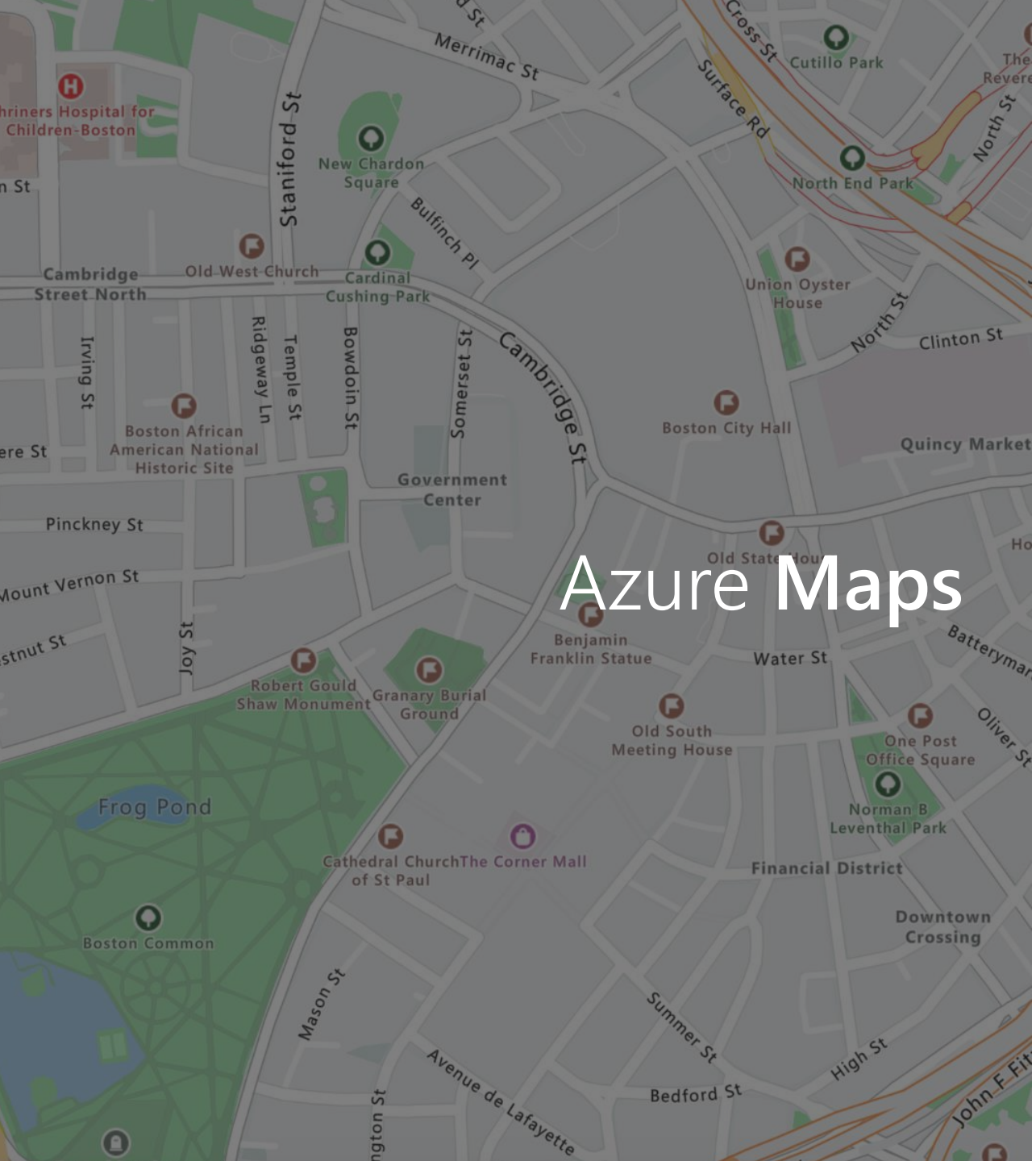
Data made available immediately, and indexed automatically

Real-time, resilient change feeds logged forever and always accessible

Native integration with Azure Functions

Actions





Actions

Maps
Render maps and satellite imagery across many geographies

Map Control
Integrate rich mapping visualizations into applications

Routing
Calculate routes from N to N points for optimal calculations

Search and Geocoding
Convert places and addresses to coordinates; or, convert coordinates to addresses or cross streets

Traffic
Show real time traffic information

Time Zones
Obtain time zone and current time information

Agenda

What is IoT?

Azure Components Overview

Lab 1 - Getting Data from a Pi

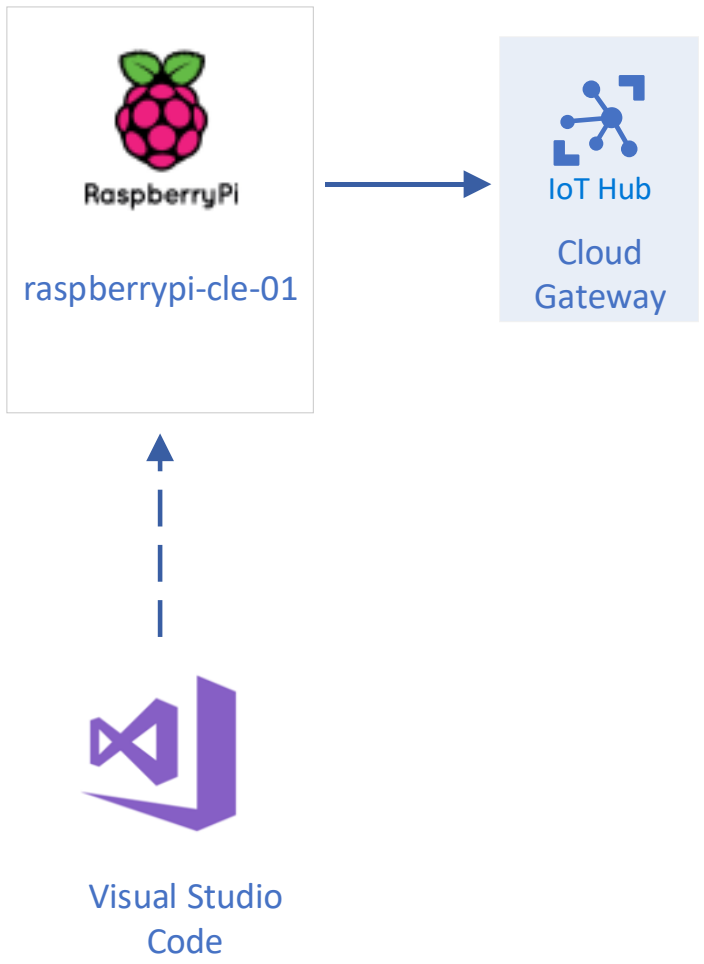
Real-Life Scenario Overview

Lab 2 – End-to-End Solution

Wrap Up and More

Lab 1 – Getting Data From a Pi Into Azure

In this lab, you will be using a SenseHAT on a Raspberry Pi Device to send environment data to an Azure IOT Hub



STEPS:

- Create an IOT Hub in Azure
- Connect the pi to IOT Hub in Azure
- Update code with Connection String from Azure
- Deploy code to the Pi device
- Run the code on the Pi to collect telemetry data
  - Temperature
  - Humidity
  - Pressure
- See the collected data in your Azure IOT Hub

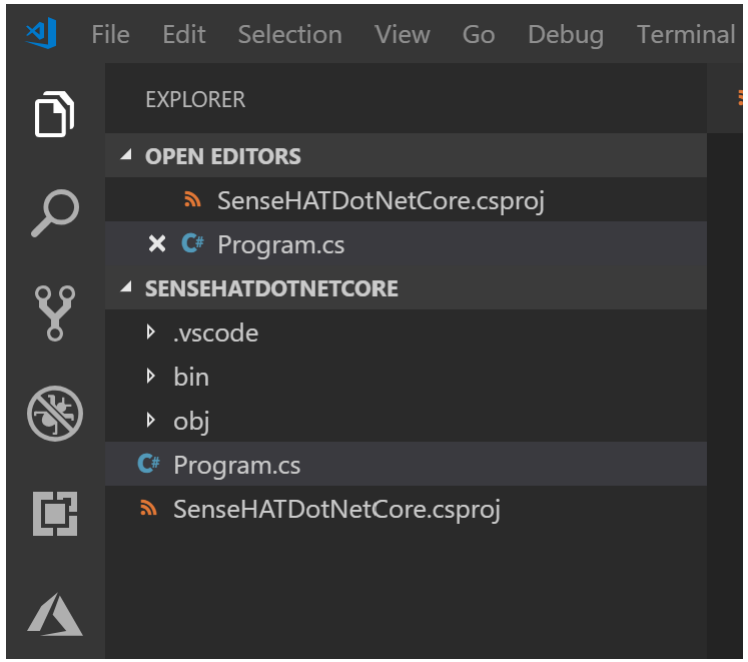
Lab1

Code Walk-Through

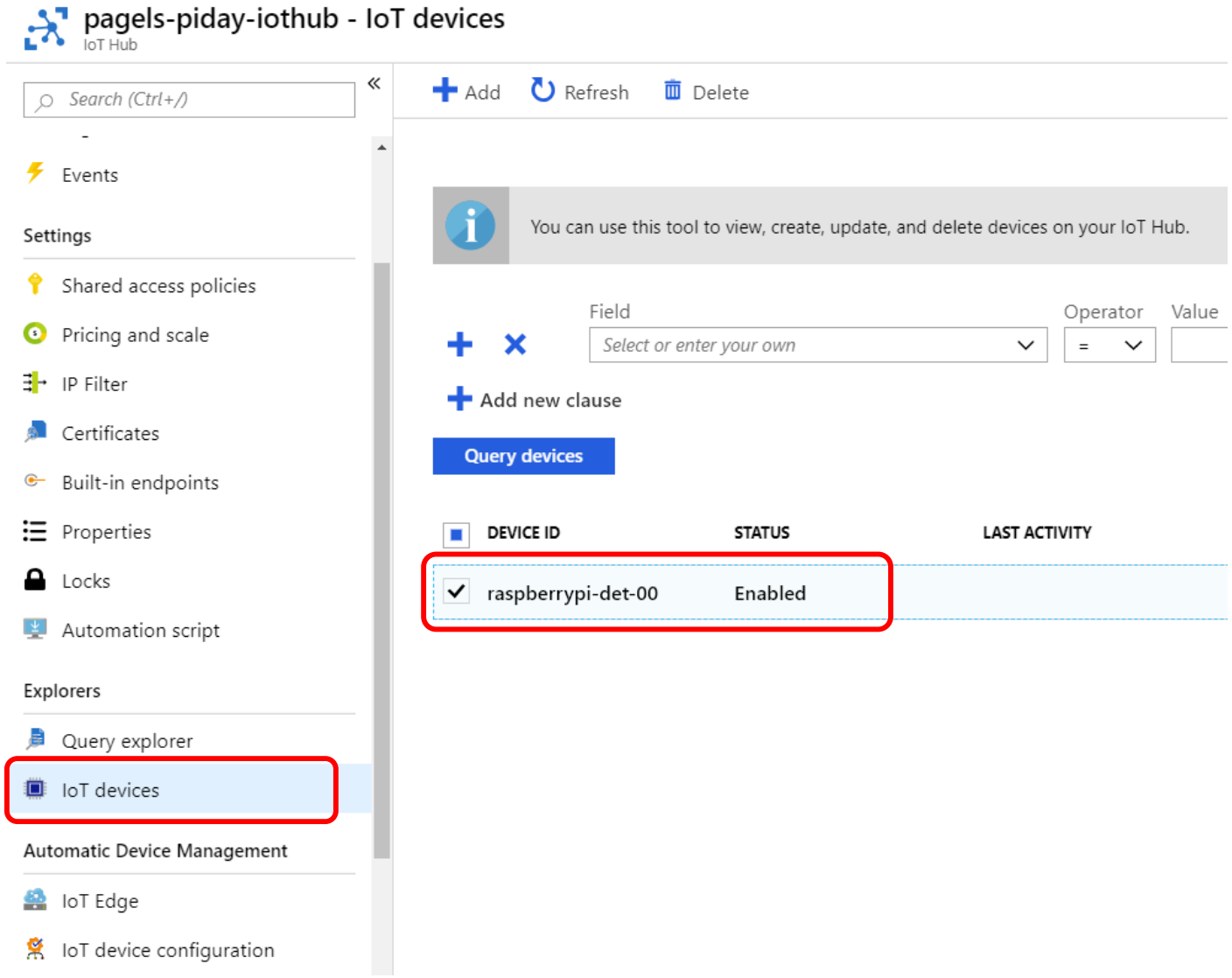
Lab 1.1: SenseHAT Source Code

```
C:\Workspace\IoT-Pi-Day-master\Lab 1 - Getting started with the Sense HAT\Solution\SenseHATDotNetCore>code .
```

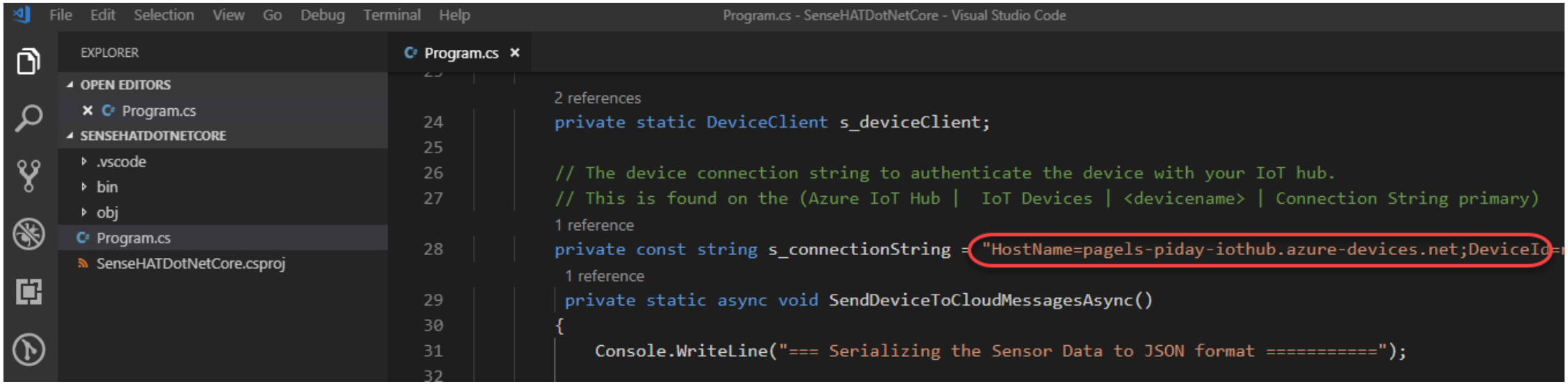
SenseHATDotNetCore.csproj



Lab 1.1: Connect the IOT Hub to the Device



Lab 1.1: Connect the Device to the IOT Hub



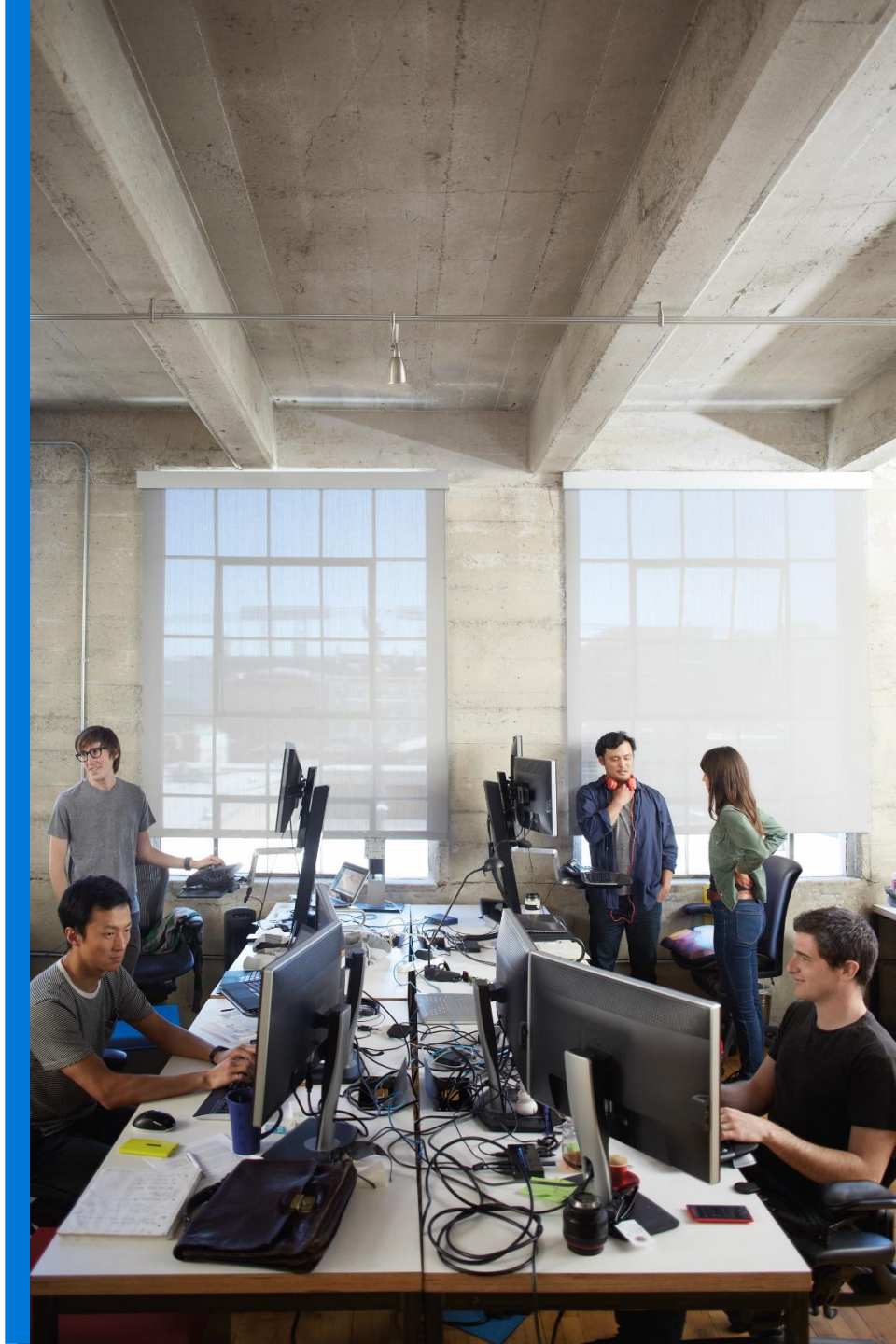
Lab 1

~ 20 Minutes

- Make sure you have completed the laptop setup
- Make sure you have downloaded the GitHub repository
  - https://github.com/Azure/IoT-Pi-Day
- Make sure you know the name and IP address of your pi device
- Make sure you have completed the Laptop setup steps
- Raise your hand if you have any issues – proctors will be walking around

TIPS:

- Put everything in a SINGLE resource group!
- Don't use the same Azure region for deployment as your neighbor!
- Follow the naming suggestions! (if you have a long last name, consider using your initials instead)
- Take note of what you name things and what region you pick
- If there is more than one person per device, you may have an issue getting SenseHAT data in Lab 1.1 due to device sharing limitations – only one person at a time will be able to get info.



Lab 1 – Recap

- Why did we put everything in a single resource group?
- What happened to the data you sent to the IoT hub?
- What was the surprise, and what was the code that drove it?
- Is there a better way to get code to a device?
- Why did we have to copy the Connection String from the IOT Hub service in Azure into the code?

Agenda

What is IoT?

Azure Components Overview

Lab 1 - Getting Data from a Pi

Real-Life Scenario Overview

Lab 2 – End-to-End Solution

Wrap Up and More

Everything’s Better On a Bike

Welcome to BLUEbikes, your public bike share system in Boston, Brookline, Cambridge and Somerville.

How it works:

Unlock - Pick up a bike at any station around Boston, Brookline, Cambridge and Somerville.

Ride - Take a quick one-way trip or a leisurely ride around town. Commute to work or school, run errands and explore the city.

Return - Return your bike to any station. Slide the bike firmly into an empty dock and wait for the green light to make sure it's locked

System Data

Where do Bluebikes riders ride? When do they ride? How far do they go? Which stations are most popular? On what days of the week are most rides taken? We've heard all of these questions and more, which is why we are pleased to provide the data sets to help you discover the answers to these questions and more. We invite developers, engineers, statisticians, artists, academics and other creative members of the public to use the data we provide for analysis, development, visualization, and whatever else moves you.

Blue Bikes Comprehensive Trip Histories

We publish downloadable files of Bluebikes trip data each quarter. This data is provided according to the Bluebikes Data License Agreement.

The data includes:

- Trip Duration (seconds)
- Start Time and Date
- Stop Time and Date
- Start Station Name & ID
- End Station Name & ID
- Bike ID
- User Type (Casual = Single Trip or Day Pass user; Member = Annual or Monthly Member)
- Birth Year
- Gender, self-reported by member

Our Task:

Use BLUEbikes system data to determine which how many trips originate from each bike station today (Start Station id).

“Liberties”:

- We are using the system data as a simulation of real-time IOT device data

https://www.bluebikes.com/system-data

Agenda

Why are we here?

Azure Components Overview

Lab 1 - Getting Data from a Pi into Azure

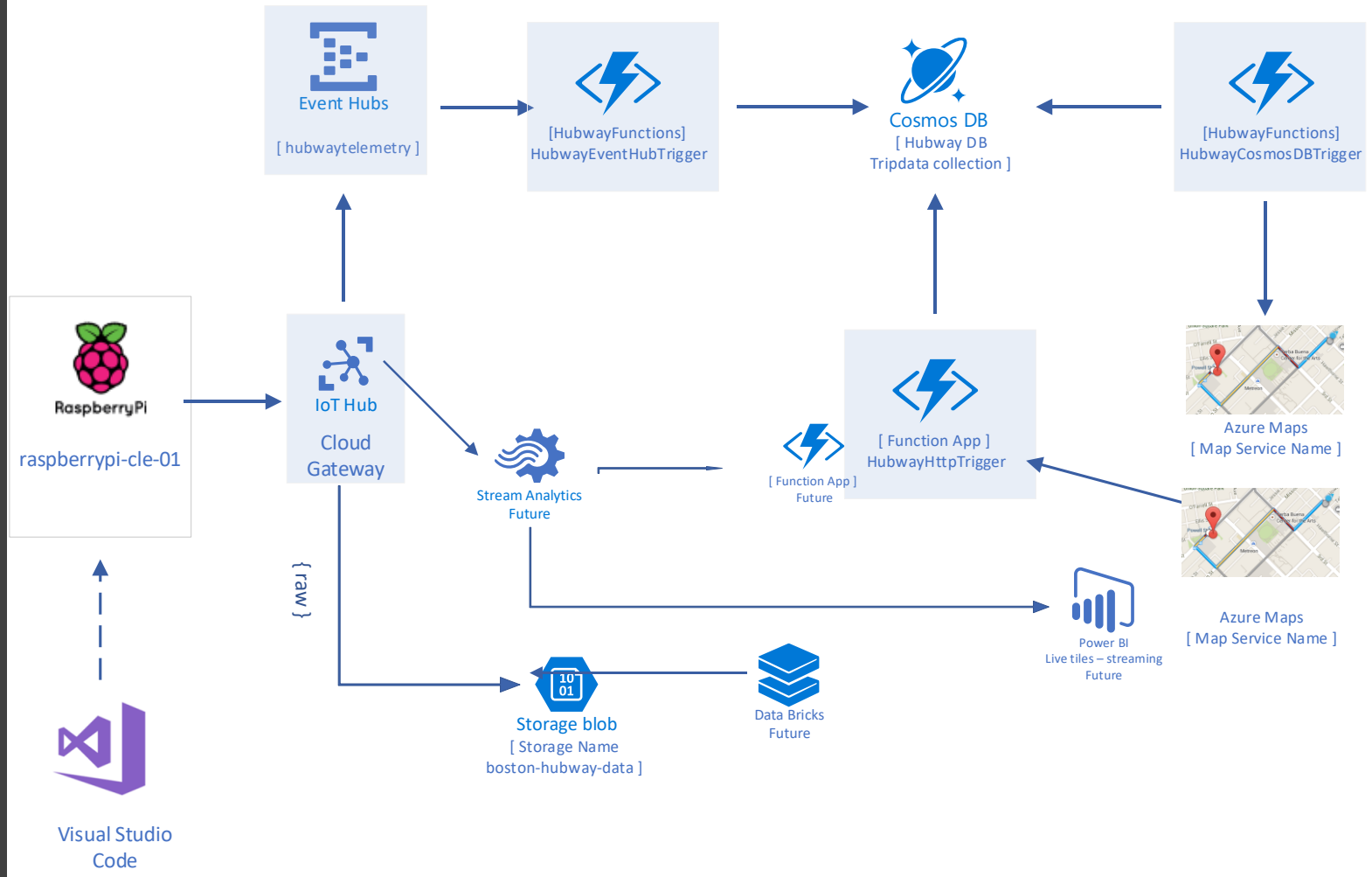
Real-Life Scenario Overview

Lab 2 – A Full Solution (Part1)

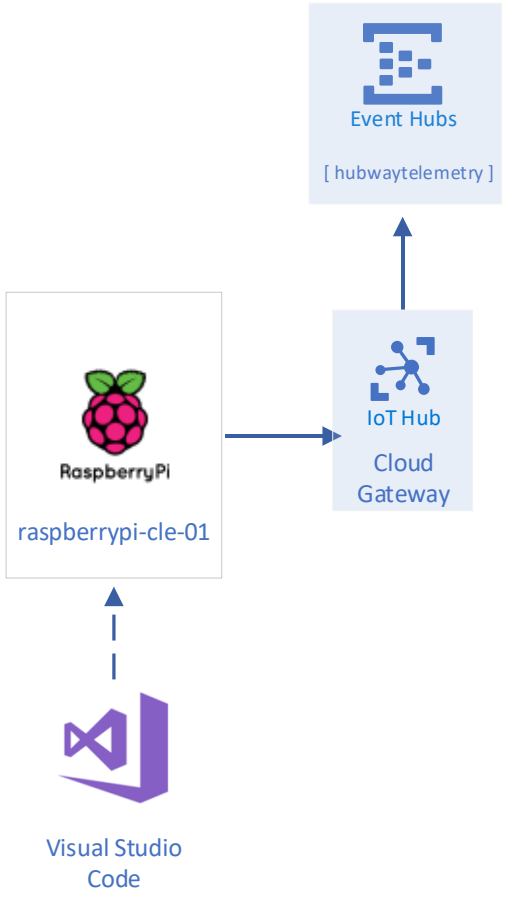
Wrap Up and More

Lab 2 Reference Architecture

- Stateless Architecture
- Allows for fast, real-time analysis of records
- Can be expanded to be stateful



Lab 2 – Part 1 Overview



In these first set of labs for Lab2, you are going to configure a number of Azure services that will read the IOT data “real-time”. You will also configure a “bucket” in Azure Event Hubs and will specify that route for messages coming into the IOT Hub.

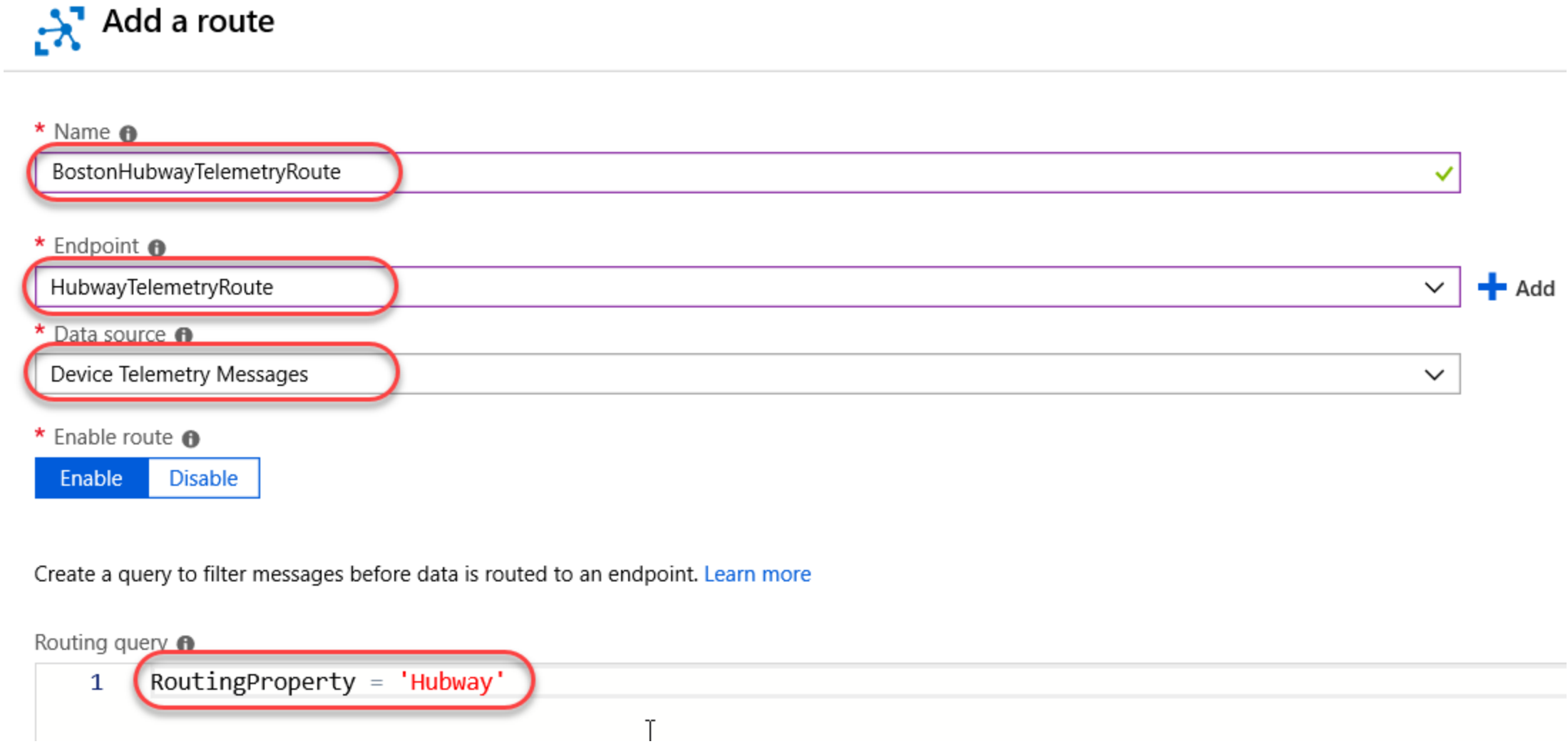
You will be completing the following labs:

- Lab 2.0 Resource Configuration
- Lab 2.1 Define Message Routing for Device
- Lab 2.2 Send Hubway Data to IOT Hub
- Lab 2.3 Setting Up Event Hub

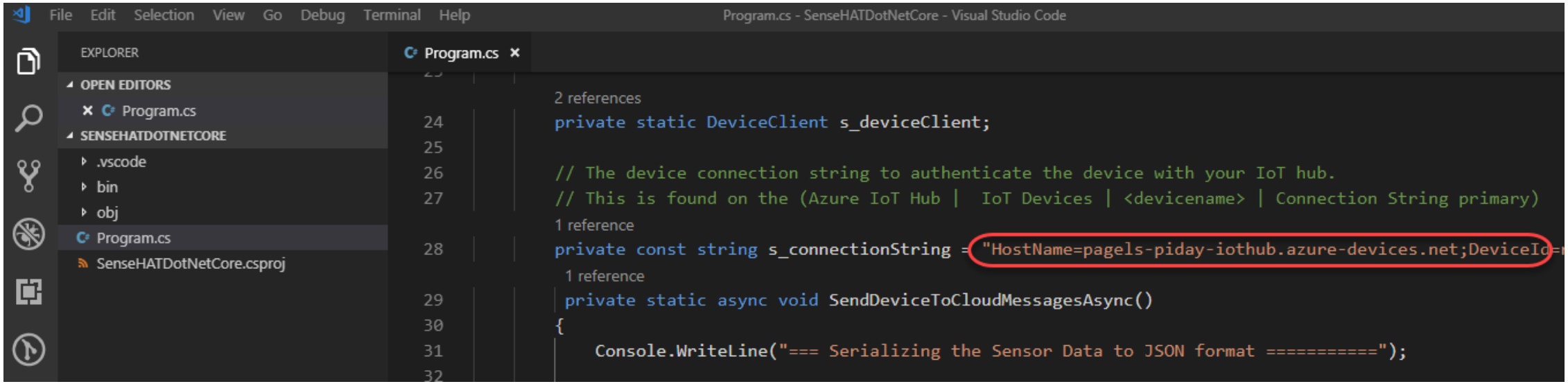
Lab2 (part one)

Code Walk-Through

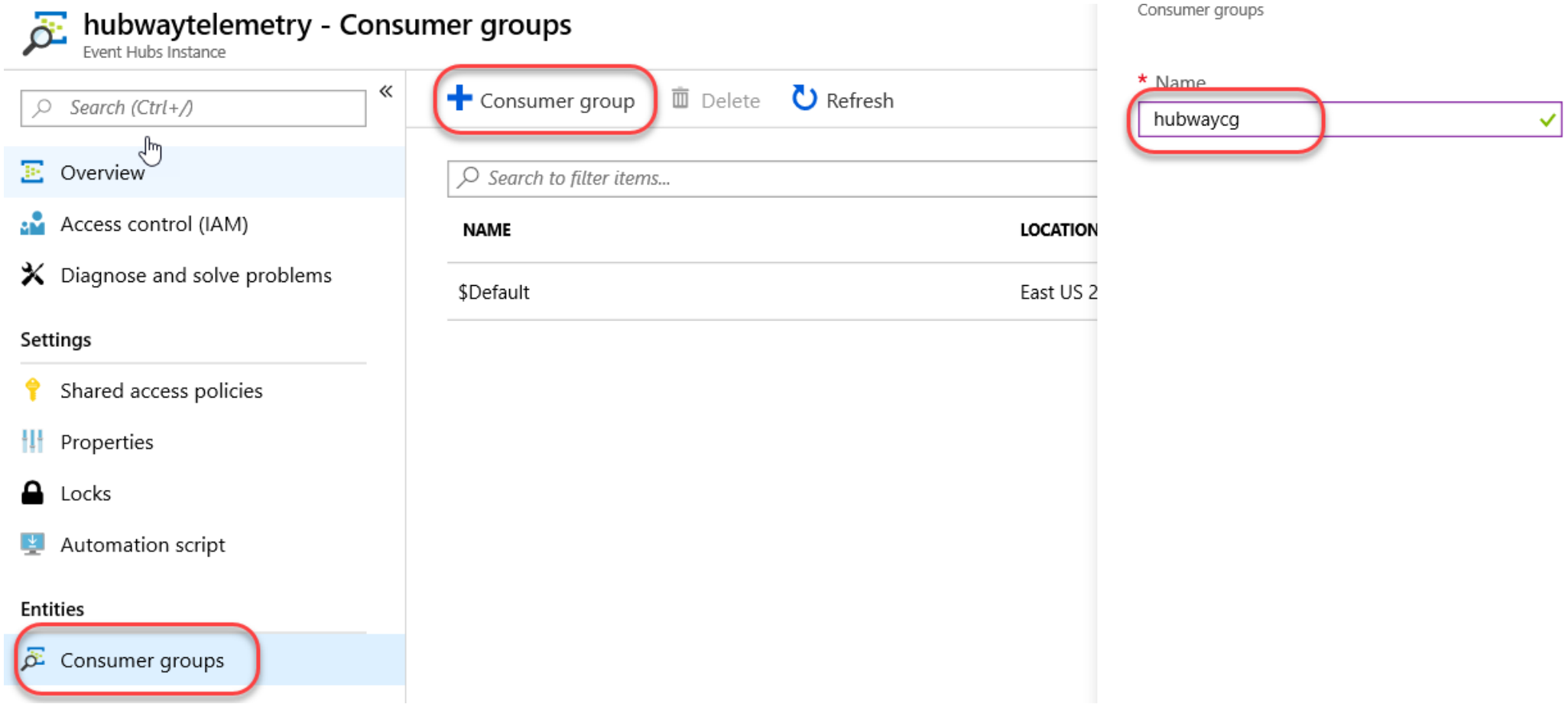
Lab 2.1: Connect IOT Hub to Event Hub



Lab 2.2: Connect the Device to the IOT Hub



Lab 2.3: Add an Event Hub Consumer Group



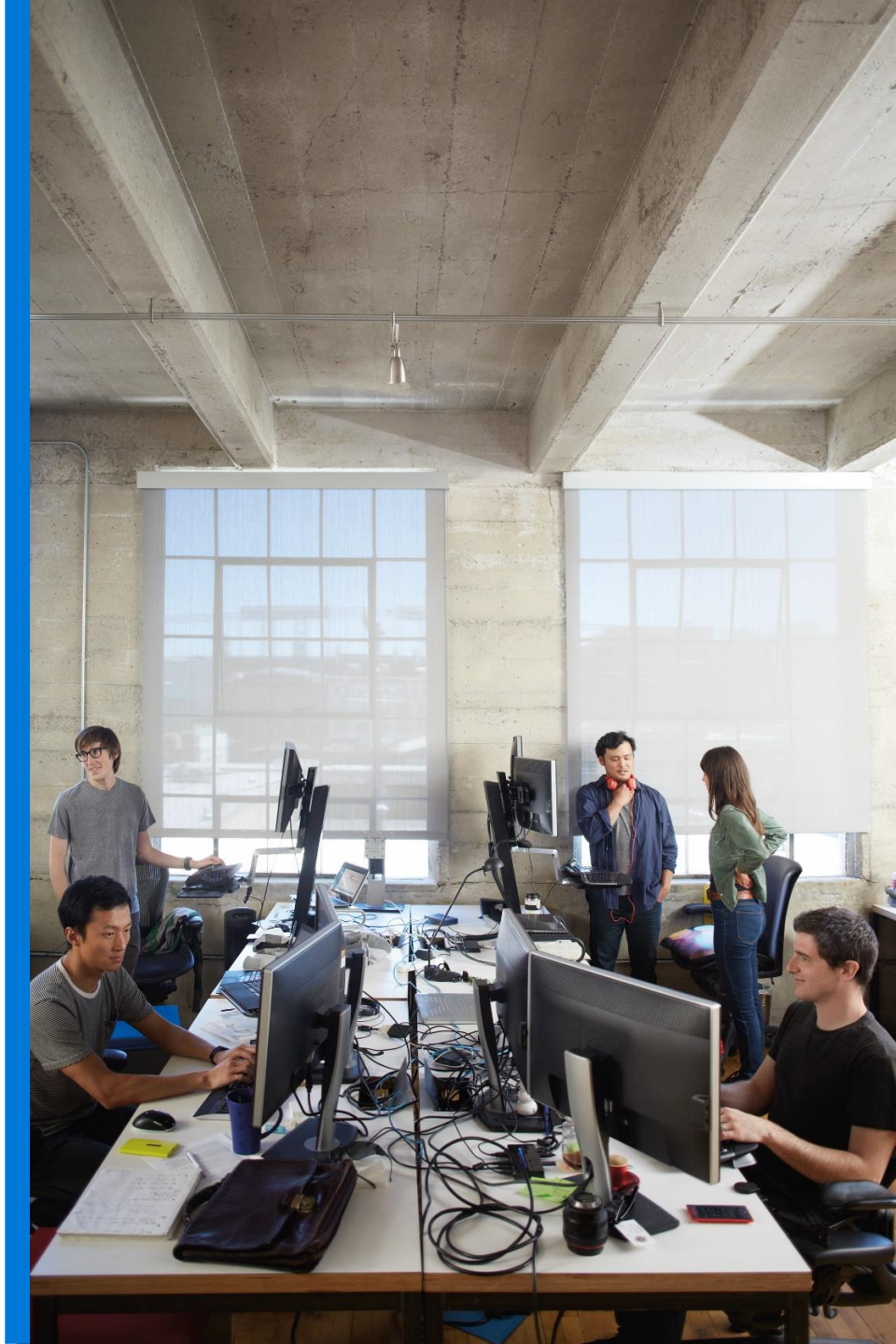
~ 45 Minutes

Lab 2 – Part 1 (2.0 – 2.3)

- Make sure you have completed Lab1 and you complete the Lab2 components in order
  - https://github.com/Azure/IoT-Pi-Day
- ONLY do Labs 2.0 – 2.3
- Raise your hand if you have any issues – proctors will be walking around

TIPS:

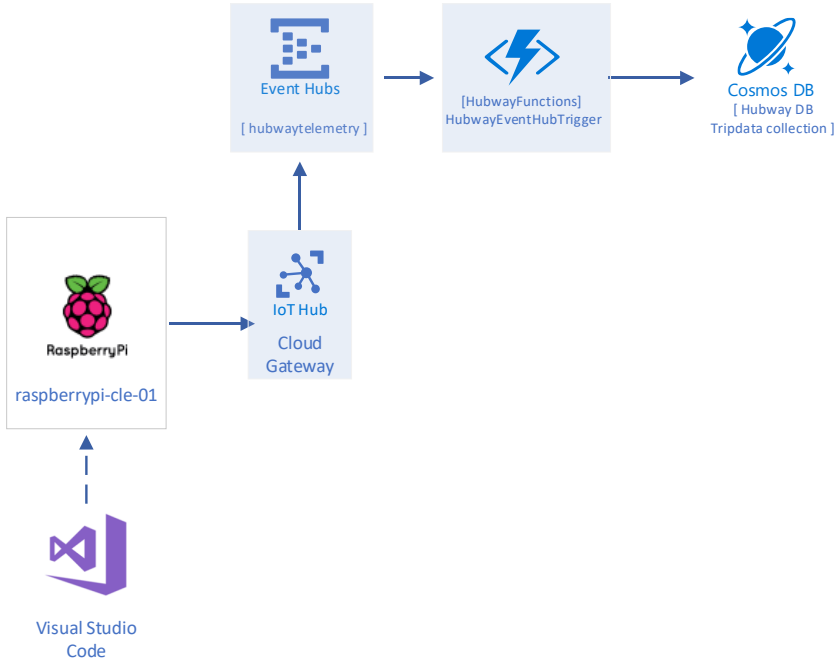
- Let us know if you have any tips. We think these labs are pretty solid!



Lab2 (part 1) Recap

- Why are we creating custom endpoints in the IoT Hub?
- What does the Routing Property = “Hubway” do?
- What is a Consumer Group and why are we creating one?

Lab 2 – Part 2 Overview



In this second set of labs for Lab2, you are going to create a number of Azure Function triggers. These functions will act on data coming in, transform the data and create records in a CosmosDb.

You will be completing the following lab:

- Lab 2.4 Create Event Hub Trigger Function

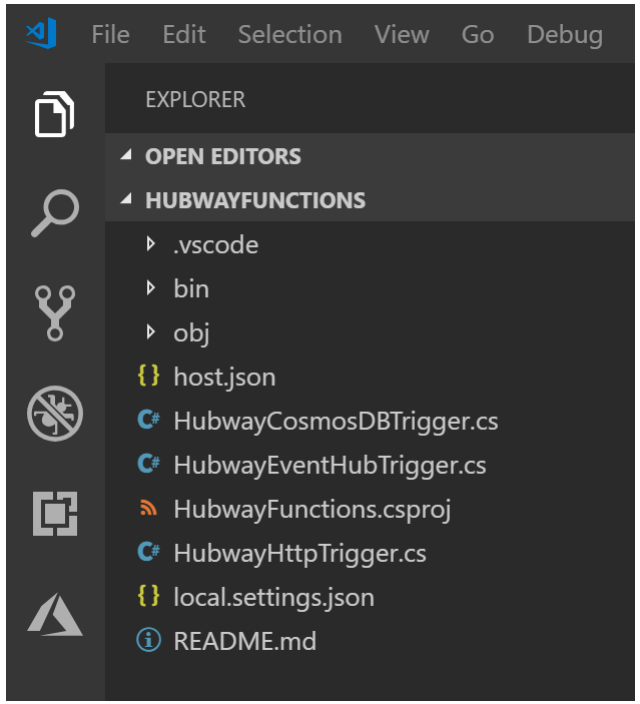
Lab2 (part two)

2.4 Code Walk-Through

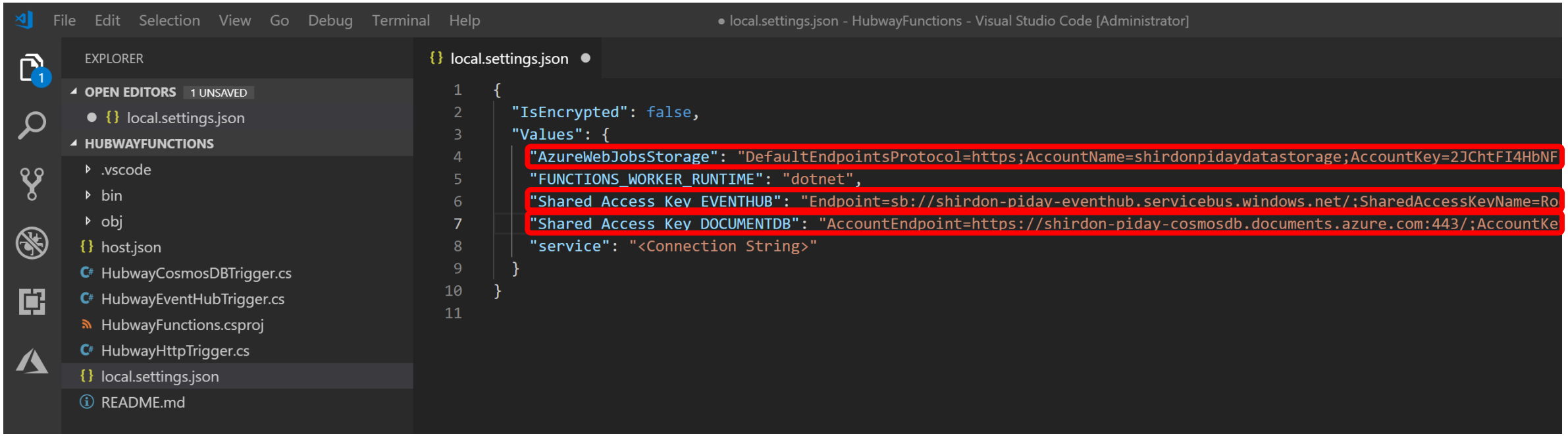
Lab 2.4: Functions Application Source Code

```
C:\Workspace\IoT-Pi-Day-master\Lab 2 - Working with Hubway Data\Solution\HubwayFunctions>code .
```

HubwayFunctions.csproj



Lab 2.4: Set the Event Hub, Storage Account, and CosmosDB Connection Strings



Lab 2.4: HubwayEventHubTrigger.cs

HubwayEventHubTrigger.cs

```
ing Microsoft.Azure.WebJobs;
ing Newtonsoft.Json.Linq;
ing Microsoft.Azure.WebJobs.Host;
ing Microsoft.Extensions.Logging;

mespace Company.function

    0 references
    public static class HubwayEventHubTrigger
    {
        [FunctionName("HubwayEventHubTrigger")]
        0 references
        public static void Run([EventHubTrigger("hubwaytelemetry", Connection = "Shared_Access_Key_EVENTHUB", ConsumerGroup = "hubwaycg")]
        string myEventHubMessage, [CosmosDB(databaseName: "Hubway",collectionName: "Tripdata",
        ConnectionStringSetting = "Shared_Access_Key_DOCUMENTDB")] out dynamic outputDocument, ILogger log)
        {
            log.LogInformation($"C# Event Hub trigger function processed a message: {myEventHubMessage}");

            dynamic msg = JObject.Parse(myEventHubMessage);

                outputDocument = new  {startTime = msg.GetValue("starttime").ToString(),
                                       stopTime = msg.GetValue("stoptime").ToString(),
                                       tripDuration = msg.GetValue("tripduration").ToString(),
                                       startStationID = msg.GetValue("start_station_id").ToString(),
                                       startStationName = msg.GetValue("start_station_name").ToString(),
                                       startStationLatitiude = msg.GetValue("start_station_latitude").ToString(),
                                       startStationLongitude = msg.GetValue("start_station_longitude").ToString(),
                                       endStationID = msg.GetValue("end_station_id").ToString(),
                                       endStationName = msg.GetValue("end_station_name").ToString(),
                                       endStationLatitude = msg.GetValue("end_station_latitude").ToString(),
                                       endStationLongitude = msg.GetValue("end_station_longitude").ToString(),
                                       bikeID = msg.GetValue("bikeid").ToString(),
                                       userType = msg.GetValue("usertype").ToString(),
                                       gender = msg.GetValue("gender").ToString()};

        }
    }
```

- Picks up event and details from Event Hub
- Performs data transformation
- Creates a corresponding CosmosDB document
- Writes the record to the document

Lab 2 – Part 2 (Lab2.4)

~ 45 Minutes

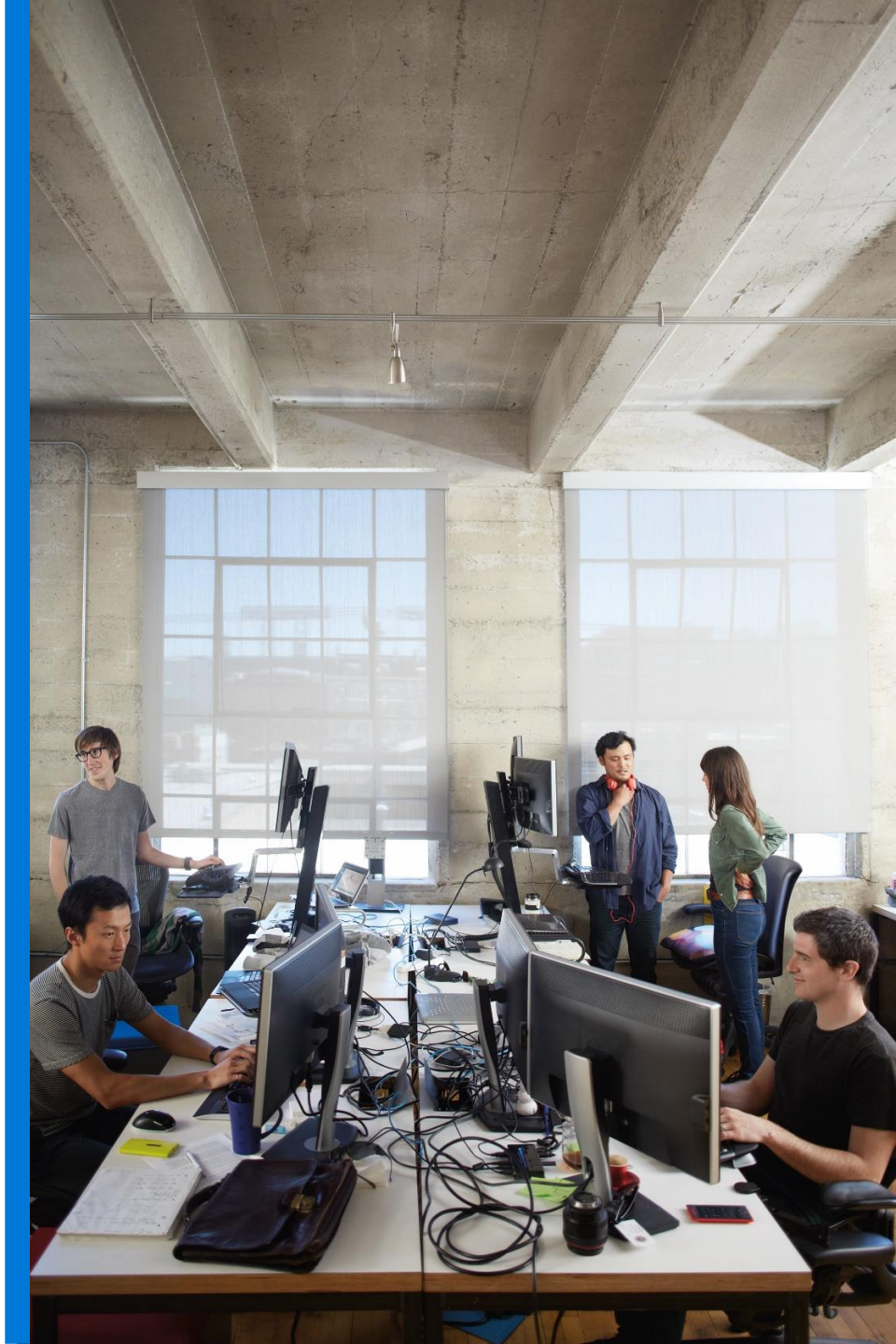
- Make sure you have completed Lab1 and you complete the Lab2 components in order
  - https://github.com/Azure/IoT-Pi-Day

- ONLY do Lab 2.4

- Raise your hand if you have any issues – proctors will be walking around

TIPS:

- Be careful that you're copying the correct connection strings from Azure portal



LUNCH

Video from Howard – Bar Mitzvah Lights with Raspberry Pi

Lab2 (part 2) Recap

- Can you have multiple functions per function app? Why would you do this?
- What does the Event Hub trigger do?
- What does the CosmosDb trigger do?

Agenda

Why are we here?

Azure Components Overview

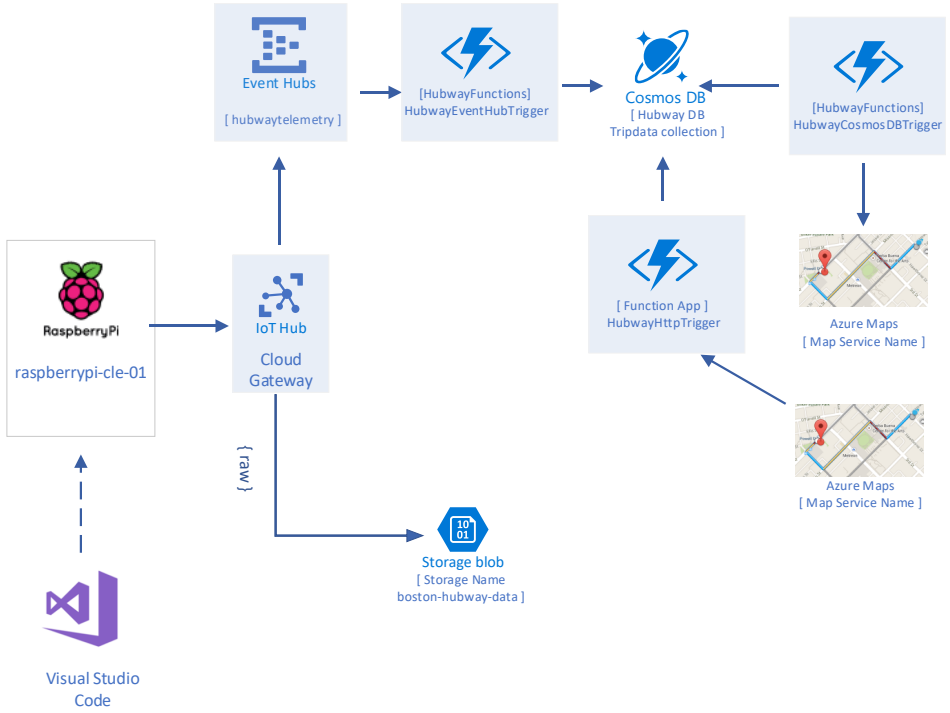
Lab 1 - Getting Data from a Pi into Azure

Real-Life Scenario Overview

Lab 2 – A Full Solution (Continued)

Wrap Up and More

Lab 2 – Part 3 Overview



In these last set of labs for Lab2, you are going to configure a blob storage account. You are also going to leverage the Azure Maps APIs to create a static map with bubbles and counts on bike station names to depict bike starting locations.

You will be completing the following labs:

- Lab 2.5 Update the CosmosDBTrigger Function
- Lab 2.6 Configure Storage
- Lab 2.7 Working with Azure Maps

Lab2 (part 3)

Code Walk-Through

Lab 2.5: HubwayCosmosDBTrigger.cs

```
19          public static async Task Run([CosmosDBTrigger(
20              databaseName: "Hubway",
21              collectionName: "Tripdata",
22              ConnectionStringSetting = "Shared_Access_Key_DOCUMENTDB",
23              LeaseCollectionName = "leases",
24              CreateLeaseCollectionIfNotExists = true)]IReadOnlyList<Document> input, ILogger log)
```

```
29              foreach (var doc in input)
30              {
31
32                  lat = doc.GetPropertyValue<string>("startStationLatitiude");
33                  lon = doc.GetPropertyValue<string>("startStationLongitude");
34
35                  log.LogInformation("Start Station Latitude variable lat: " + lat);
36                  log.LogInformation("Start Station Longitude variable lon: " + lon);
37
38                  // Create a New HttpClient object and dispose it when done, so the app doesn't leak resources
39                  using ( HttpClient http = new HttpClient() )
40
41                  // Perform the Azure Map Search passing lat/lon
42                  try
43                  {
44
45                      //var http = new HttpClient();
46                      var url = string.Format("https://atlas.microsoft.com/search/address/reverse/json?subscription-key=<jk3CPEu
47
48                      log.LogInformation("Formatted Map URL: " + url);
49
50                      var response = await http.GetAsync(url);
51                      var result = await response.Content.ReadAsStringAsync();
52
53                      log.LogInformation("Azure Maps search result is: " + result );
54
55                      }
```

- Retrieve CosmosDB record
- Get startStation latitude and longitude to send to Azure Map search
- Retrieve street address of Hubway Station

Lab 2.7.1 HubwayHTTPTrigger.cs

```
16          public static TripDataGeoJson Run(
17          [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post",  Route = null)]HttpRequest req,
18          [CosmosDB(
19              databaseName: "Hubway",
20              collectionName: "Tripdata",
21              ConnectionStringSetting = "Shared_Access_Key_DOCUMENTDB",
22              SqlQuery = "SELECT * FROM c order by c.startStationID")]
23              IEnumerable<TripItems> tripItems,
24          ILogger log)
```

```
53                  //compare station ID's
54                  if ( sCurrentStationID == sLastStationID )
55                  {
56                      // they are in the same array
57                      //log.LogInformation("Station id's match: "+ sCurrentStationID );
58
59                      // increment the counter
60                      iCounter += 1;
61                  }
```

```
31              // return GeoJson object
32              TripDataGeoJson tdGeoJson = new TripDataGeoJson();
33
```

- Reads start station data records in Cosmos DB
- Counts number of instances a particular station was utilized as a start location
- Determines the location of the start station and returns it as a GeoJson object

Lab 2.7.2 Create the Web Page

```
26    function GetMap() {
27
28        //Add your Azure Maps subscription key to the map SDK. Get an Azure Maps key at https://azure.com/maps
29        atlas.setSubscriptionKey('Azure Maps subscription');
30
```

- Update your Azure Maps Subscription key

```
23        // change to your api
24        var hubwayFeed = "https://hubwayfunctions.azurewebsites.net/api/HubwayHttpTrigger";
25
```

- Point to the HubwayHTTPTrigger function

- Set aesthetics of map

Lab 2 – Part 3 (2.5-2.7)

~40 Mins

- Make sure you complete the labs in order
  - https://github.com/Azure/IoT-Pi-Day

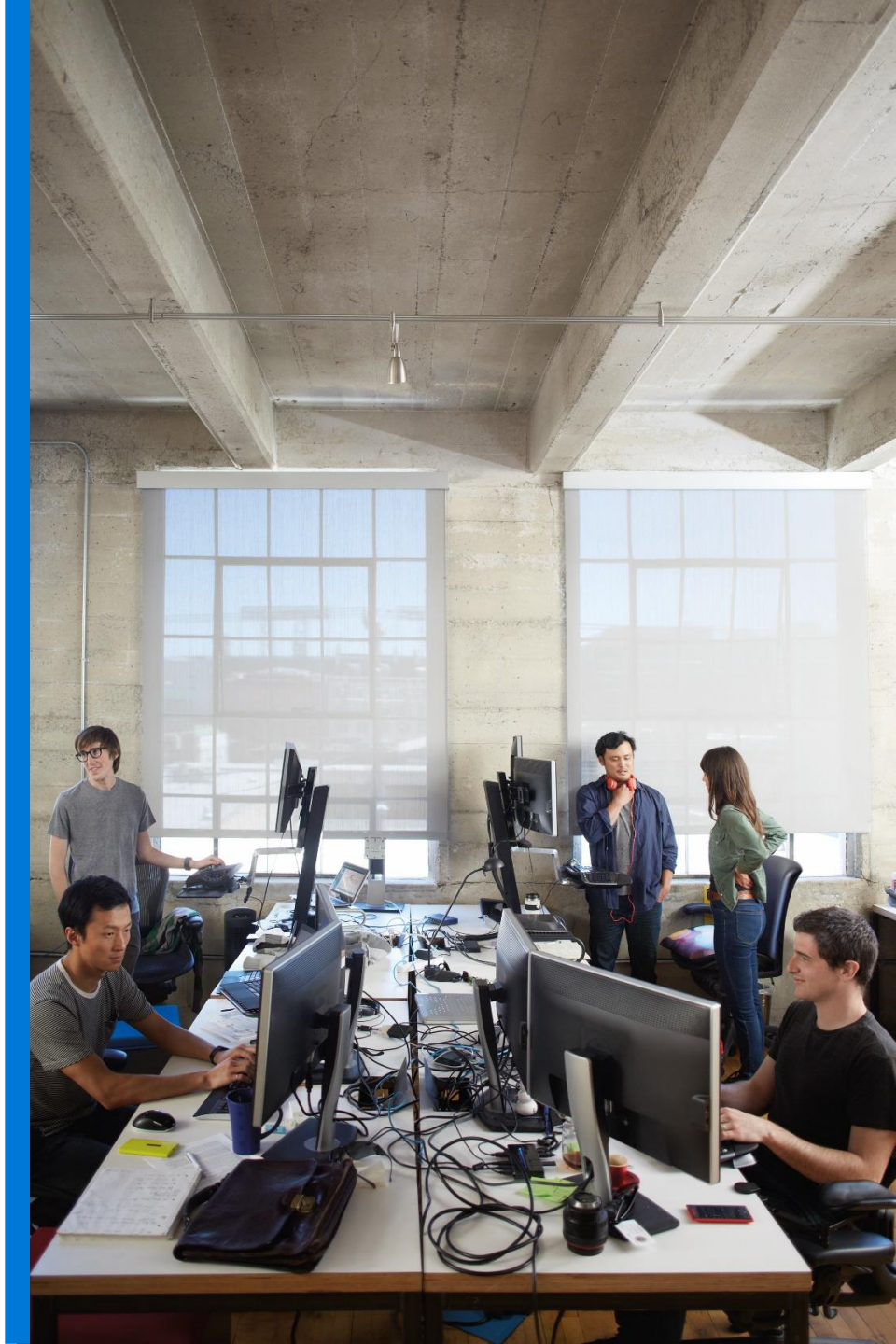
- If you have any issues or questions, raise your hand – proctors will be walking around

TIPS:

- Make sure your storage acct doesn't have any capital letters or special characters.

CHALLENGE:

- Can you change the color/size, etc of the "dots" on the Azure Map?



Lab2 (part 3) Recap

- Did you try the challenge? How can you change the format of the Azure Map bubbles, etc?
- What else could we show on the map?
- How is this data useful to BLUEbikes?

Agenda

Why are we here?

Azure Components Overview

Intro Lab - Setup

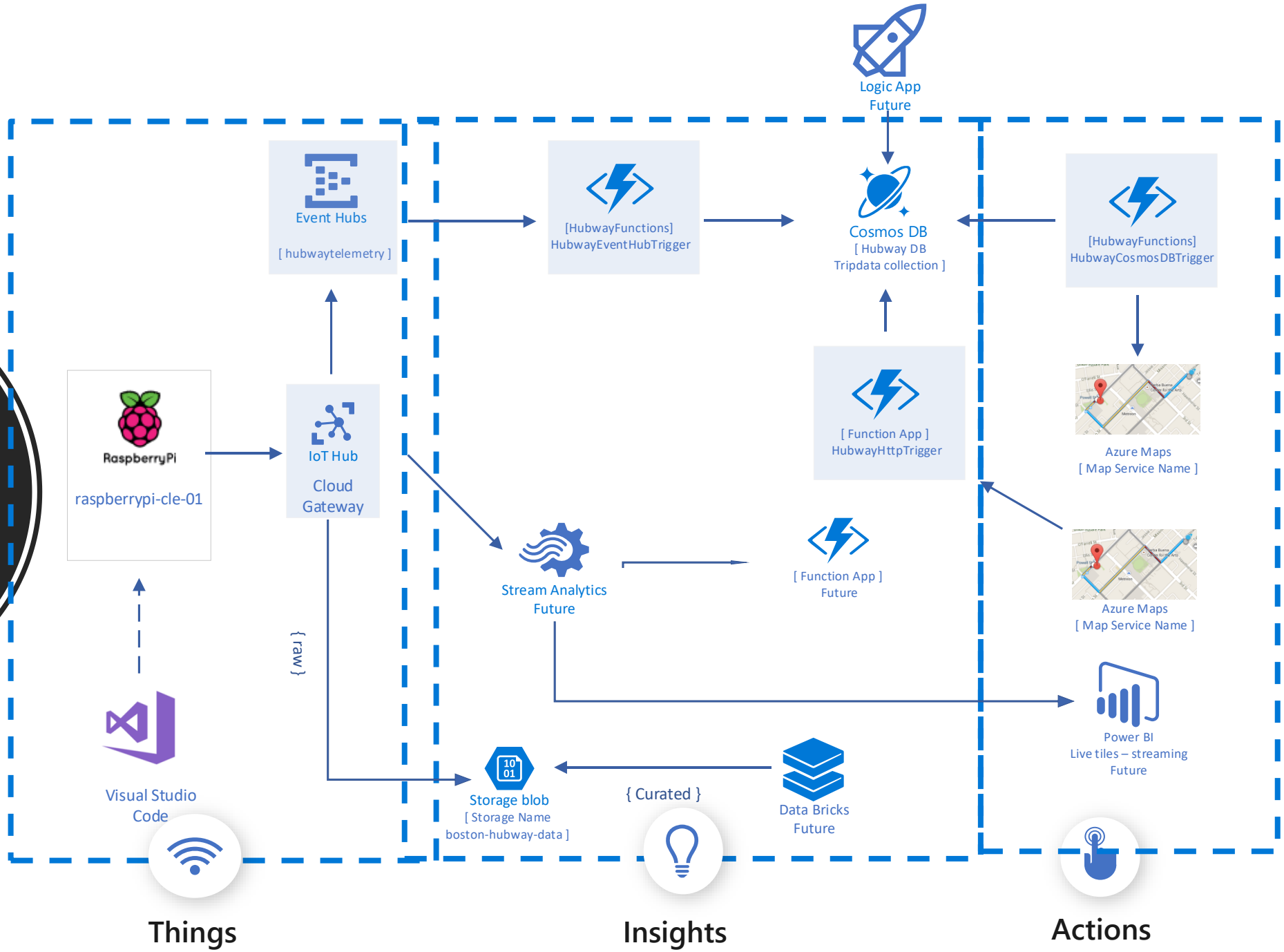
Lab 1 - Getting Data from a Pi

Real-Life Scenario Overview

Lab 2 – A Full Solution

Wrap Up and More

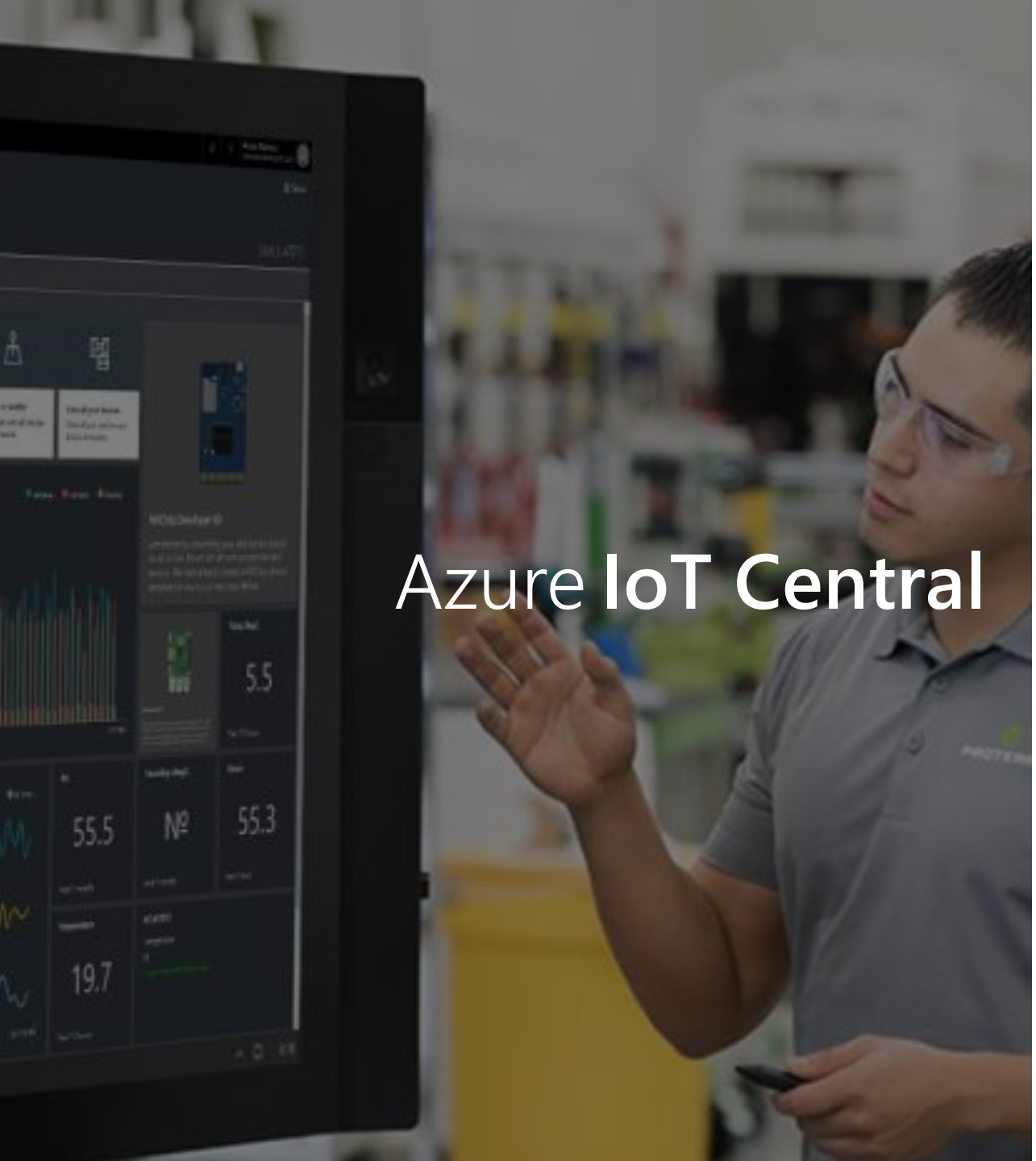
Remember this?



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
- Device Simulation



Azure IoT Central

- Fully hosted and managed by Microsoft
- No cloud development expertise required
- Device connectivity and management
- Monitoring rules and triggered actions
- User roles and permissions
- Analytics, dashboards and visualization
- Risk-free trial with simplified pricing

Get Started Today

Looking to USE an IoT Solution?

Use managed and industry-specific solutions to get started quickly and easily. Try IoT solutions

Ready to BUILD IoT Applications?

Find everything you need to develop advanced IoT apps using familiar languages and tools. Build IoT apps

1. Go to Azure.com/IoT
2. Skill up at IoT School
3. Select a partner
4. Contact Us

Resources

- IOT Suite For Developers Video: https://www.youtube.com/watch?v=5ES-1g_mGxY
- IOT customer videos: https://azure.microsoft.com/en-us/resources/videos/azure-iot-customer-stories/
- IOT Accelerators: https://docs.microsoft.com/en-us/azure/iot-accelerators/

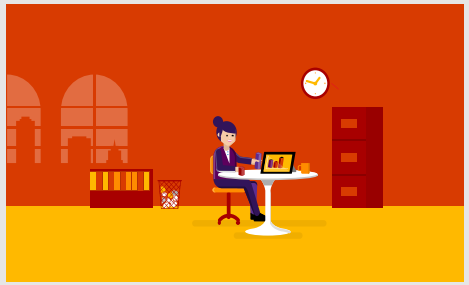
- Analyze Boston link: https://data.boston.gov/dataset/hubway-system-data

- Blue Bikes System Data: https://www.bluebikes.com/system-data

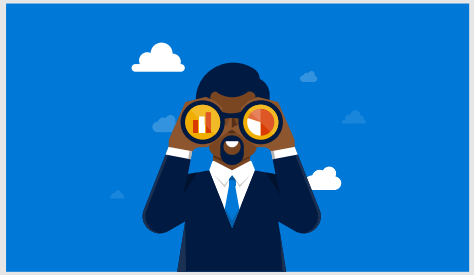
Thank You

Please take the survey!

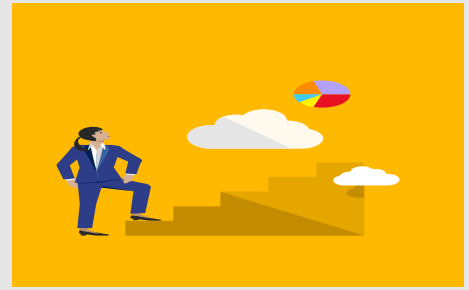
Use Case



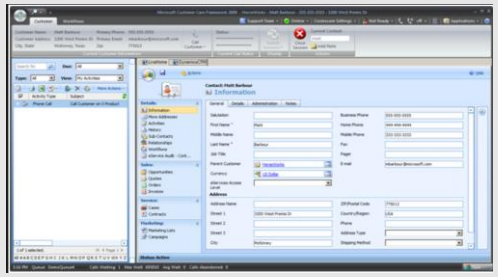
Kris is a building operations manager needs to **trigger downstream processes** once a new fire sensor is created in her IoT Hub



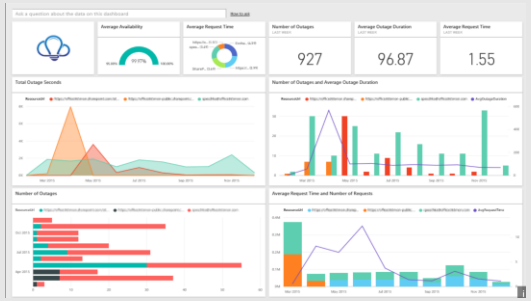
COMPLIANCE: She uses IoT Hub to kick off an Azure Functions to check compliance of the new device



Kris is able to **quickly react to device lifecycle events**, **integrate events in her business applications** with ease and save costs on polling services



TICKETING: Through IoT Hub Events she can trigger Azure Logic Apps to update her CRM system with information on the new device and open a ticket for the engineer to configure the device

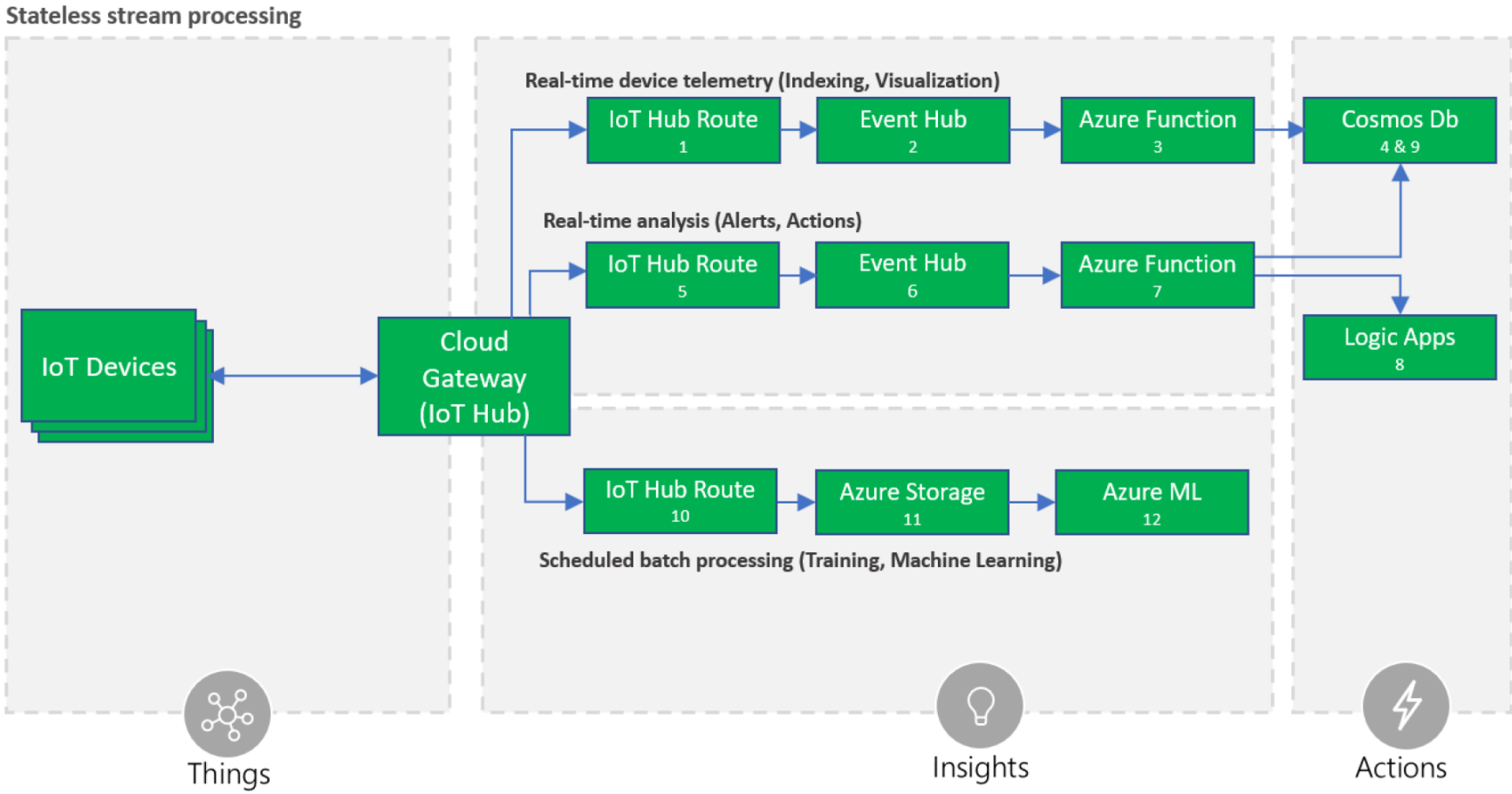


REPORTING: Her Power BI dashboard also gets updated, through a webhook, to display the new device and its status

Stateless Stream Processing

This architecture provides a solution for fast and scalable real-time analysis of ingested data records, in scenarios where only stateless analysis is required, using a small set of simple logic rules.

Also included is a slower path, allowing execution of more complex analysis, for instance machine learning jobs, without the speed limitations of the fast path.



Stateless Stream Processing Requirements

Real-Time Device Telemetry:

- Input data records are serialized in JSON
- Processing rules take in input one message at a time
- Define routing conditions in Azure IoT Hub in order to forward only specific messages

Batch Processing:

- Input record storage where it can be archived indefinitely at low cost
- Device data analysis by M/L systems

Architecture Benefits

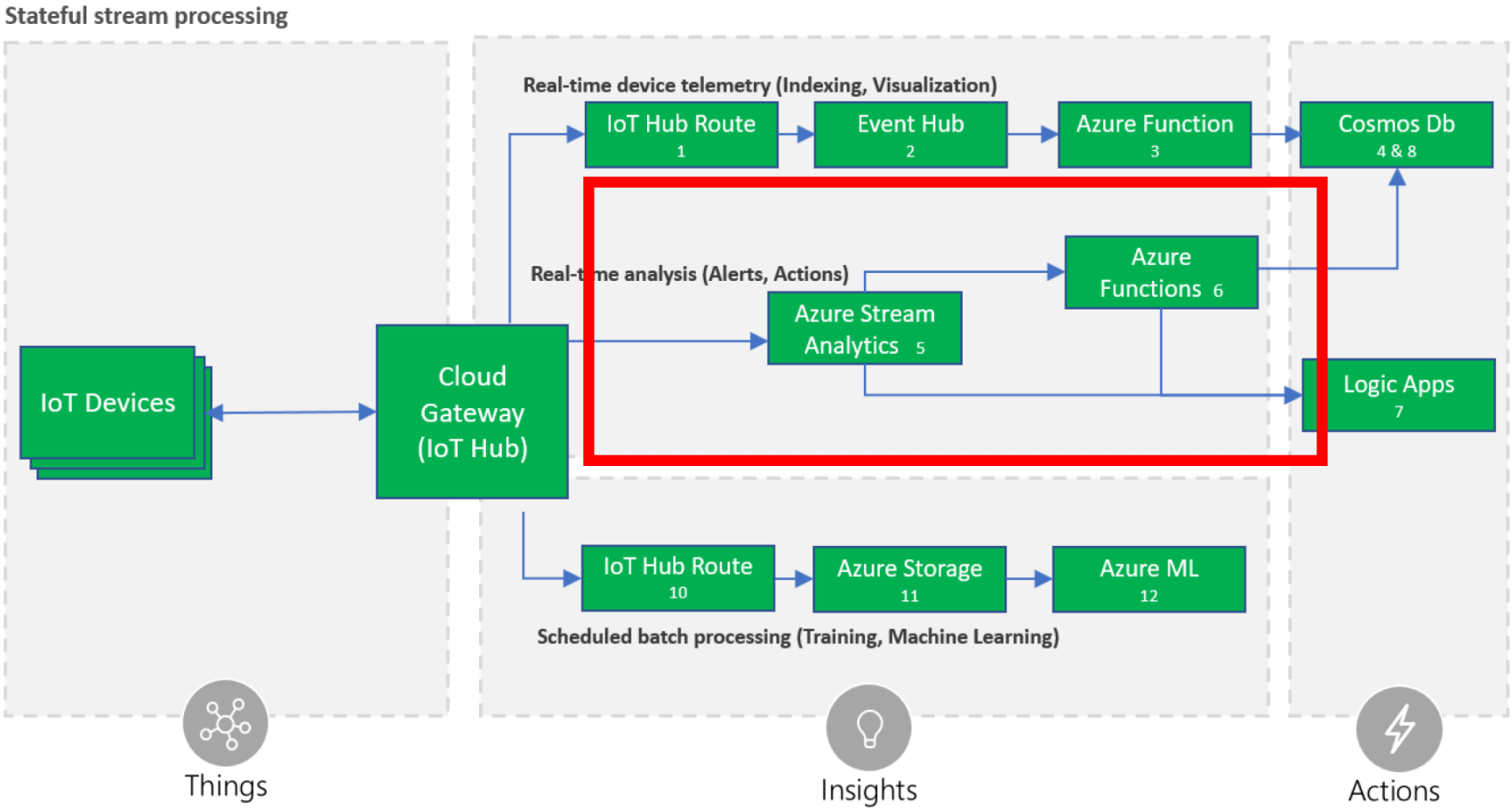
- High availability due to geographic redundancy and quick disaster recovery features of the Azure services.
- Low cost: most of the components automatically scale, adapting to variable work load, minimizing the cost whenever there is no data to process.
- Minimal operational costs, because all the components are managed Azure services.
- Flexibility: Azure Functions and Azure Cosmos DB allow transformation of ingested data to any desired schema, supporting multiple access patterns and APIs like MongoDB, Cassandra and Graph APIs.
- Actions and Business Integration: A wide choice of integrations are available via Logic Apps and Azure ML.

Stateful Stream Processing

This architecture describes a fast, flexible and scalable solution for stateful real-time analysis of ingested data records in multiple formats, with the ability to reference external data.

The architecture includes the same slow path seen in the previous architecture for use with machine learning and other complex analysis not possible in the fast path.

The architecture is similar to the solution recommended for stateless processing **only the analysis path is replaced with Azure Stream Analytics (ASA) that is designed for hyper-scale analysis and routing of data records, in a stateful fashion**, with the ability to apply complex queries over time periods and multiple streams.



Architecture Benefits

- High availability due to geographic redundancy and quick disaster recovery features of Azure services.
- Minimal operational costs, because all the components are managed Azure services.
- Azure Stream Analytics ability to execute complex analysis at scale, for instance use of tumbling/sliding/hopping windows, stream aggregations, and external data source joins.
- Flexibility: Azure Functions and Cosmos DB allow transformation of ingested data to any desired schema, supporting multiple access patterns and APIs like MongoDB, Cassandra and Graph APIs.
- Actions and Business Integration: A wide choice of integrations are available via Logic Apps and Azure ML.
- Performance: Support for binary data streams, in order to reduce latency.