Azure IoT Developer Specialty Crash Course

Microsoft Azure IoT Developer (AZ-220)

December/2020

Reza Salehi

Cloud Consultant and Trainer
Pluralsight author, O'Reilly Media instructor

@zaalion

Microsoft CERTIFIED Trainer

2008 - 2018

Microsoft CERTIFIED
AZURE IOT DEVELOPER
SPECIALTY

Course Overview

AI-220 Candidate Profile

- Should have subject matter expertise in:
  - Developing cloud and edge components of an Azure IoT solution.
  - Other related Azure services
    - AI, storage, security
    - Integration, monitoring, etc.

Azure IoT Developer Role

- Using cloud services and other tools
  - Manage the IoT device lifecycle,
  - Set-up, configuration and maintenance
  - Implement designs for Azure IoT solutions (device topology, connectivity, debugging, and security)
  - Manage, monitor, and transform IoT-related data pipelines
  - Deploy Azure IoT Edge components and configure device networking on the edge.

Skills Measured in AI-220

- Implement the IoT solution infrastructure (15-20%)
- Implement Edge (15-20%)
- Process and manage data (15-20%)
- Monitor, troubleshoot, and optimize IoT solutions (15-20%)
- Implement security (15-20%)

Skills Measured in AI-220

- Exam page:
  - https://docs.microsoft.com/en-us/learn/certifications/exams/az-220
- Skills outline:
  - https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE4nBeC

IoT (Internet of Things)

“Describes the network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the Internet.”

https://en.wikipedia.org/wiki/Internet_of_things

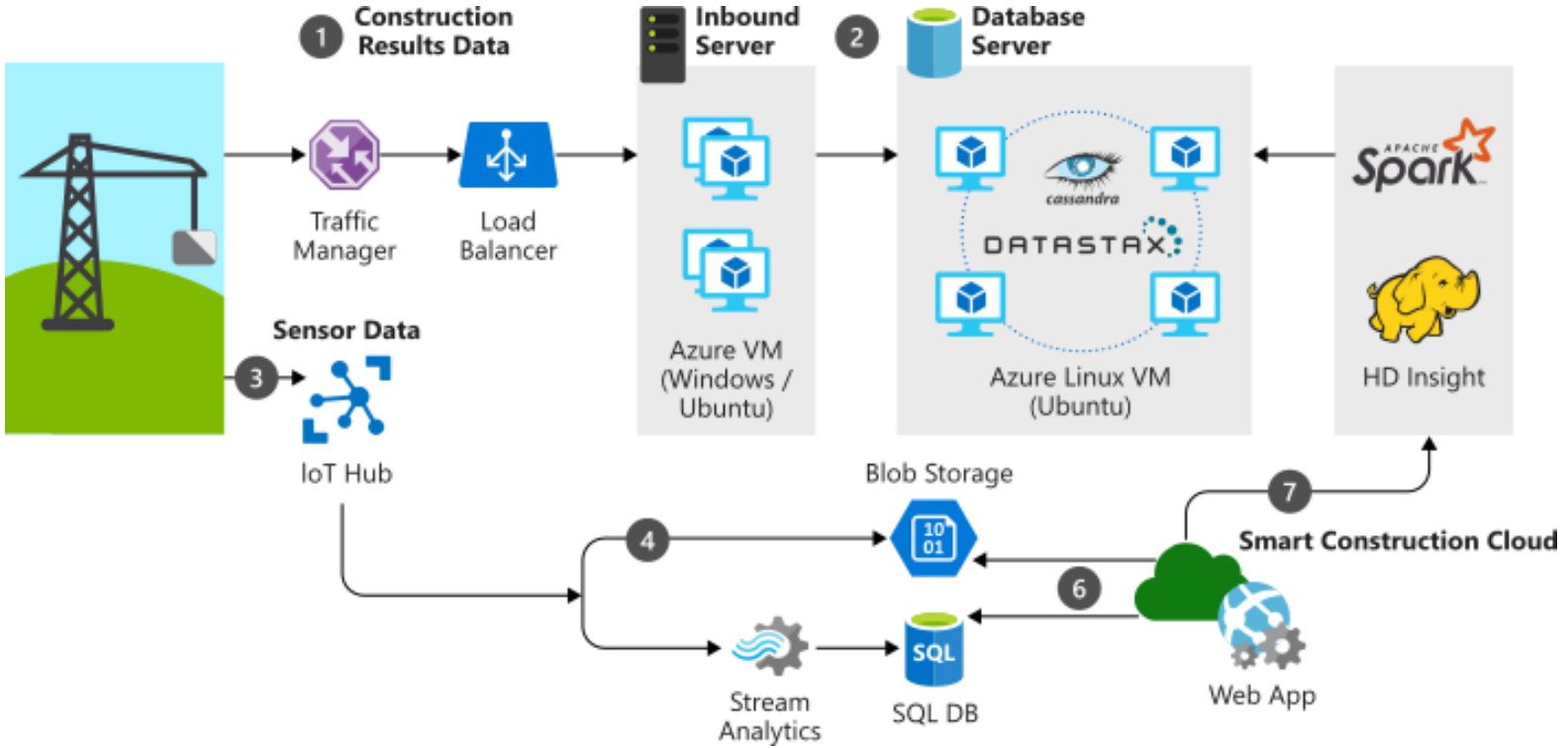
IoT (Internet of Things)

- Connected devices
- Using the public Internet
- Some sort of sensor (camera, heat, pressure, sounds, magnetic, etc.)
- May sends data in high volumes, frequently
- Talks with a backend service, which collects and processes the sensor data

IoT (Internet of Things)

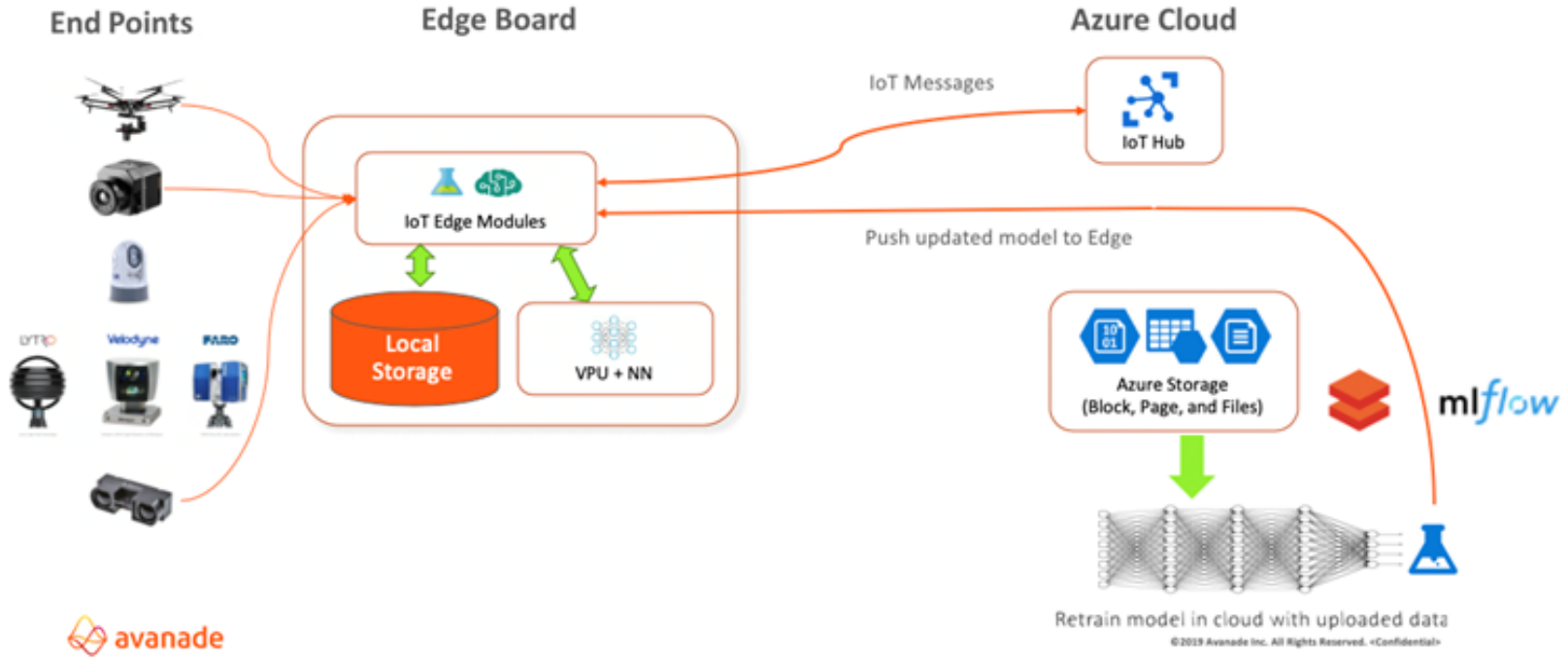
- Azure Architecture Center
  - https://docs.microsoft.com/en-us/azure/architecture/browse/#internet-of-things

IoT (Internet of Things)



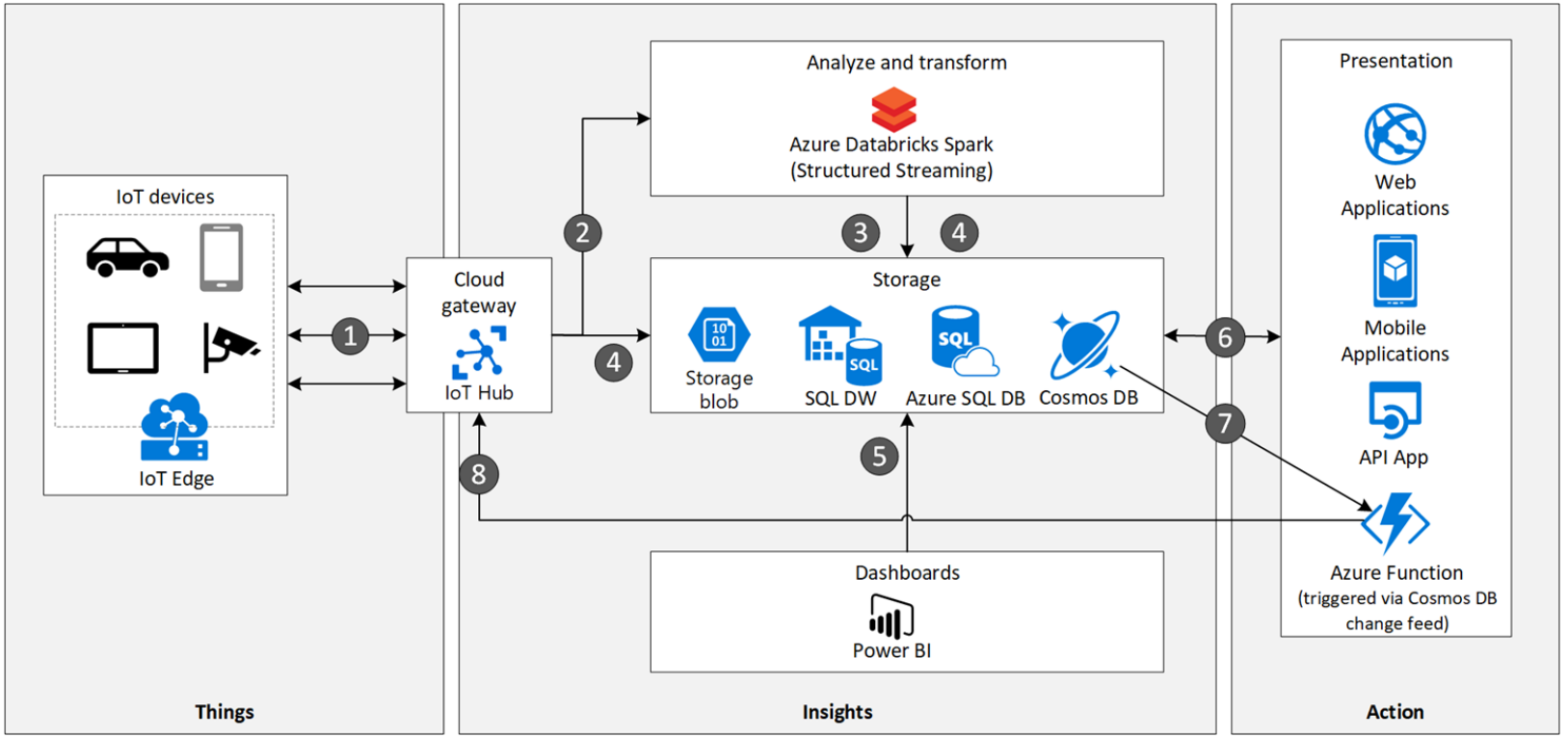
https://docs.microsoft.com/en-us/azure/architecture/example-scenario/data/big-data-with-iot

IoT (Internet of Things)



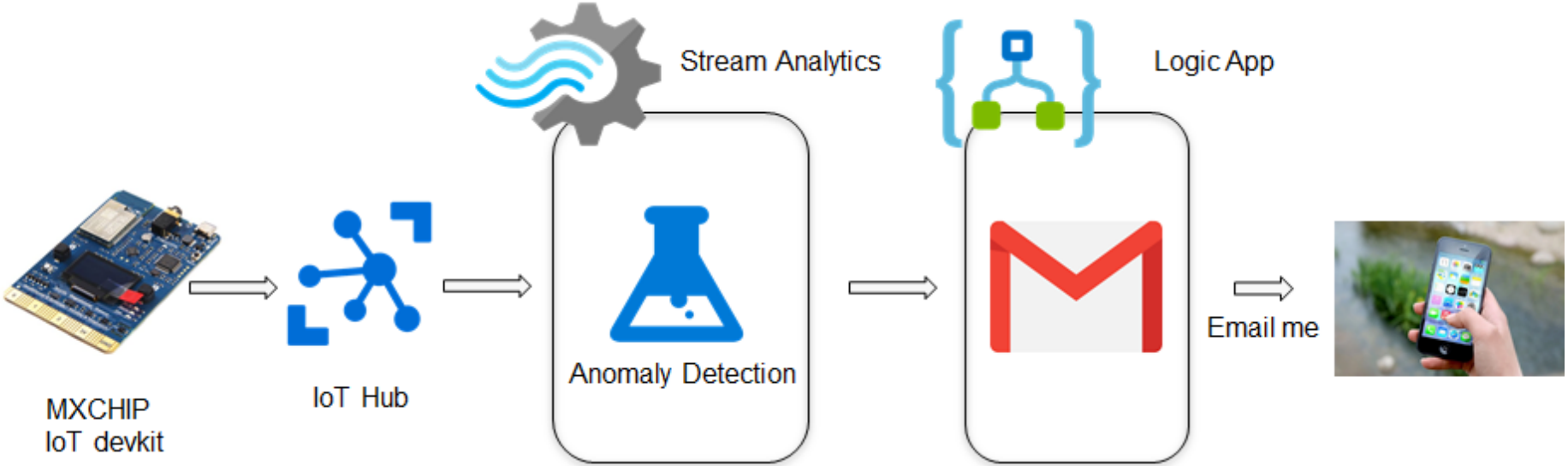
https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/contactless-interfaces

IoT (Internet of Things)



https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/iot-using-cosmos-db

IoT (Internet of Things)



https://microsoft.github.io/azure-iot-developer-kit/

IoT Hub

IoT (Internet of Things)

- An Azure IoT solution main components:
  - An IoT device
  - IoT Hub
  - IoT Edge
  - Azure Stream Analytics (or other live stream processing)
  - Data Storage (Azure SQL, Cosmos DB, Azure Storage Account, etc.)
  - Further processing (Azure Functions, App Services, Data Factory, etc.)

The Course Structure

- The topics are based on the exam skills.
  - https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE4nBeC

Questions & Resources

- Post questions in the QnA box
- Resources in the course repository
  - https://github.com/zaalion/oreilly-az-220

- Reach out:
  - Twitter: **@zaalion**

Implement the IoT Solution Infrastructure

Implement the IoT Solution Infrastructure

- Create and configure an IoT Hub
- Build device messaging and communication
- Configure physical IoT devices

Create and Configure an IoT Hub

- Create an IoT Hub
- Register a device
- Configure a device twin
- Configure IoT Hub tier and scaling

IoT Hub

- A managed service, hosted in the cloud
- A central message hub for **bi-directional** communication between your IoT application and the devices.
- Reliable and secure communications between **millions** of IoT devices

IoT Hub Scaling and Tiers

- Scales to millions of simultaneously connected devices
- Millions of events per second
    - Basic and standard tiers
    - Partitions
        - Contain core components of Azure Event Hubs

IoT Hub Scaling and Tiers

- Tier upgrade (with no interruptions)
  - Add **units** within the IoT hub. For example: each additional unit in a B1 IoT hub allows for an additional 400,000 messages per day.
  - Change the **size** of the IoT hub. For example, migrate from the B1 tier to the B2 tier to increase the number of messages that each unit can support per day.
  - Upgrade to a higher tier. For example, upgrade from the B1 tier to the S1 tier for access to advanced features with the same messaging capacity.

IoT Hub Scaling and Tiers

- Message throughput
  - Traffic is measured on a per-unit basis.
  - When creating, you choose tier and edition, and set the number of units available.
  - Purchase up to 200 units for the B1, B2, S1, or S2 edition, or up to 10 units for the B3 or S3 edition
  - After creation, change the number of units available (within its edition), upgrade or downgrade between editions within its tier (B1 to B2), or upgrade from the basic to the standard tier (B1 to S1) without interrupting.

IoT Hub Scaling and Tiers

Tier edition	Sustained throughput	Sustained send rate
B1, S1	Up to 1111 KB/minute per unit (1.5 GB/day/unit)	Average of 278 messages/minute per unit (400,000 messages/day per unit)
B2, S2	Up to 16 MB/minute per unit (22.8 GB/day/unit)	Average of 4,167 messages/minute per unit (6 million messages/day per unit)
B3, S3	Up to 814 MB/minute per unit (1144.4 GB/day/unit)	Average of 208,333 messages/minute per unit (300 million messages/day per unit)

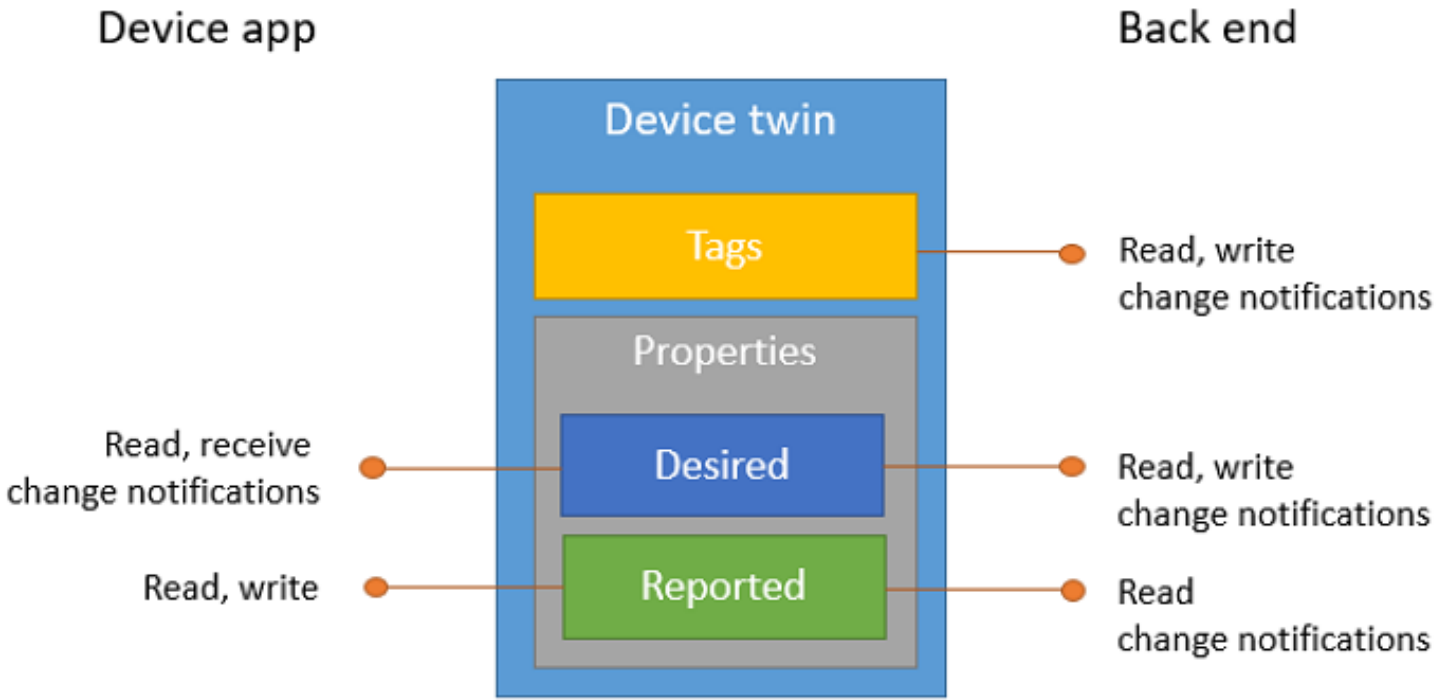
Register a Device

- You need to register your devices with the IoT Hub before being able to communicate with them.

Device Twin

- Device twins are JSON documents that store device state information including metadata, configurations, and conditions.
- Azure IoT Hub maintains a device twin for each device.

Device Twin


Device app
Back end
Device twin
Tags
Properties
Desired
Reported
Read, write
change notifications
Read, receive
change notifications
Read, write
change notifications
Read, write
Read
change notifications

Device Twin

- Use Device Twins to
  - Store device-specific metadata in the cloud. For example, the deployment location of a vending machine.
  - Report current state information such as available capabilities and conditions from your device app. For example, a device is connected to your IoT hub over cellular or WiFi.
  - Synchronize the state of long-running workflows between device app and back-end app. For example, when the solution back end specifies the new firmware version to install, and the device app reports the various stages of the update process.
  - Query your device metadata, configuration, or state.

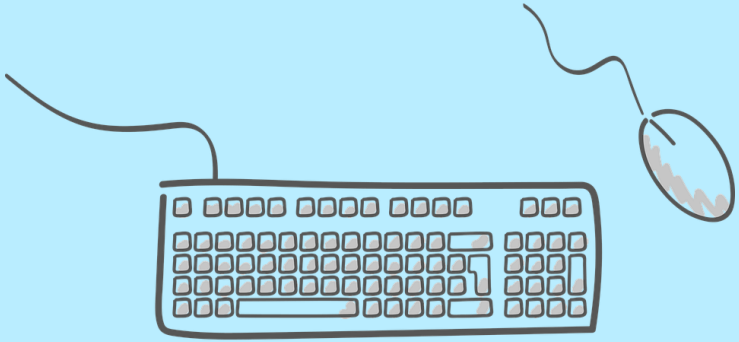
Device Twin

- Device Twins include:
  - **Tags**. A section of the JSON document that the solution back end can read from and write to. Tags are not visible to device apps.
  - **Desired properties**. Used along with reported properties to synchronize device configuration or conditions. The solution back end can set desired properties, and the device app can read them. The device app can also receive notifications of changes in the desired properties.
  - **Reported properties**. Used along with desired properties to synchronize device configuration or conditions. The device app can set reported properties, and the solution back end can read and query them.
  - **Device identity properties**. The root of the device twin JSON document contains the read-only properties from the corresponding device identity stored in the identity registry. Properties connectionStateUpdatedTime and generationId will not be included.

Device Twin


{
"deviceId": "devA",
"etag": "AAAAAAAAAAc=",
"status": "enabled",
"statusReason": "provisioned",
"statusUpdateTime": "0001-01-01T00:00:00",
"connectionState": "connected",
"lastActivityTime": "2015-02-30T16:24:48.789Z",
"cloudToDeviceMessageCount": 0,
"authenticationType": "sas",
"x509Thumbprint": {
"primaryThumbprint": null,
"secondaryThumbprint": null
},
"version": 2,
"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
}
}
}

Demo



- Create an IoT Hub
- Configure IoT Hub tier and scaling
- Register a device
- Device Twin

Build Device Messaging and Communication

- Build messaging solutions by using SDKs (device and service)
- Implement device-to-cloud communication
- Implement cloud-to-device communication
- Configure file upload for devices

Azure IoT Hub SDKs

- Two categories of software development kits (SDKs) for working with IoT Hub:
  - IoT Hub Device SDKs: build apps that run on your IoT devices using device client or module client.
  - IoT Hub Service SDKs: build backend applications to manage your IoT Hub, and optionally send messages, schedule jobs, invoke direct methods, or send desired property updates to your IoT devices or modules.

Azure IoT Hub SDKs

- In addition, we also provide a set of SDKs for working with the Device Provisioning Service. (DPS)
  - Provisioning Device SDKs: build apps that run on your IoT devices to communicate with the Device Provisioning Service.
  - Provisioning Service SDKs: build backend applications to manage your enrollments in the Device Provisioning Service.

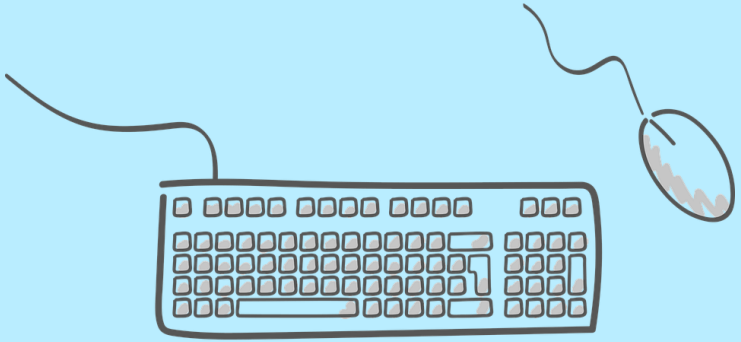
Device-to-cloud Communications

- When sending information from the device app to the solution back end, IoT Hub exposes three options:
  - Device-to-cloud messages for time series telemetry and alerts.
  - Device twin's reported properties for reporting device state information such as available capabilities, conditions, or the state of long-running workflows. For example, configuration and software updates.
  - File uploads for media files and large telemetry batches uploaded by intermittently connected devices or compressed to save bandwidth.

Cloud-to-device Communications

- IoT Hub provides three options for device apps to expose functionality to a device:
  - Direct methods for communications that require immediate confirmation of the result. Direct methods are often used for interactive control of devices such as turning on a fan.
  - Twin's desired properties for long-running commands intended to put the device into a certain desired state. For example, set the telemetry send interval to 30 minutes.
  - Cloud-to-device messages for one-way notifications to the device app.

Demo

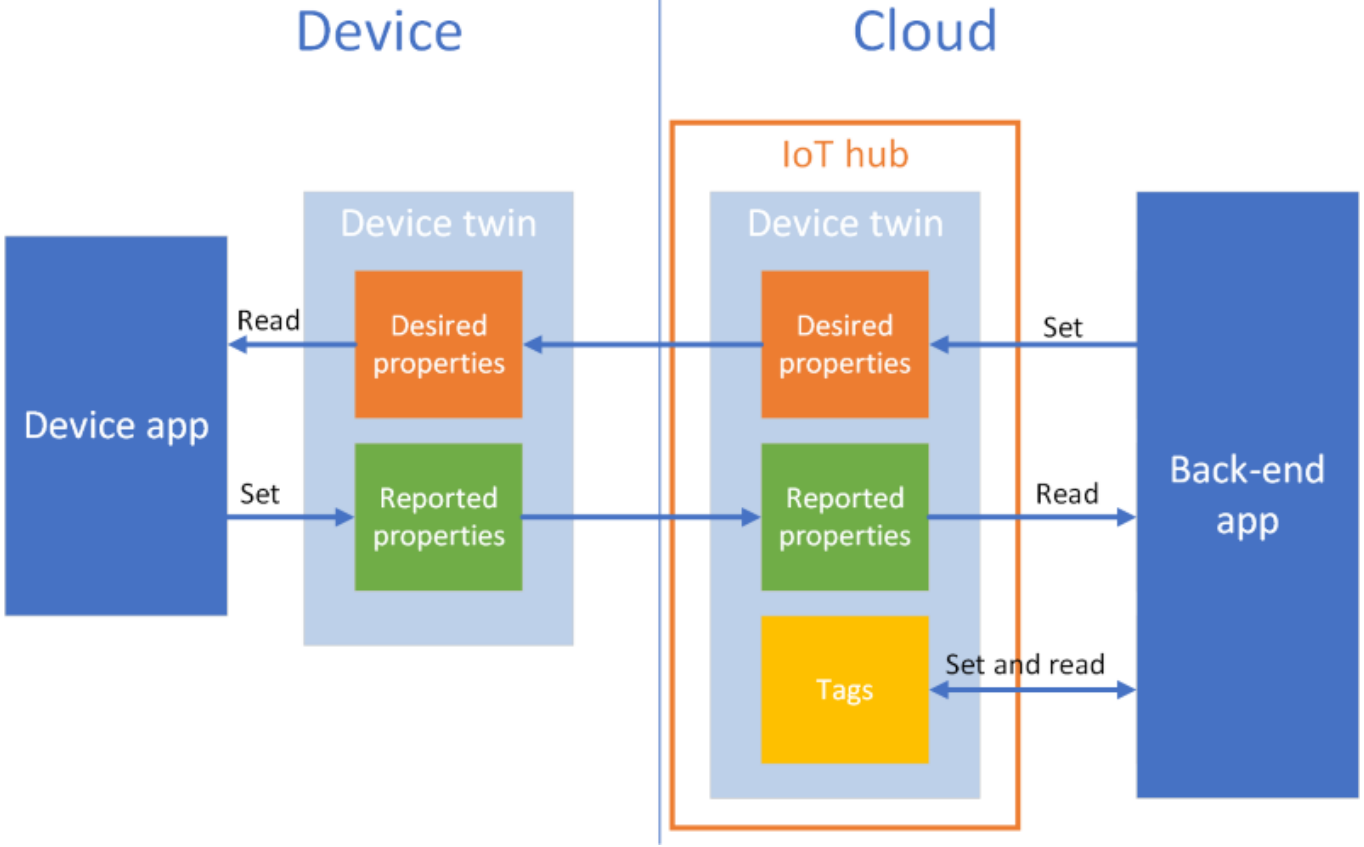


- IoT Hub SDKs (device and service)
- Device-to-cloud communication (D2C)
- Cloud-to-device communication (C2D)

Configure Physical IoT Devices

- Recommend an appropriate protocol based on device specifications
- Configure device networking, topology, and connectivity

IoT Hub Networking



Cloud-to-device Communications

- IoT Hub allows devices to use the following protocols for device-side communications:
  - MQTT
  - MQTT over WebSockets
  - AMQP
  - AMQP over WebSockets
  - HTTPS

Cloud-to-device Communications

Protocol	When you should choose this protocol
MQTT MQTT over WebSocket	Use on all devices that do not require to connect multiple devices (each with its own per-device credentials) over the same TLS connection.
AMQP AMQP over WebSocket	Use on field and cloud gateways to take advantage of connection multiplexing across devices.
HTTPS	Use for devices that cannot support other protocols.

Cloud-to-device Communications

- Consider the following points when you choose your protocol for device-side communications:
  - **Cloud-to-device pattern**.
  - **Field gateways**.
  - **Low resource devices**.
  - **Network traversal**.
  - **Payload size**.

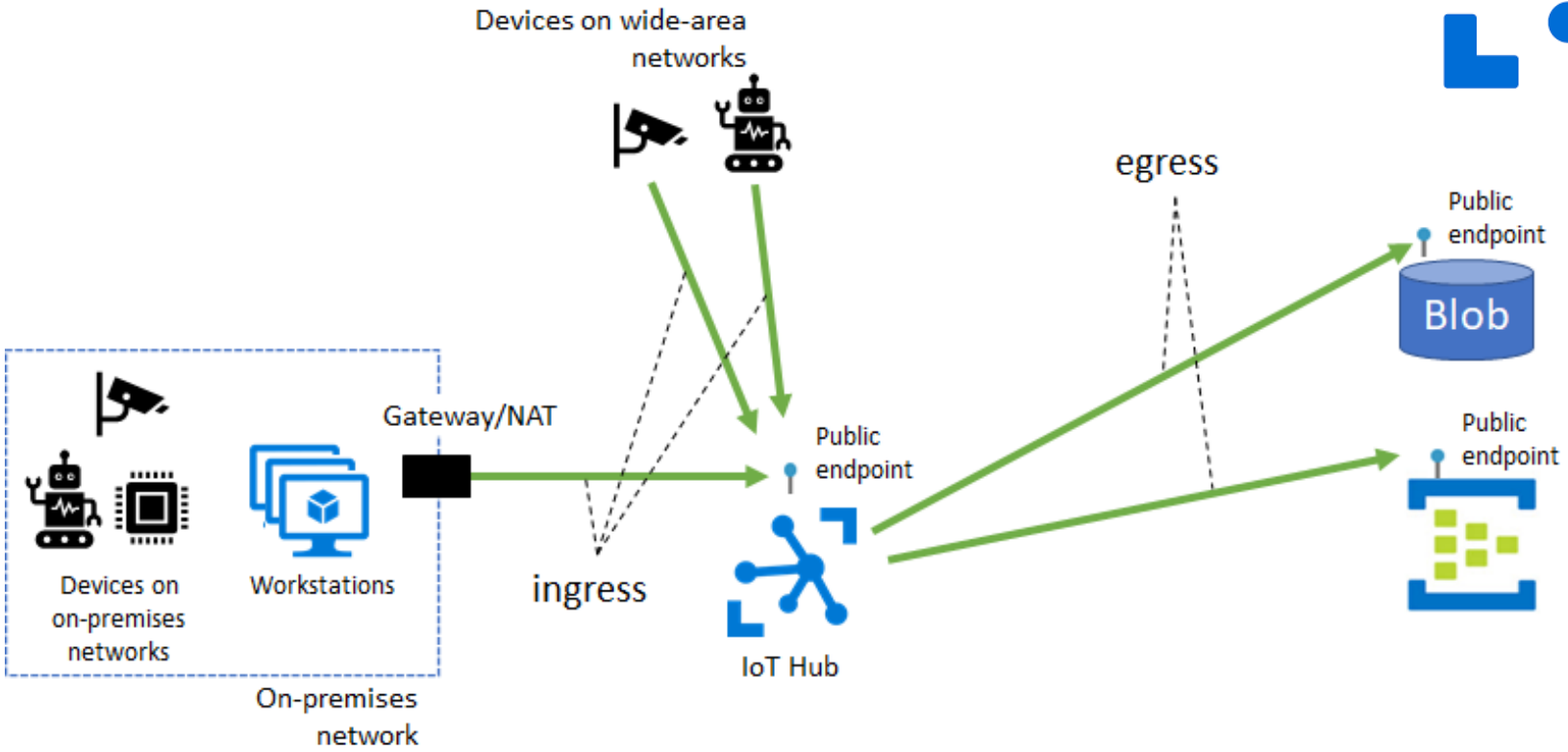
Protocol Limitations

- MQTT and HTTPS support only a single device identity.
- HTTPS does not support server push. When using HTTPS, devices poll IoT Hub for cloud-to-device messages. Poll for messages every 25 minutes or more.
- MQTT and AMQP support server push
- The MQTT and HTTPS libraries have a smaller footprint than the AMQP libraries.
- AMQP protocol uses port 5671, and MQTT listens on port 8883. Use MQTT over WebSockets and AMQP over WebSockets when firewall issues
- MQTT and AMQP are binary protocols, which result in more compact payloads than HTTPS.

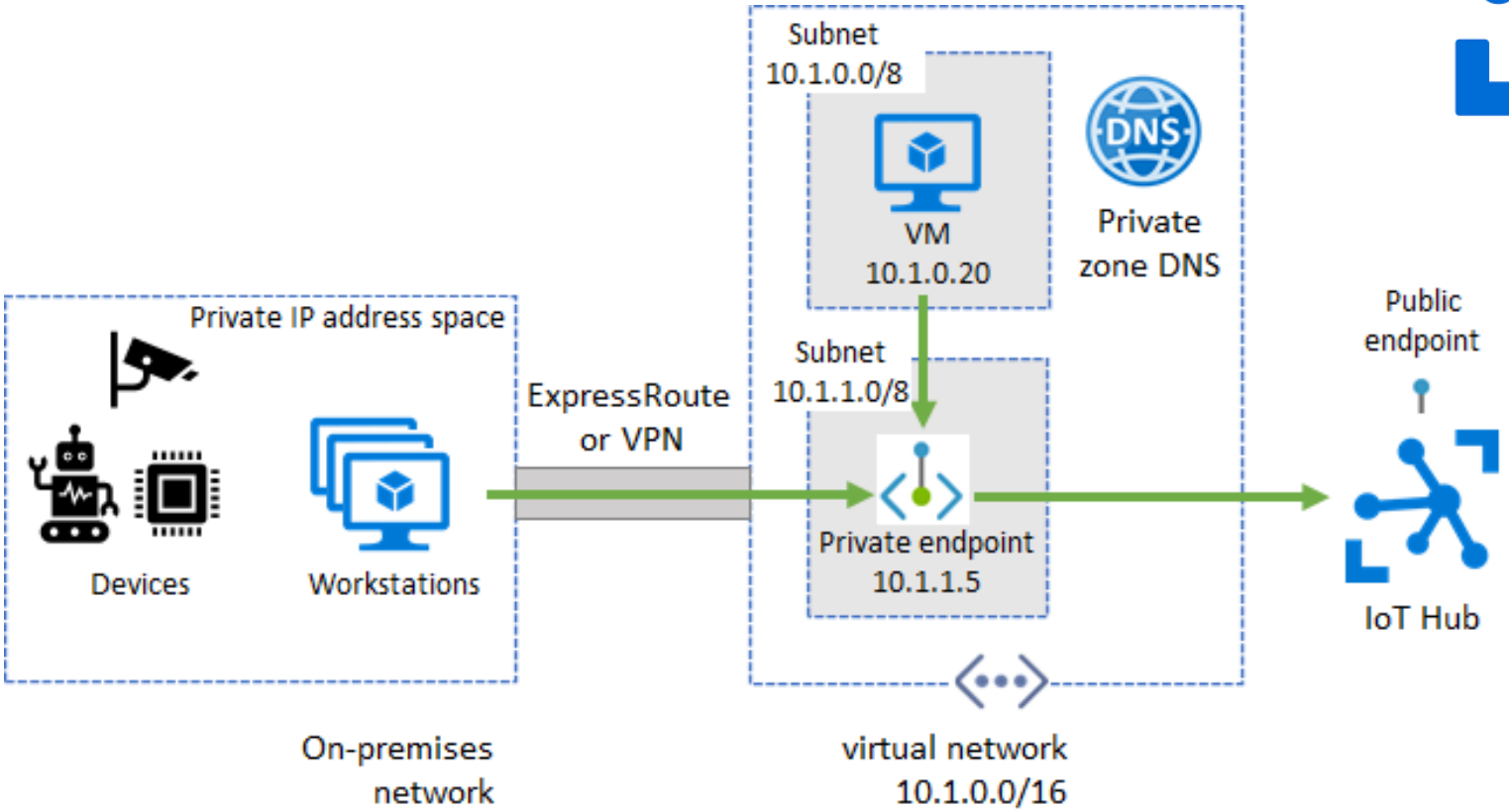
IoT Hub Networking

- By default, IoT Hub hostnames map to a public endpoint with a publicly routable IP address over the internet.
- You can restrict connectivity to your Azure resources (including IoT Hub) through a VNet that you own and operate.
- Setup ingress connectivity to IoT Hub using Azure Private Link.

IoT Hub Networking



IoT Hub Networking



Provision and Manage Devices

Provision and Manage Devices

- Implement the Device Provisioning Service (DPS)
- Manage the device lifecycle
- Manage IoT devices by using IoT Hub
- Build a solution by using IoT Central

Implement the Device Provisioning Service (DPS)

- Create a Device Provisioning Service
- Create a new enrollment in DPS (try simulated device here)
- Link an IoT Hub to the DPS
- Manage allocation policies by using Azure Functions

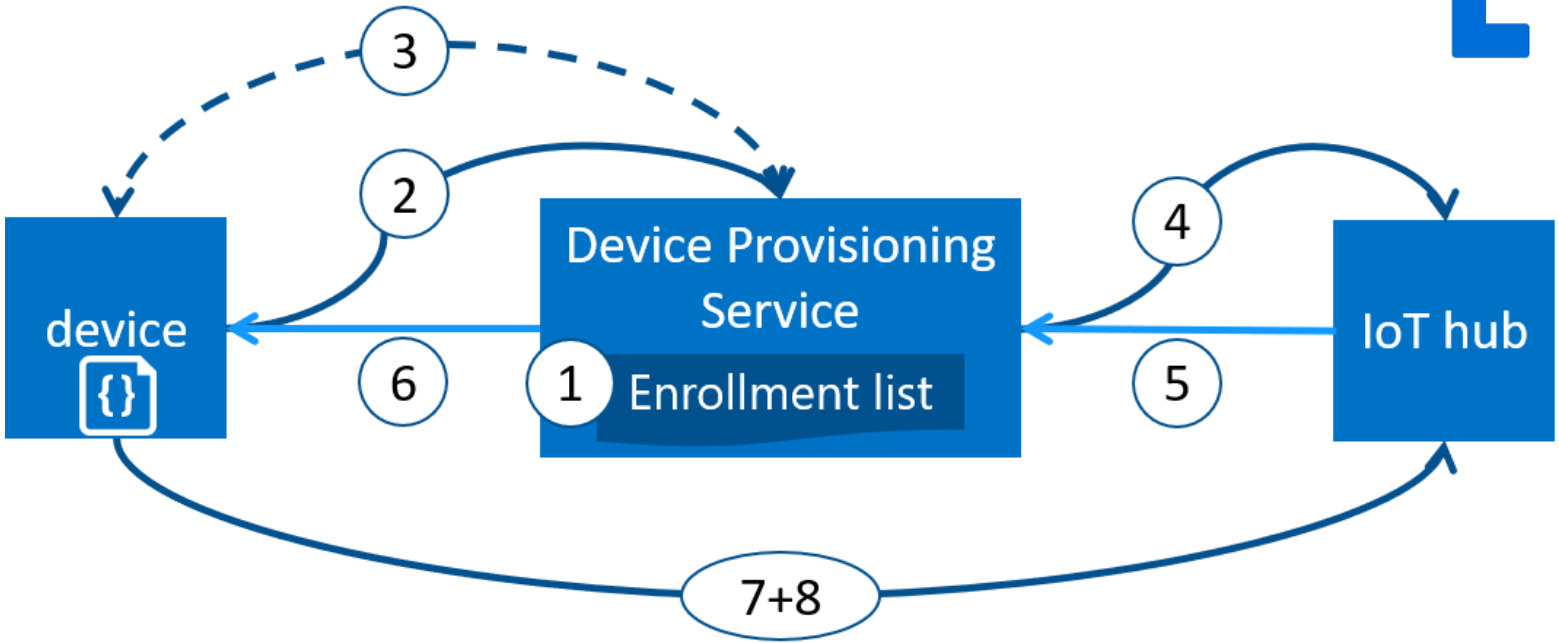
Device Provisioning Service (DPS)

- The IoT Hub Device Provisioning Service (DPS) is a helper service for IoT Hub
- DPS enables zero-touch, just-in-time provisioning to the right IoT hub without requiring human intervention
- DPS enables the provisioning of millions of devices in a secure and scalable manner

Device Provisioning Service (DPS)

- When to use DPS?
  - Provisioning to a single IoT solution without hardcoding IoT Hub connection information at the factory.
  - 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)

Device Provisioning Service (DPS)



https://docs.microsoft.com/en-us/azure/iot-dps/about-iot-dps#behind-the-scenes

Device Provisioning Service (DPS)

- DPS device provisioning steps:
    a. Device manufacturer adds the device registration information to the enrollment list in the Azure portal.
    b. Device contacts the DPS endpoint set at the factory. The device passes the identifying information to DPS to prove its identity.
    c. DPS validates the identity of the device by validating the registration ID and key against the enrollment list entry using either a nonce challenge (Trusted Platform Module) or standard X.509 verification (X.509).
    d. DPS registers the device with an IoT hub and populates the device's desired twin state.
    e. The IoT hub returns device ID information to DPS.
    f. DPS returns the IoT hub connection information to the device. The device can now start sending data directly to the IoT hub.
    g. The device connects to IoT hub.
    h. The device gets the desired state from its device twin in IoT hub.

Device Provisioning Service (DPS)

- DPS device provisioning process:
    a. **The manufacturing step** in which the device is created and prepared at the factory, and
    b. **The cloud setup step** in which the Device Provisioning Service is configured for automated provisioning.

Device Provisioning Service (DPS)

- Features of the Device Provisioning Service
    a. **Secure attestation** support for both X.509 and TPM-based identities.
    b. **Enrollment list** containing the complete record of devices/groups of devices that may at some point register.
    c. **Multiple allocation policies** to control how DPS assigns devices to IoT hubs in support of your scenarios: Lowest latency, evenly weighted distribution (default), and static configuration via the enrollment list.
    d. **Monitoring and diagnostics logging** to make sure everything is working properly.
    e. **Multi-hub support** allows DPS to assign devices to more than one IoT hub. DPS can talk to hubs across multiple Azure subscriptions.
    f. **Cross-region support** allows DPS to assign devices to IoT hubs in other regions.
    g. **Encryption for data at rest** allows data in DPS to be encrypted and decrypted transparently using 256-bit AES encryption.

Device Provisioning Service (DPS)

- Cross-platform support
    a. DPS works cross-platform with a variety of operating systems
    b. Azure offers open-source SDKs in a variety of languages to facilitate connecting devices and managing the service
    c. DPS supports the following protocols for connecting devices:
        i. HTTPS
        ii. AMQP
        iii. AMQP over web sockets
        iv. MQTT
        v. MQTT over web sockets
    d. DPS only supports HTTPS connections for service operations.

Device Provisioning Service (DPS)

- DPS custom allocation policies:
    a. DPS works cross-platform with a variety of operating systems
    b. Azure offers open-source SDKs in a variety of languages to facilitate connecting devices and managing the service
    c. DPS supports the following protocols for connecting devices:
        i. HTTPS
        ii. AMQP
        iii. AMQP over web sockets
        iv. MQTT
        v. MQTT over web sockets
    d. DPS only supports HTTPS connections for service operations.

Device Provisioning Service (DPS)

- Custom allocation policies
  - A custom allocation policy gives more control over how devices are assigned to an IoT hub.
  - This is accomplished by using custom code in an Azure Function to assign devices to an IoT hub.
  - The device provisioning service calls your Azure Function code providing all relevant information about the device and the enrollment.
  - The function code is executed and returns the IoT hub information used to provision the device.
  - By using custom allocation policies, you define your own allocation policies when the policies provided by the Device Provisioning Service don't meet the requirements of your scenario.

Device Provisioning Service (DPS)

- Custom allocation policies

Device

DPS

Assigns device to the selected IoT Hub

IoT Hub 1

IoT Hub 2

Custom allocation logic

Manage the Device Lifecycle

- Provision a device by using DPS
- Deprovision an autoenrollment
- Decommission (disenroll) a device

Device Provisioning Service (DPS)

- Auto-provision a device by using DPS
    - Using TPM attestation (Windows, Linux)
    - Using X.509 certificates attestation
    - Using symmetric key attestation
- Both individual and group enrollment

Device Provisioning Service (DPS)

- Deprovision devices that were previously **auto-provisioned**
  - You may find it necessary to deprovision devices that were previously auto-provisioned through the Device Provisioning Service.
    - For example, a device may be sold or moved to a different IoT hub, or it may be lost, stolen, or otherwise compromised.

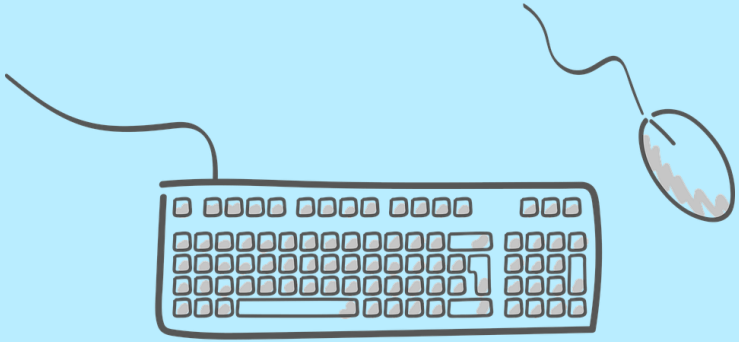
Device Provisioning Service (DPS)

- Deprovision devices that were previously auto-provisioned
  - Steps:
    i. Disenroll the device from your provisioning service, to prevent future auto-provisioning.
    ii. Deregister the device from your IoT Hub, to prevent future communications and data transfer.
  - Review steps for both individual device removal and Enrollment groups

Device Provisioning Service (DPS)

- Disenroll a device from Azure IoT Hub Device Provisioning Service (DPS)
  - A best practice for IoT systems is to have a clear plan of how to revoke access for devices when their credentials, whether a shared access signatures (SAS) token or an X.509 certificate, might be compromised.
    i. Disallow devices by using an individual enrollment entry
    ii. Disallow an X.509 intermediate or root CA certificate by using an enrollment group
    iii. Disallow specific devices in an enrollment group

Demo



- Create a Device Provisioning Service
- Create a new enrollment in DPS (try simulated device here)
- Link an IoT Hub to the DPS
- Provision a device by using DPS

Manage IoT Devices by Using IoT Hub

- Manage devices list in the IoT Hub device registry
- Modify device twin tags and properties
- Trigger an action on a set of devices by using IoT Hub Jobs and Direct Methods
- Set up Automatic Device Management of IoT devices at scale

IoT Hub Device Registry

- Identity registry in IoT Hub
    - Every IoT hub has an identity registry that stores information about the devices and modules permitted to connect to the IoT hub.
    - Before a device or module can connect to an IoT hub, there must be an entry for that device or module in the IoT hub's identity registry
    - A device or module must also authenticate with the IoT hub based on credentials stored in the identity registry.
        i. The device or module ID stored in the identity registry is case-sensitive.

IoT Hub Device Registry

- Use the identity registry when you need to:
  - Provision devices or modules that connect to your IoT hub.
  - Control per-device/per-module access to your hub's device or module-facing endpoints.

IoT Hub Device Registry

- Identity registry operations
  - Create device or module identity
  - Update device or module identity
  - Retrieve device or module identity by ID
  - Delete device or module identity
  - List up to 1000 identities
  - Export device identities to Azure blob storage **(The only way to retrieve all identities in an IoT hub's identity registry is to use the Export functionality.)**
  - Import device identities from Azure blob storage

IoT Hub Device Registry

> **Important**
>
> Only use the identity registry for device management and provisioning operations. High throughput operations at run time should not depend on performing operations in the identity registry. For example, checking the connection state of a device before sending a command is not a supported pattern. Make sure to check the **throttling rates** for the identity registry, and the **device heartbeat** pattern.

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-identity-registry#identity-registry-operations

Device Twins

- Modify device twin tags and properties
  - Device twins are JSON documents that store device state information including **metadata**, **configurations**, and **conditions**.
  - IoT Hub maintains a device twin for each device that you connect to IoT Hub.
- **Device Twins** are only available in the **standard** tier of IoT Hub

Device Twins

- Use device twins to:
  - Store device-specific metadata in the cloud.
  - Report current state information such as available capabilities and conditions from your device app.
  - Synchronize the state of long-running workflows between device app and back-end app.
  - Query your device metadata, configuration, or state.

Device Twins

- A device twin is a JSON document that includes:
  - Tags.
  - Desired properties.
  - Reported properties.
  - Device identity properties.

Device Twins: Back-end Operations

- Back-end operations to work with Device Twins
  - Retrieve device twin by ID
  - Partially update device twin
  - Replace desired properties
  - Replace tags
  - Receive twin notifications
  - In addition to these operations, the solution back end can:
    i. Query the device twins using the SQL-like IoT Hub query language.
    ii. Perform operations on large sets of device twins using jobs.

Device Twins: Device Operations

- The device app operates on the device twin using the following atomic operations:
  - Retrieve device twin.
  - Partially update reported properties.
  - Observe desired properties.

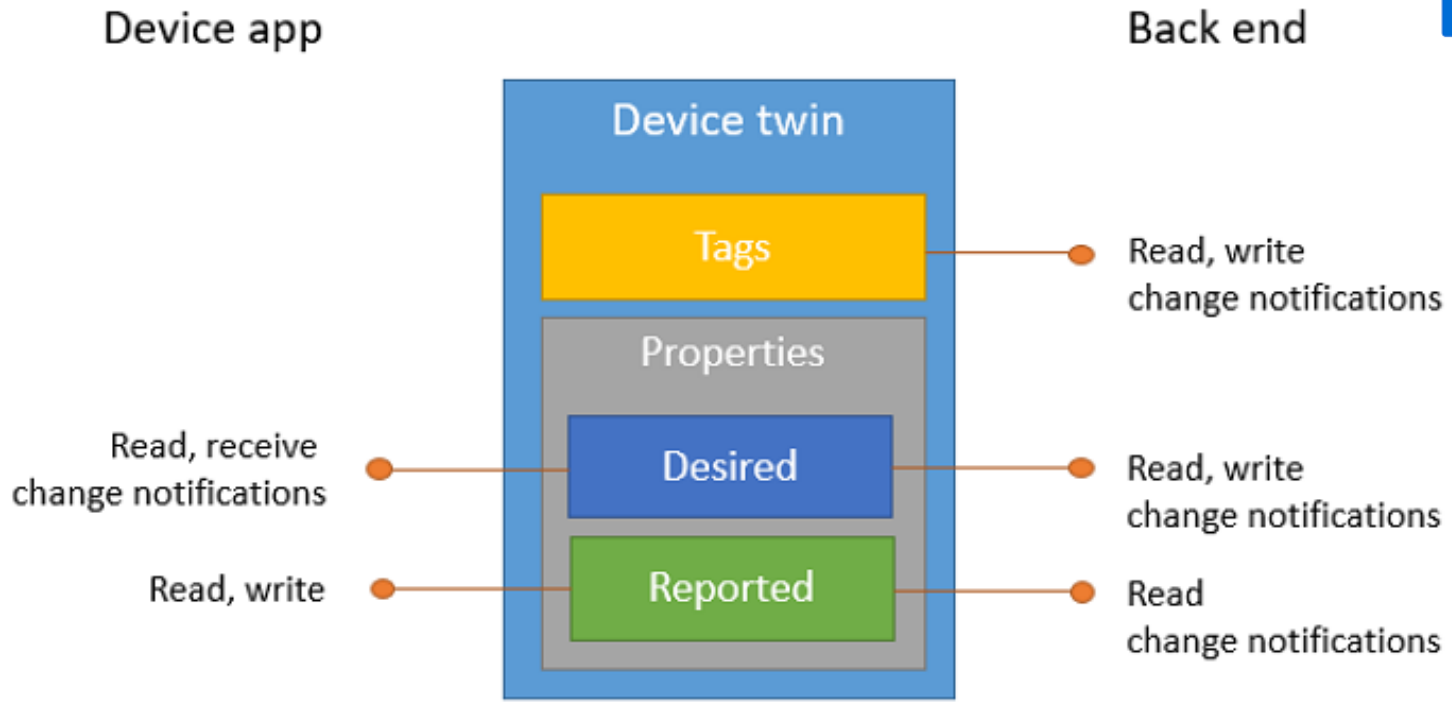
Device Twins: Facts

- Tags and properties format
  - Tags, desired properties, and reported properties are JSON objects with the following restrictions:
    i. **Keys**: All keys in JSON objects are UTF-8 encoded, case-sensitive, and up-to 1 KB in length. Allowed characters exclude UNICODE control characters (segments C0 and C1), and ., $, and SP.
    ii. **Values**: All values in JSON objects can be of the following JSON types: boolean, number, string, object. Arrays are also supported.
      - Integers can have a minimum value of -4503599627370496 and a maximum value of 4503599627370495.
      - String values are UTF-8 encoded and can have a maximum length of 4 KB.
    iii. **Depth**: **The maximum depth of JSON objects in tags, desired properties, and reported properties is 10**.

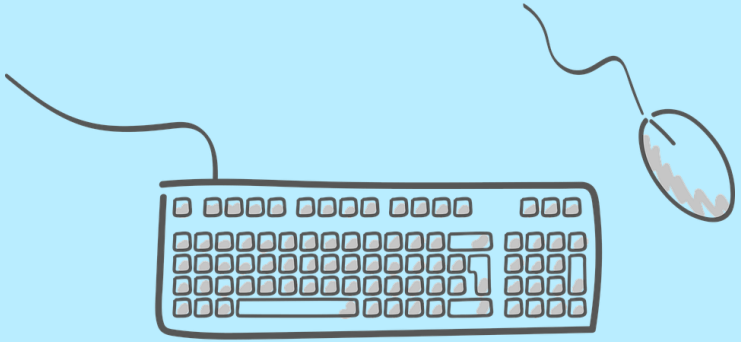
Device Twins: Facts

- Device twin size
  - IoT Hub enforces an 8 KB size limit on the value of tags, and a 32 KB size limit each on the value of desired and reported properties.
    i. These totals are exclusive of read-only elements like $etag, $version, and $metadata/$lastUpdated.

Device Twins



Demo



- Modify device twin tags and properties

Trigger an Action on a Device

- Trigger an action on a set of devices/device
  - Direct Methods
  - IoT Hub Jobs (Node.js sample)

Trigger an Action on a Device

- Direct Methods
  - This approach is useful for scenarios where the course of immediate action is different depending on whether the device was able to respond.
  - Each device method targets a **single** device.
  - Anyone with **service connect** permissions on IoT Hub may invoke a method on a device.
  - Direct methods follow a request-response pattern and are meant for communications that require **immediate confirmation** of their result
    i. See details

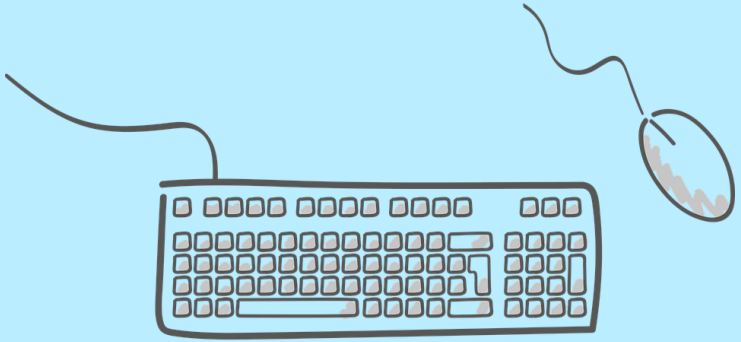
Trigger an Action on a Device

- Schedule a job on multiple devices
  - Jobs execute device twin updates and direct methods against a **set of devices** at a scheduled time.
  - Consider using jobs when you need to schedule and track progress any of the following activities on a set of devices:
    i. Update desired properties
    ii. Update tags
    iii. Invoke direct methods
  - See details

Automatic IoT Device Management

- Set up Automatic Device Management of IoT devices at scale
  - Automatic device management in Azure IoT Hub automates many of the repetitive and complex tasks of managing large device fleets.
  - Target a set of devices based on their properties, define a desired configuration, and then let IoT Hub update the devices when they come into scope.
  - This update is done using an **automatic device configuration** or **automatic module configuration**, which lets you **summarize completion and compliance**, h**andle merging and conflicts**, and **roll out configurations in a phased approach**.
- Create, monitor, modify and delete a device configuration in the portal

Demo



- Trigger an action on a set of devices/device using Direct Methods

Build a Solution by Using IoT Central

- Define a device type in Azure IoT Central
- Configure rules and actions in Azure IoT Central
- Define the operator view
- Add and manage devices from IoT Central
- Monitor devices
- Custom and industry-focused application templates
- Monitor application health using metrics

IoT Central

- IoT Central is a SaaS IoT platform
  - Reduces the burden and cost of developing, managing, and maintaining enterprise-grade IoT solutions. *(SaaS offering)*
  - Choosing to build with IoT Central allows you to focus more on the business problem and less on the complex IoT infrastructure.
  - The web UI lets you monitor device conditions, create rules, and manage millions of devices and their data throughout their life cycle.
  - It enables you to act on device insights by extending IoT intelligence into line-of-business applications.

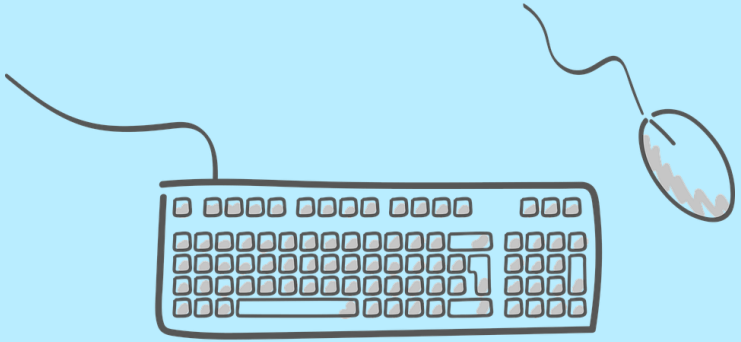
IoT Central Personas

- Personas who interact with an IoT Central application:
  - A **solution builder** is responsible for defining the types of devices that connect to the application and customizing the application for the operator.
  - An **operator** manages the devices connected to the application.
  - An **administrator** is responsible for administrative tasks such as managing user roles and permissions within the application.
  - A **device developer** creates the code that runs on a device or IoT Edge module connected to your application.

IoT Central Personas

- Create an IoT Central application
  - As a **solution builder**, you use IoT Central to create a custom, cloud-hosted IoT solution for your organization. A custom IoT solution typically consists of:
    i. A cloud-based application that receives telemetry from your devices and enables you to manage those devices.
    ii. Multiple devices running custom code connected to your cloud-based application.
  - Quickly deploy a new IoT Central application and then customize it to your specific requirements. You can start with a generic application template or with one of the industry-focused application templates for Retail, Energy, Government, or Healthcare.

Demo



- IoT Central overview

Process and Manage Data

Process and Manage Data

- Configure routing in Azure IoT Hub
- Configure stream processing
- Configure an IoT solution for Time Series Insights (TSI)

Configure Routing in Azure IoT Hub

- Implement message enrichment in IoT Hub
- Configure routing of IoT Device messages to endpoints
- Define and test routing queries
- Integrate with Event Grid

Implement Message Enrichment in IoT Hub

- Message enrichments for **device-to-cloud** IoT Hub messages
  - Message enrichments is the ability of the IoT Hub to stamp messages with additional information before the messages are sent to the designated endpoint.
    i. One reason to use message enrichments is to include data that can be used to simplify downstream processing.
       - For example, enriching device telemetry messages with a device twin tag can reduce load on customers to make device twin API calls for this information.

Implement Message Enrichment in IoT Hub

- A message enrichment has three key elements:
  - Enrichment name or key
    i. Is a string. A key can only contain alphanumeric characters or these special characters: hyphen (-), underscore (_), and period (.).
  - A value
    i. Any static string. Dynamic values such as conditions, logic, operations, and functions are not allowed.
    ii. The name of the IoT hub sending the message. This value is *$iothubname*.
    iii. Information from the device twin, such as its path. e.g. *$twin.tags.field* and *$twin.tags.latitude*.

Implement Message Enrichment in IoT Hub

- The messages can come from any data source supported by IoT Hub message routing, including the following examples:
  - device telemetry, such as temperature or pressure
  - device twin change notifications -- changes in the device twin
  - device life-cycle events, such as when the device is created or deleted

Implement Message Enrichment in IoT Hub

- Enrichments can be configured using the the following methods:
  - Azure portal
  - Azure CLI
  - Azure PowerShell

Implement Message Enrichment in IoT Hub

- Enrichments limitations
  - Add up to 10 enrichments per IoT Hub (standard or basic tier). For the free tier, add up to 2 enrichments.
  - Updates to a device twin can take up to five minutes to be reflected in the corresponding enrichment value.
  - The total message size, including the enrichments, can't exceed 256 KB. If a message size exceeds 256 KB, the IoT Hub will drop the message.
  - Message enrichments don't apply to digital twin change events.

Implement Message Enrichment in IoT Hub

- Enrichments limitations
  - In some cases, if you are applying an enrichment with a value set to a tag or property in the device twin, the value will be stamped as a string value. For example, if an enrichment value is set to $twin.tags.field, the messages will be stamped with the string "$twin.tags.field" rather than the value of that field from the twin. This happens in the following cases:
    i. Your IoT Hub is in the basic tier. Basic tier IoT hubs do not support device twins.
    ii. Your IoT Hub is in the standard tier, but the device sending the message has no device twin.
    iii. Your IoT Hub is in the standard tier, but the device twin path used for the value of the enrichment does not exist. For example, if the enrichment value is set to $twin.tags.location, and the device twin does not have a location property under tags, the message is stamped with the string "$twin.tags.location".

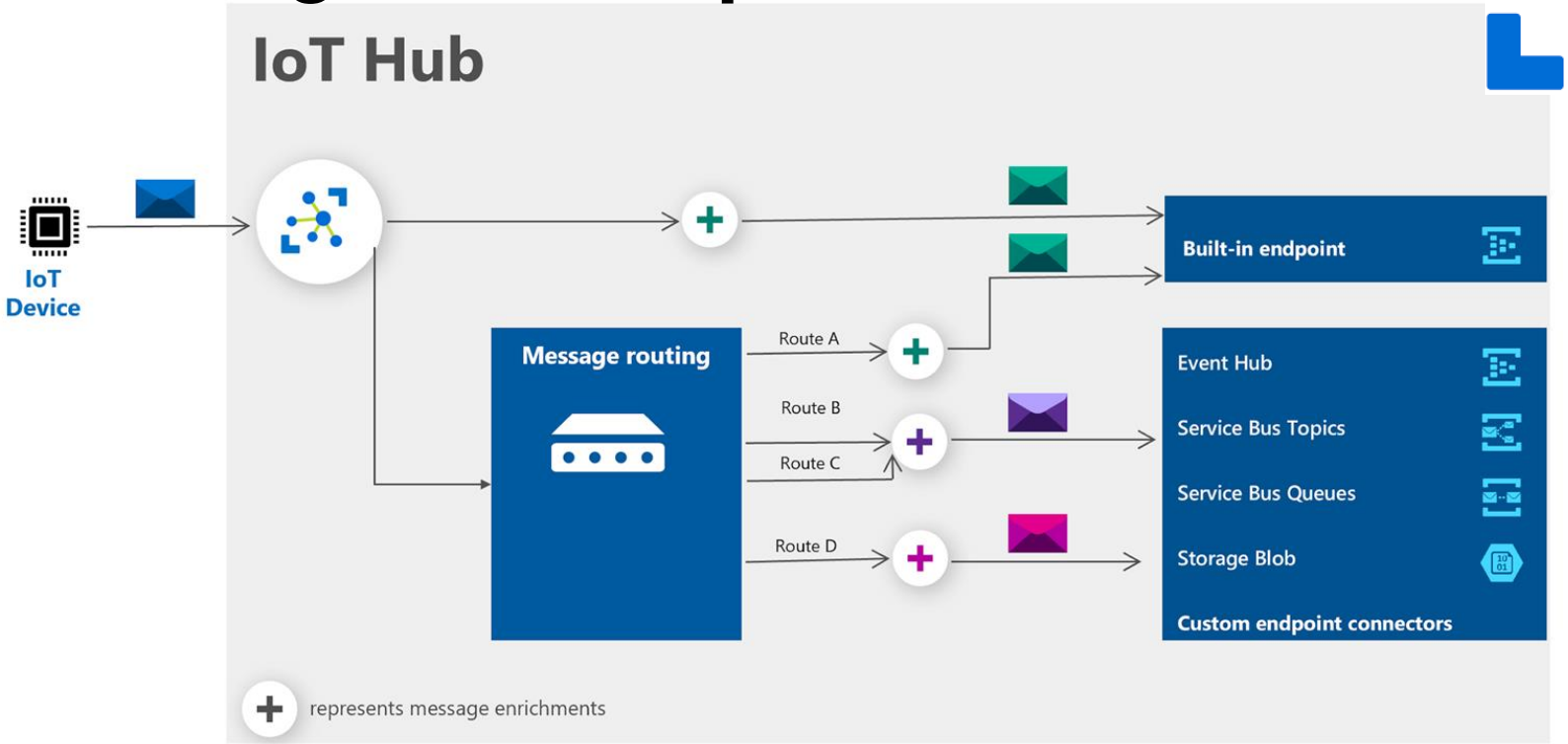
Implement Message Enrichment in IoT Hub

- Pricing
  - Message enrichments are available for no additional charge.

Configure Routing of IoT Device Messages to Endpoints

- Message routing enables sending telemetry data from your IoT devices to
  - Built-in Event Hub-compatible endpoints, or
  - Custom endpoints such as blob storage, Service Bus Queues, Service Bus Topics, and Event Hubs.
    i. To configure custom message routing, you create routing queries to customize the route that matches a certain condition.

Configure Routing of IoT Device Messages to Endpoints

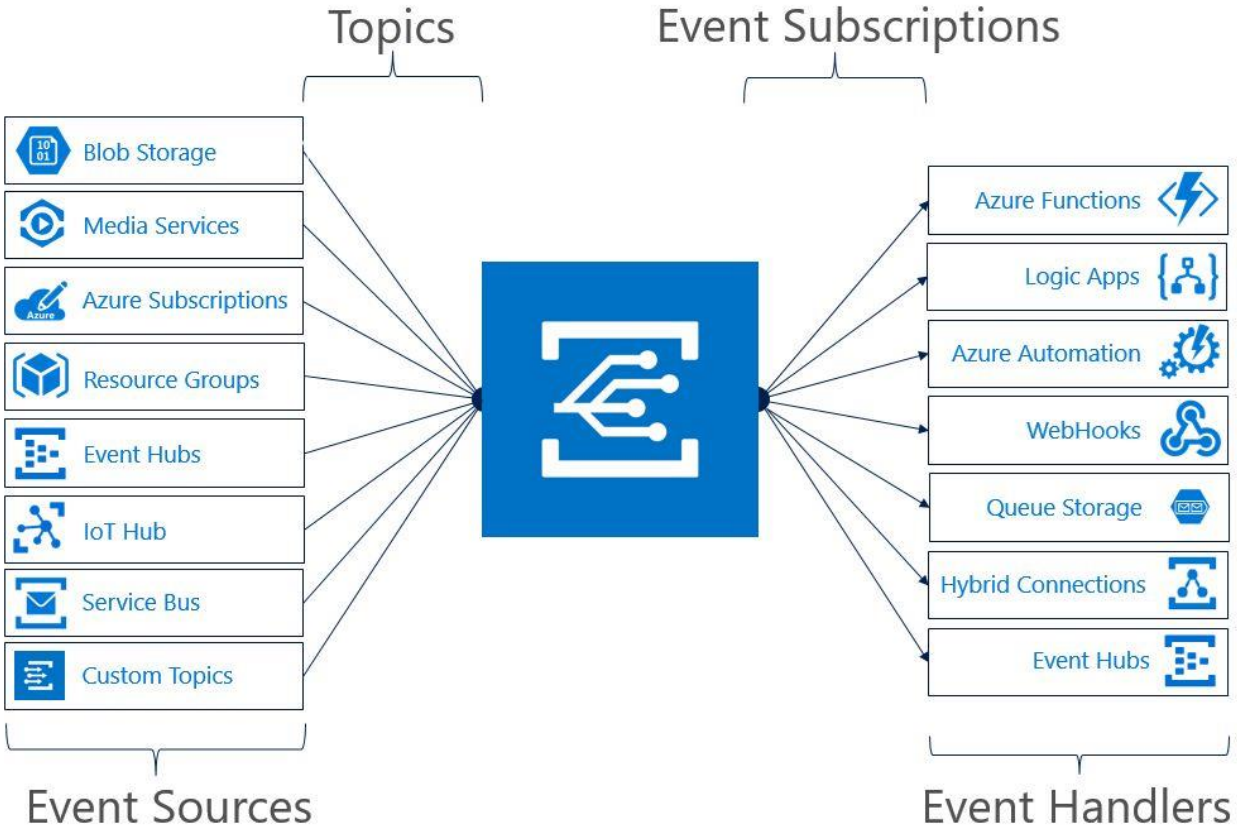


https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-message-enrichments-overview

Define and Test Routing Queries

- Message routing enables users to route different data types to various endpoints.
  - You can also apply rich queries to this data before routing it to receive the data that matters to you.
    i. Message routing query based on message properties
    ii. Message routing query based on message body
    iii. Message routing query based on device twin

Integrate IoT Hub with Event Grid

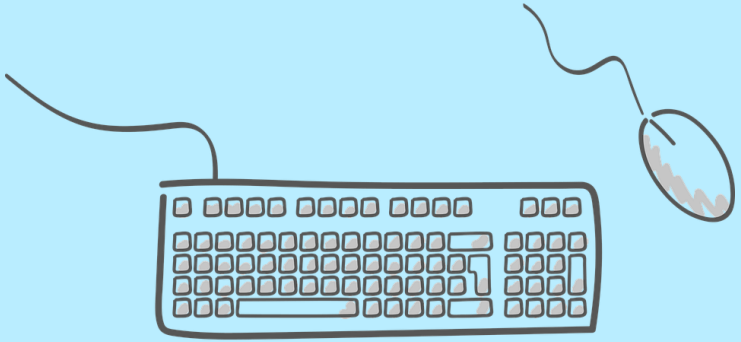


https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-event-grid

Integrate IoT Hub with Event Grid

- Azure IoT Hub integrates with Azure Event Grid so that you can send event notifications to other services and trigger downstream processes.
  - Configure your business applications to listen for IoT Hub events so that you can react to critical events in a reliable, scalable, and secure manner.
    i. For example, build an application that updates a database, creates a work ticket, and delivers an email notification every time a new IoT device is registered to your IoT hub.

Demo

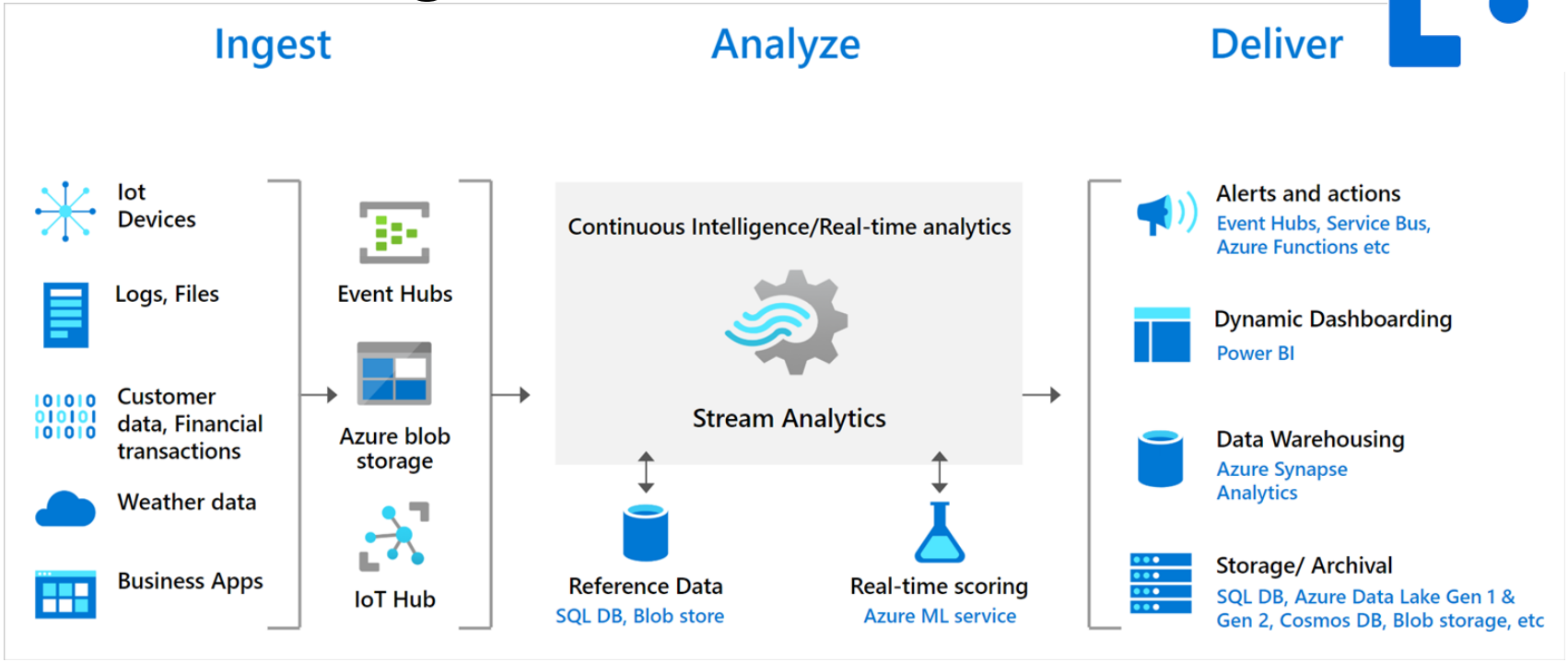


- Azure IoT Hub routing
- Message enrichment

Configure Stream Processing

- Create ASA for data and stream processing of IoT data
- Process and filter IoT data by using Azure Functions
- Configure Stream Analytics outputs

Create ASA for Data and Stream Processing of IoT Data



https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-introduction

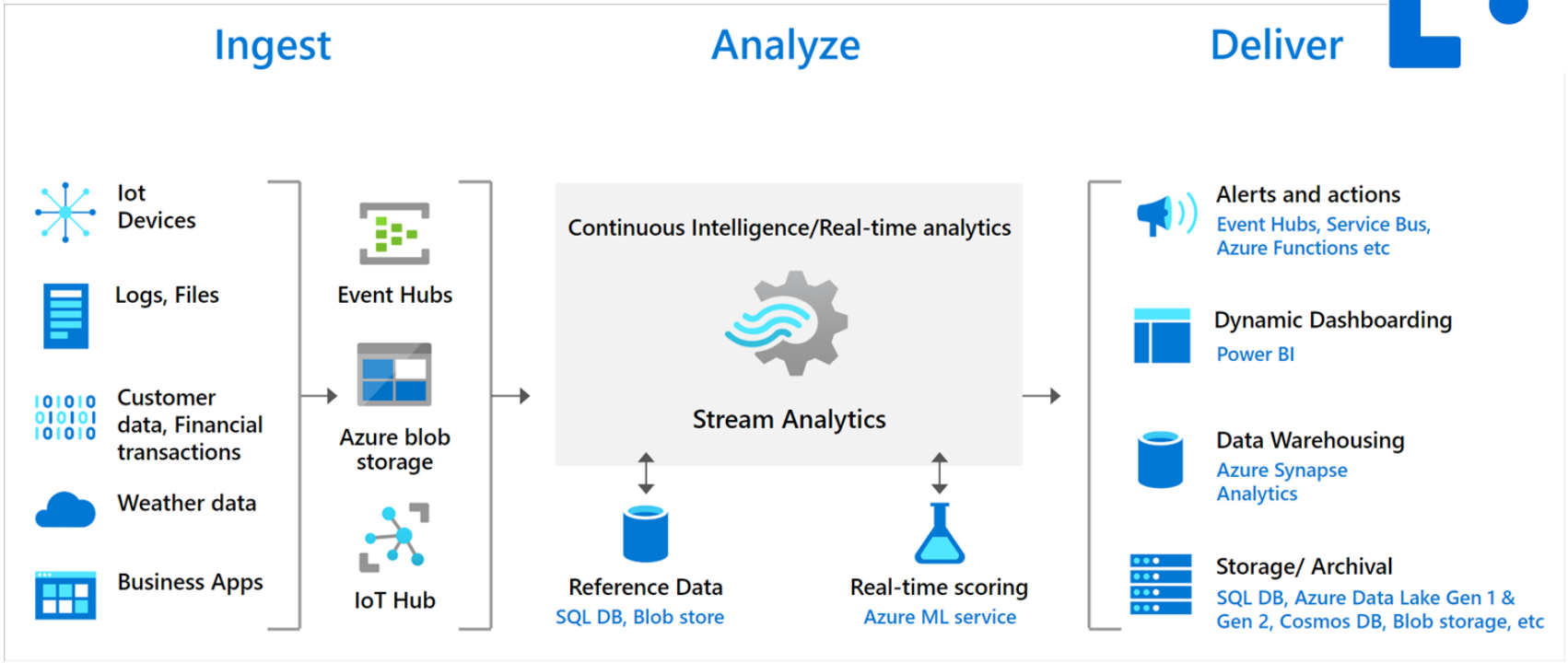
Create ASA for Data and Stream Processing of IoT Data

- Process real-time IoT data streams with Azure Stream Analytics
  - Create a Stream Analytics job
  - Create an Azure Stream Analytics query

Processing Data from IoT Hub with Azure Functions

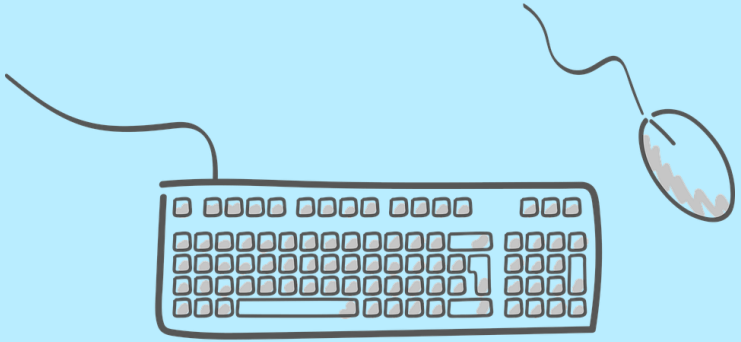
- Set up the integration between IoT Hub and Azure Functions
  - Setup IoT Hub
  - Azure Functions triggered by IoT Hub
  - Save/sent Function App output into the desired storage/service

Configure Stream Analytics Outputs



https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-define-outputs

Demo



- Azure Stream Analytics

Monitor, Troubleshoot, and Optimize IoT Solutions

Monitor, Troubleshoot, and Optimize IoT Solutions

- Configure health monitoring
- Troubleshoot device communication
- Perform end-to-end solution testing and diagnostics

Configure Health Monitoring

- Configure metrics in IoT Hub
- Set up diagnostics logs for Azure IoT Hub
- Query and visualize tracing by using Azure Monitor
- Use Azure Policy definitions for IoT Hub

IoT Hub Metrics

- IoT Hub metrics give you information about the state of the Azure IoT resources in your Azure subscription.
  - Assess the overall health of the IoT Hub service and the devices connected to it.
  - See what is going on with your IoT hub and help perform root-cause analysis on issues without needing to contact Azure support
- **Metrics are enabled by default. You can view IoT Hub metrics from the Azure portal.**

Set up Diagnostics Logs for Azure IoT Hub

- If you have an IoT Hub solution running in production, you want to set up some metrics and enable diagnostic logs
  - If a problem occurs, you have data to look at that will help you diagnose the problem and fix it more quickly.

Query and Visualize Tracing by Using Azure Monitor

- Log queries help you to fully leverage the value of the data collected in Azure Monitor Logs.

Use Azure Policy Definitions for IoT Hub

- Regulatory Compliance in Azure Policy provides Microsoft created and managed initiative definitions, known as built-ins, for the compliance domains and security controls related to different compliance standards.
  - Azure Security Benchmark
  - HIPAA HITRUST 9.2

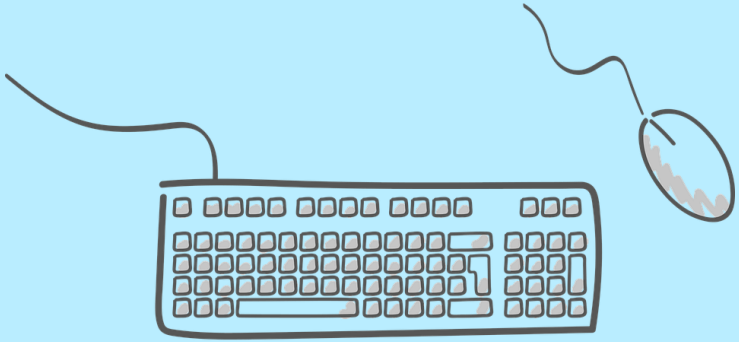
Troubleshoot Device Communication

- Establish maintenance communication
- Verify device telemetry is received by IoT Hub
- Validate device twin properties, tags and direct methods
- Troubleshoot device disconnects and connects

Perform End-to-end Solution Testing and Diagnostics

- Estimate the capacity required for each service in the solution
- Conduct performance and stress testing

Demo



- Configure metrics in IoT Hub
- Set up diagnostics logs for Azure IoT Hub
- Use Azure Policy definitions for IoT Hub

Implement Security

Implement Security

- Implement device authentication in the IoT Hub
- Implement device security by using DPS
- Implement Azure Security Center (ASC) for IoT

Implement Device Authentication in the IoT Hub

- Choose an appropriate form of authentication
- Manage the X.509 certificates for a device
- Manage the symmetric keys for a device

Choose an Appropriate Form of Authentication

- Options for securing your IoT Hub
  - Access control and permissions
  - Authentication
  - Protocol specifics
  - Scope IoT hub-level credentials
  - Security tokens
  - Supported X.509 certificates
  - Custom device and module authentication

Implement Device Security by Using DPS

- Configure different attestation mechanisms with DPS
- Generate and manage x.509 certificates for IoT Devices
- Configure enrollment with x.509 certificates
- Generate a TPM endorsements key for a device
- Configure enrollment with symmetric keys

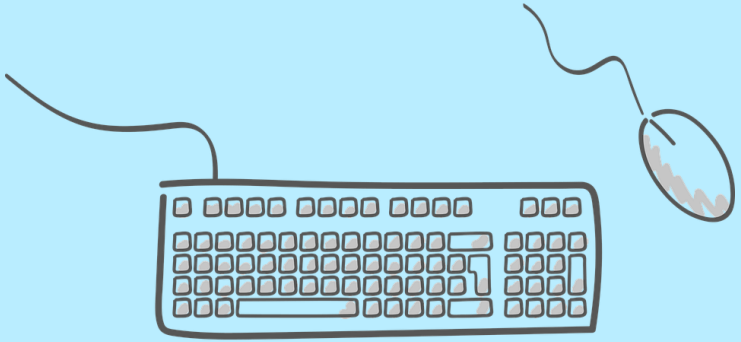
Configure Different Attestation Mechanisms with DPS

- Choose an attestation mechanism
  - Trusted Platform Module (TPM): TPM is an established standard for most Windows-based device platforms, as well as a few Linux/Ubuntu based devices.
  - X.509: X.509 certificates can be stored in relatively newer chips called Hardware Security Modules (HSM).

Implement Azure Security Center (ASC) for IoT

- Enable ASC for IoT in Azure IoT Hub
- Create security modules
- Configure custom alerts

Demo



- Azure Security Center and IoT Hub

Implement Edge

Azure IoT Edge

- Azure IoT Edge moves cloud analytics and custom business logic to devices
- Packaging your business logic into standard containers,
  - Then deploy those containers to any of your devices and monitor it all from the cloud.

Azure IoT Edge

- Azure IoT Edge is made up of three components:
  - **IoT Edge modules** are containers that run Azure services, third-party services, or your own code. Modules are deployed to IoT Edge devices and execute locally on those devices.
  - The **IoT Edge runtime** runs on each IoT Edge device and manages the modules deployed to each device.
  - A **cloud-based interface** enables you to remotely monitor and manage IoT Edge devices.

Set up and Deploy an IoT Edge Device

- Create a device identity in IoT Hub
- Deploy a single IoT device to IoT Edge
- Create a deployment for IoT Edge devices
- Install container runtime on IoT devices
- Define and implement deployment manifest
- Update security daemon and runtime
- Provision IoT Edge devices with DPS

Set up an Azure IoT Edge device

- With symmetric key authentication
- With X.509 certificate authentication

Deploy Azure IoT Edge Modules

- Each IoT Edge device runs at least two modules: $edgeAgent and $edgeHub, which are part of the IoT Edge runtime
- IoT Edge device can run multiple additional modules for any number of processes.
- Use a deployment manifest to tell your device which modules to install and how to configure them to work together.

Deployment Manifest

- The deployment manifest is a JSON document that describes:
    i. The **IoT Edge agent module twin**, which includes three components:
        - The container image for each module that runs on the device.
        - The credentials to access private container registries that contain module images.
        - Instructions for how each module should be created and managed.
    ii. The **IoT Edge hub module twin**, which includes how messages flow between modules and eventually to IoT Hub.
    iii. The desired properties of any additional module twins (optional).

Install or uninstall the Azure IoT Edge runtime

- The Azure IoT Edge runtime is what turns a device into an IoT Edge device.
- The runtime can be deployed on devices as small as a Raspberry Pi or as large as an industrial server.
- Once a device is configured with the IoT Edge runtime, you can start deploying business logic to it from the cloud

Update security daemon and runtime

- As the IoT Edge service releases new versions, you'll want to update your IoT Edge devices for the latest features and security improvements.
- Two components of an IoT Edge device need to be updated if you want to move to a newer version.
  - The first is the security daemon, which runs on the device and starts the runtime modules when the device starts.
  - The second component is the runtime, made up of the IoT Edge hub and IoT Edge agent modules.
- Depending on how you structure your deployment, the runtime can be updated from the device or remotely.

Set up and Deploy an IoT Edge Device

- IoT Edge automatic deployments
- Deploy on constrained devices
- Secure IoT Edge solutions
- Deploy production certificates

IoT Edge Automatic Deployments

- Azure IoT Edge provides two ways to configure the modules to run on IoT Edge devices.
  - The first method is to deploy modules on a per-device basis. You create a deployment manifest and then apply it to a particular device by name.
  - The second method is to deploy modules automatically to any registered device that meets a set of defined conditions. You create a deployment manifest and then define which devices it applies to based on tags in the device twin.

Deploy on Constrained Devices

- If you're deploying constrained devices with limited memory available, you can configure IoT Edge hub to run in a more streamlined capacity and use less disk space.
  - Don't optimize for performance on constrained devices
  - Disable unused protocols
  - Reduce storage time for messages

Secure IoT Edge Solutions

- Security standards for Azure IoT Edge
  - Standards
  - Authentication
  - Authorization
  - Attestation
  - Static attestation
  - Runtime attestation
  - Software attestation
  - Hardware root of trust
  - Certification
  - Extensibility

Deploy IoT Edge Solutions

- Deploy your IoT Edge solution in production
  - IoT Edge devices can be anything from a Raspberry Pi to a laptop to a virtual machine running on a server.
    i. Install production certificates
    ii. Have a device management plan
    iii. Use Moby as the container engine

Develop Modules

- Create and configure an Edge module
- Deploy a module to an Edge device
- Publish an IoT Edge module to an Azure Container Registry

IoT Edge Modules

- Azure IoT Edge lets you deploy and manage business logic on the edge in the form of modules.
- Azure IoT Edge modules are the smallest unit of computation managed by IoT Edge
- Can contain
  - **Azure services (such as Azure Stream Analytics)**, or
  - **Your own solution-specific code**.

IoT Edge Modules

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

Deploy IoT Edge Modules

- Deploy IoT Edge Modules
  - Individual devices
    i. Azure portal, CLI, Visual Studio Code
  - At scale
    i. Azure portal, CLI, Visual Studio Code

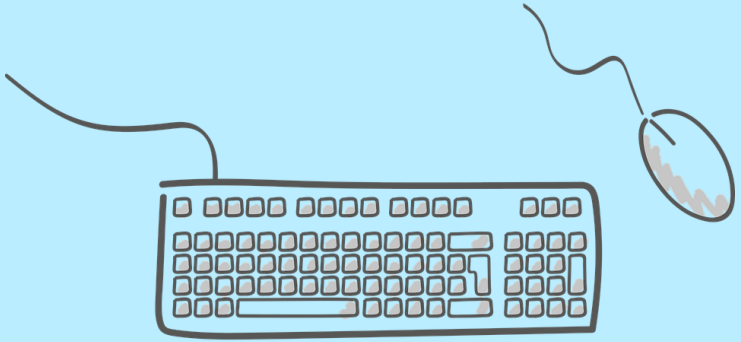
Configure an IoT Edge Device

- Select and deploy an appropriate gateway pattern
- Implement Industrial IoT solutions with modules like Modbus and OPC
- Implement module-to-module communication
- Implement and configure offline support (including local storage)

Select and Deploy an Appropriate Gateway Pattern

- Configure an IoT Edge device to function as a gateway
  - To act as a transparent gateway
  - Authenticate a downstream device to Azure IoT Hub
  - Connect a downstream device to an Azure IoT Edge gateway
  - Connect Modbus TCP devices through an IoT Edge device gateway

Demo



- IoT Edge overview

The Exam

Questions in AZ-220

- Multiple choice
- Drag and drop
- Scenario based
- There will be hands-on labs

AZ-220

- Exam AI-100 : https://docs.microsoft.com/en-us/learn/certifications/exams/az-220
- Skills measured : https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE4nBeC

solution, including data storage options, data analysis, data processing, and platform-as-a-service options. This role should also be able to recognize Azure IoT service configuration settings within the code portion of an IoT solution and perform specific IoT coding tasks in at least one Azure-supported language, including C#, Node, C, or Python.

Part of the requirements for: Microsoft Certified: Azure IoT Developer Specialty
Related exams: none
Important: See details
Go to Certification Dashboard

Schedule exam

Exam AZ-220: Microsoft Azure IoT Developer

United States

Languages: English, Japanese, Chinese (Simplified), Korean
Retirement date: none

This exam measures your ability to implement the IoT solution infrastructure; provision and manage devices; implement solutions on Azure IoT Edge devices; process and manage data; monitor, troubleshoot, and optimize IoT solutions; and implement security.

$165 USD*

Price based on the country in which the exam is proctored.

Schedule exam

Official practice test for Microsoft Azure IoT Developer
All objectives of the exam are covered in depth so you'll be ready for any question on the exam.

Skills measured

- The content of this exam was updated on September 24, 2020. Please download the exam skills outline below to see what changed.
- Implement the IoT solution infrastructure (15-20%)
- Provision and manage devices (20-25%)
- Implement Edge (15-20%)

Verify exam discount eligibility

For Microsoft employees

Microsoft employees are eligible for discounted exams. The discount will be reflected at the end of the checkout process. For MOS exams at Certiport, please request a voucher through the Microsoft Employee Voucher Portal.

To verify you are a Microsoft employee, link your Microsoft work account (alias@microsoft.com).

Link account

For Microsoft event attendees

If you recently attended a Microsoft event, you may be eligible for a discounted Microsoft Certification exam. To check eligibility, select an event you attended and verify the account used to register for the event.
Terms and Conditions apply.

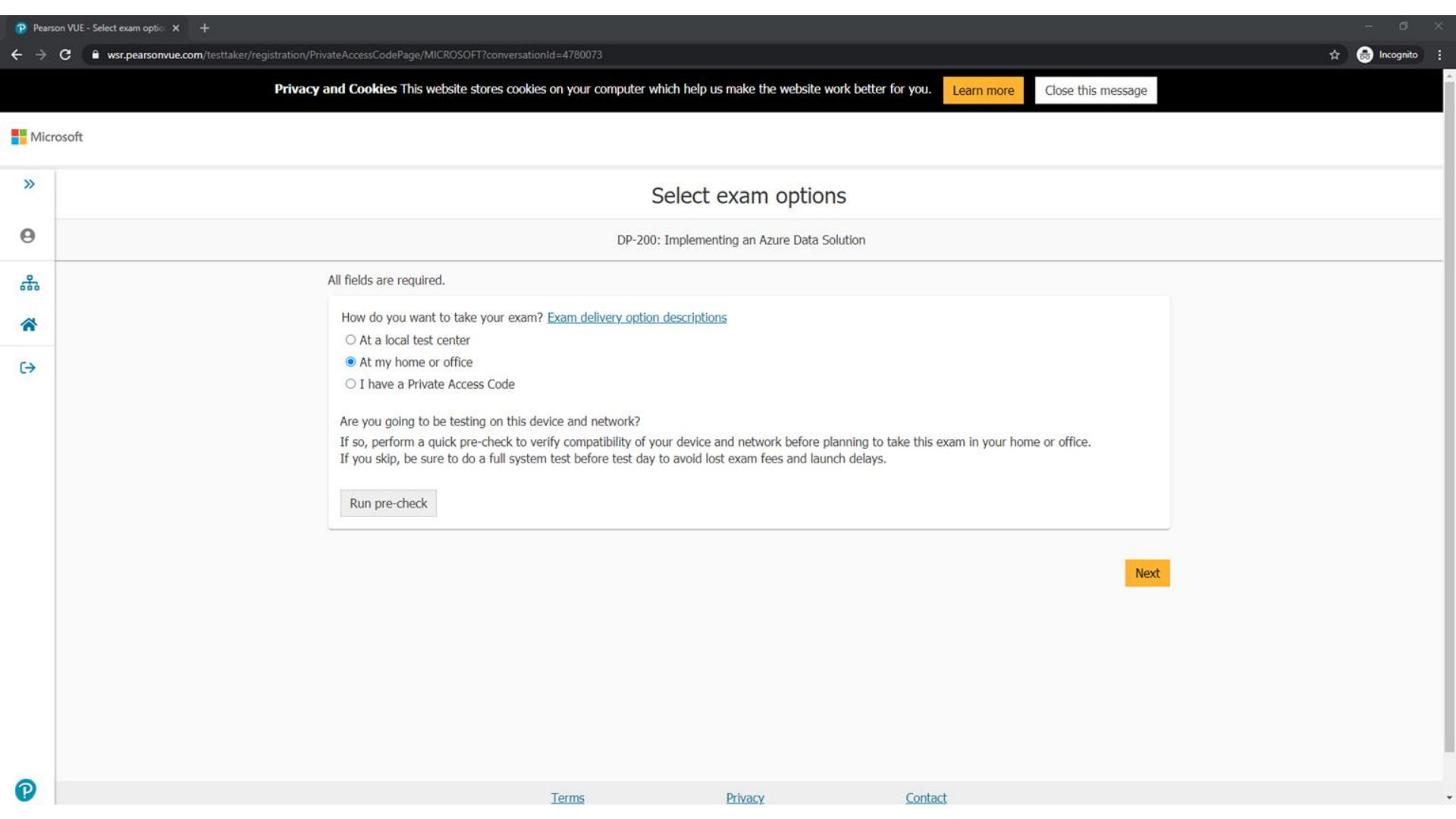
Microsoft Ignite 2019, Orlando

Verify account

Continue scheduling exam

Proceed to the Pearson VUE website to complete the exam scheduling process.

Go to Pearson VUE


Pearson VUE - Select exam optio
wsr.pearsonvue.com/testtaker/registration/PrivateAccessCodePage/MICROSOFT?conversationId=4780073
Incognito
Privacy and Cookies This website stores cookies on your computer which help us make the website work better for you.
Learn more
Close this message
Microsoft
Select exam options
DP-200: Implementing an Azure Data Solution
All fields are required.
How do you want to take your exam? Exam delivery option descriptions
At a local test center
At my home or office
I have a Private Access Code
Are you going to be testing on this device and network?
If so, perform a quick pre-check to verify compatibility of your device and network before planning to take this exam in your home or office.
If you skip, be sure to do a full system test before test day to avoid lost exam fees and launch delays.
Run pre-check
Next
Terms
Privacy
Contact

Pearson VUE - Select exam optic | System Check

systemcheck.proctorcam.com/systemcheck?candidateId=247036059

Incognito

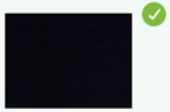
Pearson | VUE

System check - Checking your requirements

Microphone

Default - Microphone (SI

Internet speed



Webcam

Integrated Webcam (0c4

Next

Course Repository

https://github.com/zaalion/oreilly-az-220

Q&A

O'REILLY®

Thank you!

Reza Salehi

@zaalion