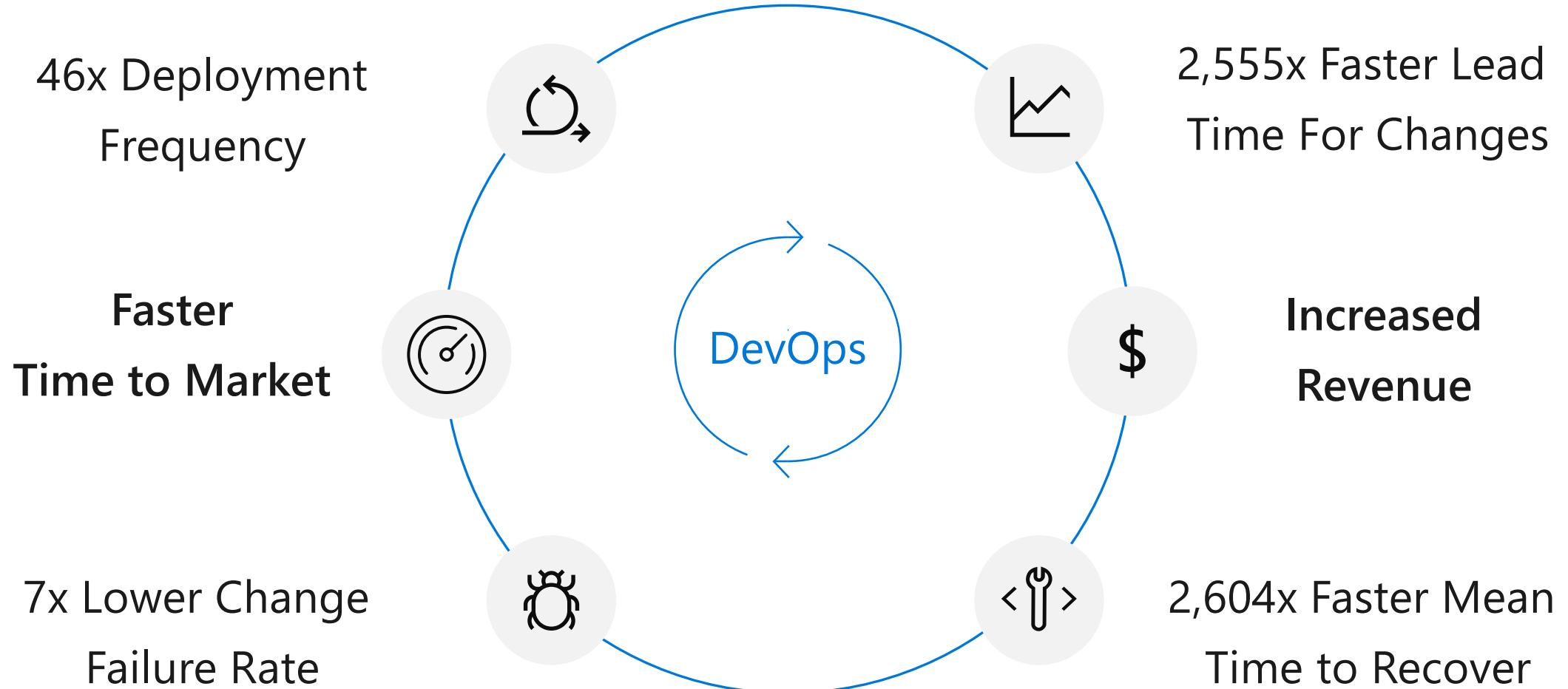


# DevOps for IoT

Alex Yochev  
@alxayo



# High Performance DevOps Companies Achieve...



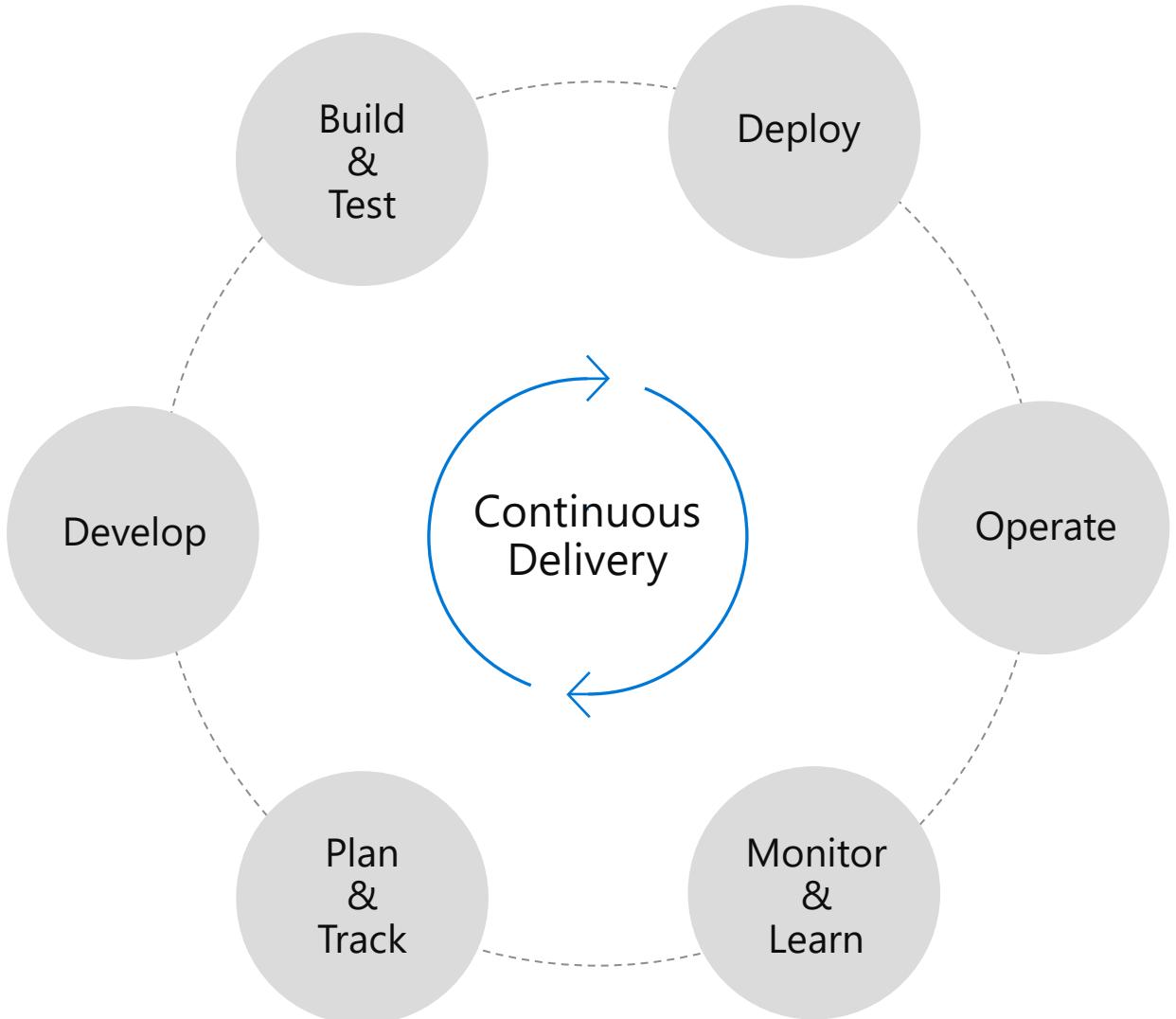
# What is DevOps?

People. Process. Products.

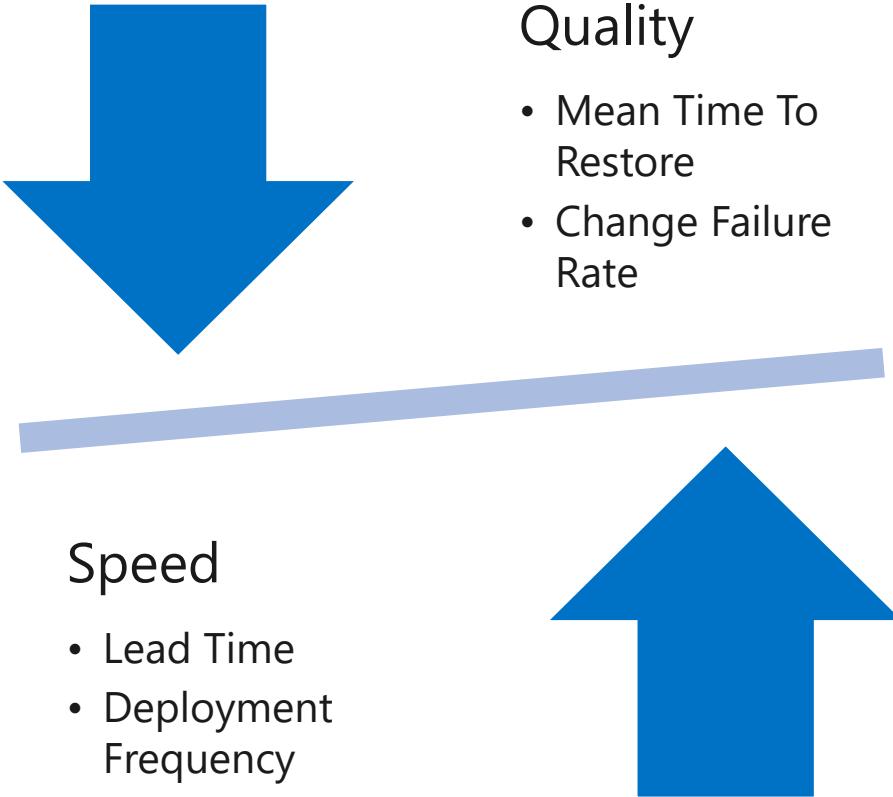
“

DevOps is the union of **people**, **process**, and **products** to enable continuous delivery of value to your end users.”

”



# Balancing Quality and Speed



## LEAD TIME

The time it takes to design and validate a product or feature, and the time to deliver the feature to customers.

## DEPLOYMENT FREQUENCY

The frequency with which software is deployed to the end users (i.e. in production or in an app store)

## MEAN TIME TO RESTORE (MTTR)

How quickly is the service restored in the event of (an inevitable) failure

## CHANGE FAILURE RATE

The percentage of changes to production (code and infrastructure) that fail

# DevOps for IoT and IIoT

"Using a well-designed automated **DevOps toolchain, you can streamline security testing, performance testing and other testing processes that are much more difficult to do manually**," said David Linthicum, chief cloud strategy officer at **Deloitte Consulting**.

What's more, **deployment tools can deal with the complexity of pushing code and data out to thousands of devices**, using automation to administer updates and checking they have the correct device configuration, Linthicum said.

The upshot is that DevOps improves stability and security while reducing deployment and operating costs.

Assessing the Advantages of DevOps For IoT Performance Testing (2021, iotworldtoday.com):  
<https://www.iotworldtoday.com/2021/07/06/assessing-the-advantages-of-devops-for-iot-performance-testing>



# DevOps capabilities and practices

<b>Continuous planning</b> 	<b>Continuous integration</b> 	<b>Continuous delivery</b> 	<b>Continuous operations</b> 
<ul style="list-style-type: none"><li>Objectives &amp; Key Results (OKR)</li><li>Lean product discovery</li><li>Lean product definition</li><li>Release planning</li><li>Sprint planning</li><li>Agile requirements</li><li>Security requirements</li><li>Architecture design</li><li>Capacity planning</li><li>UX architecture design</li><li>Threat modeling</li><li>Prioritization &amp; estimation</li><li>Demos &amp; Retrospectives</li></ul>	<ul style="list-style-type: none"><li>Behavior-driven development (BDD)</li><li>Test-driven development (TDD)</li><li>Microservices &amp; container development</li><li>Mono-repo &amp; Multi-repo</li><li>Unit testing &amp; code coverage</li><li>Version control</li><li>Git pull request</li><li>Trunk-based policies</li><li>Security static code scan</li><li>CredScan</li><li>Open Source software (OSS) component compliance</li><li>Build parallel &amp; serial pipeline</li></ul>	<ul style="list-style-type: none"><li>Release pipeline</li><li>Secure infrastructure deployment</li><li>IaaS deployment</li><li>PaaS deployment</li><li>SaaS deployment</li><li>Shared services</li><li>Infrastructure as code (IaC)</li><li>Change management</li><li>Configuration management</li><li>Release management</li><li>Blue-green deployments</li><li>Canary deployments</li><li>Feature flags</li><li>Trunk production ready</li></ul>	<ul style="list-style-type: none"><li>Site reliability engineering (SRE)</li><li>Telemetry &amp; monitoring</li><li>Application performance monitoring</li><li>Auto Failover, scaling, &amp; DR</li><li>Modern service management</li><li>Secure access &amp; application data</li><li>High availability, security, cost &amp; performance advisory</li><li>Secure DevOps ChatOps</li><li>Shift-right testing</li><li>Secrets management</li><li>Governance &amp; GDPR support</li><li>Automation &amp; AIOps</li><li>Continuity &amp; resilience</li></ul>
<b>Continuous quality</b> 	Quality requirements Shift-left testing	Governance & standards Test automation	Compliance & audits Shift right testing
<b>Continuous security</b> 	Security architecture Access & identity management	Application & data security Security infrastructure	Secure operations Governance, risk, & compliance
<b>Continuous collaboration</b> 	Collaborative culture Alignment & autonomy	Kanban collaboration Wiki & Teams collaboration	ChatOps collaboration Feature Team & SRE
<b>Continuous improvement</b> 	Lead time & cycle time Deployment frequency	Mean time to restore (MTTR) Change fail percentage	Continuous feedback Value stream mapping

# Connection Between IoT and DevOps - CI

The **continuous integration** makes sure that the Internet of things (IoT) applications that belong to DevOps are considered as a **holistic system** or application. All these constituent parts depend upon one another, even **components that aren't under direct control, like sensors** that are embedded in machines.

**This is slightly different from the traditional methods to DevOps, where all constituents are under direct control.**

# Connection Between IoT and DevOps - CD

**Deployment** is considered to be the **most significant component** when it comes to **DevOps IoT**.

The application requires **living on the platform as well as the functioning and playing using remote devices**.

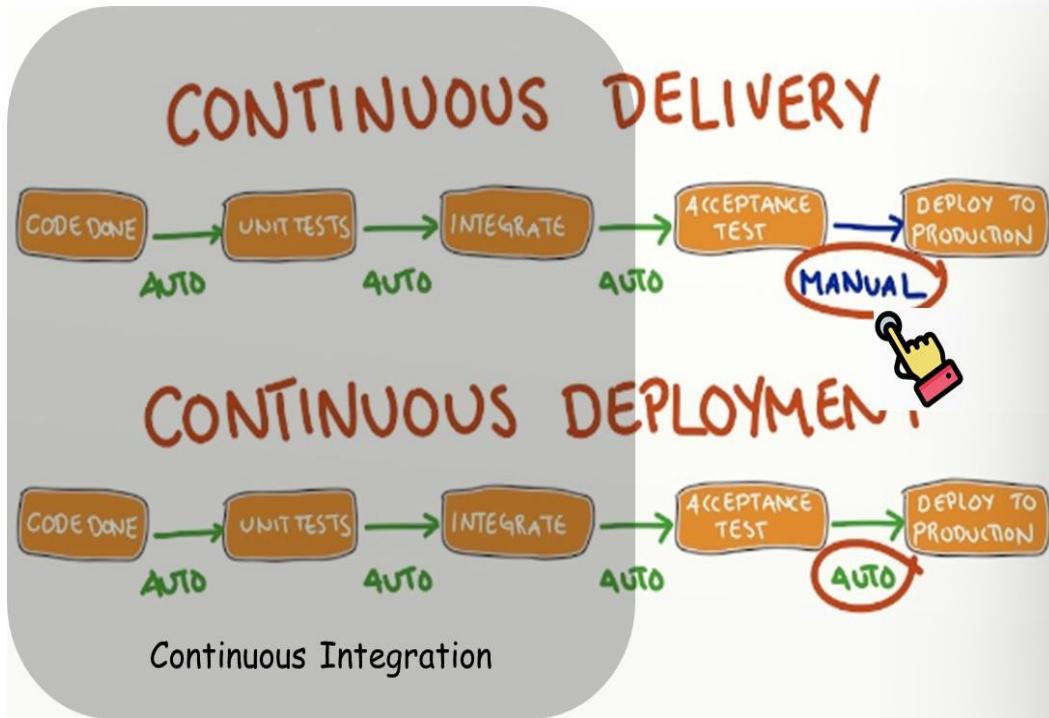
The **deployment processes get complicated** due to this. There can be some updates to the **application that exists on a central cloud** platform, and maybe **even firmware updates that exist on a device or remote sensor**

What is the Connection Between IoT and DevOps?:

<https://www.kovair.com/blog/connection-between-iot-and-devops/>



# What is Continuous Integration



## Continuous Integration is a Mindset and a Team Strategy

“... a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.”

# What are the Goals of Continuous Integration

The main purpose of Continuous Integration is to prevent developers stepping over each other code and eliminate integration issues.

## CI Goals:

- #1: Harness collaboration**
- #2: Enable parallel development**
- #3: Minimize integration debt**
- #4: Act as a quality gate**
- #5: Automate Everything!**

# Elements of Continuous Integration

## Version Control System

- Application / Test Code
- Configuration
- Database Definition
- Seed / Test Data
- Build / Deployment Scripts

## Branching Strategy

- Have a Branching strategy
- Keep changes small
- Commit frequently
- At least once per day for each developer

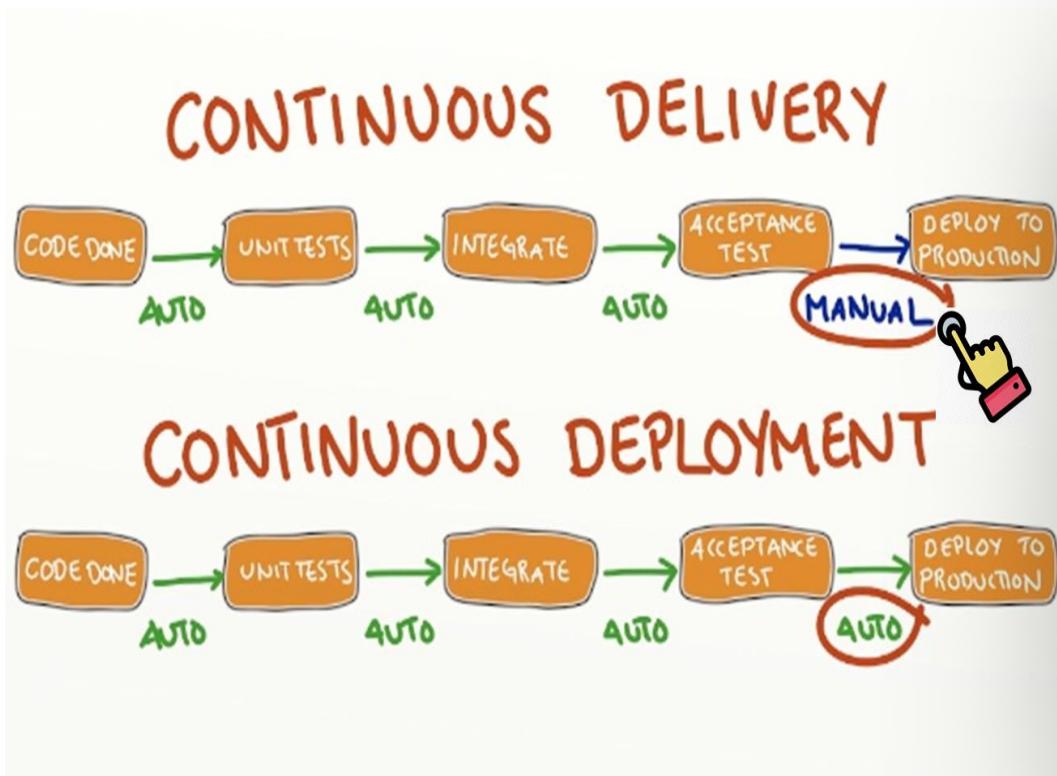
## Automated Build

- Runs after every commit/check-in
- Checks quality of the commit (Shift Left Testing)
- Separate Build machine (Build Server)
- Must run fast

## Agreement of the Team

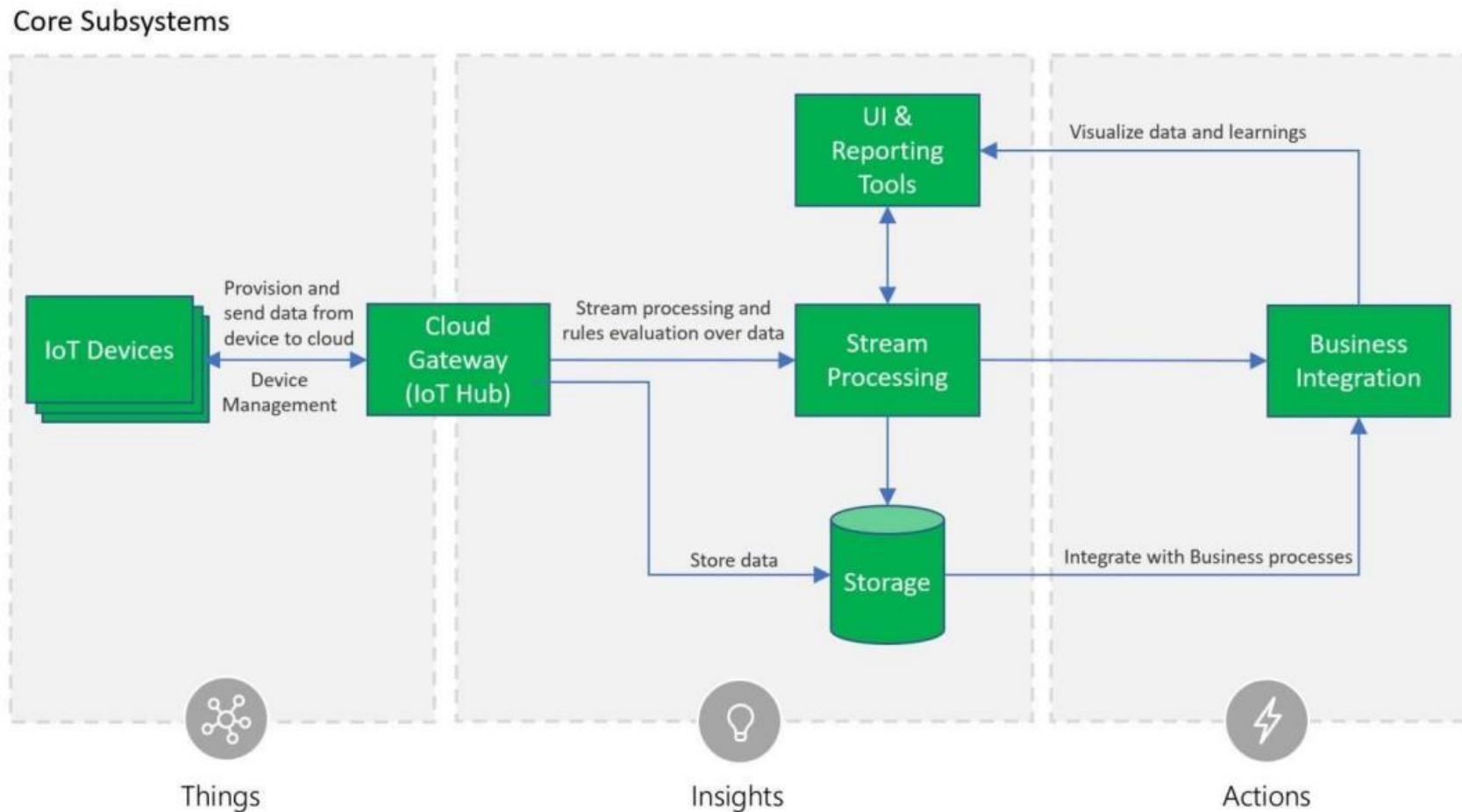
**Continuous integration is a practice, not a tool. It requires a degree of commitment and discipline from your development team.**

# What is Continuous Delivery

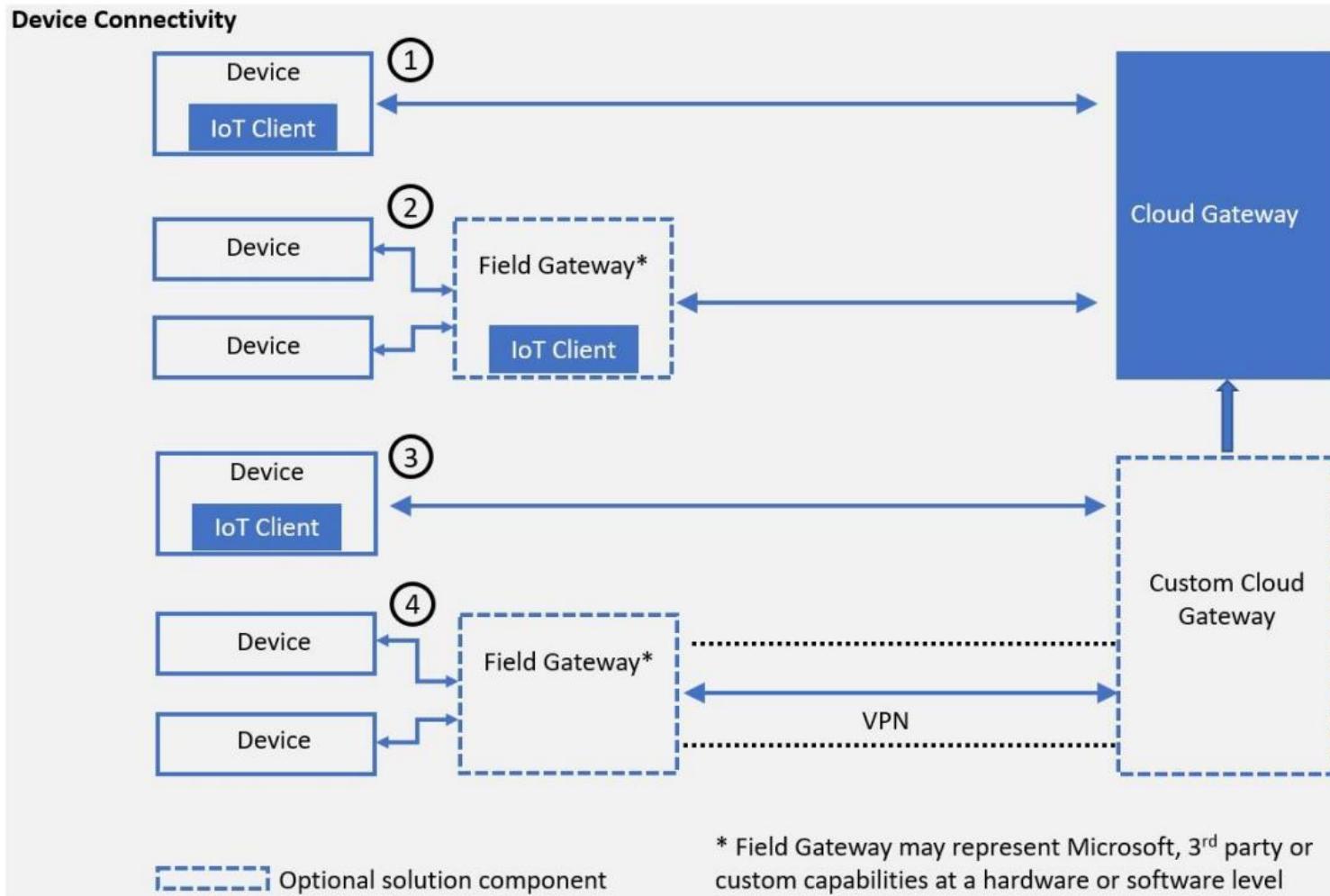


**Continuous delivery** is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so manually. It aims at building, testing, and releasing software with greater speed and frequency. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

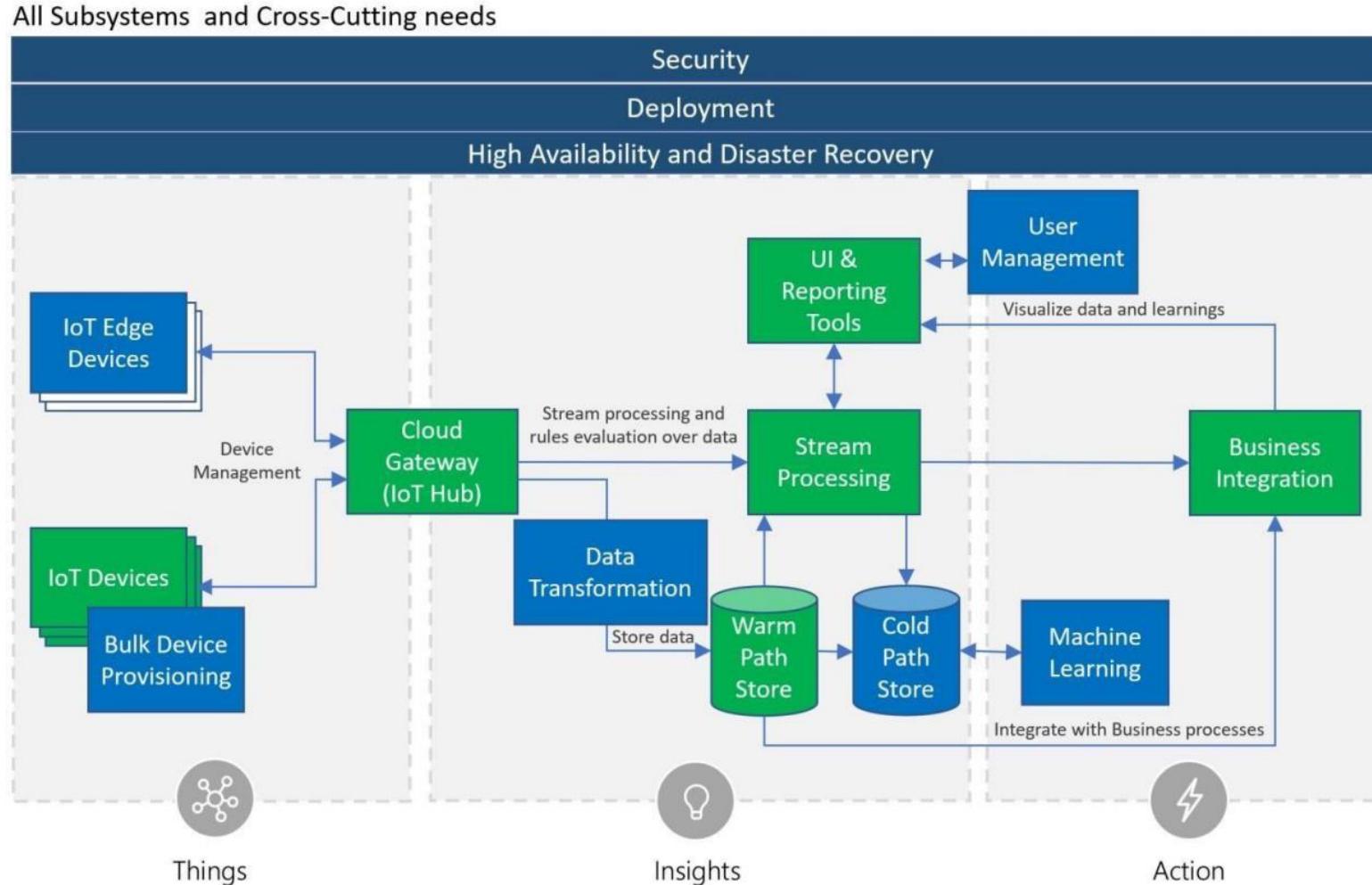
# Microsoft Azure IoT Reference Architecture



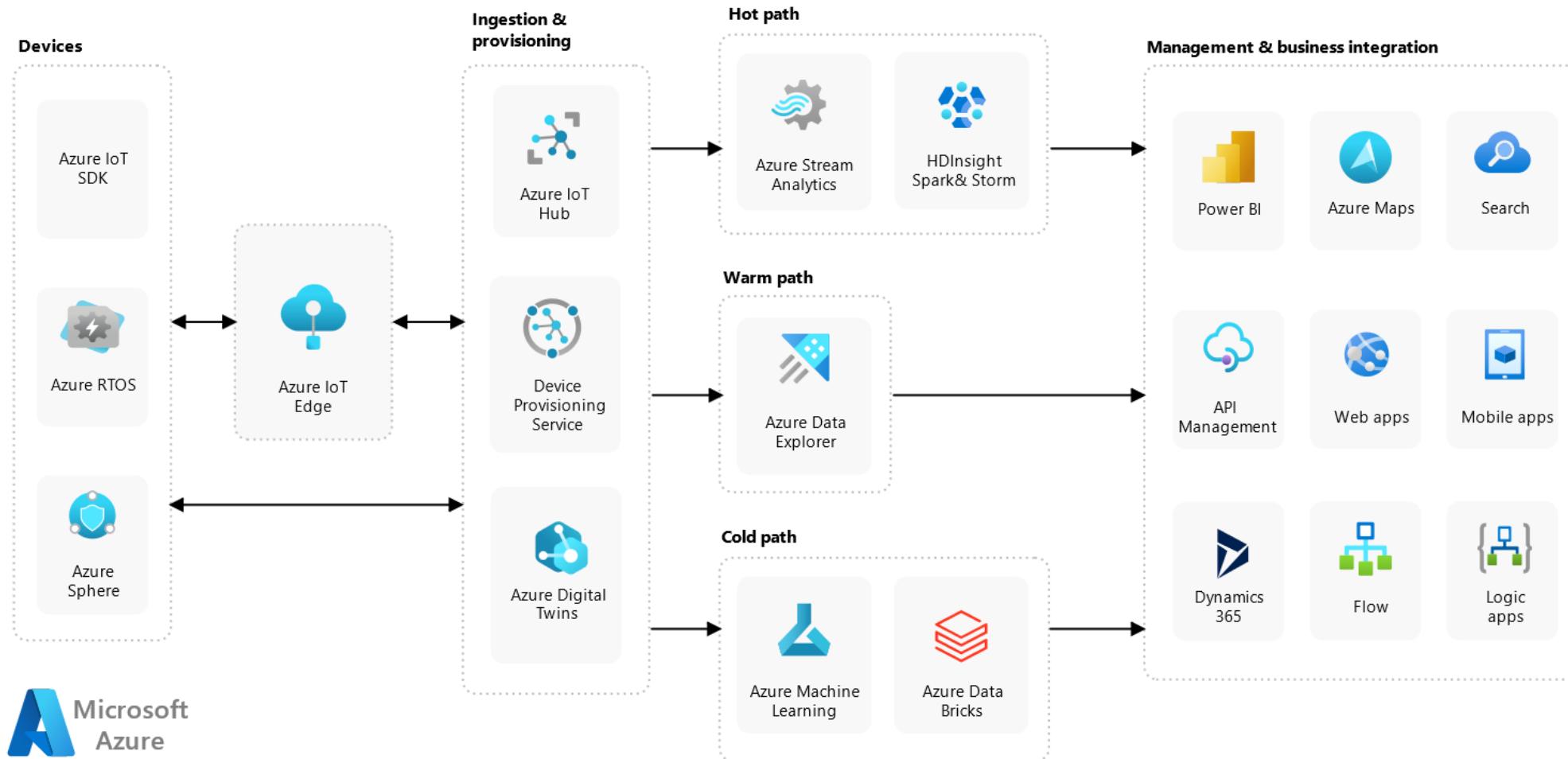
# Device Connectivity



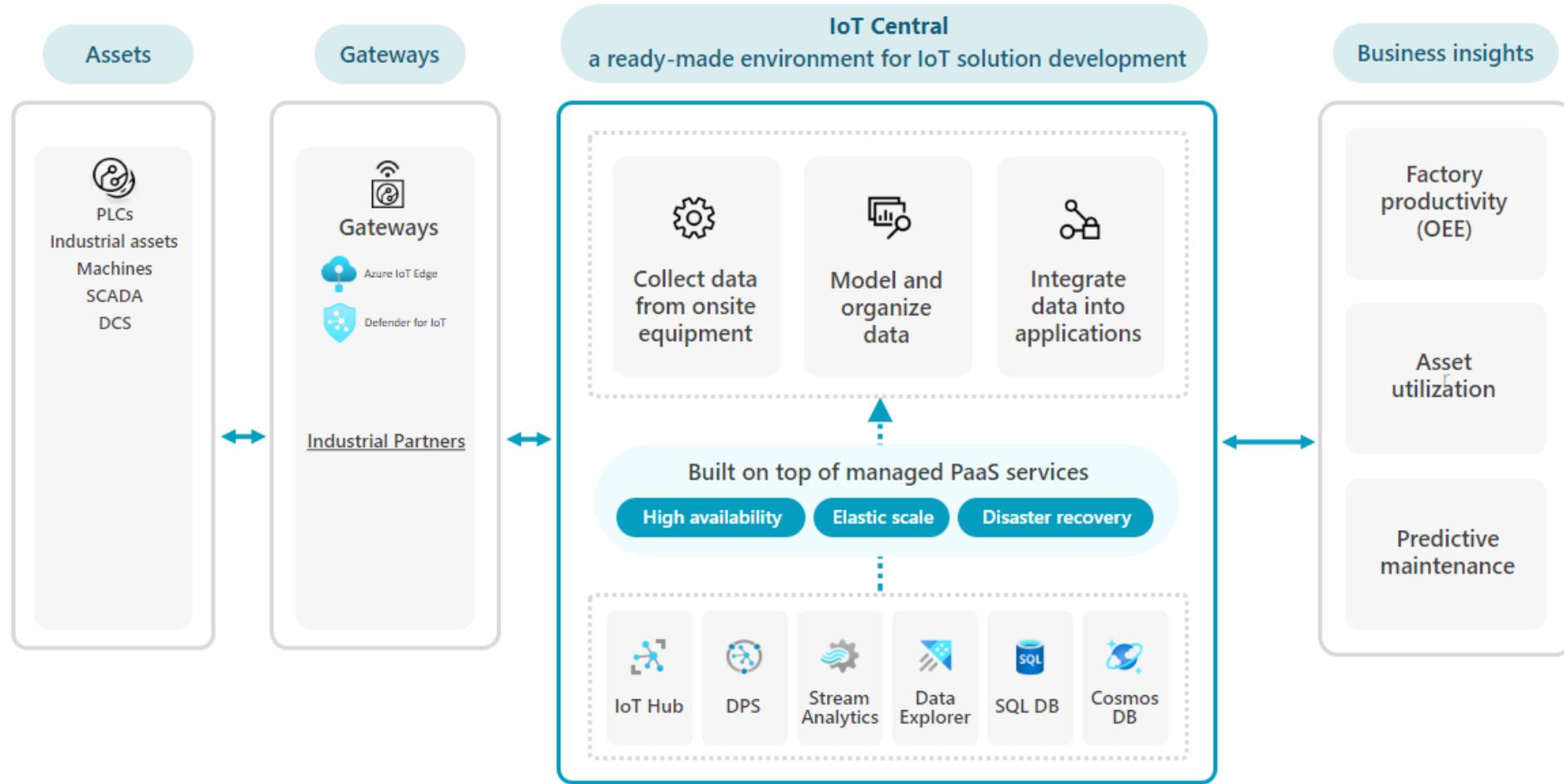
# Microsoft Azure IoT Reference Architecture



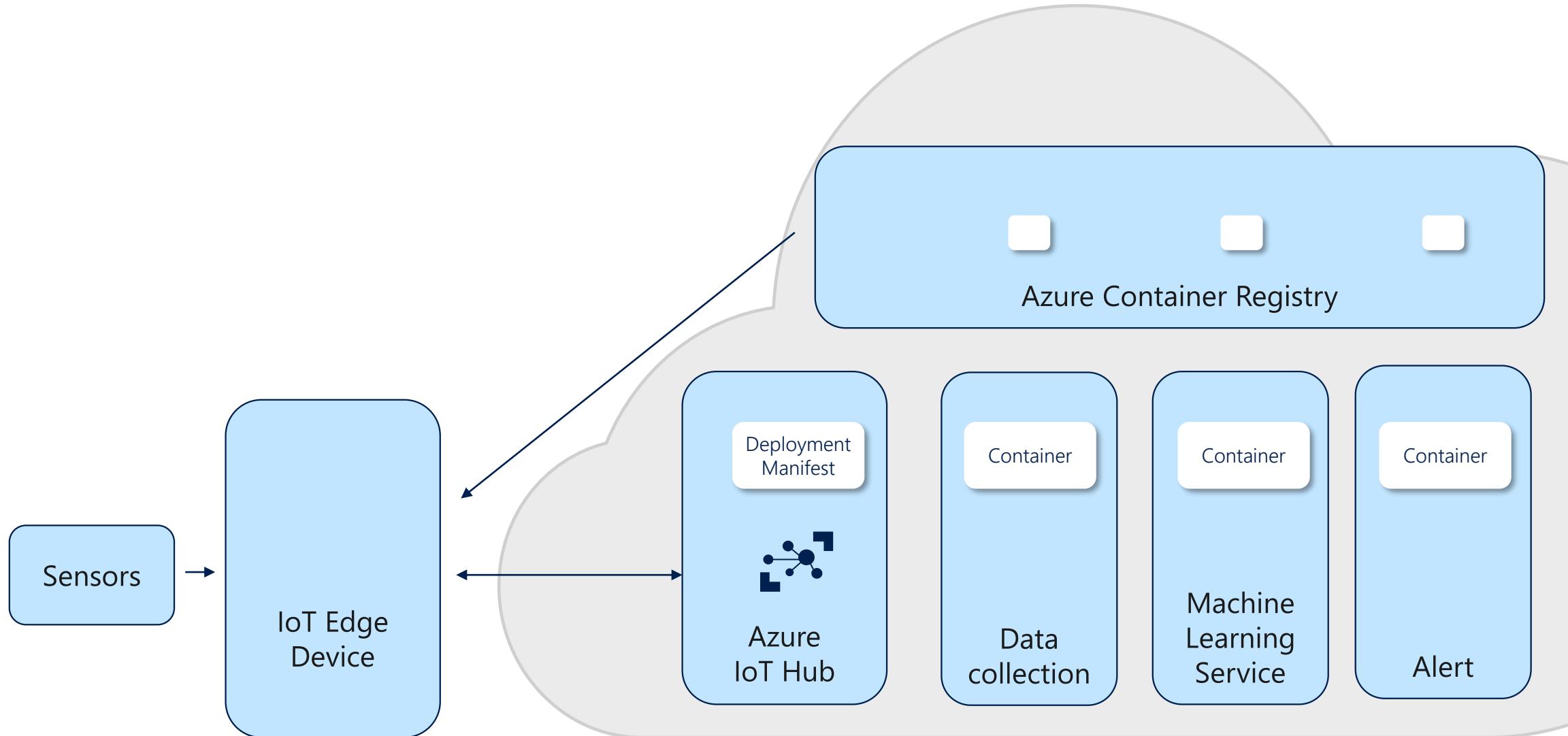
# Azure IoT Reference Architecture



# IoT (IIoT) architecture with Azure IoT Central



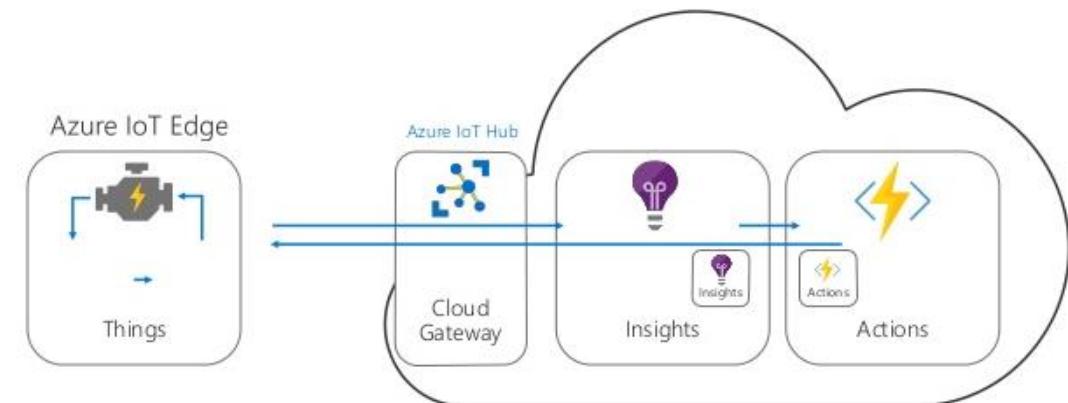
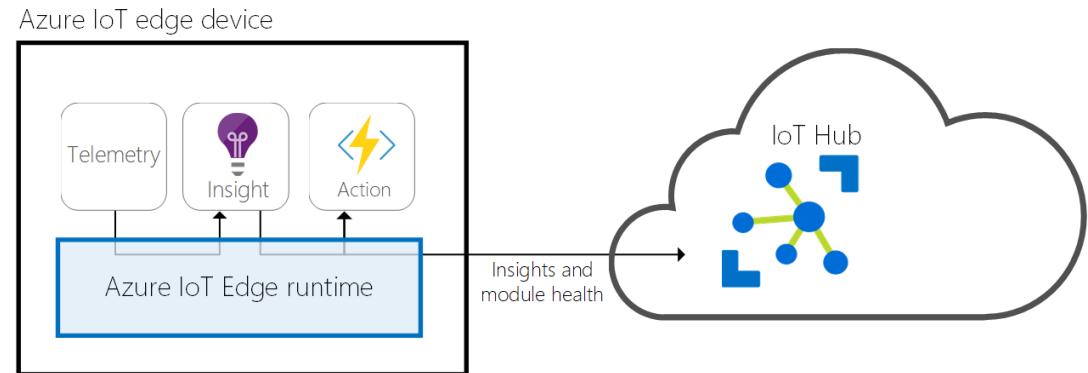
# Bringing Compute Where The Data Is



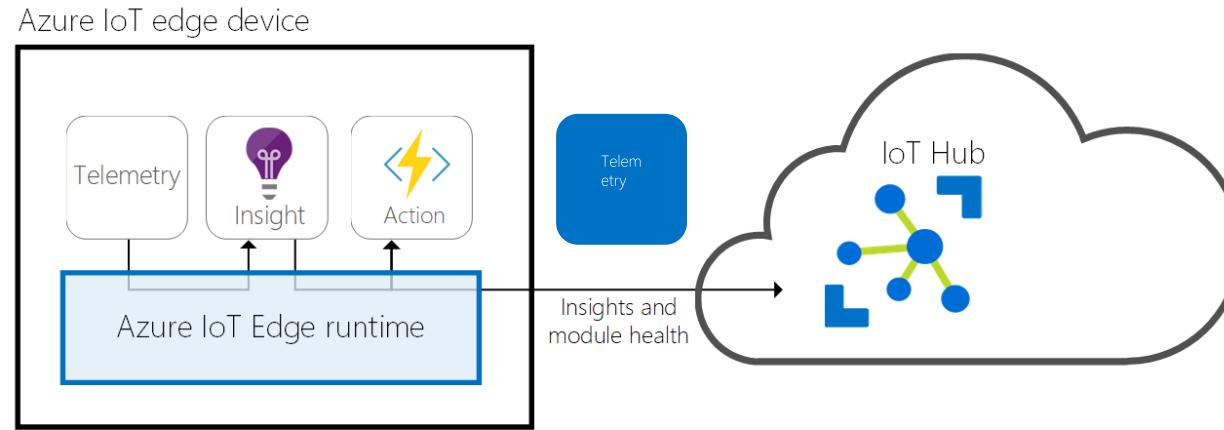
# Azure IoT Edge

- Built on container technology as 'modules'
- Modules support Python, NodeJS, .Net Core, Java, & C
- Low-latency AMQP / MQTT data transport
- Operate in offline / intermittent network conditions
- Supports Linux X64 | ARM32/64 , Windows X64

OSS and available @ <https://github.com/Azure/iotedge>



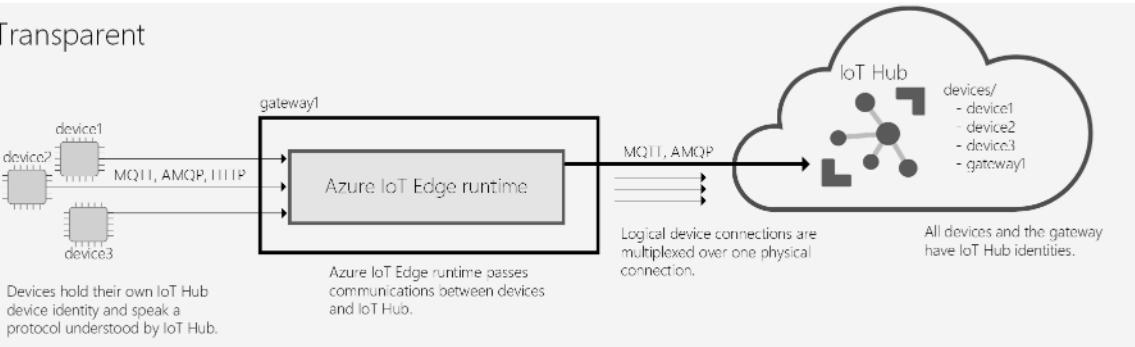
# IoT Edge Runtime



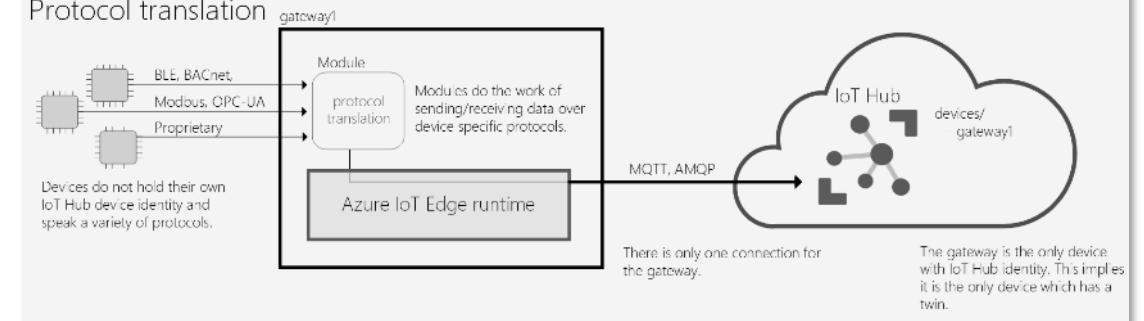
- Installs and updates workloads on the device.
- Maintains Azure IoT Edge security standards on the device.
- Ensures that IoT Edge modules are always running.
- Reports module health to the cloud for remote monitoring.
- Facilitates communication between downstream leaf devices and the IoT Edge device.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

# Azure IoT Edge as a Gateway (Patterns)

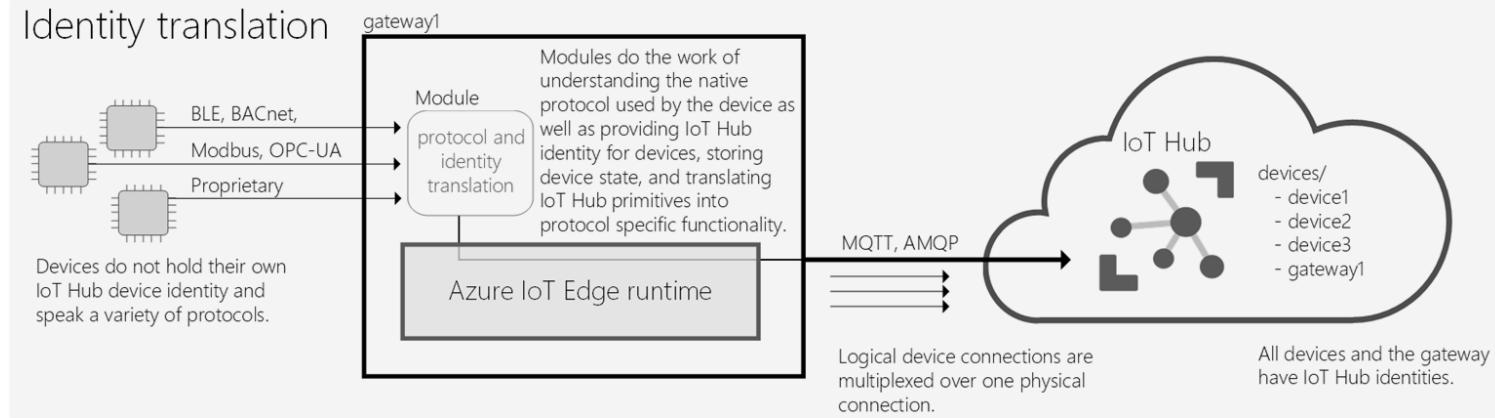
Transparent



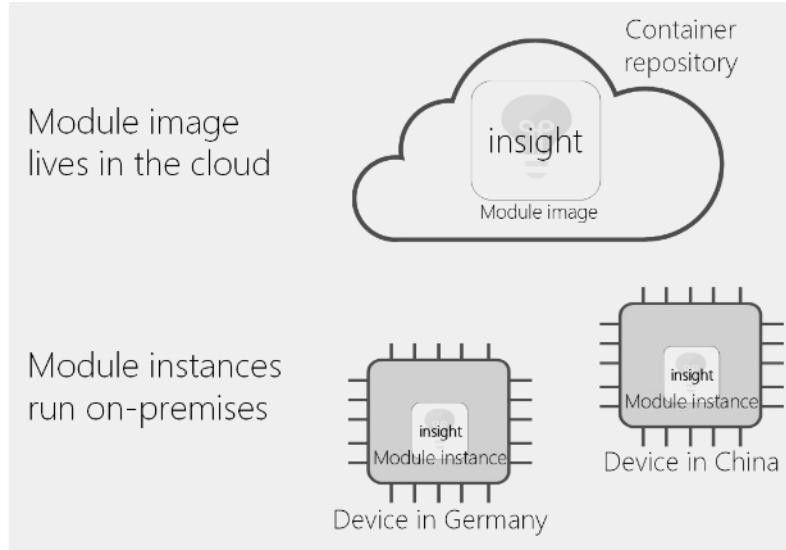
Protocol translation



Identity translation



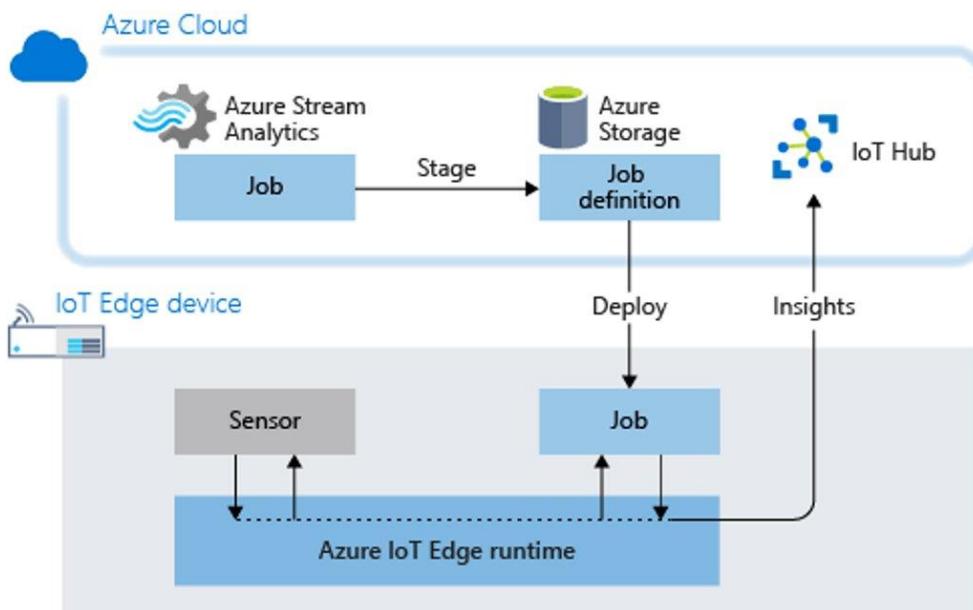
# IoT Edge Module



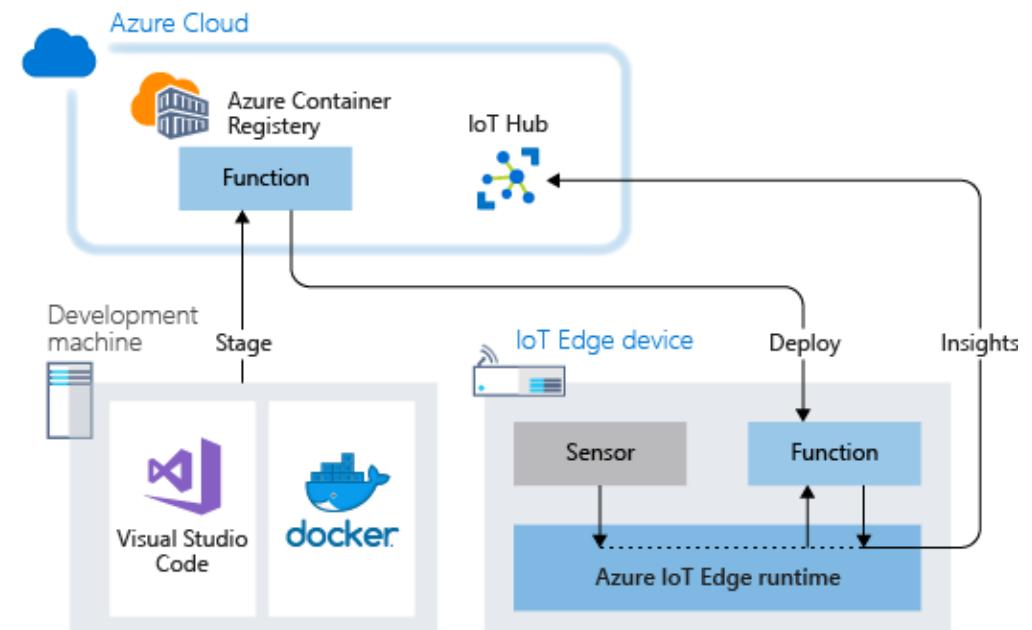
- A **module image** is a package containing the software that defines a module.
- A **module instance** is the specific unit of computation running the module image on an IoT Edge device. The module instance is started by the IoT Edge runtime.
- A **module identity** is a piece of information (including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

# Azure Services as Edge Modules

Azure Stream Analytics

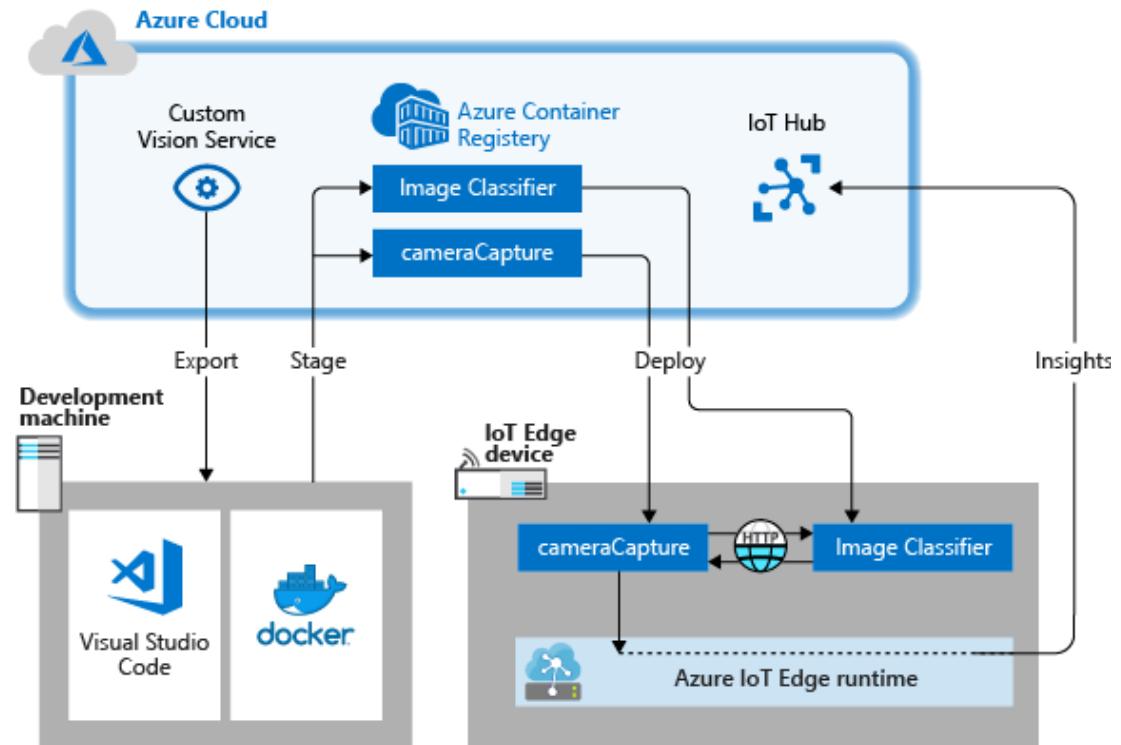


Azure Functions

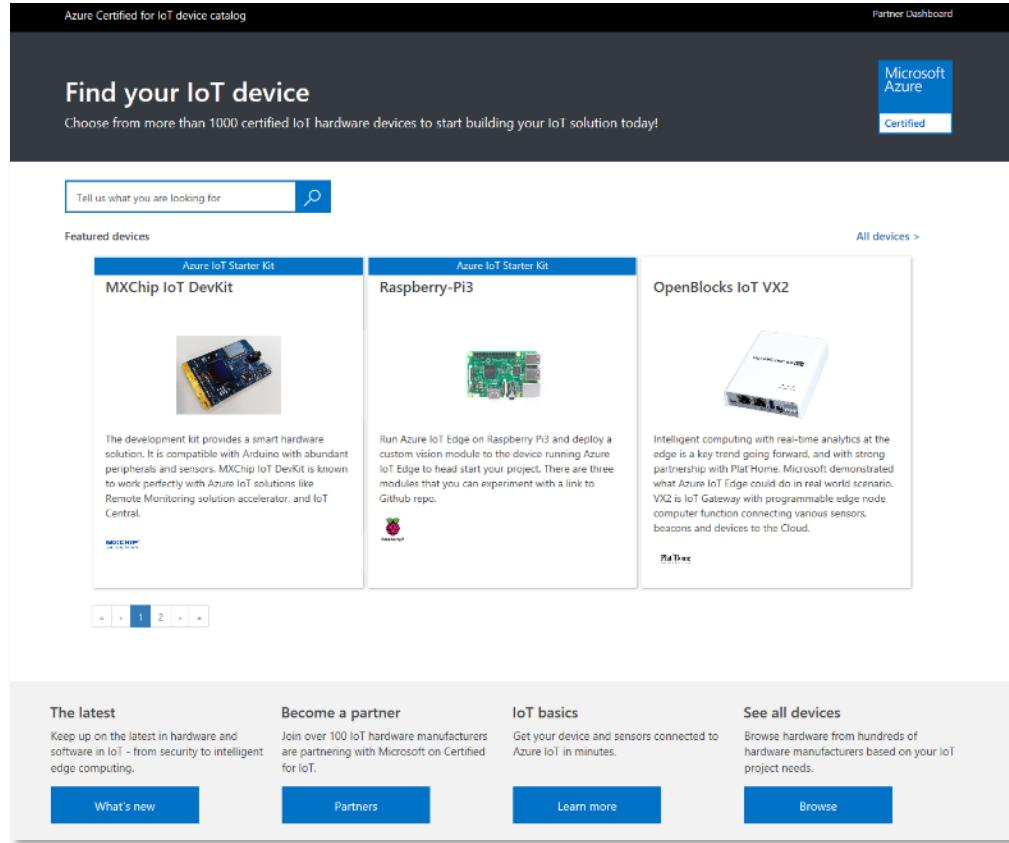


# AI/ML Modules

- Cognitive Services including:
  - Text Analytics - Key Phrase Extraction, Language Detection, Sentiment Analysis
  - Face – Detection, Verification, and Emotion
  - Computer Vision – Text Recognition, custom vision module allows to finds whatever you train it to find
- Azure Machine Learning at the Edge
  - Export AML models to run a Edge modules



# Certified Hardware for Azure IoT Edge



## Azure certified for IoT device catalog

- Provides an easy and intuitive way to discover the right IoT device for intended use case
- More than 1000 devices already listed in the device catalog
- Start at <https://catalog.azureiotsolutions.com/>

## Expanded device catalog with IoT Edge certified hardware

- Capability based certification for extensibility and long-term sustainability
- Each capability has N number of levels. Level 1 being lowest
- Select the best device most suitable for your IoT application

# Azure IoT Edge Module on Azure Marketplace

The screenshot shows the Microsoft Azure Marketplace interface. At the top, there's a navigation bar with links like 'Why Azure', 'Solutions', 'Products', 'Documentation', 'Pricing', 'Training', 'Marketplace' (which is highlighted in blue), 'Partners', 'Support', 'Blog', and 'More'. A search bar and a user profile for 'Emmanuel' are also present. Below the navigation, there are tabs for 'Azure Marketplace', 'Apps', 'Consulting services', 'Sell', and 'Learn'. A search bar with the placeholder 'Search Marketplace' is followed by a user icon and a link to 'Free account'. On the left, a sidebar lists categories such as 'Compute', 'Networking', 'Storage', 'Web', 'Mobile', 'Containers', 'Databases', 'Analytics', 'AI + Machine Learning', 'Internet of Things' (which is expanded to show 'Connectivity', 'Storage', 'Integration', 'Security', 'Identity', 'Developer tools', 'Management Tools', and 'Blockchain'), and 'Azure Active Directory apps'. The main content area displays results for 'Internet of Things' (16 items). A section titled 'IoT Edge Modules' contains three cards: 'Azure Stream Analytics on IoT Edge' (By Microsoft, Deploy near-real-time analytical intelligence to IoT Edge devices), 'SQL Server Module' (By Microsoft, Use Azure IoT Edge and SQL Server to store and query data at the edge), and 'Simulated Temperature Sensor' (By Microsoft, IoT Edge module that simulates a temperature sensor). Below this is a section titled 'Data analytics (9)' which includes cards for 'Stream Analytics job' (By Microsoft, Unlock real-time insights from streaming data), 'PI Integrator for Micros...' (By osssoft, Driving Digital Transformation and /), 'Smart oneM2M network' (By Pilot Things, Plug and Play IoT network), 'IoT Hub' (By Microsoft, Connect, monitor and manage IoT devices), and 'SpringBoard Industrial IoT System for Azure' (By Bright Wolf, A customizable solution for industrial IoT and connected product systems). There are also 'See all' and 'Software plans start at' links.

*Customers:*

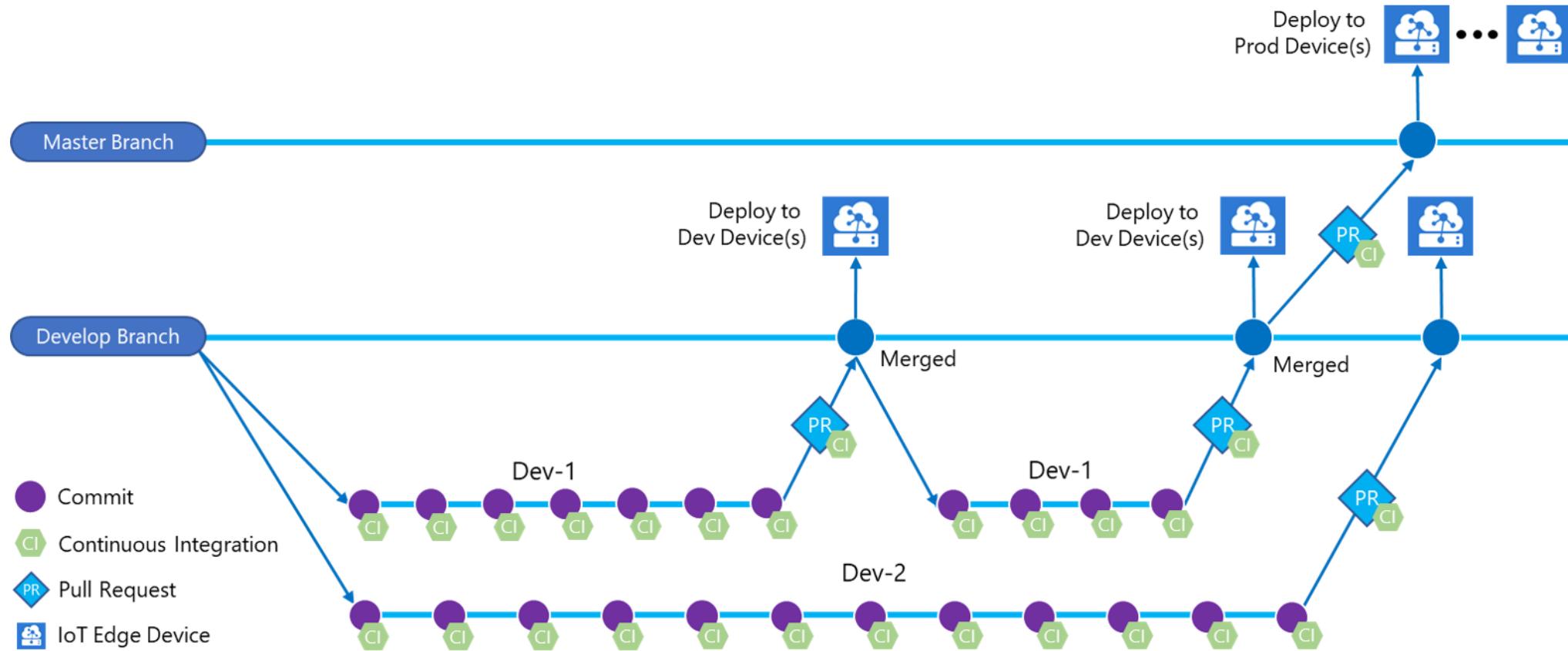
Save development effort  
Discover and integrate certified  
pre-built modules

*Partners:*

Showcase with wide reach  
Certified modules gives IoT customers  
peace of mind for their project

Work with a leader in IoT  
Market with Microsoft, and collaborate  
with other Microsoft IoT partners

# DevOps Process with IoT Edge



# Implement CI/CD for IoT Edge

The image displays two screenshots of the Azure DevOps Pipelines interface, illustrating the setup of CI/CD pipelines for IoT Edge.

**Left Screenshot:** This screenshot shows the configuration of an "iot-edge-yaml-pipeline-test" pipeline. The pipeline consists of a single "Agent job 1" step, which is set to run on an agent. The "Get sources" step is configured to pull from the "iot-edge-yaml-pipeline-test" repository's master branch. In the "Add tasks" pane, a search for "Azure IoT Edge" has been performed, resulting in two task cards: "Azure IoT Edge" (Build and deploy an Azure IoT Edge image) and "[Internal Test] Azure IoT Edge" (Build and deploy an Azure IoT Edge image). The "Azure IoT Edge" card has its "Add" button highlighted with a red box. The pipeline is currently in the "Pending" state.

**Right Screenshot:** This screenshot shows the configuration of an "iot-edge-yaml-pipeline-test" pipeline. Similar to the left, it includes a "Get sources" step for the "iot-edge-yaml-pipeline-test" repository and an "Agent job 1" step. The "Add tasks" pane shows a search for "Publish build artifacts", leading to the "Publish build artifacts" task card, which is also highlighted with a red box. This pipeline is also in the "Pending" state.

Implement CI/CD for IoT Edge:

<https://docs.microsoft.com/en-us/azure/iot-edge/how-to-continuous-integration-continuous-deployment-classic?view=iotedge-1.4>



# Implement CI/CD for IoT Edge

The screenshot shows the Azure DevOps interface for a pipeline named 'IoTEdgePhoneDevOps'. The left sidebar highlights the 'Pipelines' section. The main area displays a build run titled 'Jobs in run #202... tpc-ci-build'. A specific job, 'Azure Deployment:Create ACR', is expanded, showing its log output:

```
1 Starting: Azure Deployment:Create ACR
2 =====
3 Task      : Azure resource group deployment
4 Description: Deploy an Azure Resource Manager (ARM) template to a resource group and
5 Version   : 2.171.0
6 Author    : Microsoft Corporation
7 Help      : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-res
8 =====
9 Checking if the following resource group exists: IoTEdgePhoneDevOps-rg.
10 Resource group exists: true.
11 Creating deployment parameters.
12 Starting template validation.
13 Deployment name is arm-acr-20200630-162725-afee
14 Template deployment validation was completed successfully.
15 Starting Deployment.
16 Deployment
17 Successful
18 Finishing
```

The pipeline summary indicates a 'Deployment process' status of 'Succeeded' and an 'Agent phase' status of 'Succeeded'. The deployment was started at 8/14/2019, 10:40:36 PM and completed in 3m 51s. The agent used was 'Hosted Ubuntu 1604'.

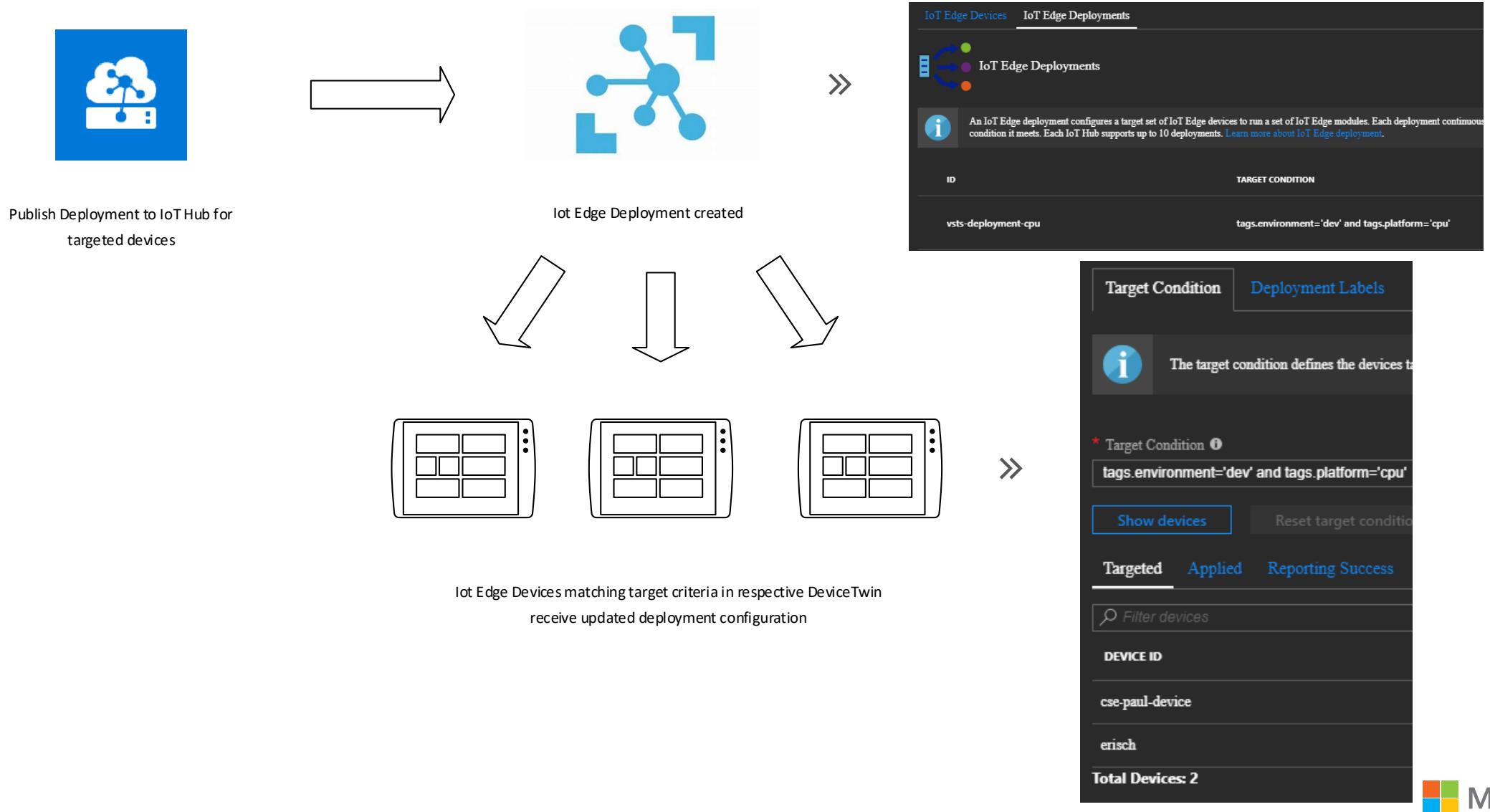
Task	Description	Duration
Initialize job	succeeded	1s
Download Artifacts	succeeded	<1s
Azure IoT Edge - Generate deployment manifest	succeeded	1m 39s
Azure IoT Edge - Deploy to IoT Edge devices	succeeded	1m 41s
Finalize Job	succeeded	<1s

Implement CI/CD for IoT Edge:

<https://docs.microsoft.com/en-us/azure/iot-edge/how-to-continuous-integration-continuous-deployment-classic?view=iotedge-1.4>



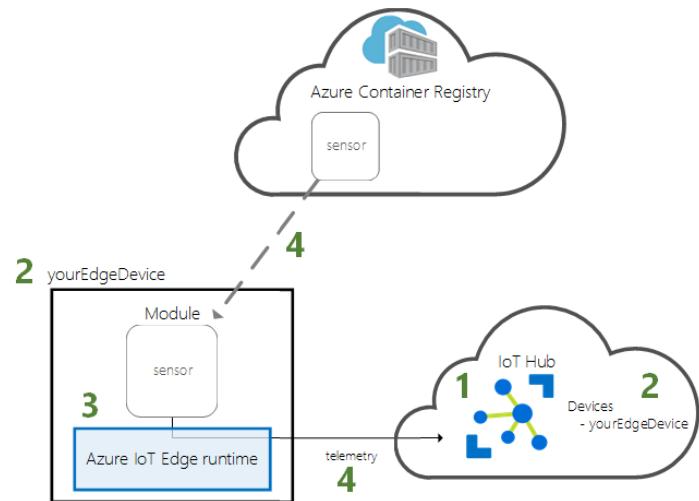
# IoT Edge Deployment Process



# System Modules

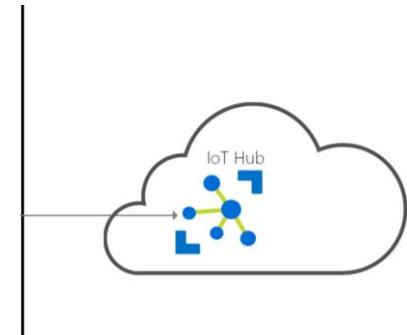
edge-agent:

Deployment & Container orchestration  
Ensures module uptime



edge-hub:

Communication to/from Azure IoT Hub  
Inter-module communication



# Implement CI/CD for IoT Edge – smoke tests

The screenshot shows the 'All pipelines > Release Pipeline - Copy' page in Azure DevOps. The 'Tasks' tab is selected, highlighted with a red border. The pipeline configuration includes:

- Create Deployment** (Deployment process)
- Agent job** (Some settings need attention)
  - Replace tokens in deployment.template.json (Replace Tokens)
  - Replace tokens in modules/\*\*/module.json (Replace Tokens)
  - Azure IoT Edge - Build module images (Azure IoT Edge)
  - Azure IoT Edge - Push module images** (Azure IoT Edge) - This task is selected and highlighted with a red border.
  - Azure IoT Edge - Deploy to IoT Edge devices (Azure IoT Edge)

Task version: 2.\*

Display name: Azure IoT Edge - Push module images

Action: Push module images

Container registry type: Azure Container Registry

Azure subscription: (Manage) [Subscription dropdown] (Scoped to subscription 'Microsoft Azure Sponsorship 2')

Azure Container Registry: IoTEdgeDevOpsMsLearn

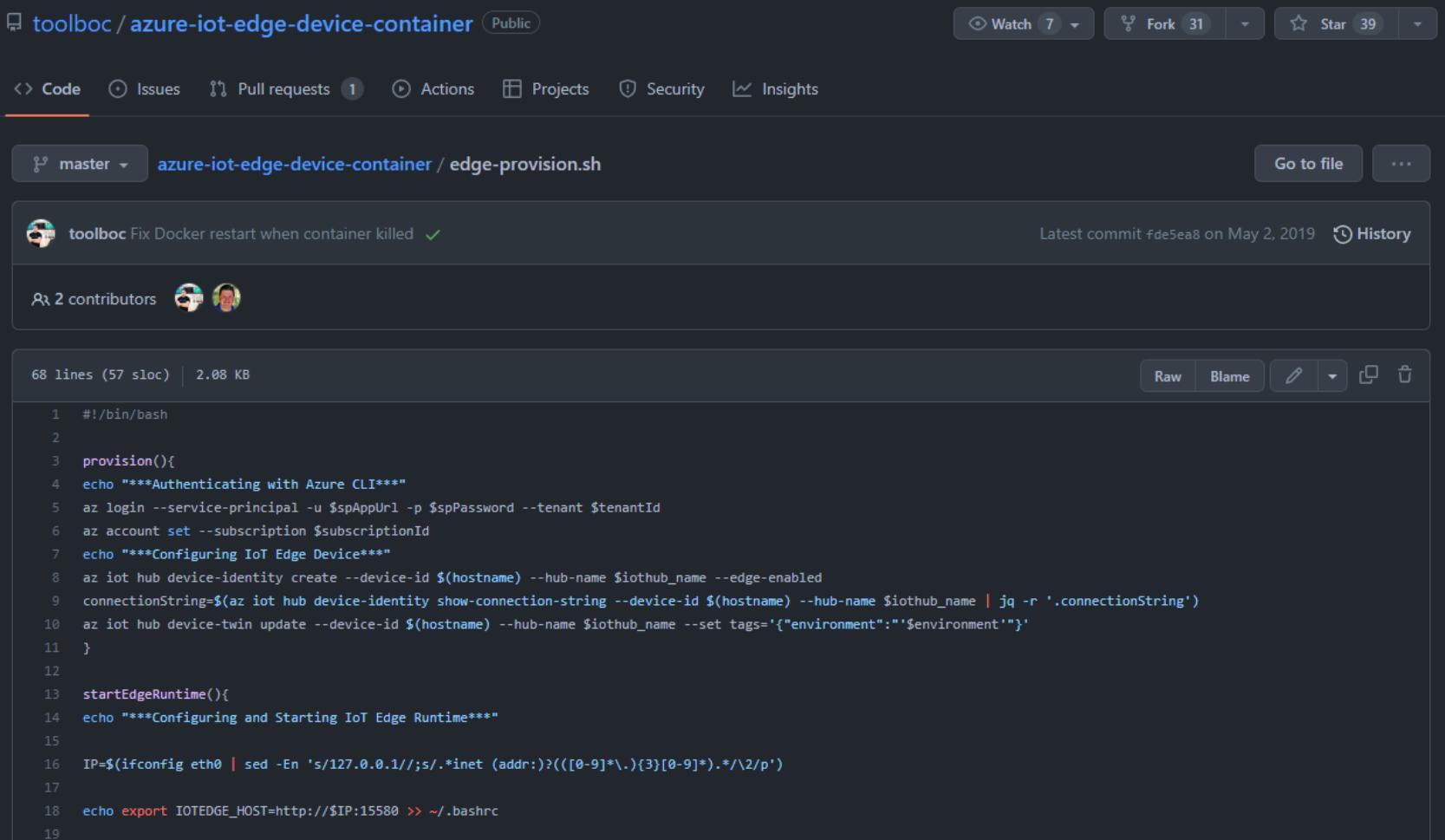
.template.json file: \$(System.DefaultWorkingDirectory)/\$(Release.PrimaryArtifactSourceAlias)/drop/EdgeSolution/deployment.template.json

Exercise - Create a CD release pipeline for IoT Edge with a smoke test:

<https://docs.microsoft.com/en-us/training/modules/implement-cicd-iot-edge/6-exercise-create-release-pipeline-iot-edge>



# Implement CI/CD for IoT Edge – device container



The screenshot shows a GitHub repository page for 'toolboc/azure-iot-edge-device-container'. The repository is public and has 7 watches, 31 forks, and 39 stars. The 'Code' tab is selected, showing the 'edge-provision.sh' file. The file contains a bash script for provisioning an IoT Edge device, including steps for authenticating with Azure CLI, creating device identities, and updating device twins. The script also includes logic to start the IoT Edge runtime and set environment variables.

```
#!/bin/bash

provision(){
    echo "****Authenticating with Azure CLI***"
    az login --service-principal -u $spAppUrl -p $spPassword --tenant $tenantId
    az account set --subscription $subscriptionId
    echo "****Configuring IoT Edge Device***"
    az iot hub device-identity create --device-id $(hostname) --hub-name $iothub_name --edge-enabled
    connectionString=$(az iot hub device-identity show-connection-string --device-id $(hostname) --hub-name $iothub_name | jq -r '.connectionString')
    az iot hub device-twin update --device-id $(hostname) --hub-name $iothub_name --set tags='{"environment":"$environment"}'
}

startEdgeRuntime(){
    echo "****Configuring and Starting IoT Edge Runtime***"
    IP=$(ifconfig eth0 | sed -En 's/127.0.0.1//;s/.*inet (addr:)?(([0-9]*\.){3}[0-9]*).*/\2/p')
    echo export IOTEDGE_HOST=http://$IP:15580 >> ~/.bashrc
}
```

azure-iot-edge-device-container

<https://github.com/toolboc/azure-iot-edge-device-container>



# Implement CI/CD for IoT Edge – smoke tests design

Designing an automated smoke-testing strategy provides many benefits.

- Deploying to virtual IoT devices is cheaper than deploying to physical IoT devices.
- A successful smoke test ensures that the program is ready for further testing.
- Automated smoke testing avoids expensive rework for more sophisticated tests.
- Optimizing the time for your developer and the test teams.

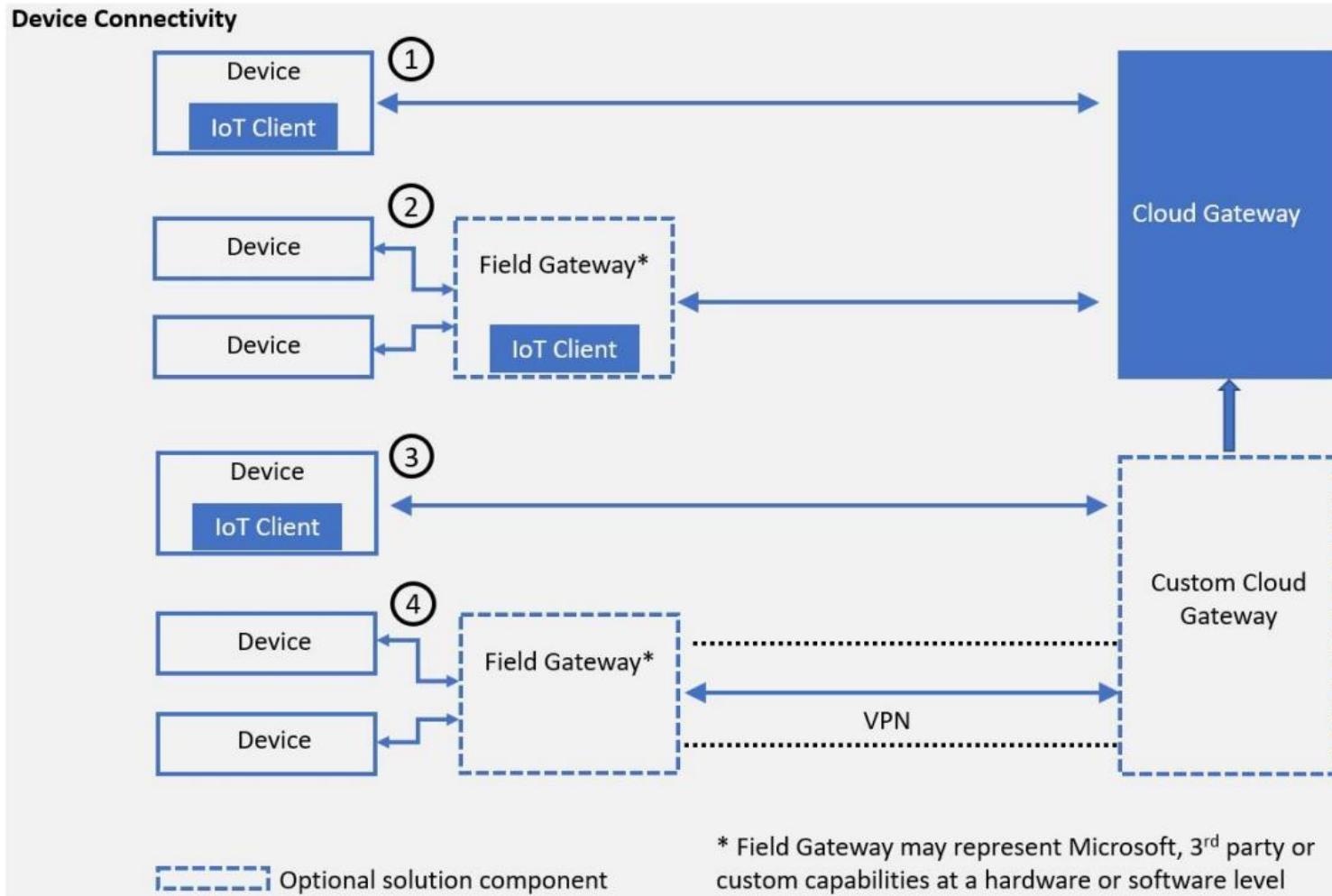
To implement this solution, you'll create a smoke test as a subset of your main functionality. The test should be self-scoring, automated, and able to run on virtual IoT devices.

Exercise - Create a CD release pipeline for IoT Edge with a smoke test:

<https://docs.microsoft.com/en-us/training/modules/implement-cicd-iot-edge/6-exercise-create-release-pipeline-iot-edge>



# Device Connectivity



# DevOps for Mobile Apps and Devices

New test run

① Select devices ② Configure ③ Submit

Which device configurations do you want to test?

Search devices

Please select at least one device to continue.

Devices

<input type="checkbox"/> Device	OS Version
<input type="checkbox"/> Google Google Pixel 2	Android 9
<input type="checkbox"/> Google Google Pixel 2 XL	Android 8.1.0
<input type="checkbox"/> Google Google Pixel 2	Android 8.1.0
...	

# Build your own -Mobile Apps Test center



Run apps on a hardware device

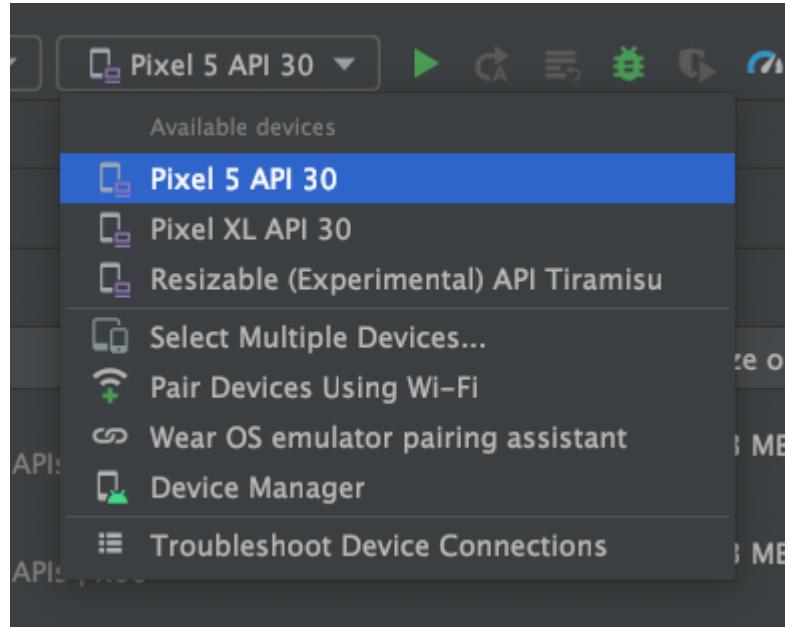
<https://developer.android.com/studio/run/device>



# Glimpse of a real - Mobile Apps Test center



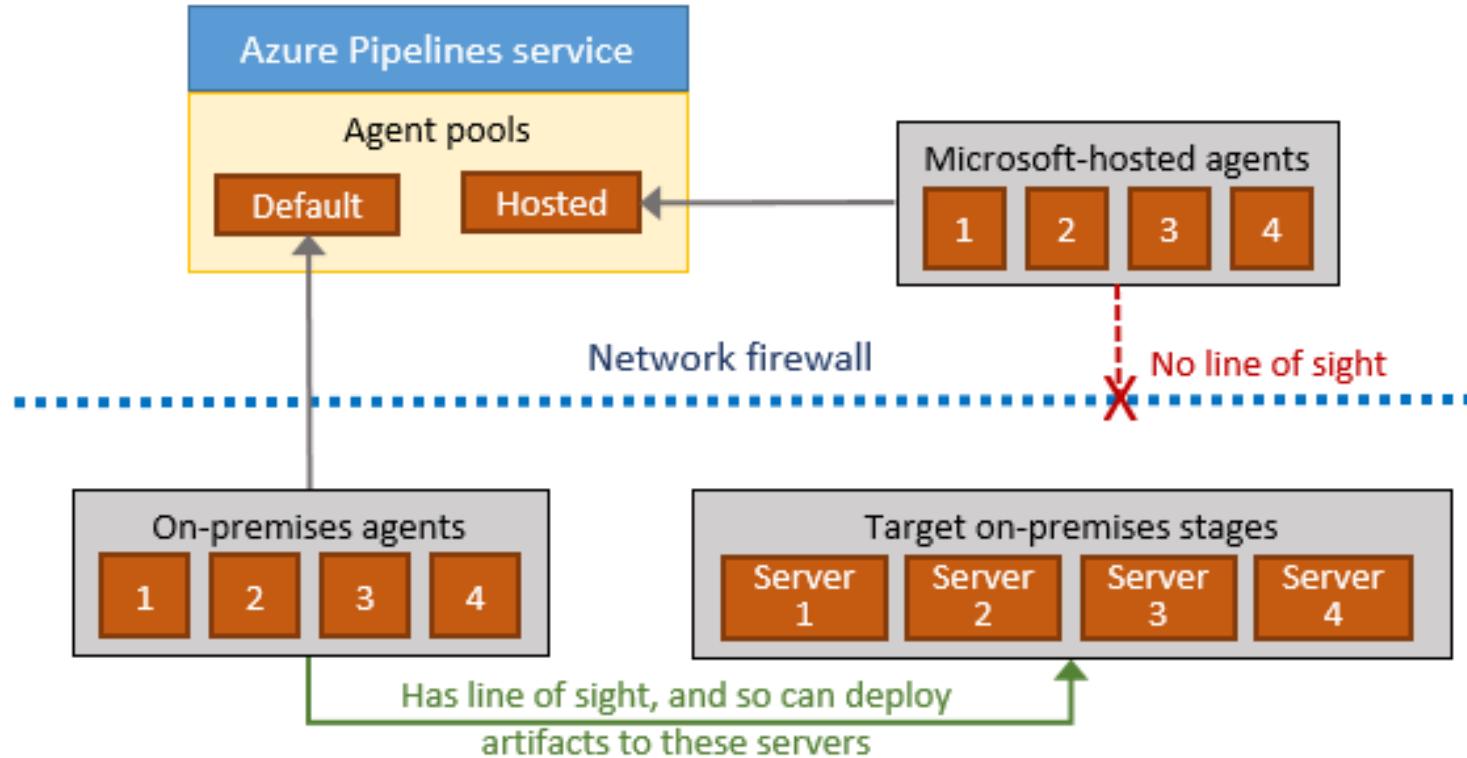
# Emulators



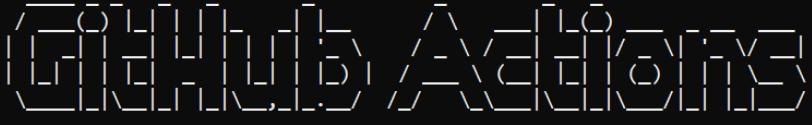
Run apps on the Android Emulator

<https://developer.android.com/studio/run/emulator>

# DevOps Self-Hosted build agents



# DevOps – Self Hosted Runner on Pi



Self-hosted runner registration

```
# Authentication

✓ Connected to GitHub

# Runner Registration

Enter the name of runner: [press Enter for ubuntu] pionubuntu

This runner will have the following labels: 'self-hosted', 'Linux', 'ARM'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip]

✓ Runner successfully added
✓ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

✓ Settings Saved.

ubuntu@ubuntu:~/actions-runner$ ./run.sh

✓ Connected to GitHub

2021-06-05 17:33:16Z: Listening for Jobs
2021-06-05 17:35:30Z: Running job: build
2021-06-05 17:36:05Z: Job build completed with result: Succeeded
```

# Mobile Devices Seamless updates

## **A/B (Seamless) system updates**

Modern Android devices have two copies of each partition (A and B) and can apply an update to the currently unused partition while the system is running but idle. A/B devices do not need space to download the update package because they can apply the update as they read it from the network; this is known as streaming A/B.

## **Non-A/B system updates**

Older Android devices have a dedicated recovery partition containing the software needed to unpack a downloaded update package and apply the update to the other partitions.

# Device update

# OTA

Over-The-Air

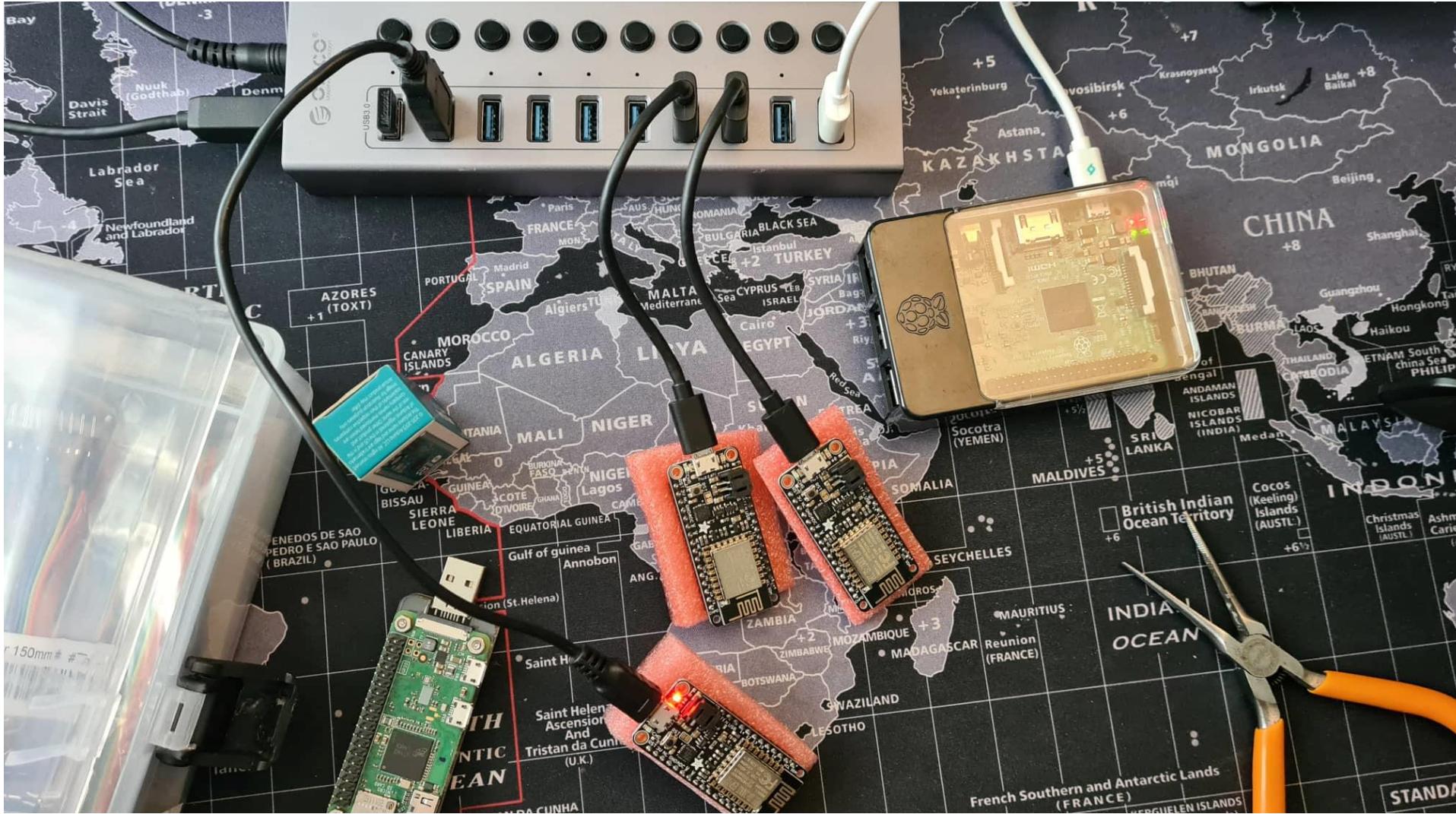
# DFU

Device Firmware  
Update

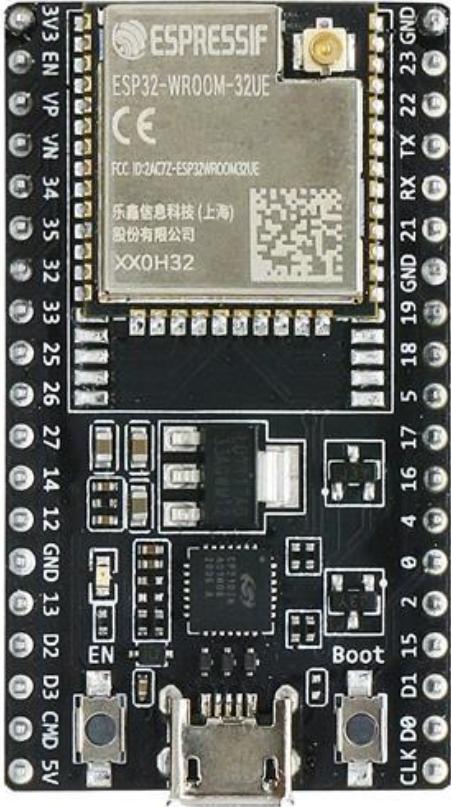
# FOTA

Firmware Updates  
Over-the-air

# Build your own – IoT Test Center ?!



# ESP32 OTA



The OTA update mechanism allows a device to update itself based on data received while the normal firmware is running (for example, over Wi-Fi or Bluetooth.)

**OTA requires configuring the Partition Table of the device with at least two “OTA app slot” partitions (i.e. `ota_0` and `ota_1`) and an “OTA Data Partition”.**

**The OTA operation functions write a new app firmware image to whichever OTA app slot that is currently not selected for booting.** Once the image is verified, the OTA Data partition is updated to specify that this image should be used for the next boot.

# ESP32 OTA - Arduino

```
#include <esp32fota.h>
#include <WiFi.h>

const char *ssid = "";
const char *password = "";

esp32FOTA esp32FOTA("esp32-fota-http", "1.0.0");

void setup()
{
    esp32FOTA.checkURL = "http://server/fota/fota.json";
    Serial.begin(115200);
    setup_wifi();
}

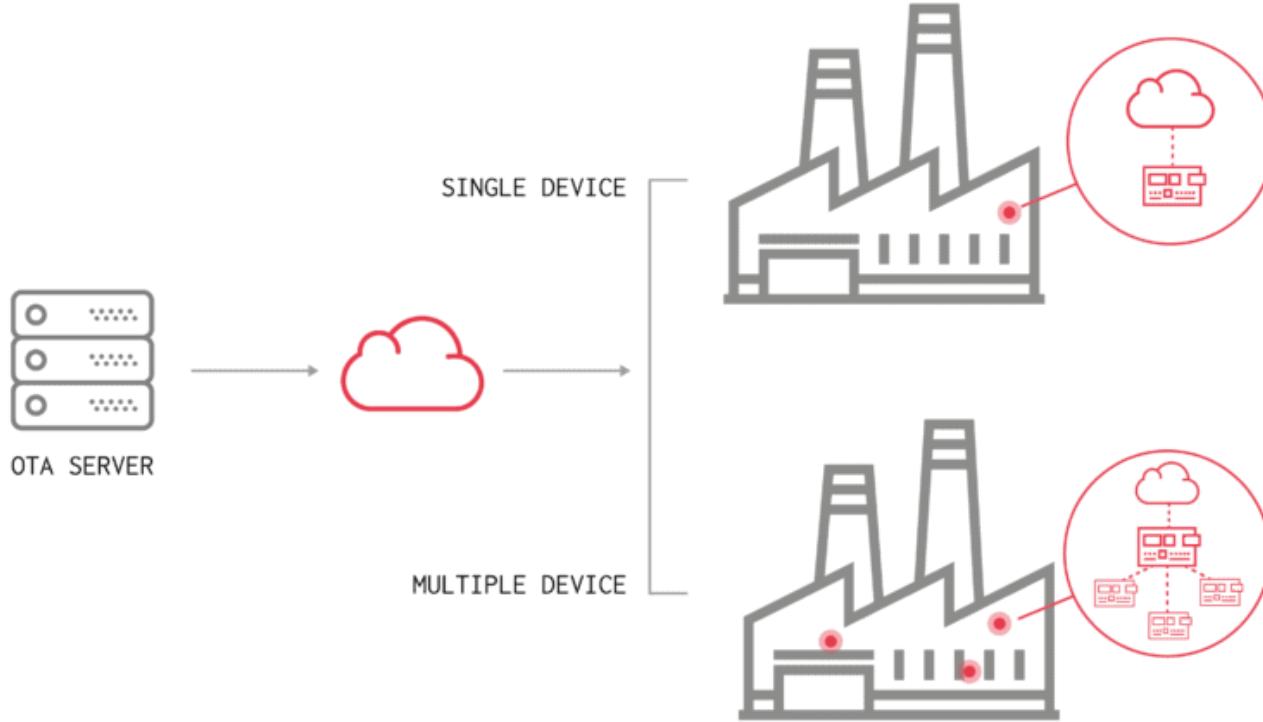
void setup_wifi()
{
    delay(10);
    Serial.print("Connecting to ");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
}

void loop()
{
    bool updatedNeeded = esp32FOTA.execHTTPcheck();
    if (updatedNeeded)
    {
        esp32FOTA.execOTA();
    }
    delay(2000);
}
```

- Web update (requires web server)
- Batch firmware sync
- Force firmware update
- https support
- Signature check
- https or https
- Signature verification
- Semantic versioning support

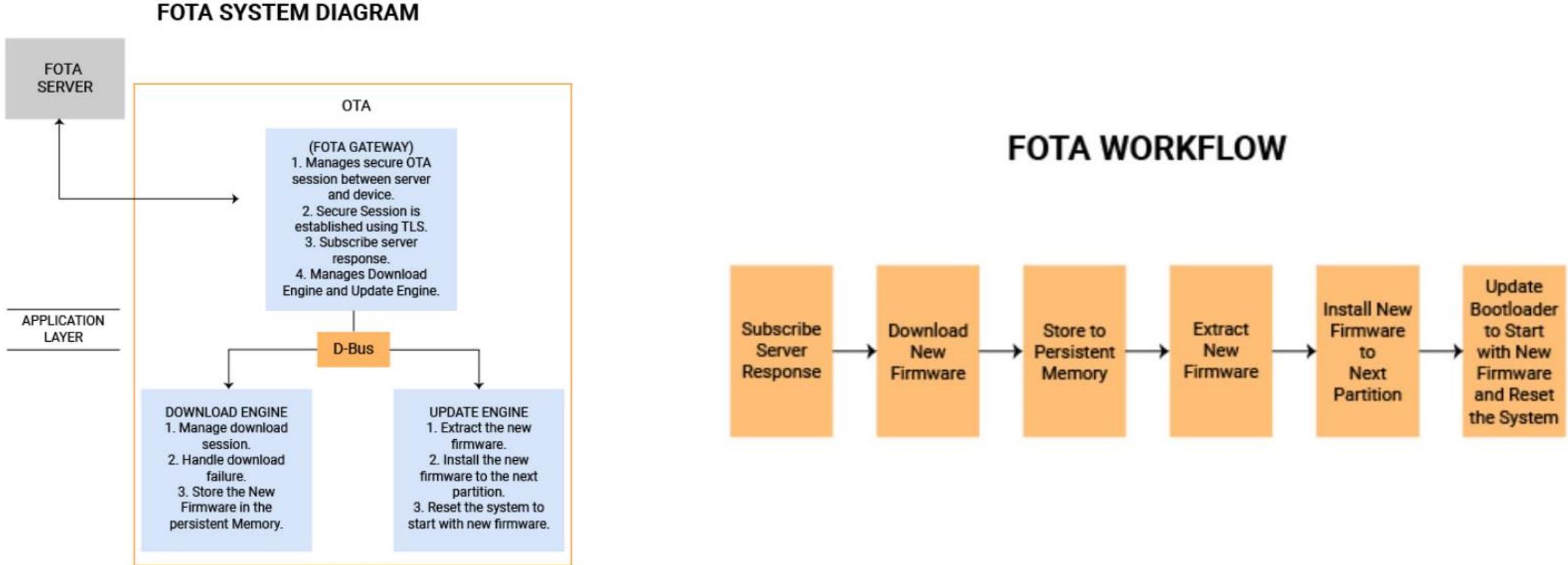
# Emulators and Virtual Devices ?!

# Benefits of OTA Updates in IoT



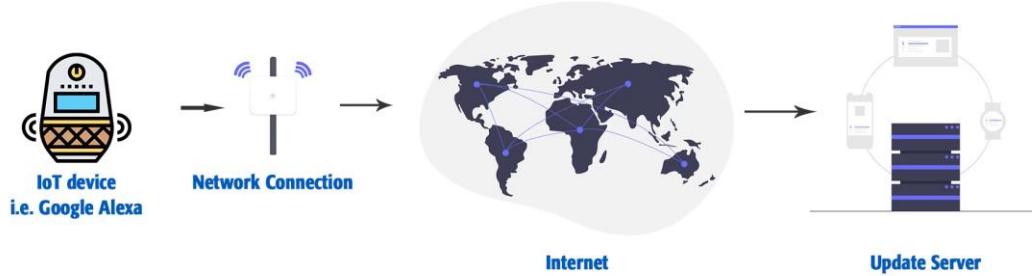
- Improved safety and compliance
- Maintenance cost reduction
- Additional revenue streams
- Reduced operational cost

# FOTA system modules and flow

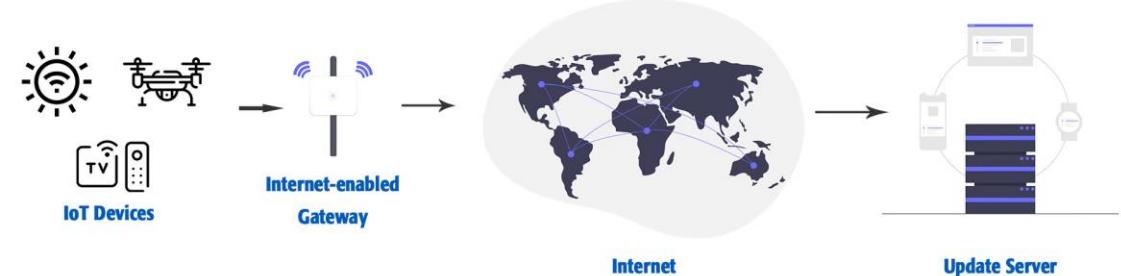


# OTA update methods

## Edge-to-Cloud updates



## Gateway-to-Cloud updates



1. Edge-to-Cloud OTA updates (E2C)
2. Gateway-to-Cloud OTA updates (G2C)
3. Edge-to-Gateway-to-Cloud OTA updates (E2G2C)

Benefits of OTA Updates in IoT

<https://jfrog.com/connect/post/ota-updates-for-remote-devices-different-methods-explained/>

# Best practices for OTA updates

**Incorporating an OTA mechanism from day-one:** adding OTA update routines to the firmware greatly improves the chance of introducing feature updates and bug fixes at a later time, during the production and even after deployment.

**Progressive development approach:** following a progressive development approach, it can not only make the development more reliable, but also streamlines the development of new features.

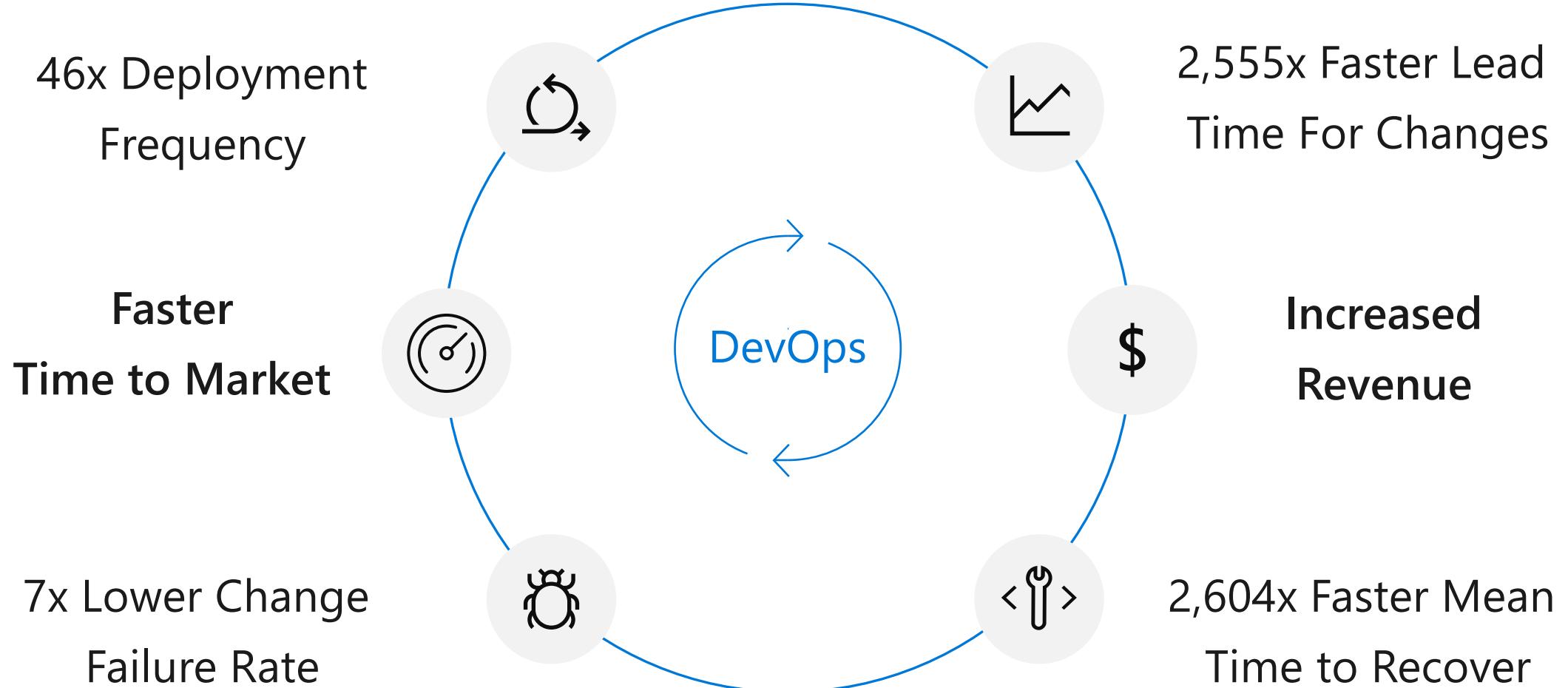
**Prioritized updates:** To make sure that the functionality of the running devices is not interrupted.

**Atomic updates with rollback:** instead of pushing large updates, micro updates

**Backing up current configuration**

**Management platform:** using a central management platform, the previous updates must be logged and managed to track the changes up to the date.

# High Performance DevOps Companies Achieve...



# Secure firmware Over-The-Air updates for IoT

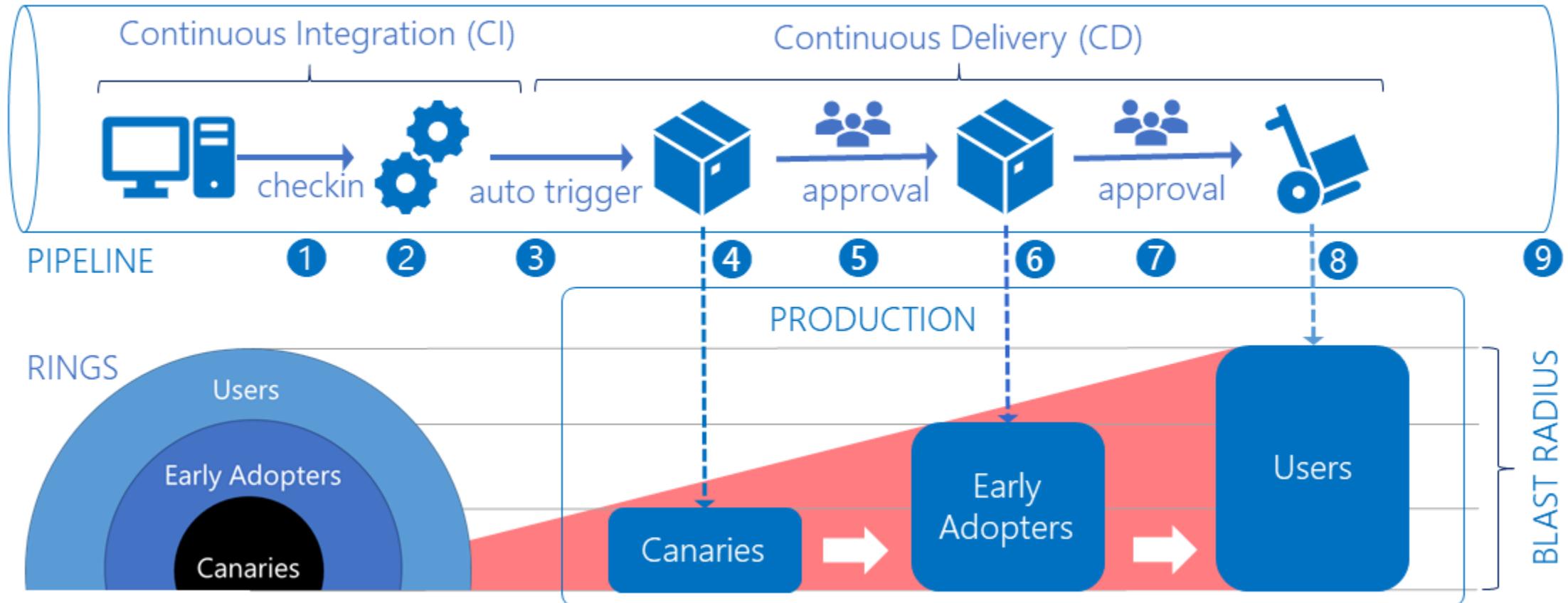
In practice, there are **multiple ways of updating an IoT device**, since **different manufacturers** follow **different approaches** depending on their tools, infrastructures, and strategies.

These ways fall in one of these **two categories**:

- (1) the **wired updates** (e.g., via a USB port) or
- (2) the **wireless Over the Air** (OTA) updates.

Also, the update can be either full or partial.

# Final Thoughts



Use deployment rings with extension releases

<https://docs.microsoft.com/en-us/azure/devops/migrate/phase-rollout-with-rings?view=azure-devops>



# ~7700 Scholar Articles

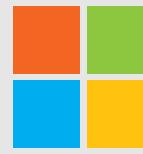
Syed, M.H. and Fernandez, E.B., 2016, April. **Cloud ecosystems support for Internet of Things and DevOps using patterns**. In 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI) (pp. 301-304). IEEE.

Bou Ghantous, G. and Gill, A.Q., 2021. **Evaluating the DevOps Reference Architecture for Multi-cloud IoT-Applications**. SN Computer Science, 2(2), pp.1-35.

Judvaitis, J., Balass, R. and Greitans, M., 2021. **Mobile IoT-Edge-Cloud Continuum Based and DevOps Enabled Software Framework**. Journal of Sensor and Actuator Networks, 10(4), p.62.

Truong, H.L. and Klein, P., 2020. **Devops contract for assuring execution of iot microservices in the edge**. Internet of Things, 9, p.100150.

López-Peña, M.A., Díaz, J., Pérez, J.E. and Humanes, H., 2020. **DevOps for IoT systems: Fast and continuous monitoring feedback of system availability**. IEEE Internet of Things Journal, 7(10), pp.10695-10707.



Microsoft

© 2018 Microsoft Corporation. All rights reserved.

Thank you.