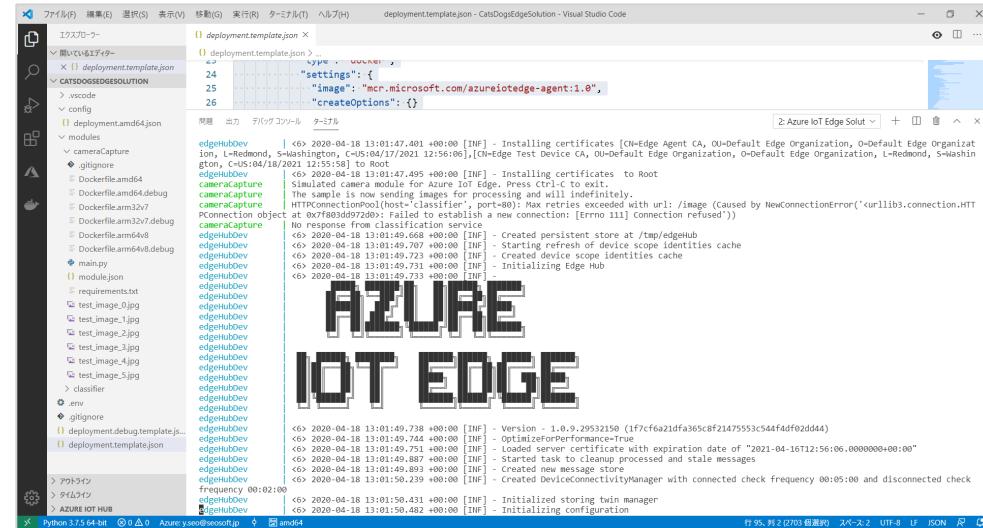


Azure IoT Edge で Custom Vision



The screenshot shows a Visual Studio Code interface with several tabs open. The main tab displays the contents of `deployment.template.json`, which includes settings for an edge agent and a camera module. Below this, a terminal window shows the logs from the Azure IoT Edge runtime. The logs detail the installation of certificates, the creation of device scope identities, and the initialization of the Edge hub. In the center of the screen, there is a watermark or logo for "AZURE IOT EDGE".

```
deployment.template.json
{
  "modules": [
    {
      "name": "edgeAgent",
      "type": "edgeAgent",
      "settings": {
        "image": "mcr.microsoft.com/azureiotedge-agent:1.0",
        "createOptions": {}
      }
    },
    {
      "name": "edgeHub",
      "type": "edgeHub",
      "settings": {}
    }
  ],
  "routes": [
    {
      "name": "AllToEdgeHub",
      "routeFrom": "/*",
      "routeTo": "edgeHub"
    }
  ]
}
```

```
[2020-04-18 13:01:47.401 +00:00] [INF] - Installing certificates [CN=Edge Agent CA, O=Default Edge Organization, O=Default Edge Organization, L=Redmond, S=Washington, C=US]@17/2021 12:56:06;[CN=Edge Test Device CA, O=Default Edge Organization, O=Default Edge Organization, L=Redmond, S=Washington, C=US]@18/2021 12:55:58] to Root
[2020-04-18 13:01:47.401 +00:00] [INF] - Installing certificates to Root
[2020-04-18 13:01:47.401 +00:00] [INF] - Simulated camera module for Azure IoT Edge. Press Ctrl+C to exit.
[2020-04-18 13:01:47.401 +00:00] [INF] - The sample is now sending images for processing and will indefinitely.
[2020-04-18 13:01:47.401 +00:00] [INF] - HTTPConnectionPool(host='classification', port=80): Max retries exceeded with url: /image (Caused by NewConnectionError('<urllib3.connection.HTTPConnectionPool object at 0x0000000000000000>': Failed to establish a new connection: [Errno 111] Connection refused'))
[2020-04-18 13:01:47.401 +00:00] [INF] - No response from classification service
[2020-04-18 13:01:49.668 +00:00] [INF] - Created persist store at /tmp/edgehub
[2020-04-18 13:01:49.668 +00:00] [INF] - Starting persist store device scope identities cache
[2020-04-18 13:01:49.723 +00:00] [INF] - Created device scope identities cache
[2020-04-18 13:01:49.731 +00:00] [INF] - Initializing Edge Hub
[2020-04-18 13:01:49.733 +00:00] [INF] - Version - 1.0-9-29532159 (1f7cf6a21dfa365c8f2147553c544fdf02dd44)
[2020-04-18 13:01:49.764 +00:00] [INF] - OptimizerPerformance=True
[2020-04-18 13:01:49.751 +00:00] [INF] - Loaded server certificate with expiration date of "2021-04-16T12:56:06.0000000+00:00"
[2020-04-18 13:01:49.887 +00:00] [INF] - Started task to cleanup processed and stale messages
[2020-04-18 13:01:49.893 +00:00] [INF] - Created message store
[2020-04-18 13:01:50.239 +00:00] [INF] - Created DeviceConnectivityManager with connected check frequency 00:05:00 and disconnected check frequency 00:02:00
[2020-04-18 13:01:50.431 +00:00] [INF] - Initialized storing twin manager
[2020-04-18 13:01:50.482 +00:00] [INF] - Initializing configuration
```

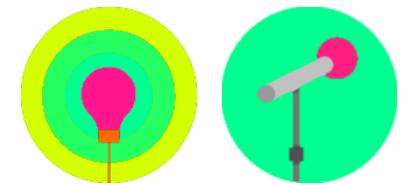
勝手に勉強会
2020年4月19日

瀬尾ソフト 瀬尾佳隆
Microsoft MVP for AI

自己紹介

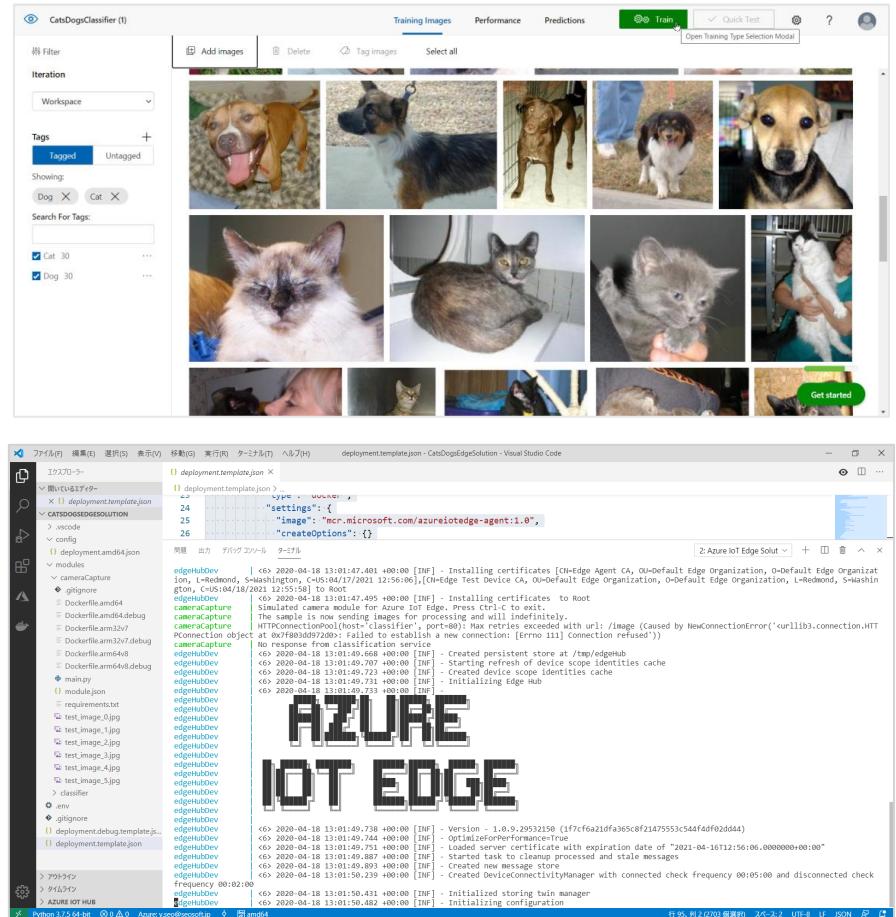
瀬尾 佳隆（せお よしたか）

- MVP for AI (Jul 2018 – Jun 2020) / MVP (Jan 2009 – Jun 2018)
 - <https://github.com/seosoft>
 - <https://yseosoft.wordpress.com/>
 - <https://twitter.com/seosoft>
 - <https://www.facebook.com/seosoft>
- Humans of IT Community Event Leader / Speaker
- Ignite The Tour Tokyo
 - BRK30055 「開発者のための機械学習：Azure Machine Learning サービスで構造化データから予測分析」
 - THR10003 「Humans of IT Empower Breakfast – 人に寄り添うアクセシビリティ技術の現在と未来」
- Tech Summit 2018 – DA09 「ユーザーインターフェースとしてのチャットボット開発手法と Microsoft Bot Framework v4」



今日の内容

- Azure IoT
 - Custom Vision
 - 開発環境
 - ソリューション開発
 - IoT Hub 操作



今日のデモ素材

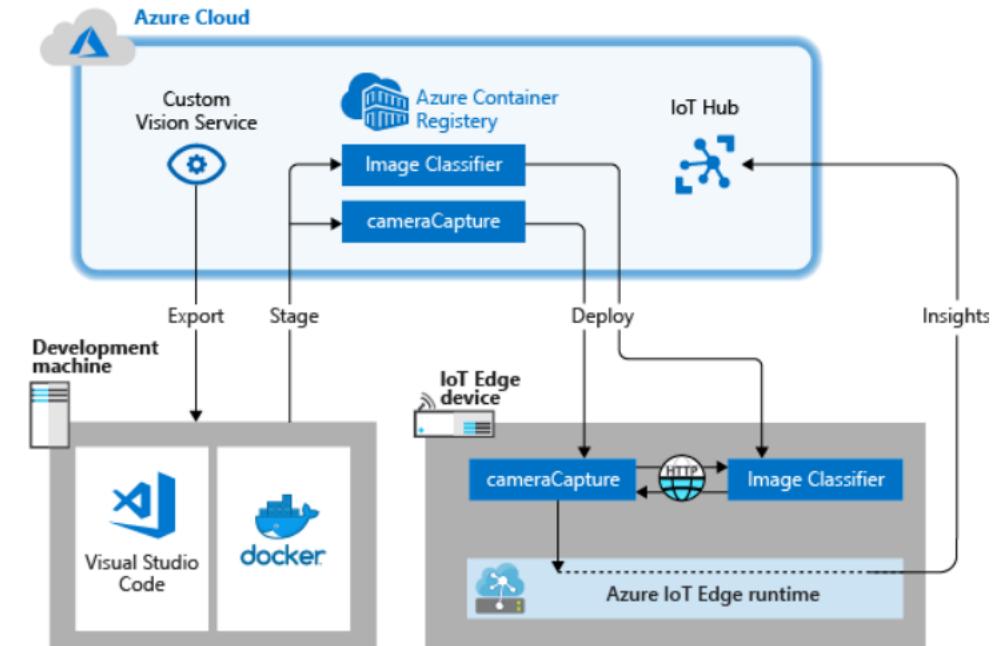
■カメラで撮影した画像を分類する

- ・今回は物理デバイスを使用しないのでカメラはエミュレーターを使う

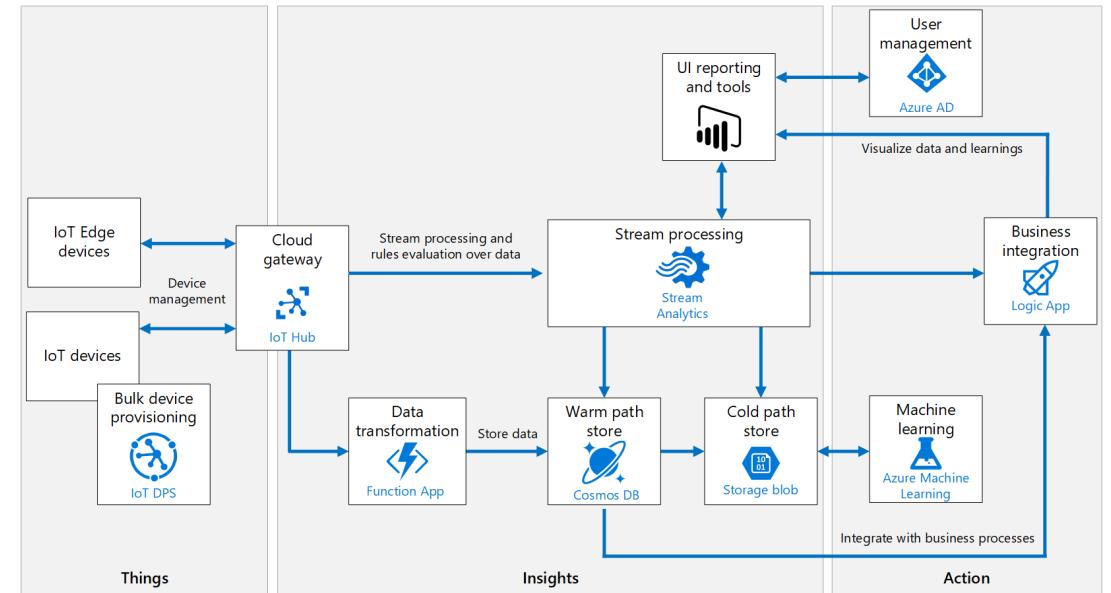
■<https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-deploy-custom-vision>

- ・手順を整理・追加
- ・アプリを少し変更

(未整理のまま、一応 GitHub に上げています。
そのうち整理するかも
<https://github.com/seosoft/IoTCvCatsDogs>)

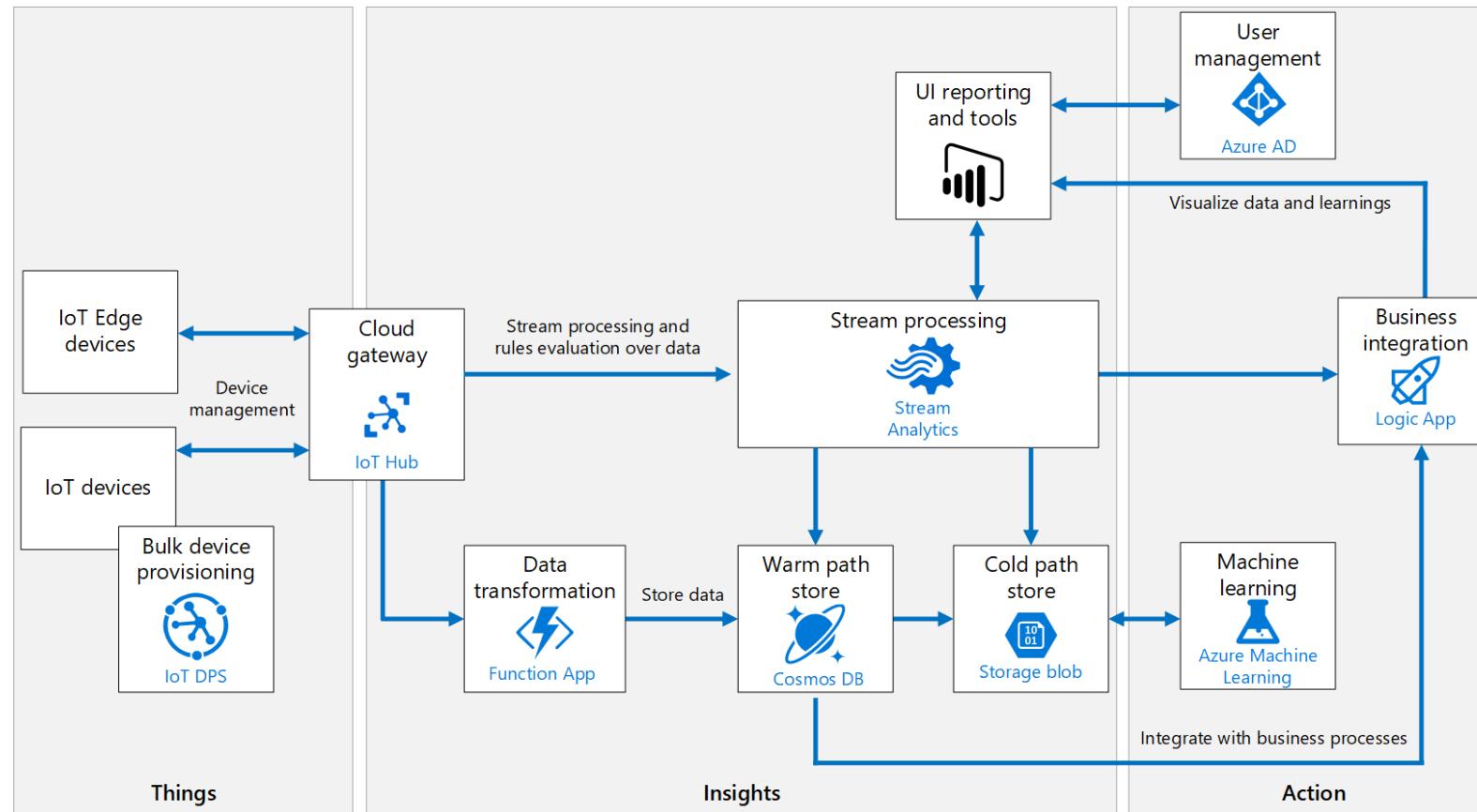


Azure IoT



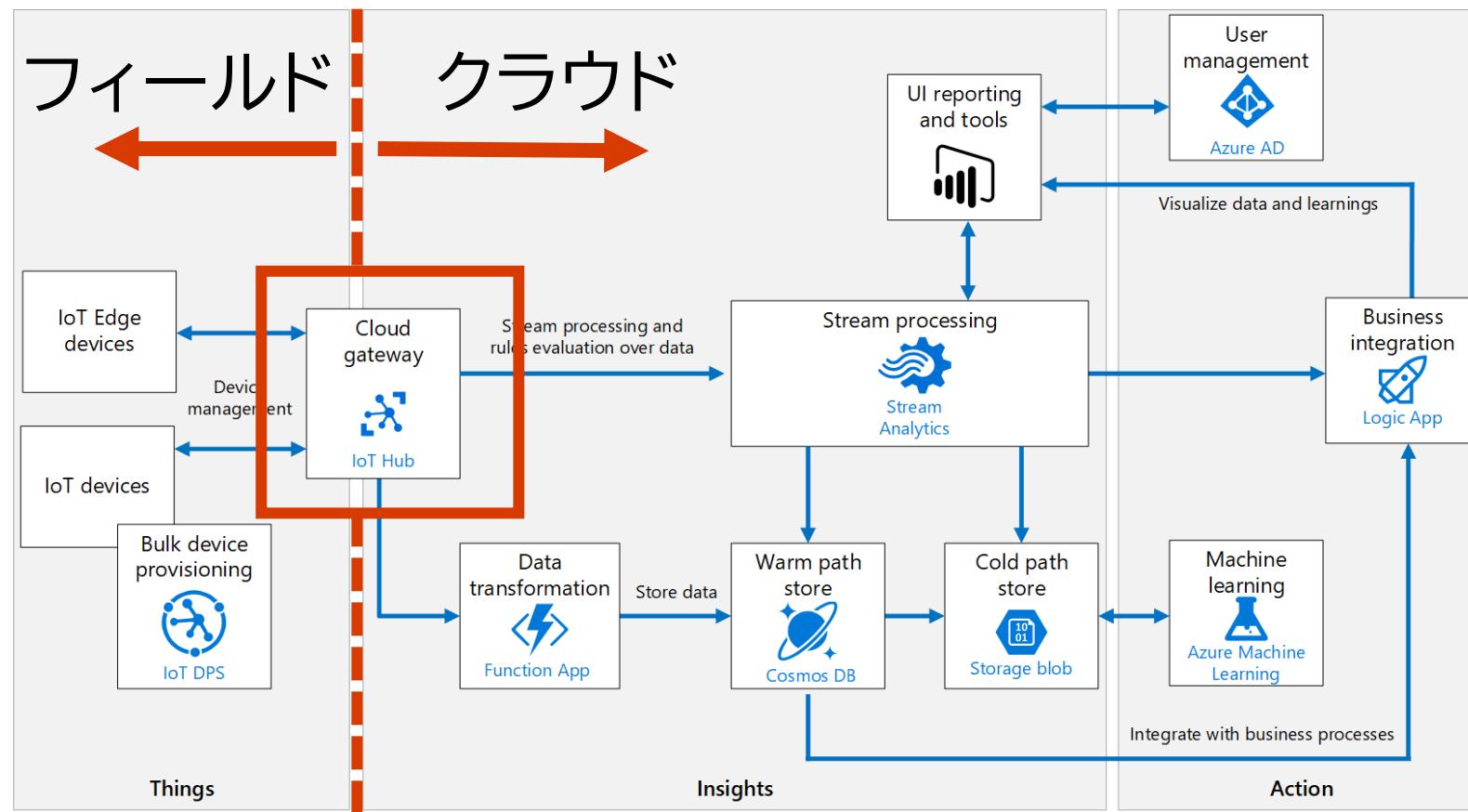
Azure IoT リファレンス アーキテクチャ

■<https://docs.microsoft.com/ja-jp/azure/architecture/reference-architectures/iot/>



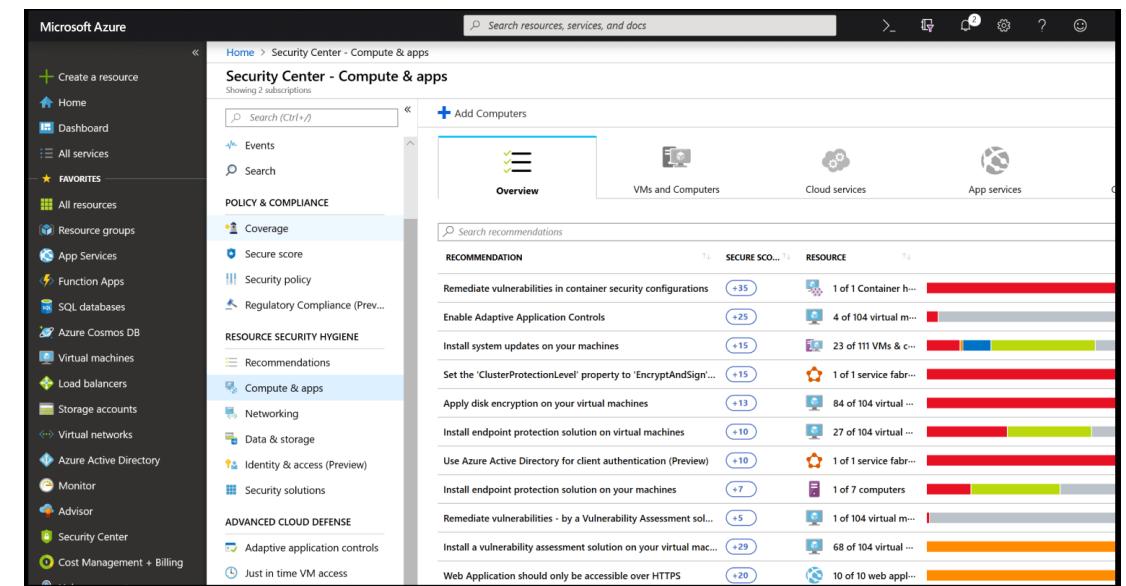
IoT Hub

■ フィールド（デバイス）とクラウドとを繋ぐ
IoT システムで中心となるサービス



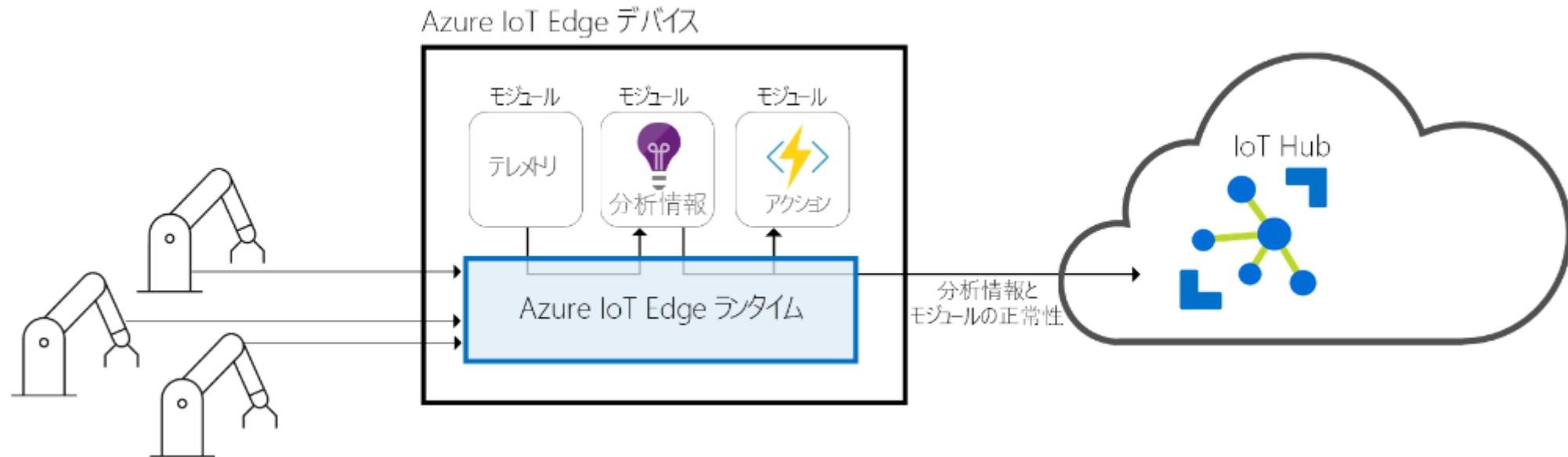
IoT Hub の特徴

- デバイスとクラウドとの間で、安全で信頼性の高い通信を実現
- クラウドでホストされるソリューション バックエンド
 - 数十億台のデバイスとの双方向通信
 - デバイスごとの認証
 - デバイス管理機能
 - スケーリングされたプロビジョニング
 - 多言語でオープンソースの SDK



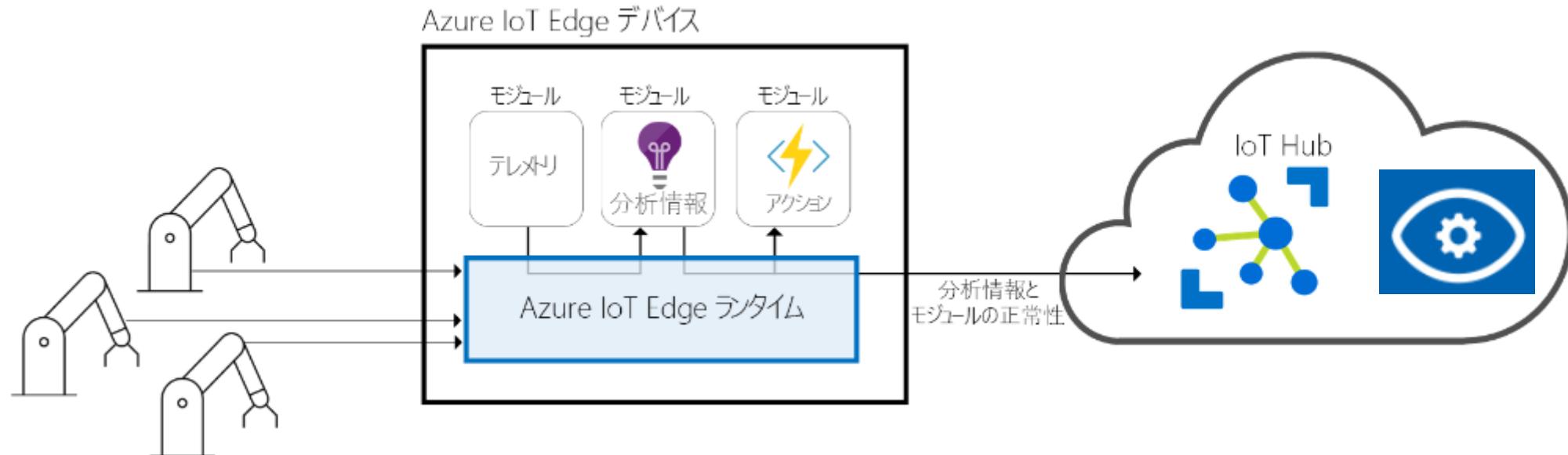
IoT Edge

- クラウドで行っていた分析とカスタム ビジネス ロジックをデバイス側で実行
- ビジネス ロジックを Docker にパッケージ化
- 任のデバイスにデプロイするエミュレータをクラウド



Custom Vision も Edge モジュール

- クラウドで行っていた分析とカスタム ビジネス ロジックをデバイス側で実行
- ビジネス ロジックを Docker にパッケージ化
- 任のデバイスにデプロイ IoT エッジ モジュールをクラウド



IoT Edge の特徴

■応答性の向上

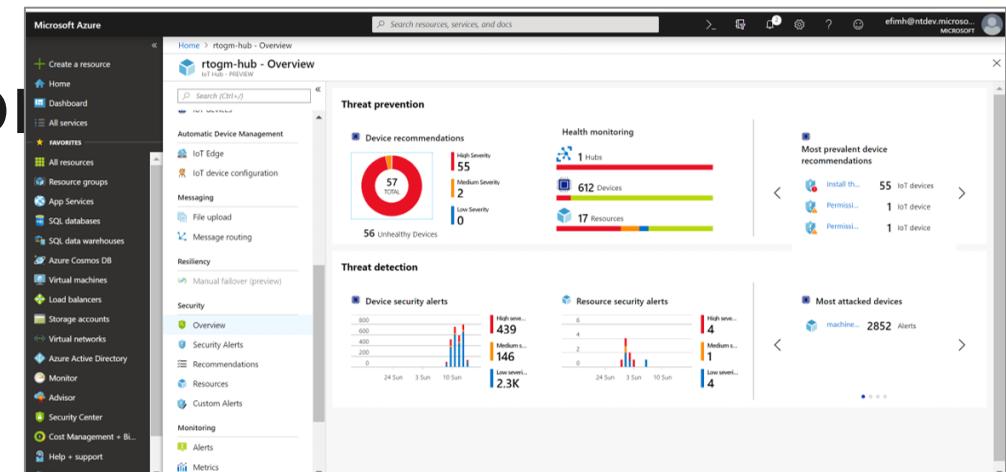
- エッジデバイス側で処理を実行するため、クラウドとの通信の時間が不要

■オフライン時、断続的な接続時でも動作可能

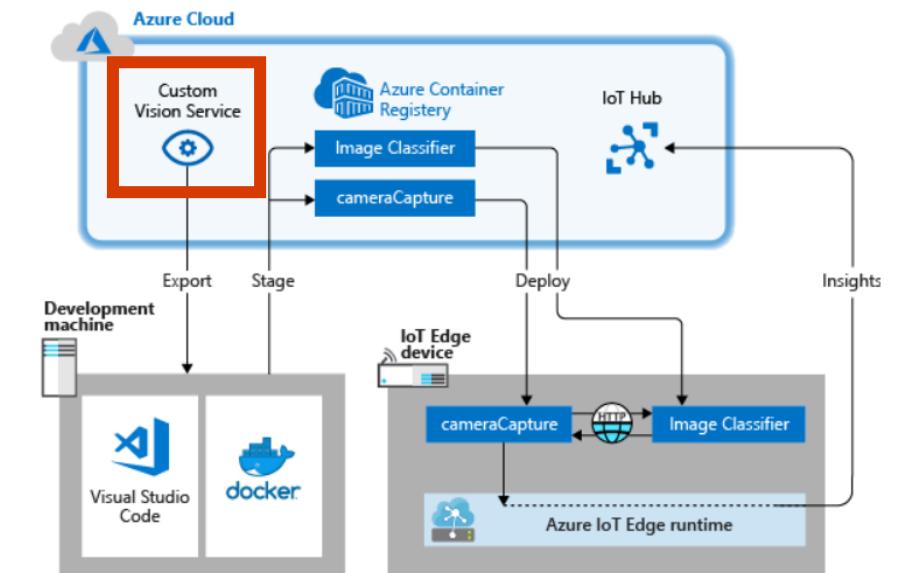
- IoT Hub との通信は IoT Edge デバイス管理によって自動的に実行

■開発の簡略化

- C#, Node.js, C, Java, Python



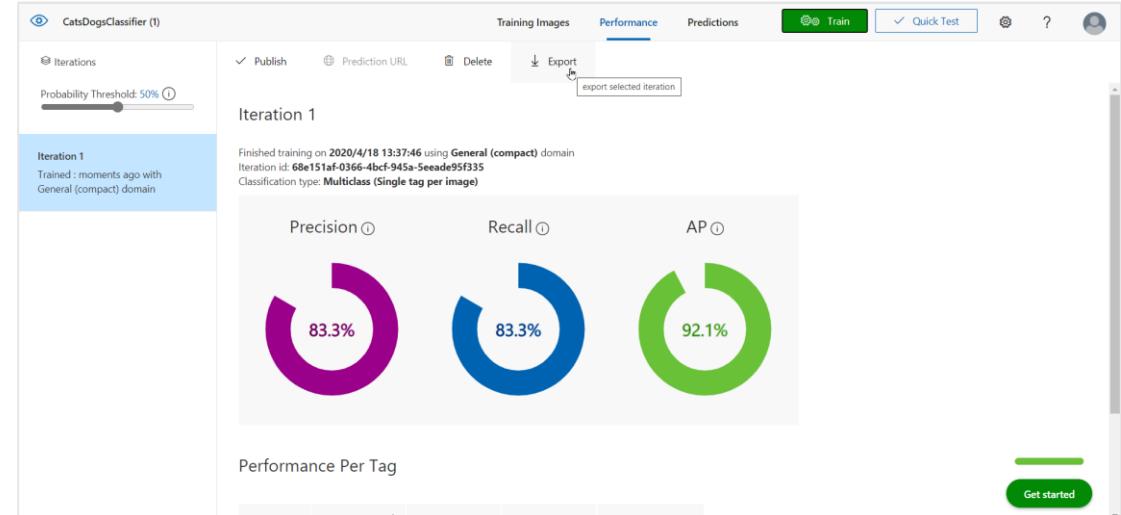
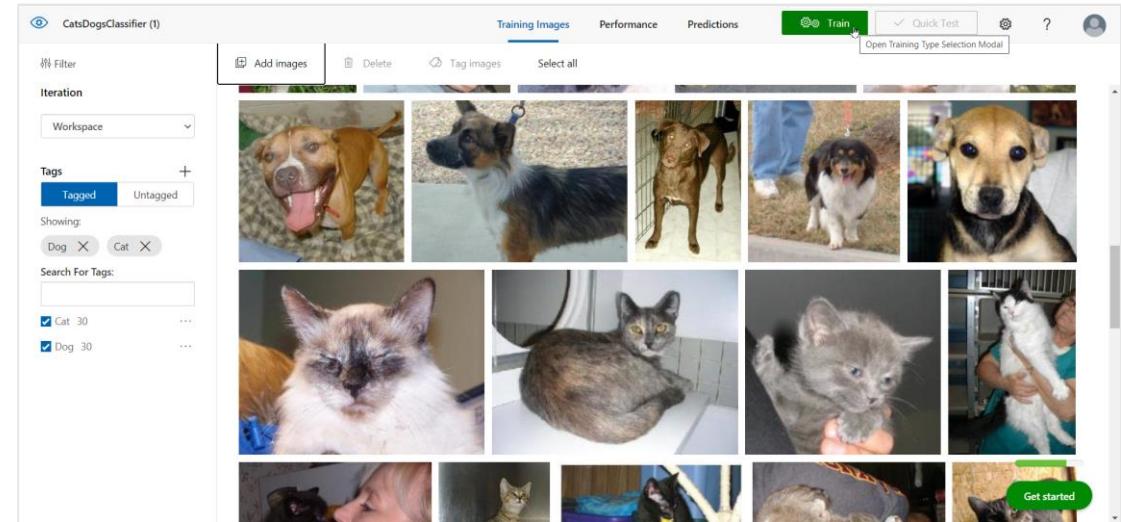
Cognitive Services - Custom Vision



Custom Vision とは

■ 独自の画像認識モデルを構築するサービス

- ・ 少量のデータで学習可能
- ・ 分類とオブジェクト検出
- ・ <https://www.customvision.ai/>



Custom Vision 操作 (1/6)

■ Azure ポータルで “Custom Vision” 作成

- Edge 用のモデルを作るなら “トレーニング” のみ

The image consists of two screenshots from the Microsoft Azure portal.

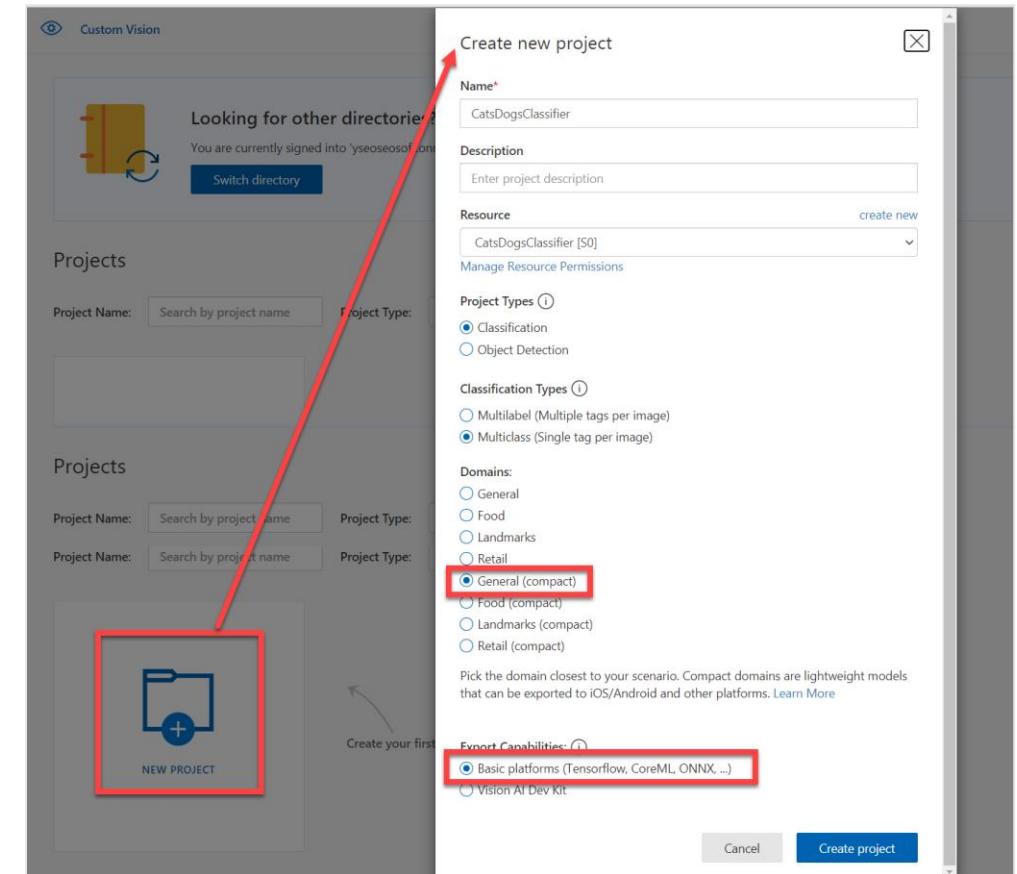
Left Screenshot: Shows the Azure Marketplace search results for "custom vision". A red box highlights the "Custom Vision" service by Microsoft. A red arrow points from the search bar to the service card.

Right Screenshot: Shows the "Create" blade for "Custom Vision". The "Training" tab is highlighted with a red box. The blade includes fields for "Project details" such as "Subscription", "Resource Group" (set to "IoTCatsDogs"), "Name" (set to "CatsDogsClassifier"), "Training Location" (set to "(US) 米国西部 2"), and "Training Price Level" (set to "S0 (10 トランザクション/秒)").

Custom Vision 操作 (2/6)

■ <https://customvision.ai/>

項目	選択する値
Domains	Compact
Export Capabilities	Basic Platforms

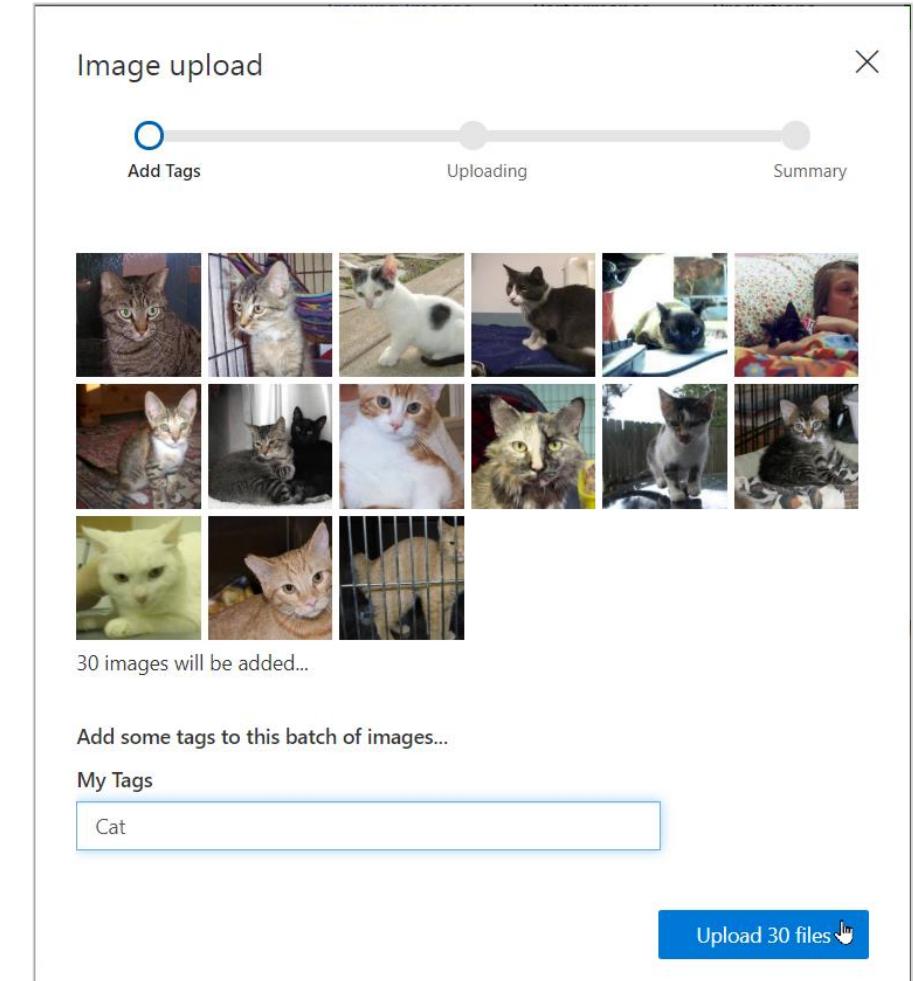


Custom Vision 操作 (3/6)

- ラベル付きで画像アップロード
- [Train] で学習

今回は “Kaggle” の “Dogs & Cats Image” を使用
(犬・猫の画像をそれぞれ 30枚)

<https://www.kaggle.com/chetankv/dogs-cats-images>

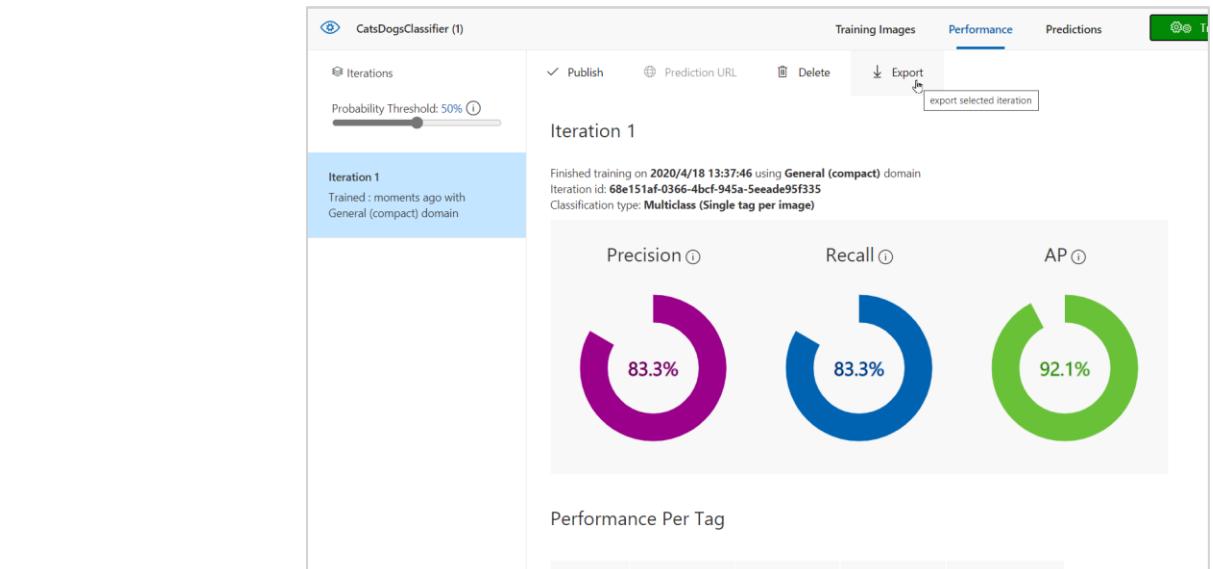
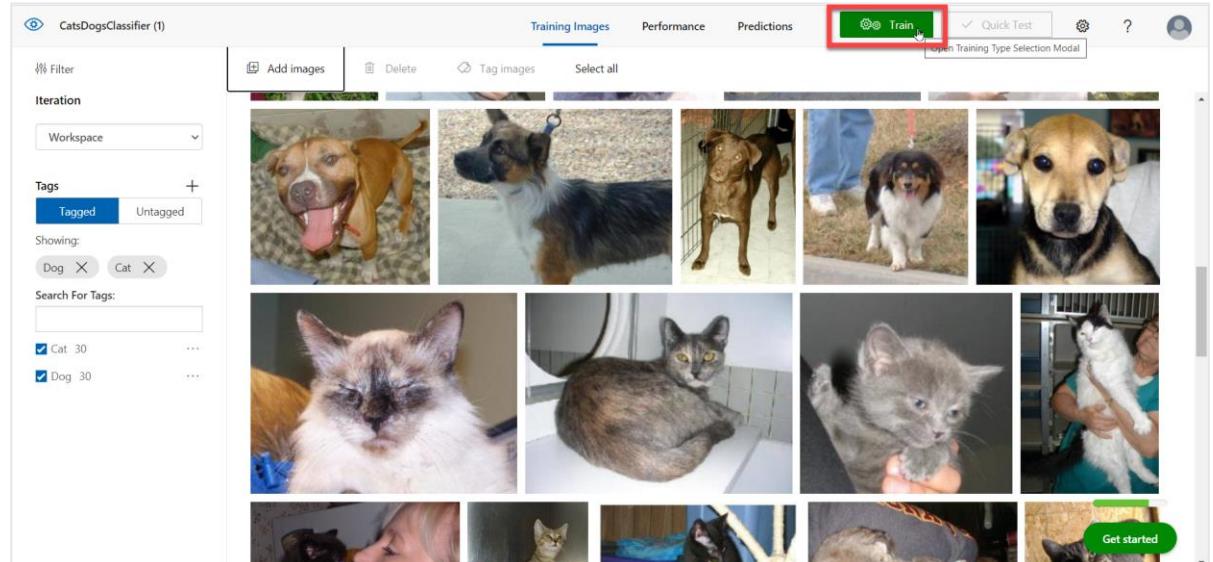


Custom Vision 操作 (4/6)

■ [Train] ボタンで学習

■ スコアを確認

- “Compact” モデルは少し品質が低くなる

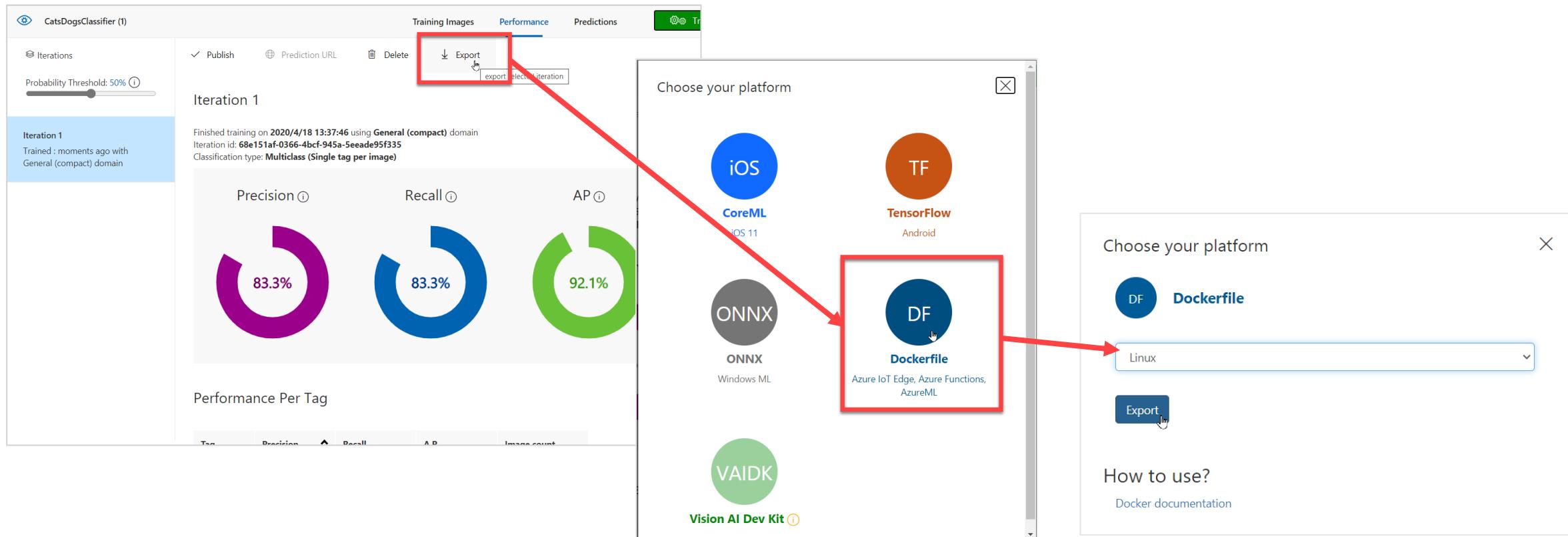


今回は“衝撃に結末”が… (あとのお楽しみ)

Custom Vision 操作 (5/6)

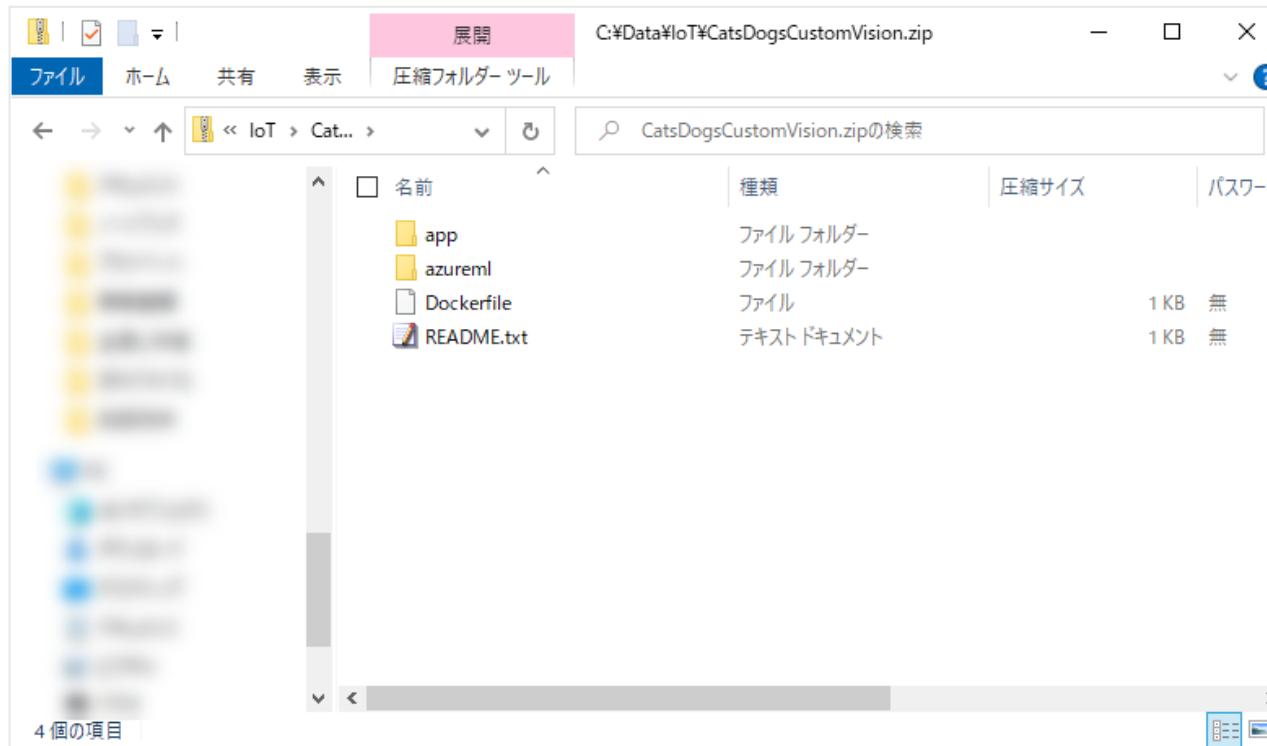
■Dockerfile をエクスポート

- ・ターゲットは今回は“Linux”

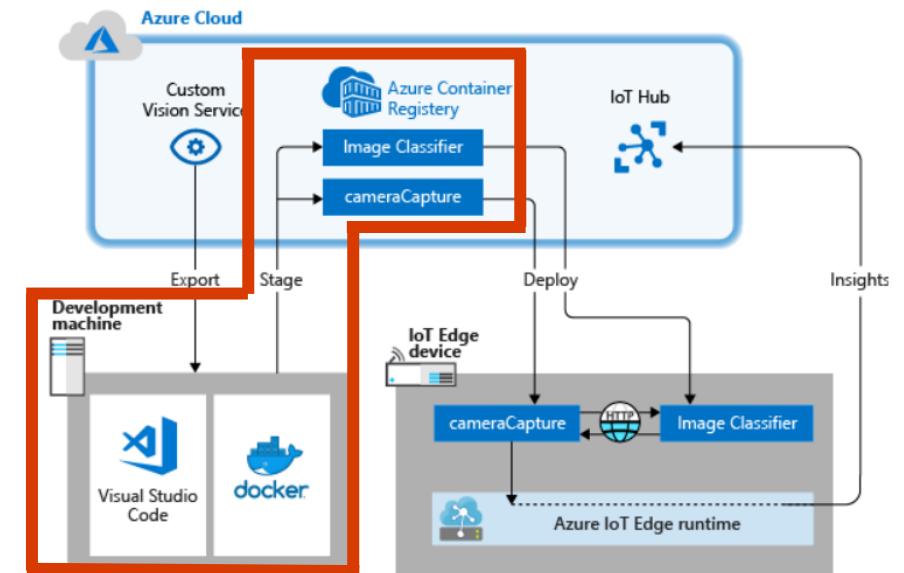


Custom Vision 操作 (6/6)

■ZIP ファイルがエクスポートされる



ソリューション開発



開発で使用するツール

■Visual Studio Code

- <https://code.visualstudio.com/>

■Azure IoT Tools (VSCode 拡張機能)

- <https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.azure-iot-tools>

■Docker Desktop

- <https://www.docker.com/products/docker-desktop>

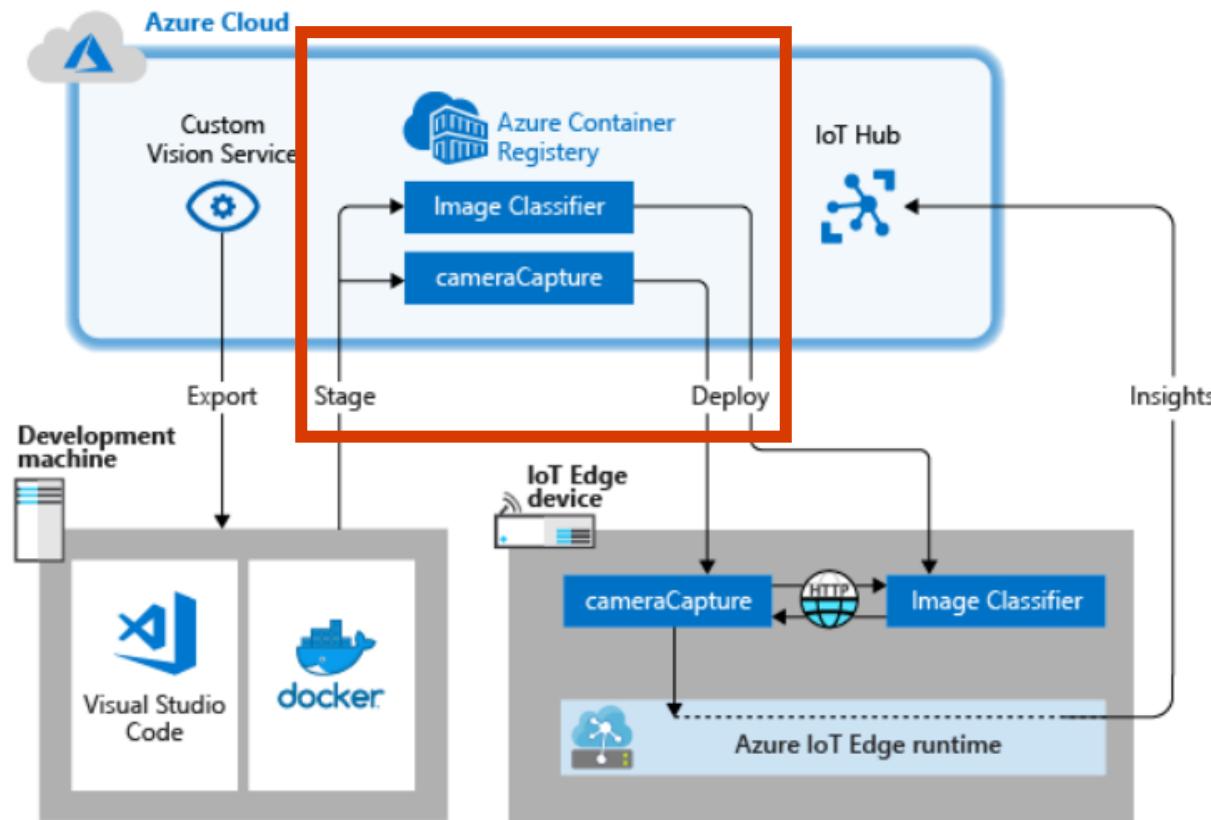
■Anaconda (または Python 3.6 以上)

- <https://www.anaconda.com/distribution/#download-section>

Azure Container Registry

■ IoT Edge 用のモジュールを管理する

- <https://azure.microsoft.com/ja-jp/services/container-registry/>



Container Registry 作成

■ Azure ポータルで作成

- ・管理者ユーザーを有効にする
- ・今回は“Basic”で十分
- ・作成したらアクセスキーを記録

Microsoft Azure リソース、サービス、ドキュメントの検索 (G+/)

ダッシュボード > IoT Cats Dogs > 新規 > Container Registry > コンテナー レジストリの作成

コンテナー レジストリの作成

基本 * 暗号化 タグ 確認および作成

Azure Container Registry を使用すると、すべての種類のコンテナー配置についてプライベート レジストリ内のコンテナー イメージと成果物をビルド、保存、管理できます。Azure コンテナー レジストリは、既存のコンテナー 開発と配置 バイブルで使用します。Azure Container Registry タスクを使用すると、オブジェクトで Azure にコンテナー イメージをビルドしたり、ソース コードの更新、コンテナー の基本 イメージの更新、またはタイマーによって自動的にビルドをトリガーしたりすることができます。 詳細情報

プロジェクトの詳細

サブスクリプション * AI Subscription

リソース グループ * IoT Cats Dogs

新規作成

インスタンスの詳細

レジストリ名 * catsdogsreg.azurecr.io

場所 * (US) 米国西部 2

管理者ユーザー * 有効 無効

SKU * Basic

確認および作成 < 戻る 次へ: 暗号化 >

Microsoft Azure リソース、サービス、ドキュメントの検索 (G+/)

catsdogsreg | アクセスキー

検索 (Ctrl+ /)

概要 アクティビティ ログ アクセス制御 (IAM) タグ クイック スタート イベント

ログイン サーバー catsdogsreg.azurecr.io

管理者ユーザー ① 有効 無効

ユーザー名 catsdogsreg

名前 パスワード
password
password2

アクセスキー

VSCode でソリューション作成 (1/2)

■ Docker Desktop を起動

■ ソリューション作成

1. コマンド パレットで “New IoT Edge Solution”
2. ソリューション名は今回は “CatsDogsEdgeSolution”
3. Module Template は “Python Module”
4. モジュール名は今回は “classifier”
5. Docker Image Repository は
<作成した ACR のログインサーバー名> + “/classifier”

VSCode でソリューション作成 (2/2)

The image consists of three vertically stacked screenshots of the Visual Studio Code interface, illustrating the steps to create an Azure IoT Edge solution.

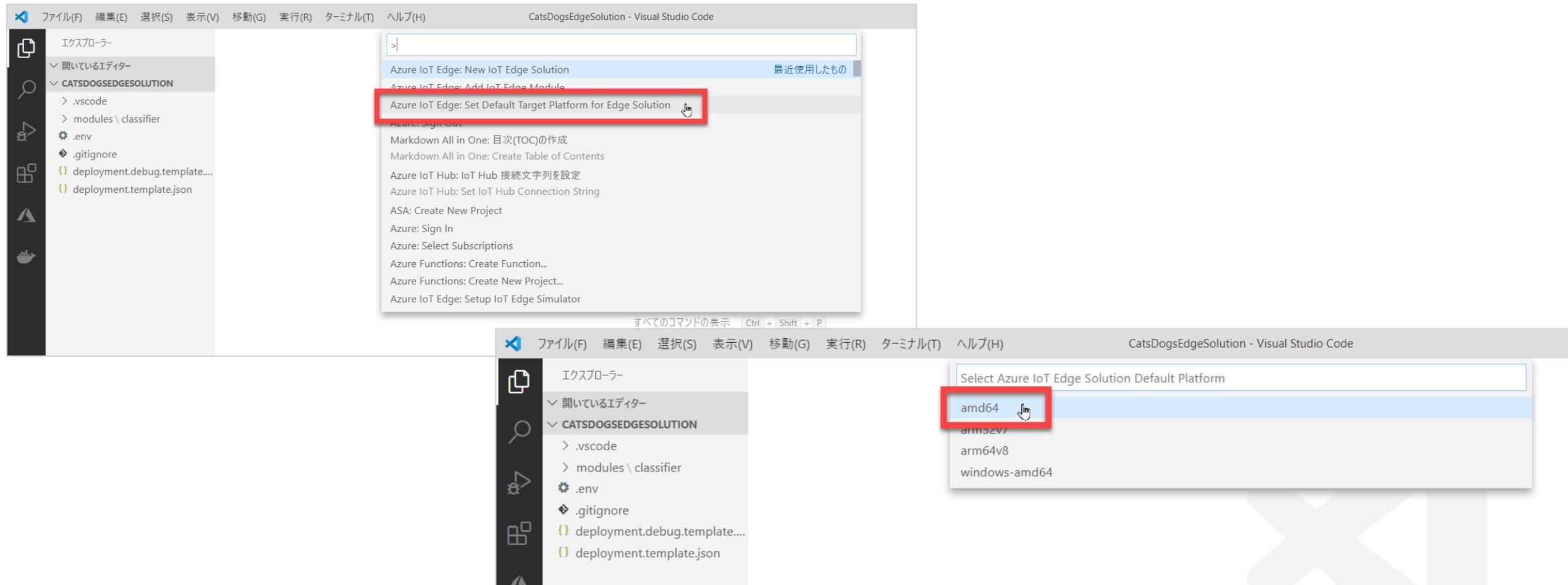
Screenshot 1: The Command Palette (Ctrl+Shift+P) is open, showing a list of Azure IoT Edge commands. The command **Azure IoT Edge: New IoT Edge Solution** is highlighted and has a red box around it, indicating it is the next step.

Screenshot 2: The Command Palette is again open, but this time the **Select Module Template** section is visible. The template **Python Module: Use Azure IoT Python SDK to build a module** is highlighted and has a red box around it.

Screenshot 3: The Command Palette is open once more, showing a Docker image repository input field. The URL **localhost:5000/classifier** is entered. A red arrow points from this field to another input field below it, which contains the URL **catsdogsreg.azurecr.io/classifier**. Both fields have red boxes around them.

ターゲット プラットフォームを設定

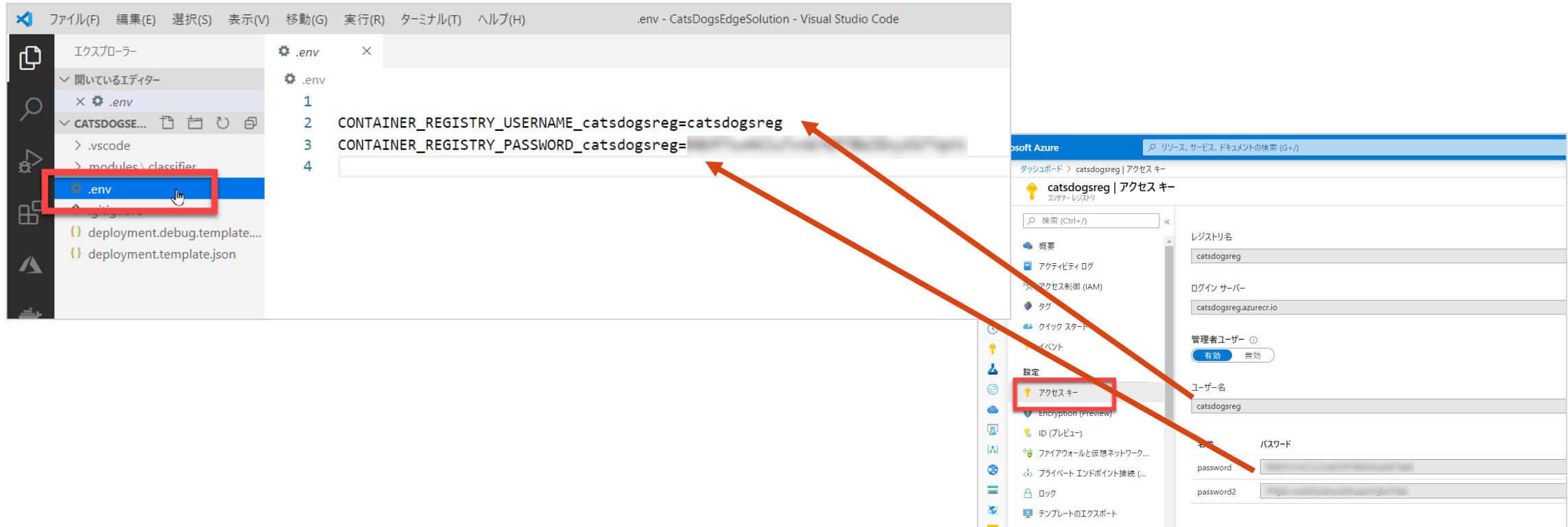
■コマンド パレットで “Set Default Platform” ・今回は “amd64” (64ビット Linux) を選択



ACR 資格情報を確認

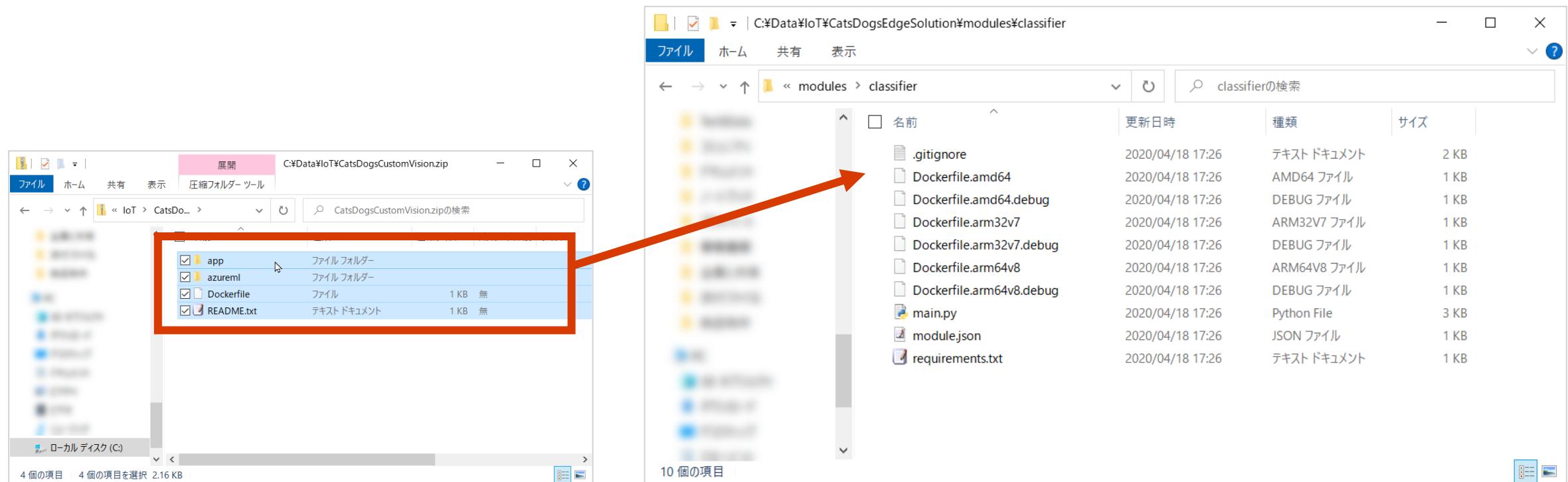
■ “.env” ファイルに入力済みのはず

- もし空ならば ACR のアクセスキー ブレードからコピー



Custom Vision のファイルをコピー

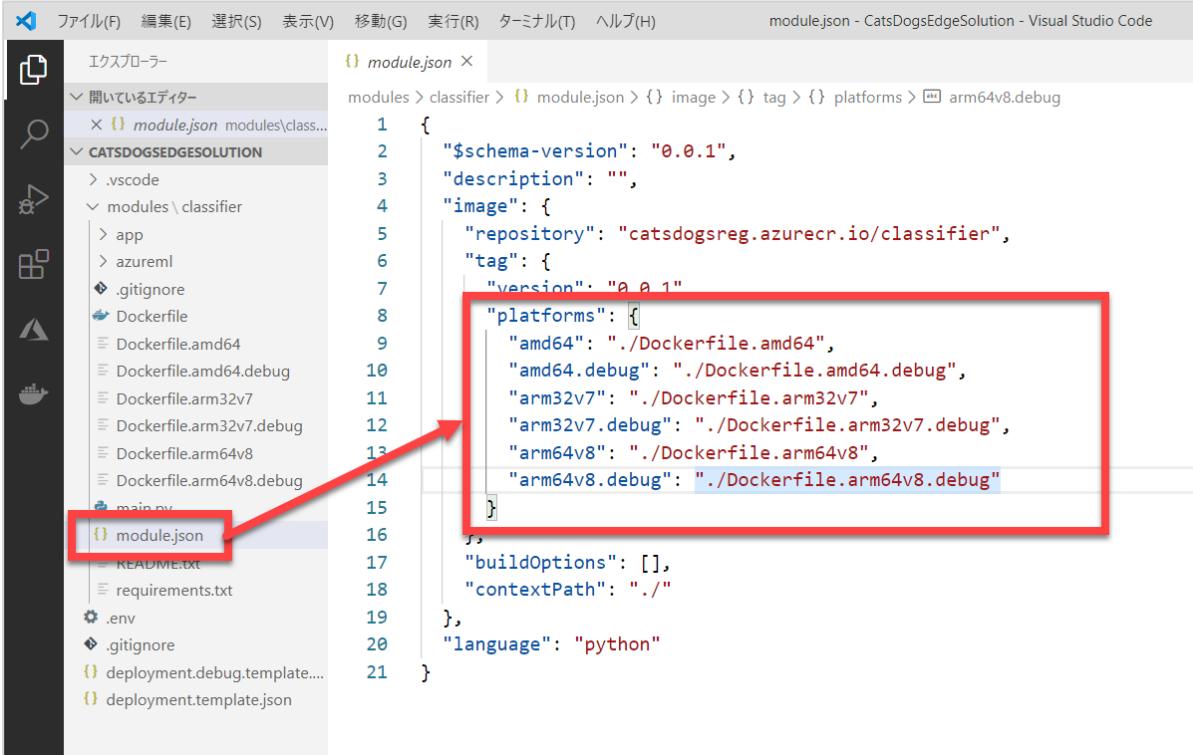
■エクスポートした ZIP ファイルの中身を
ソリューションの “modules¥classifier” にコピー



module.json を編集

■ “platforms” を編集

- ・エントリーは “amd64” のみ
- ・値は “./Dockerfile”



```
1 {
2   "$schema-version": "0.0.1",
3   "description": "",
4   "image": {
5     "repository": "catsdogsreg.azurecr.io/classifier",
6     "tag": {
7       "version": "0.0.1"
8     }
9   }
10   "platforms": [
11     "amd64": "./Dockerfile.amd64",
12     "amd64.debug": "./Dockerfile.amd64.debug",
13     "arm32v7": "./Dockerfile.arm32v7",
14     "arm32v7.debug": "./Dockerfile.arm32v7.debug",
15     "arm64v8": "./Dockerfile.arm64v8",
16     "arm64v8.debug": "./Dockerfile.arm64v8.debug"
17   ],
18   "buildOptions": [],
19   "contextPath": "./"
20 },
21   "language": "python"
}
```

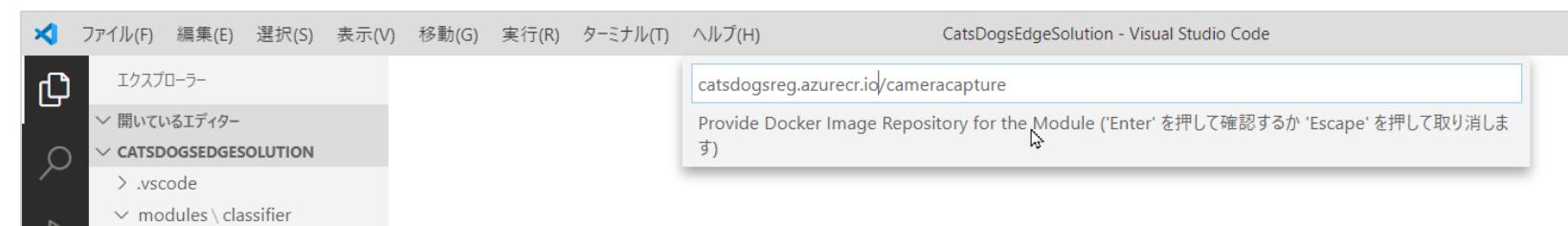
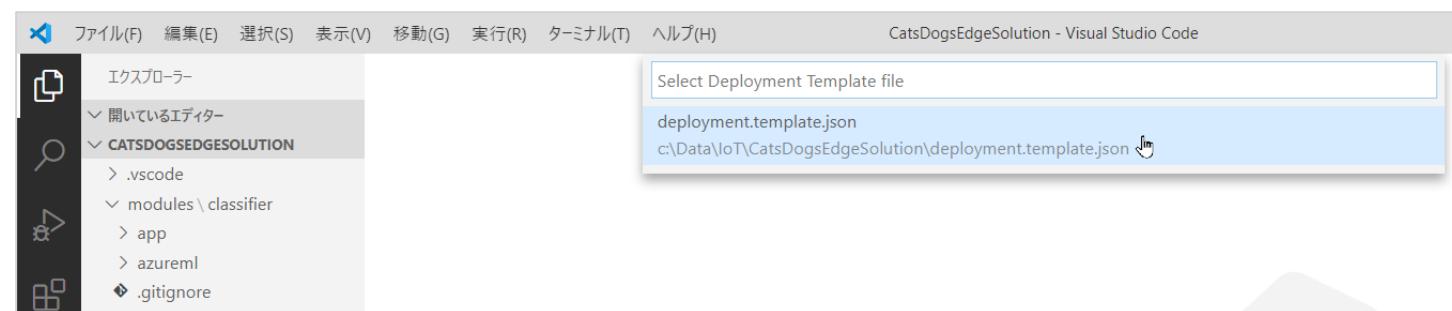
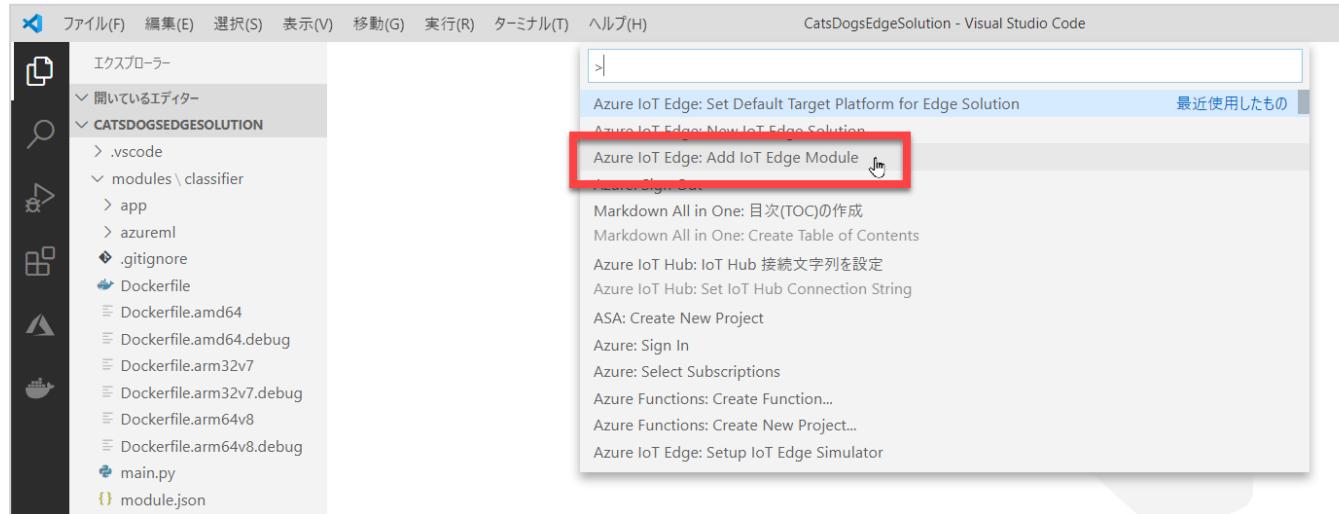
```
1 {
2   "$schema-version": "0.0.1",
3   "description": "",
4   "image": {
5     "repository": "catsdogsreg.azurecr.io/classifier",
6     "tag": {
7       "version": "0.0.1"
8     }
9   }
10   "platforms": [
11     "amd64": "./Dockerfile"
12   ],
13   "buildOptions": [],
14   "contextPath": "./"
15 },
16   "language": "python"
}
```

カメラ モジュールを追加 (1/2)

■Classifier とは別のモジュールを追加

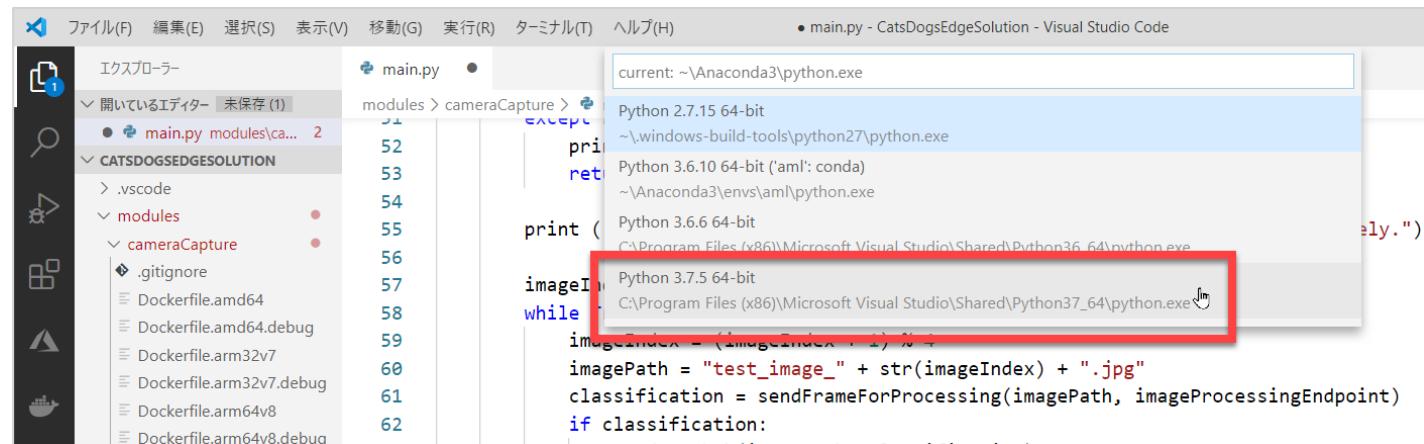
1. コマンド パレットで “Add IoT Edge Module”
2. Deployment Template は “deployment.template.json”
3. Module Template は “Python Module”
4. モジュール名は今回は “cameraCapture”
5. Docker Image Repository は
<作成した ACR のログインサーバー名> + “/cameracapture”

カメラ モジュールを追加 (2/2)



cameraCapture の main.py を編集 (1/6)

- <https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-deploy-custom-vision> の内容を参考に
- Python のインタープリターを聞かれたら 3.6 または 3.7 を適当に（必須ではない）



cameraCapture の main.py を編集 (2/6)

```
# Copyright (c) Microsoft. All rights reserved.  
# Licensed under the MIT license. See LICENSE file in the project root for  
# full license information.  
  
import time  
import sys  
import os  
import requests  
import json  
from azure.iot.device import IoTHubModuleClient, Message  
  
# global counters  
SENT_IMAGES = 0  
  
# global client  
CLIENT = None
```

cameraCapture の main.py を編集 (3/6)

```
# Send a message to IoT Hub
# Route output1 to $upstream in deployment.template.json
def send_to_hub(imagePath, strMessage):
    message = Message(bytarray(strMessage, 'utf8'))
    CLIENT.send_message_to_output(message, "output1")
    global SENT_IMAGES
    SENT_IMAGES += 1
    print("imagePath: " + imagePath)
    print("Total images sent: {}".format(SENT_IMAGES))
```

cameraCapture の main.py を編集 (4/6)

```
# Send an image to the image classifying server
# Return the JSON response from the server with the prediction result
def sendFrameForProcessing(imagePath, imageProcessingEndpoint):
    headers = {'Content-Type': 'application/octet-stream'}

    with open(imagePath, mode="rb") as test_image:
        try:
            response = requests.post(imageProcessingEndpoint, headers = headers, data =
test_image)
            print("Response from classification service: (" + str(response.status_code) + ") " +
json.dumps(response.json()) + "\n")
        except Exception as e:
            print(e)
            print("No response from classification service")
            return None

    return json.dumps(response.json())
```

cameraCapture の main.py を編集 (5/6)

```
def main(imageProcessingEndpoint):
    try:
        print ( "Simulated camera module for Azure IoT Edge. Press Ctrl-C to exit." )

    try:
        global CLIENT
        CLIENT = IoTHubModuleClient.create_from_edge_environment()
    except Exception as iothub_error:
        print ( "Unexpected error {} from IoTHub".format(iothub_error) )
        return

    print ( "The sample is now sending images for processing and will indefinitely." )

    imageIndex = -1
    while True:
        imageIndex = (imageIndex + 1) % 6
        imagePath = "test_image_" + str(imageIndex) + ".jpg"
        classification = sendFrameForProcessing(imagePath, imageProcessingEndpoint)
        if classification:
            send_to_hub(imagePath, classification)
            time.sleep(10)

    except KeyboardInterrupt:
        print ( "IoT Edge module sample stopped" )
```

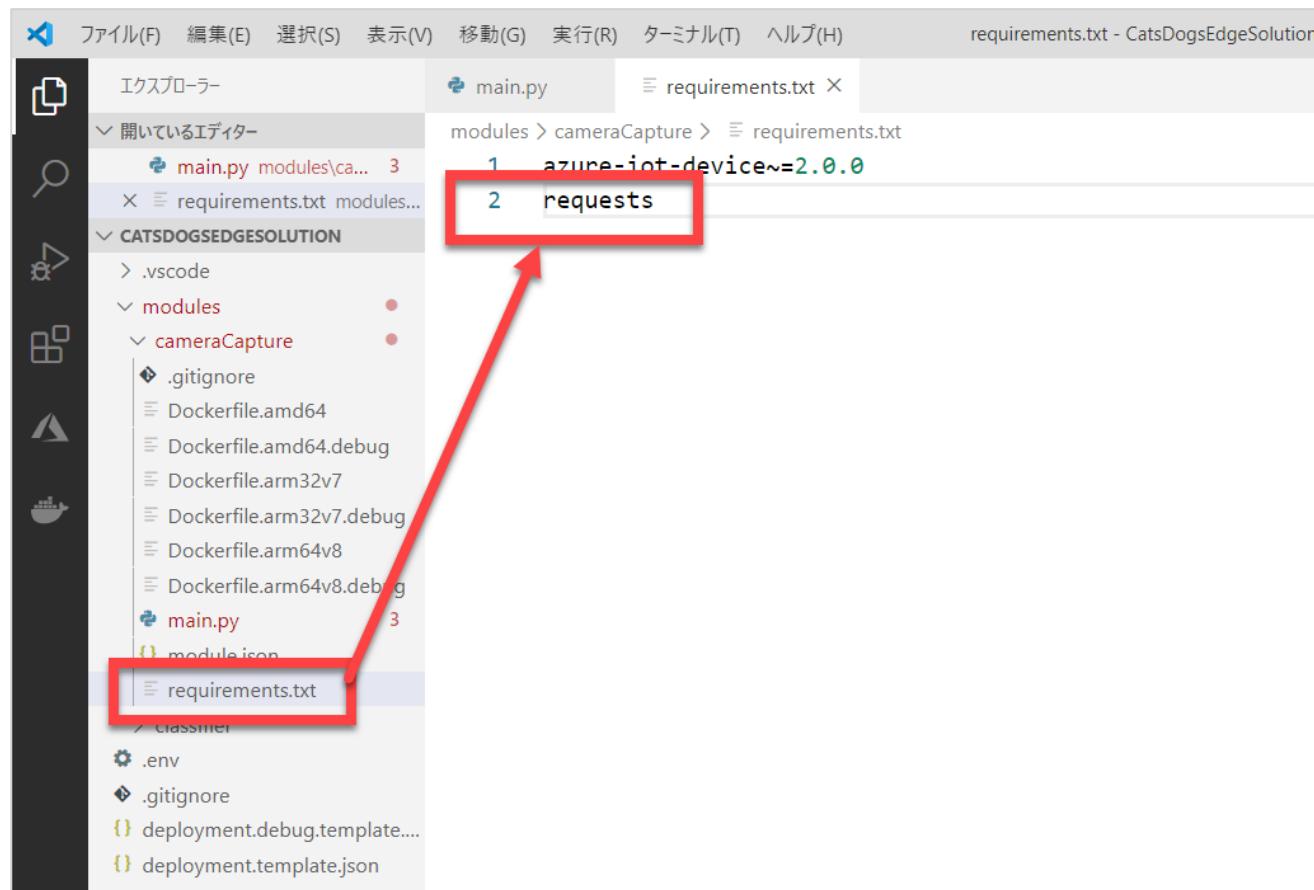
cameraCapture の main.py を編集 (6/6)

```
if __name__ == '__main__':
    try:
        # Retrieve the image location and image classifying server endpoint from container
        # environment
        # IMAGE_PATH = os.getenv('IMAGE_PATH', "")
        IMAGE_PROCESSING_ENDPOINT = os.getenv('IMAGE_PROCESSING_ENDPOINT', "")
    except ValueError as error:
        print( error )
        sys.exit(1)

    # if ((IMAGE_PATH and IMAGE_PROCESSING_ENDPOINT) != ""):
    if (IMAGE_PROCESSING_ENDPOINT != ""):
        main(IMAGE_PROCESSING_ENDPOINT)
    else:
        print( "Error: Image path or image-processing endpoint missing" )
```

requirements.txt を編集

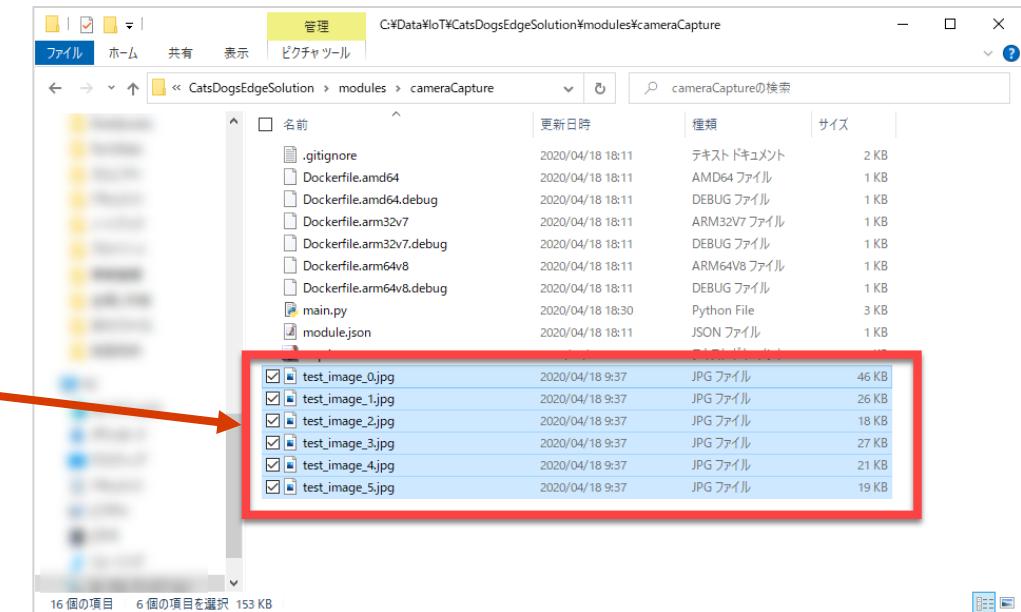
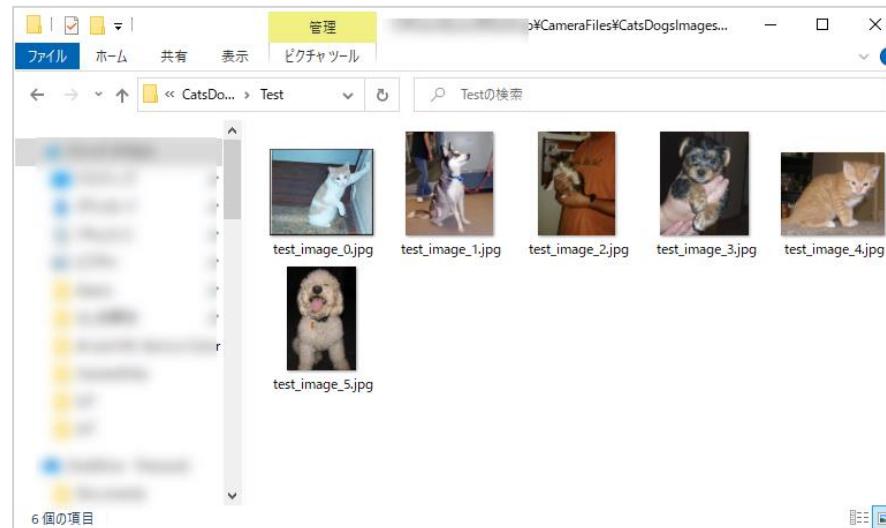
- cameraCapture の “requirements.txt”
 - ・ 2行目に “requests” を追加



エミュレート用のテスト画像を追加

■ 今回は 6枚の画像を定期的に分類するものとする

- ・学習に使用したもの以外のファイルを 6枚適当に選択
- ・ファイル名は “test_image_0.jpg” ~ “test_image_5.jpg”
- ・“modules/cameraCapture” フォルダーにコピー

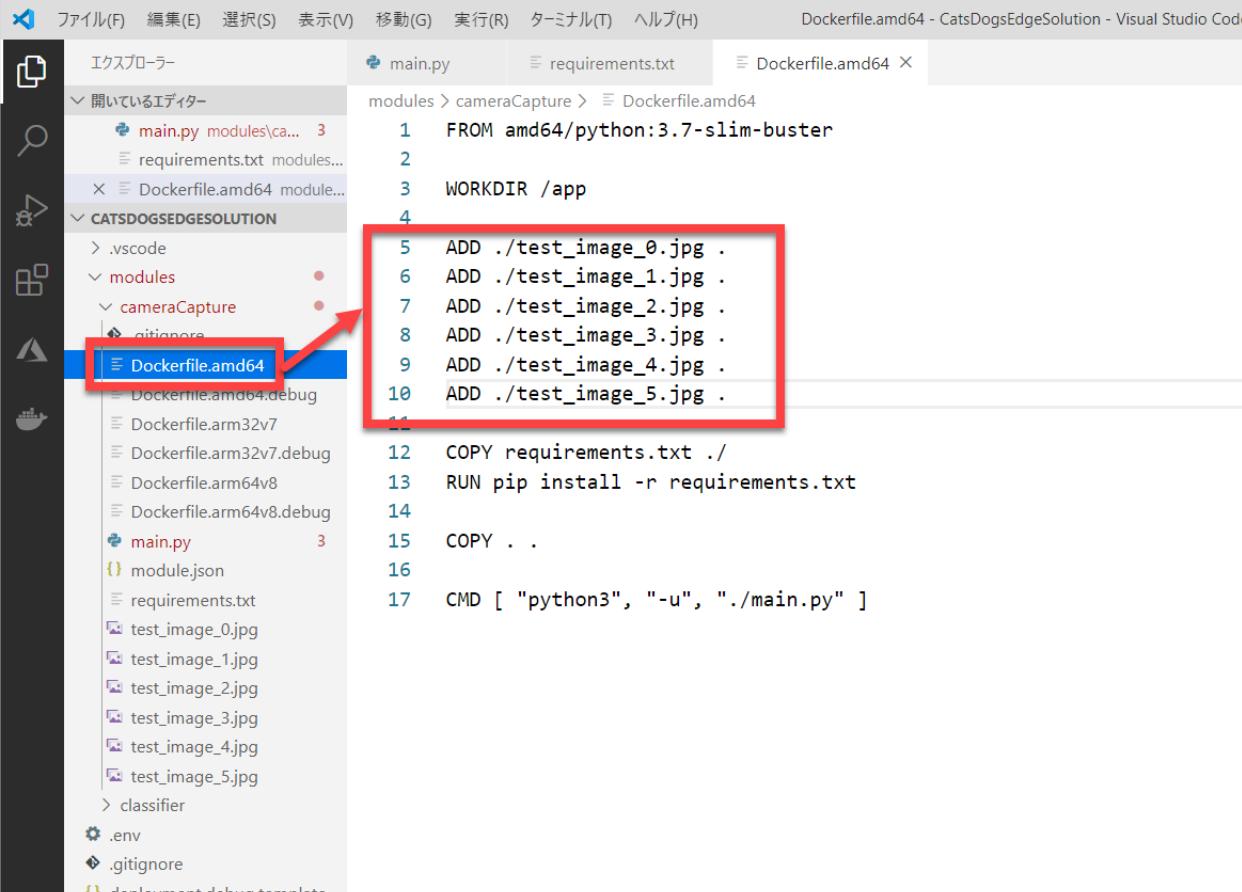


Dockerfile.amd64 を編集

■cameraCapture の “Dockerfile.amd64”

- ・テスト画像を Docker イメージに含める

```
ADD ./test_image_0.jpg .
ADD ./test_image_1.jpg .
ADD ./test_image_2.jpg .
ADD ./test_image_3.jpg .
ADD ./test_image_4.jpg .
ADD ./test_image_5.jpg .
```



```
FROM amd64/python:3.7-slim-buster
WORKDIR /app
ADD ./test_image_0.jpg .
ADD ./test_image_1.jpg .
ADD ./test_image_2.jpg .
ADD ./test_image_3.jpg .
ADD ./test_image_4.jpg .
ADD ./test_image_5.jpg .
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY .
CMD [ "python3", "-u", "./main.py" ]
```

マニフェスト ファイルを編集 (1/3)

■ deployment.template.json

- “SimulatedTemperatureSensor” は不要なので
ブロックごと削除

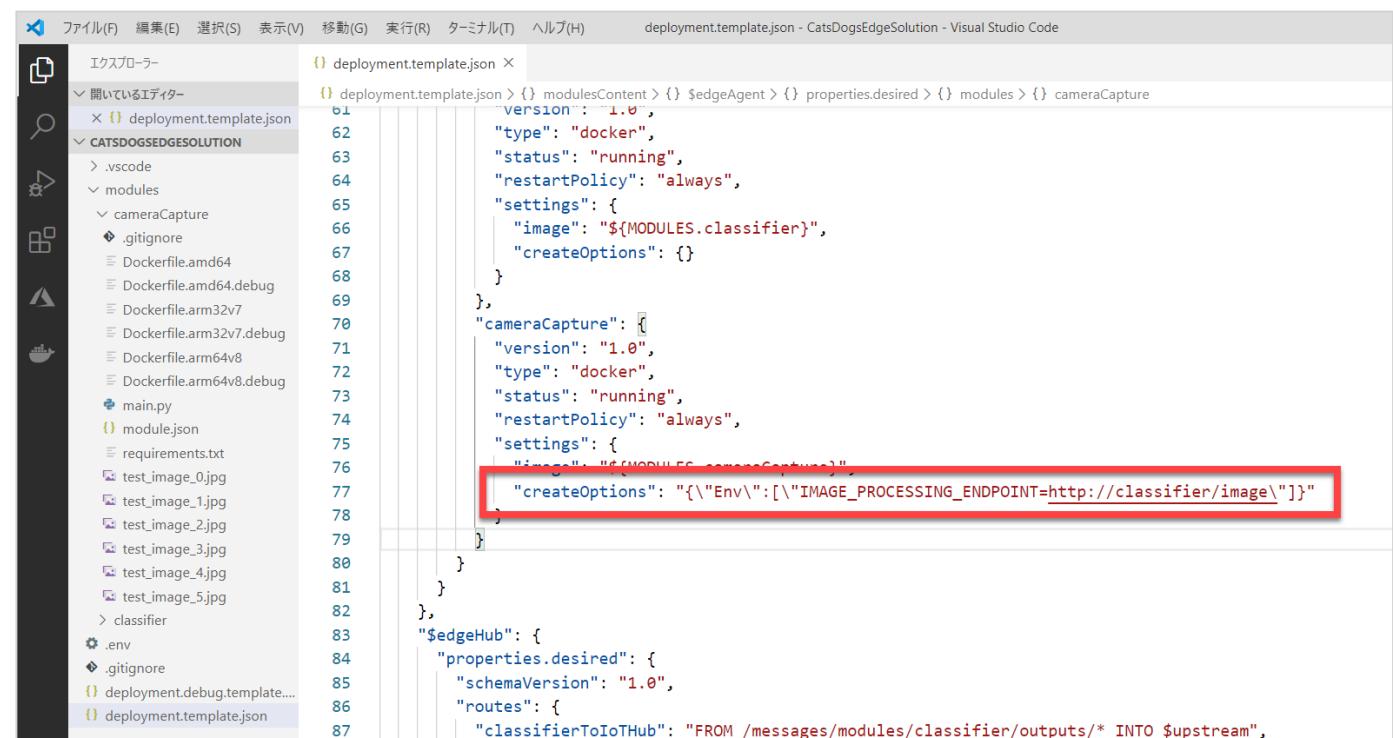
```
deployment.template.json - CatsDogsEdgeSolution - Visual Studio Code
```

```
58 },
59 },
60 "modules": {
61   "classifier": {
62     "version": "1.0",
63     "type": "docker",
64     "status": "running",
65     "restartPolicy": "always",
66     "settings": {
67       "image": "${MODULES.classifier}",
68       "createOptions": {}
69     }
70   },
71   "SimulatedTemperatureSensor": [
72     {
73       "version": "1.0",
74       "type": "docker",
75       "status": "running",
76       "restartPolicy": "always",
77       "settings": {
78         "image": "mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0",
79         "createOptions": {}
80       }
81     }
82   ],
83   "cameraCapture": [
84     {
85       "version": "1.0",
86       "type": "docker",
87       "status": "running",
88       "restartPolicy": "always",
89       "settings": {
90         "image": "${MODULES.cameraCapture}",
91         "createOptions": {}
92       }
93     }
94   ]
95 }
```

マニフェスト ファイルを編集 (2/3)

■cameraCapture/settings/createOptions を編集

```
"createOptions":  
  "{\"Env\":[\"IMAGE_PROCESSING_ENDPOINT=http://classifier/image\"]}"
```



```
deployment.template.json - CatsDogsEdgeSolution - Visual Studio Code  
deployment.template.json  
version: "1.0",  
type: "docker",  
status: "running",  
restartPolicy: "always",  
settings: {  
  "image": "${MODULES.classifier}",  
  "createOptions": {}  
},  
cameraCapture:  
  version: "1.0",  
  type: "docker",  
  status: "running",  
  restartPolicy: "always",  
  settings: {  
    "image": "${MODULES.imageProcessor}",  
    "createOptions": "{\"Env\":[\"IMAGE_PROCESSING_ENDPOINT=http://classifier/image\"]}"  
  },  
$edgeHub:  
  properties.desired:  
    schemaVersion: "1.0",  
    routes:  
      classifierToIoTHub: "FROM /messages/modules/classifier/outputs/* INTO $upstream",
```

マニフェスト ファイルを編集 (3/3)

■メッセージルーティングを編集

- “\$edgeHub” の “routes” を編集

```
"routes": {  
    "CameraCaptureToIoTHub": "FROM  
/messages/modules/cameraCapture/outputs/* INTO $upstream"  
},
```



```
エクスプローラー ファイル(F) 編集(E) 選択(S) 表示(V) 移動(G) 実行(R) ターミナル(T) ヘルプ(H) deployment.template.json - CatsDogsEdgeSolution - Visual Studio Code

エクスプローラー
開いているエディター
deployment.template.json
CATSDOGSEDGESOLUTION
> .vscode
modules
cameraCapture
.DS_Store
.Dockerfile.amd64
.Dockerfile.amd64.debug
.Dockerfile.arm32v7
.Dockerfile.arm32v7.debug
.Dockerfile.arm64v4
.Dockerfile.arm64v4.debug
main.py
module.json
requirements.txt
test_image_0.jpg
test_image_1.jpg
test_image_2.jpg
test_image_3.jpg
test_image_4.jpg
test_image_5.jpg
> classifier
.env
.DS_Store
deployment.debug.template...
deployment.template.json

deployment.template.json ×
deployment.template.json > { modulesContent > { $edgeHub > {} properties.desired
  "settings": {
    "image": "${MODULES.cameraCapture}",
    "createOptions": "{\"Env\":{\"IMAGE_PROCESSING_ENDPOINT=http://classifier/image\"}}"
  }
}
$edgeHub: {
  "properties.desired": {
    "schemaVersion": "1.0",
    "routes": {
      "CameraCaptureToIoTHub": "FROM /messages/modules/cameraCapture/outputs/* INTO $upstream"
    },
    "storeAndForwardConfiguration": {
      "timeToLiveSecs": 7200
    }
  }
}
```

Docker ログイン

■ターミナルを開いて Docker ログイン

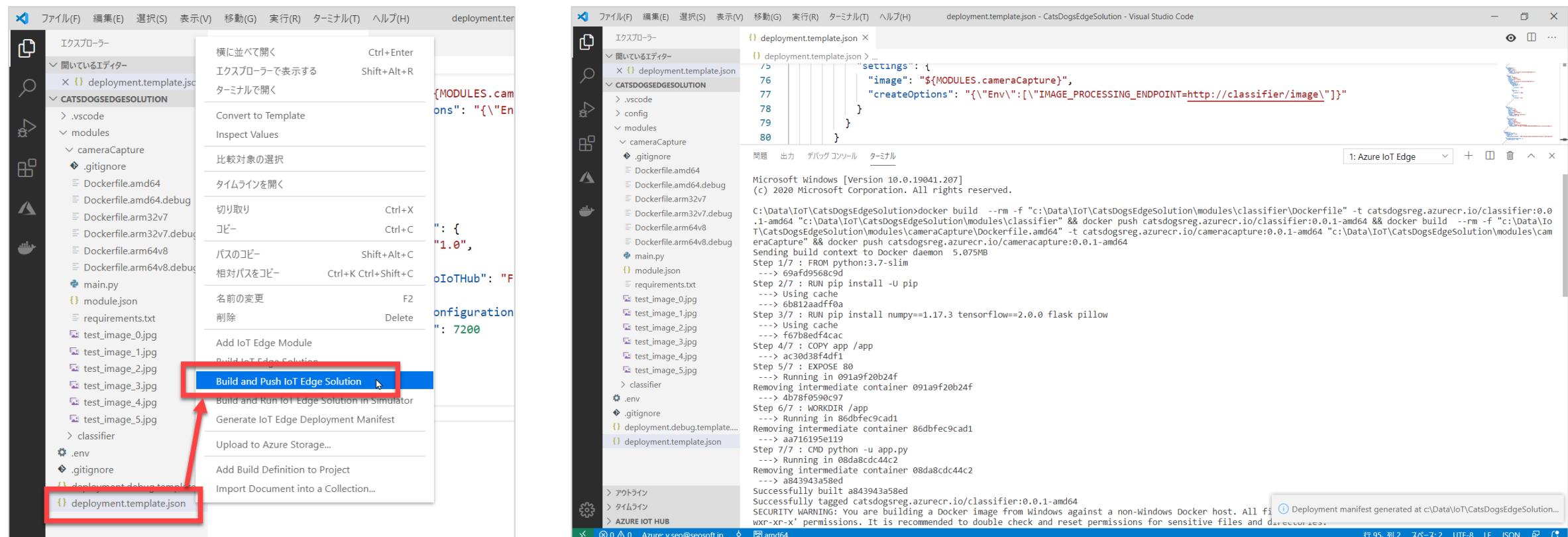
```
docker login -u <ACR ユーザー名> -p <ACR パスワード> <ACR ログインサーバー>
```

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with project files like 'deployment.template.json', '.vscode', 'modules', 'cameraCapture', 'main.py', 'module.json', 'requirements.txt', and several image files. The main editor area displays a JSON configuration file with code highlighting. The bottom right contains a terminal window showing the command 'docker login' being run and its output, which includes a warning about password security and a success message.

```
C:\Data\IoT\CatsDogsEdgeSolution>docker login -u catsdogsreg -p [REDACTED] catsdogsreg.azurecr.io
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
C:\Data\IoT\CatsDogsEdgeSolution>
```

ソリューションをビルドして ACR に発行

■ deployment.template.json の “Build and Push IoT Edge Solution”



ACR に発行されていることを確認 (1/2)

■ Azure ポータルで確認

The screenshot shows the Microsoft Azure portal interface. The top navigation bar is blue with the text "Microsoft Azure" and a search bar "リソース、サービス、ドキュメントの検索 (G+ /)". Below the navigation bar, the breadcrumb trail shows "ダッシュボード > catsdogsreg | リポジトリ > classifier".

The left sidebar has several icons: + (New), Home, Groups, Services, Star (Favorites), Key (Encryption), Task (Tasks), and Container Registry (highlighted with a red box). The main content area shows the "catsdogsreg | リポジトリ" container registry. A red arrow points from the "リポジトリ" icon in the sidebar to the "リポジトリ" section in the center.

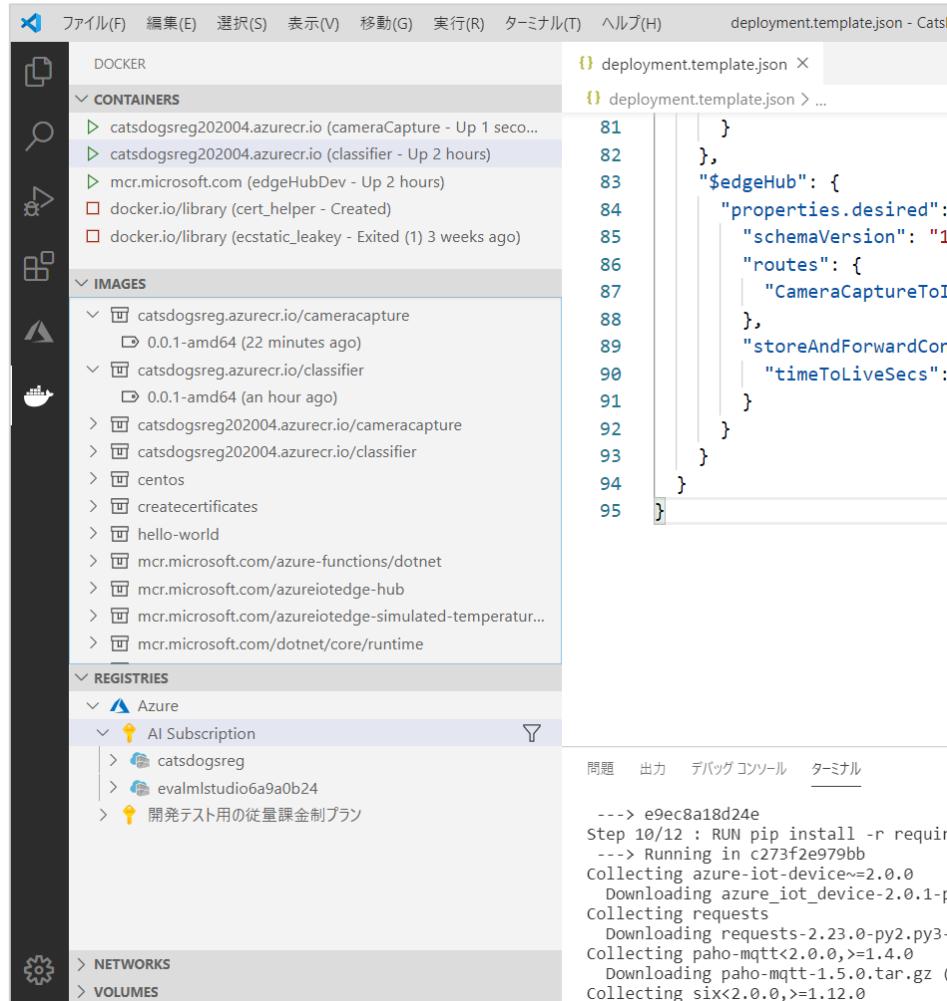
In the center, there is a search bar "検索してリポジトリをフィルター処理..." and a list of repositories. One repository, "classifier", is highlighted with a red box and has a red arrow pointing to its details. The details pane shows:

- リポジトリ名: classifier
- 最終更新日: 2020/4/18 20:27 JST
- タグ数: 1
- マニフェスト数: 1

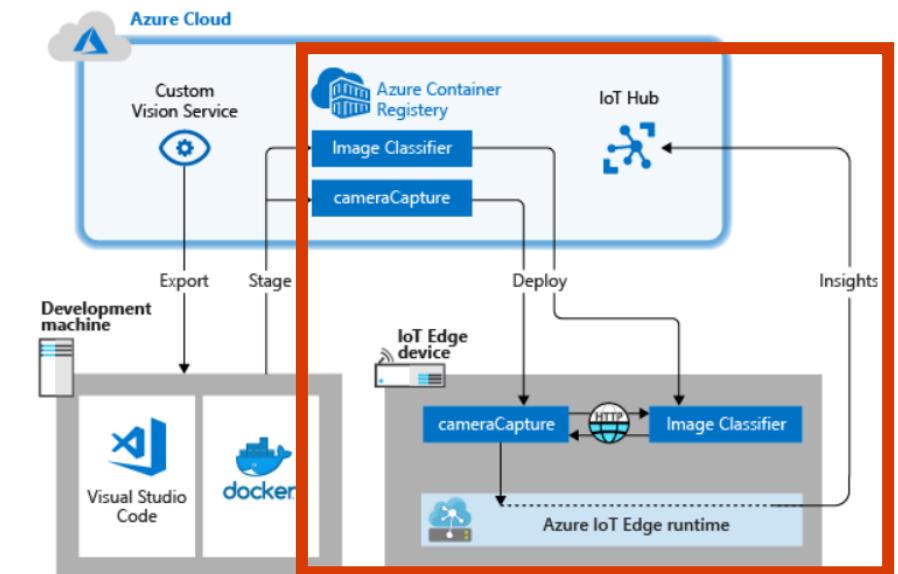
At the bottom of the details pane, there is another search bar "検索してタグをフィルター処理..." and a list of tags: "0.0.1-amd64".

ACR に発行されていることを確認 (2/2)

■VSCode + Docker 拡張機能でも確認可能



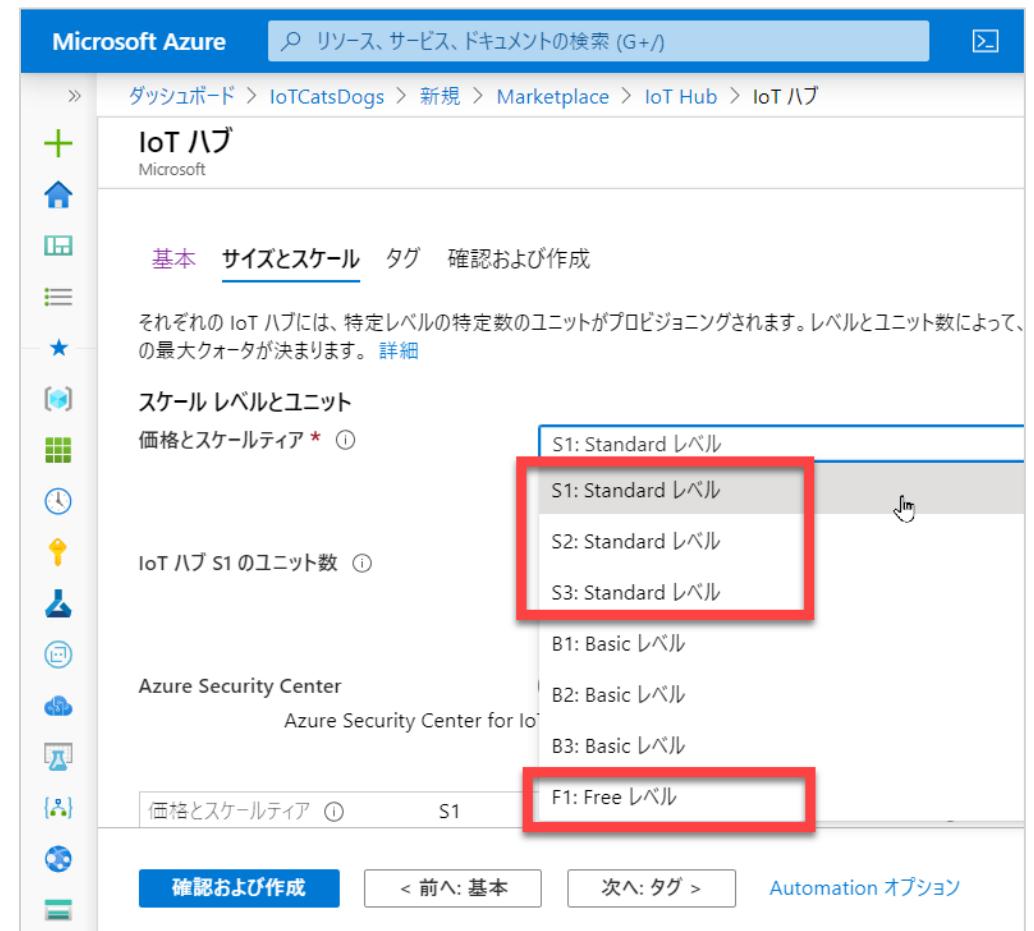
IoT Hub 操作



IoT Hub 作成

■ Azure ポータルで IoT Hub を作成

- ・スケールティアは “S” または “F”
- ・“B” では IoT Edge は使用不可



共有アクセス ポリシーをメモ

■ IoT Hub への展開やメッセージのモニタリングなど
で必要

The screenshot shows the Microsoft Azure IoT Hub management interface. On the left, there's a sidebar with various settings like Overview, IAM, and Events. The '共有アクセス ポリシー' (Shared Access Policy) option is highlighted with a red box and has a red arrow pointing to it. The main area displays a list of policies:

名前	権限
iothubowner	レジストリ読み取り レジストリ書き込み サービス接続 デバイス接続
service	
device	
registryRead	
registryReadWrite	

To the right, a detailed view of the 'iothubowner' policy is shown. It includes fields for 'アクセスポリシー名' (Access Policy Name), '権限' (Permissions), and two sets of '共有アクセスキー' (Shared Access Keys). The '権限' section shows checkboxes for Registry Read, Registry Write, Service Connection, and Device Connection, all of which are checked.

IoT Edge 追加 (1/2)

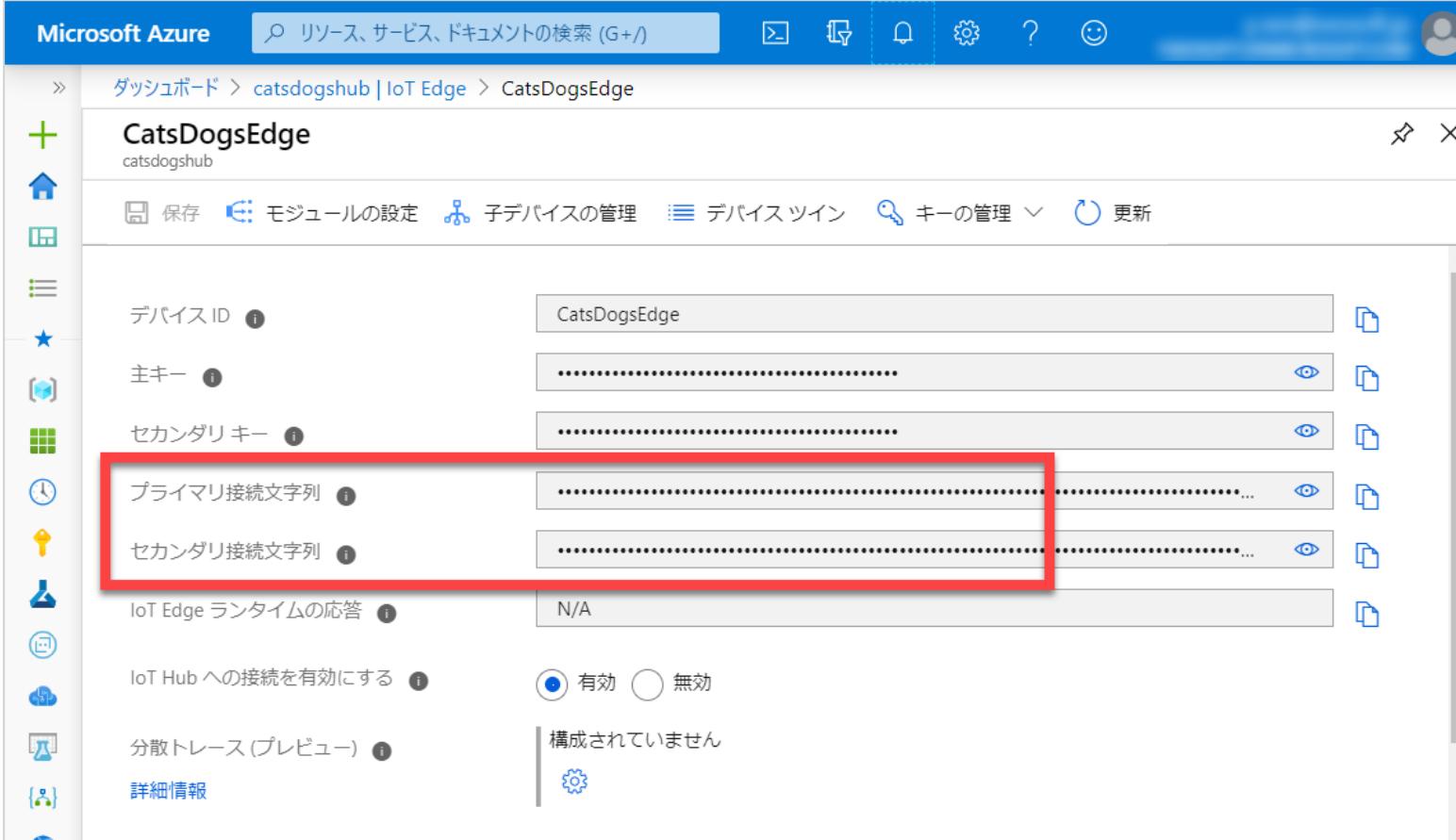
■Edge デバイスを IoT Hub に追加

The image shows two screenshots of the Microsoft Azure IoT Edge portal. The left screenshot displays the 'IoT Edge' section of the 'catsdogshub | IoT Edge' dashboard. A red box highlights the '+ IoT Edge デバイスを追加する' (Add IoT Edge device) button. The right screenshot shows the 'デバイスの作成' (Device creation) blade. A red arrow points from the 'Add IoT Edge device' button in the left screenshot to the 'デバイス ID' input field in the right screenshot. The 'デバイス ID' field contains 'CatsDogsEdge'. Other fields visible include '認証の種類' (Authentication type) set to '対称キー X.509 自己署名済み' (Symmetric key X.509 self-signed), and 'このデバイスを IoT ハブに接続する' (Connect this device to an IoT hub) switch set to '有効化' (Enabled). The bottom right corner of the right screenshot has a '保存' (Save) button.

IoT Edge 追加 (2/2)

■ IoT Edge の接続文字列をメモ

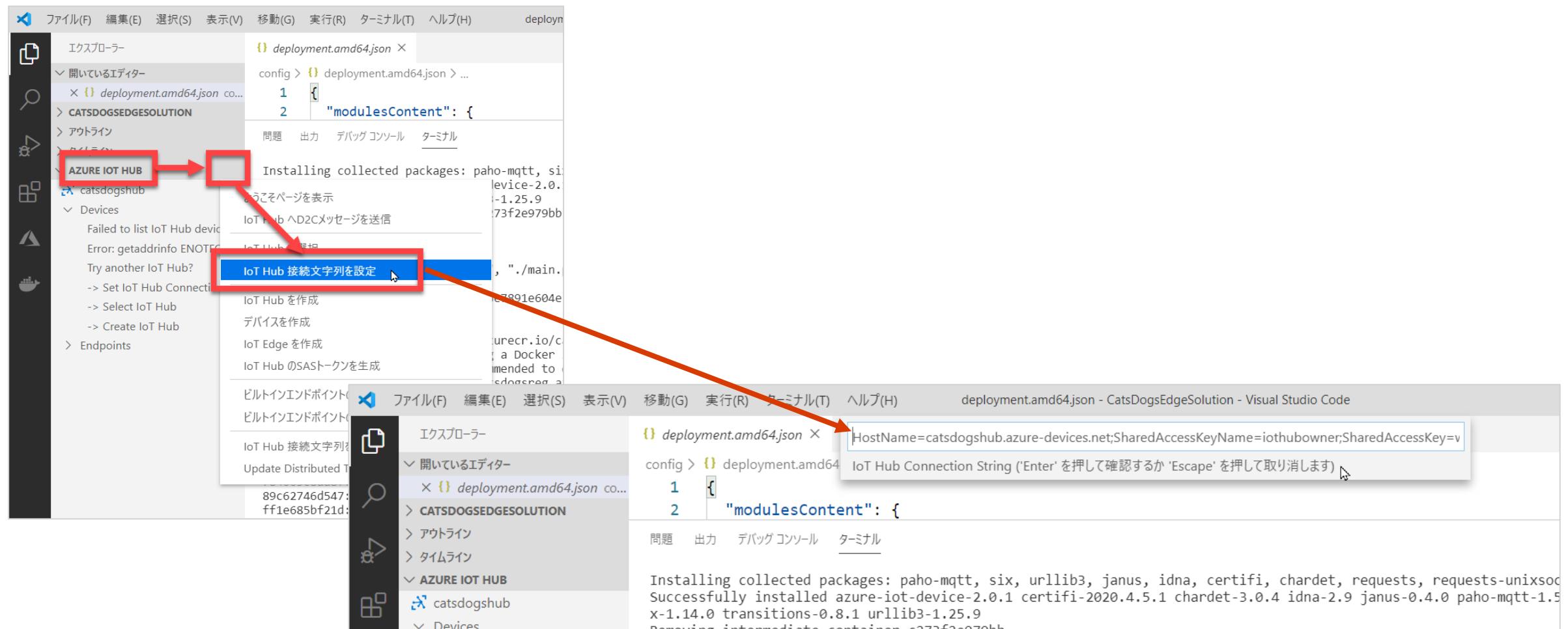
- ・デバイスでの実行のために必要



The screenshot shows the Microsoft Azure portal interface for managing an IoT Edge device. The top navigation bar includes the Microsoft Azure logo, a search bar, and various icons for account management and help. The main title is "ダッシュボード > catsdogshub | IoT Edge > CatsDogsEdge". On the left, there is a sidebar with icons for creating new resources, home, recent, favorites, and more. The main content area displays the device configuration for "CatsDogsEdge". It includes fields for "デバイス ID" (Device ID) set to "CatsDogsEdge", "主キー" (Primary key), "セカンダリ キー" (Secondary key), "プライマリ接続文字列" (Primary connection string), and "セカンダリ接続文字列" (Secondary connection string). The "Primary connection string" field is highlighted with a red rectangle. Below these fields are sections for "IoT Edge ランタイムの応答" (IoT Edge Runtime Response), "IoT Hub への接続を有効にする" (Enable connection to IoT Hub), and "分散トレース (プレビュー)" (Distributed Tracing (Preview)). A radio button for "有効" (Enabled) is selected under the IoT Hub connection section. At the bottom, there is a "構成されていません" (Not configured) message next to a gear icon.

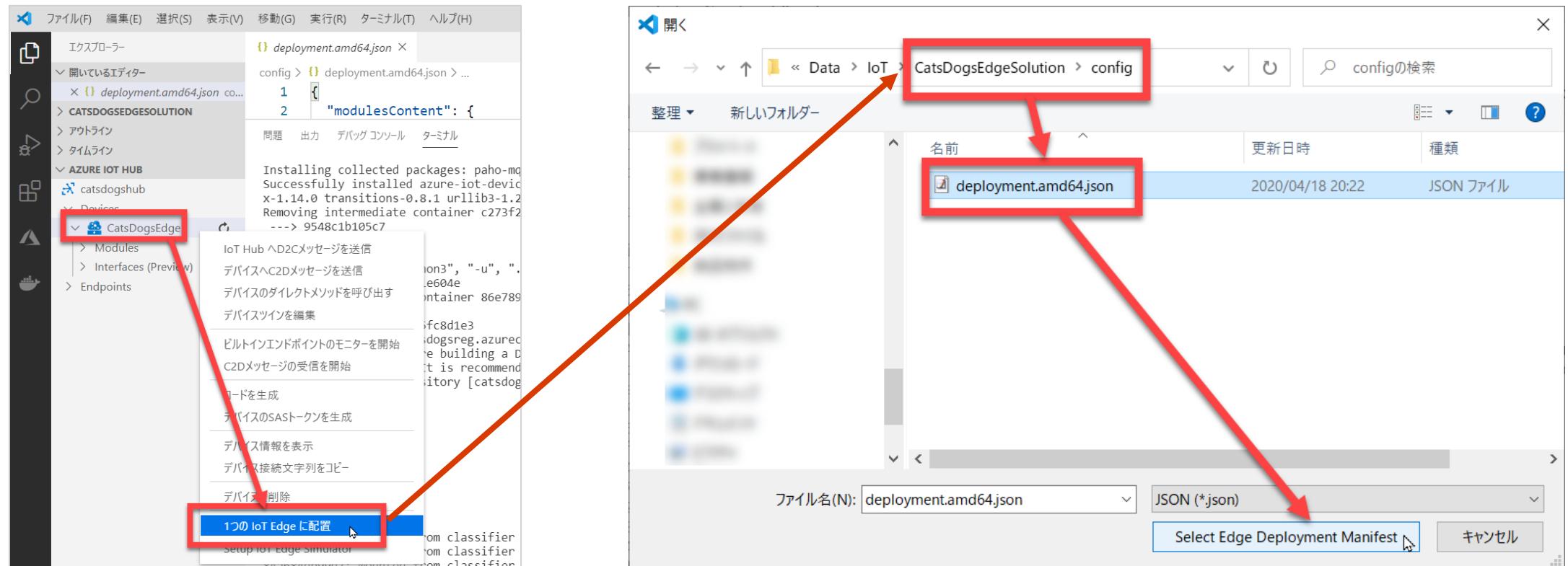
ソリューションを IoT Edge にデプロイ (1/3)

■VSCode で IoT Hub の接続文字列を入力



ソリューションを IoT Edge にデプロイ (2/3)

- IoT Hub, IoT Edge デバイスが表示されるので Edge デバイスの “1つの IoT Edge に配置”
 - config フォルダーの “deployment.amd64.json” を指定



ソリューションを IoT Edge にデプロイ (3/3)

■ Azure ポータルでデプロイを確認

- モジュール数が増えている

The screenshot shows the Microsoft Azure IoT Edge device blade for the 'CatsDogsEdge' device. On the left, the navigation menu is expanded, with the 'IoT Edge' item highlighted by a red box. In the center, the 'IoT Edge デバイス' section displays a table with the following data:

デバイス ID	ランタイムの応答	IoT Edge モジュールの数	接続済みクライアントの数
CatsDogsEdge	該当なし	4	0

A red arrow points from the 'IoT Edge' menu item in the navigation bar to the 'IoT Edge モジュールの数' value in the table.

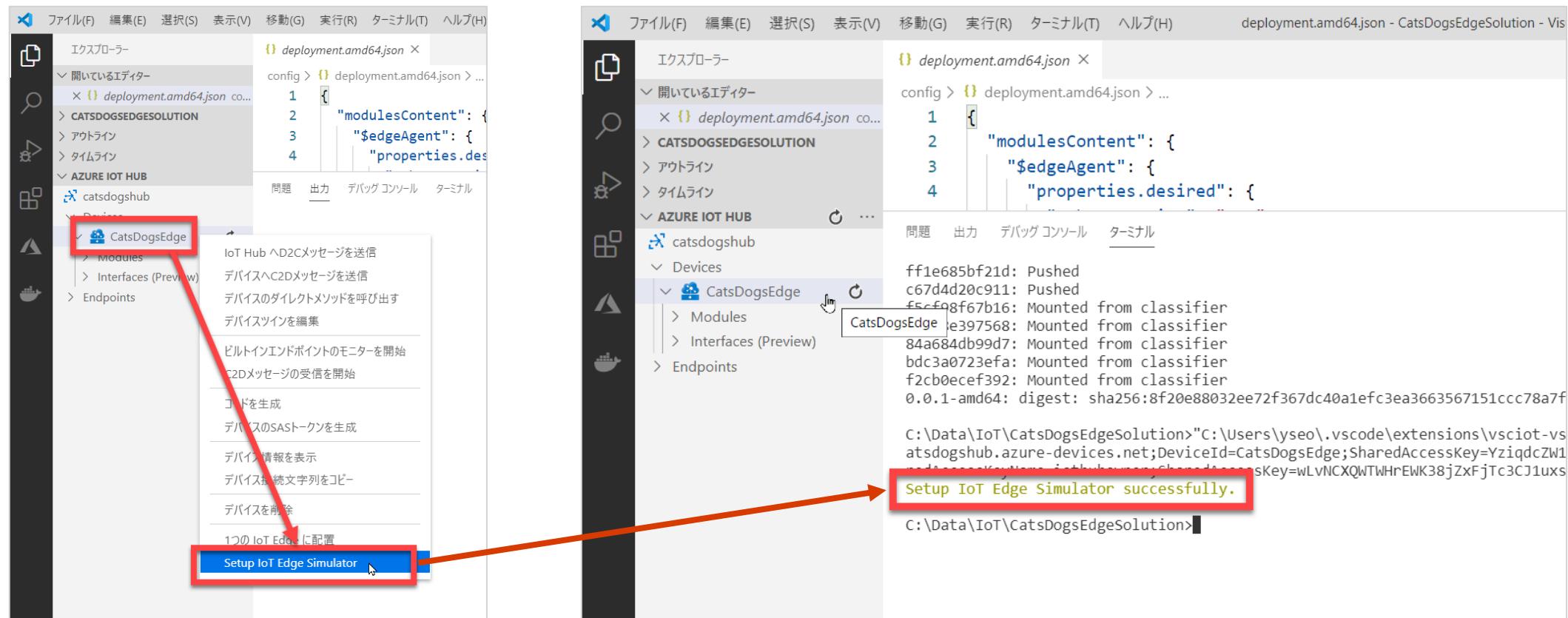
The screenshot shows the Microsoft Azure IoT Edge module settings blade for the 'CatsDogsEdge' device. The right-hand pane displays a table of deployed modules:

名前	種類	デプロイで指定
\$edgeAgent	IoT Edge システム モジュール	✓ はい
\$edgeHub	IoT Edge システム モジュール	✓ はい
classifier	IoT Edge のカスタム モジュール	✓ はい
cameraCapture	IoT Edge のカスタム モジュール	✓ はい

A red box highlights the entire table, and a red arrow points from the 'IoT Edge モジュール' tab in the top navigation to the table.

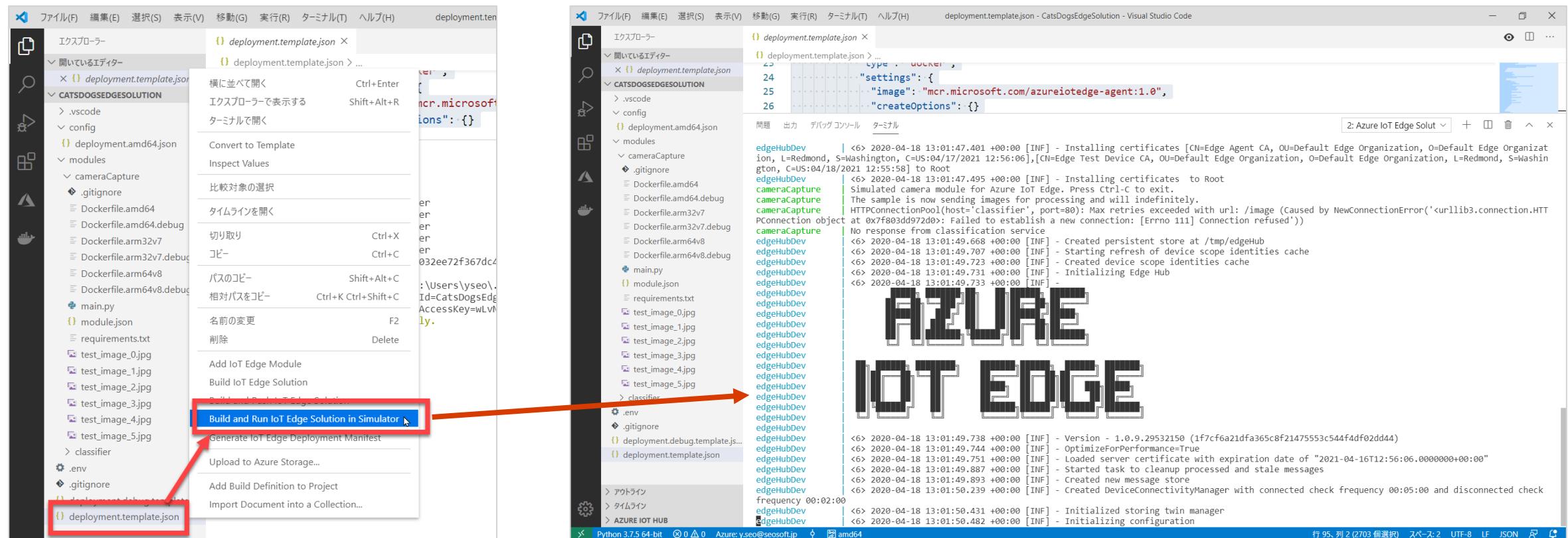
□—カル エミュレーターで実行 (1/4)

■Edge デバイスで“Setup IoT Edge Emulator”



□ カルエミュレーターで実行 (2/4)

■ deployment.template.json で “Build and Run IoT Edge Solution in Simulator”



□ - カルエミュレーターで実行 (3/4)

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "CATSDOGSEGESOLUTION".
- Editor:** Displays the file "deployment.template.json" with the following content:

```
23     "type": "Docker",  
24     "settings": {  
25       "image": "mcr.microsoft.com/azureiotedge-agent:1.0",  
26       "createOptions": {}  
}
```

- Terminal:** Shows logs from the "edgeHubDev" device. The logs indicate the device connected to the cloud, processed subscriptions, and performed image classification tasks for images named "test_image_0.jpg" through "test_image_5.jpg". The logs also show responses from the "classifier" and "cameraCapture" modules.
- Bottom Status Bar:** Shows the Python version (3.7.5), the file path (deployment.template.json), and the terminal tab (amd64).
- Page Number:** 58

□—カル エミュレーターで実行 (4/4)

■以下の例だと
“test_image_2.jpg” は「100% 猫」

```
eageHDDdev | <6> 2020-04-18 13:02:04.166 +00:00 [INF] - created cloud connection for
classifier | 2020-04-18 13:02:11.832740: Predicting image
classifier | 2020-04-18 13:02:11.832852: Image size: 375x499
classifier | 2020-04-18 13:02:11.832885: Convert to numpy array
classifier | 2020-04-18 13:02:11.839444: crop_center: 375x499 and resize to 256x256
classifier | 2020-04-18 13:02:14.693762: crop_center: 256x256 to 224x224
classifier | 2020-04-18 13:02:15.140712: Results: {'id': '', 'project': '', 'iteration': 0, 'tagName': 'Cat', 'probability': 1.0, 'tagId': '', 'boundingBox': None}]}
classifier | 172.18.0.4 - - [18/Apr/2020 13:02:15] "POST /image HTTP/1.1" 200 -
cameraCapture | Response from classification service: (200) {"created": "2020-04-18T13:02:15Z", "id": null, "imagePath": "test_image_2.jpg", "label": "Cat", "name": "Cat", "project": "", "probability": 1.0, "tagId": "", "tagName": "Cat"}, "project": ""}
cameraCapture | imagePath: test_image_2.jpg
cameraCapture | Total images sent: 2
cameraCapture |
```

Edge デバイスの仮想マシンを用意

■ “Ubuntu Server 16.04 LTS + Azure IoT Edge runtime” イメージで仮想マシンを作成

The screenshot shows the Microsoft Azure portal interface for creating a new virtual machine. The 'Marketplace' tab is selected in the sidebar. In the main area, the search bar contains 'iot edge'. A red box highlights the 'Ubuntu Server 16.04 LTS + Azure IoT Edge runtime' item from the list, which is described as 'Ubuntu Server 16.04 LTS based virtual machine with Azure IoT Edge runtime preinstalled'. Below the search bar, there are categories like Analytics, Blockchain, Compute, Containers, Databases, Developer Tools, DevOps, Identity, Integration, Internet of Things, IT & Management Tools, Media, Mixed Reality, Networking, Security, Software as a Service (SaaS), Storage, and Web. At the bottom of the page, there are navigation buttons for '確認および作成' (Review and Create) and '次: ディスク >' (Next: Disk).

The screenshot shows the Microsoft Azure Marketplace search results for 'iot edge'. The search bar at the top also contains 'iot edge'. A red arrow points from the highlighted item in the previous screenshot to this one. The 'Ubuntu Server 16.04 LTS + Azure IoT Edge runtime' item is highlighted with a red box. It is listed under the 'AI + Machine Learning' category. The item details are: 'Ubuntu Server 16.04 LTS based virtual machine with Azure IoT Edge runtime preinstalled'. Below this, other items are listed, such as 'KeyScaler - IoT Security for Microsoft Azure IoT', 'nio - Distributed IoT Platform', 'FogHorn Edge Device Manager', 'Smart oneM2M network', 'DiagPortal', and 'Jenkins on Windows Server 2019'.

Edge デバイスで実行 (1/4)

■仮想マシンに ssh 接続してコマンドを実行

- 初回に一度だけ実行すればよい

```
sudo /etc/iotedge/configedge.sh '<IoT Edge の接続文字列>'  
sudo systemctl restart iotedge  
sudo systemctl status iotedge  
sudo iotedge list
```

Edge デバイスで実行 (2/4)

```
iotuser@CatsDogsEdgeLinux: ~
C:\$Data\$IoTssh iotuser@
The authenticity of host '..... (.....)' can't be established.
ECDSA key fingerprint is SHA256:tpyIrp+E+H4735EEMFctUZeifdr90osgbWV0WeexY7c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '.....' (ECDSA) to the list of known hosts.
iotuser@.....'s password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.0.0-1035-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Apr 18 13:22:21 UTC 2020

System load: 0.04      Processes:          115
Usage of /: 4.0% of 28.90GB  Users logged in: 0
Memory usage: 4%           IP address for eth0: 10.0.3.4
Swap usage: 0%

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
```

See "man sudo_root" for details.

```
iotuser@CatsDogsEdgeLinux:~$ sudo ./etc/iotedge/configedge.sh 'HostName=catsdogshub.azure-devices.net;DeviceId=CatsDogsEdge;SharedAccessKey=.....'
Sat Apr 18 13:38:43 UTC 2020 Connection string set to HostName=catsdogshub.azure-devices.net;DeviceId=CatsDogsEdge;SharedAccessKey=.....
iotuser@CatsDogsEdgeLinux:~$ sudo systemctl restart iotedge
iotuser@CatsDogsEdgeLinux:~$
```

Edge デバイスで実行 (3/4)

```
iotuser@CatsDogsEdgeLinux: ~
See "man sudo_root" for details.

iotuser@CatsDogsEdgeLinux: ~$ sudo /etc/iotedge/configedge.sh 'HostName=catsdogshub.azure-devices.net;DeviceId=CatsDogsEdge;SharedAccessKey='
Sat Apr 18 13:38:43 UTC 2020 Connection string set to HostName=catsdogshub.azure-devices.net;DeviceId=CatsDogsEdge;SharedAccessKey=
iotuser@CatsDogsEdgeLinux: ~$ sudo systemctl restart iotedge
iotuser@CatsDogsEdgeLinux: ~$ sudo systemctl status iotedge
● iotedge.service - Azure IoT Edge daemon
  Loaded: loaded (/lib/systemd/system/iotedge.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2020-04-18 13:38:50 UTC; 15s ago
    Docs: man:iotedged(8)
   Main PID: 4079 (iotedged)
     Tasks: 8
    Memory: 5.2M
      CPU: 26ms
     CGroup: /system.slice/iotedge.service
             └─4079 /usr/bin/iotedged -c /etc/iotedge/config.yaml

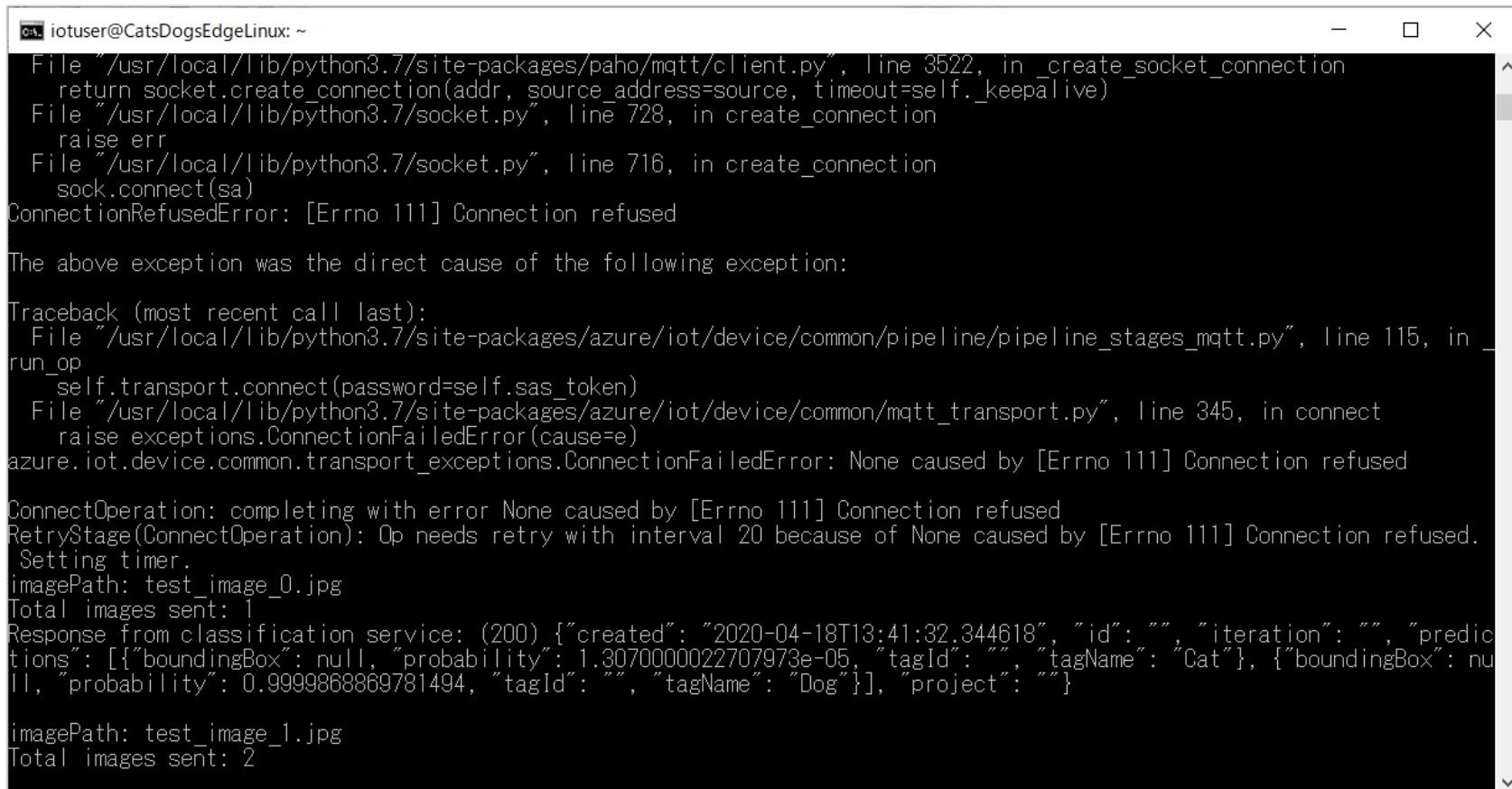
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Starting workload API...
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Starting watchdog with 60 second frequency
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Listening on fd://iotedge.mgmt.socket/ with 1 connection
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Listening on fd://iotedge.socket/ with 1 connection
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Checking edge runtime status
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Creating and starting edge runtime module
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Updating identity for module $edgeAgent
Apr 18 13:38:50 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:38:50Z [INFO] - Pulling image mcr.microsoft.com/azureiotedge-agent:1.0
Apr 18 13:39:04 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:39:04Z [INFO] - Successfully pulled image mcr.microsoft.com/azureiotedge-agent:1.0
Apr 18 13:39:04 CatsDogsEdgeLinux iotedged[4079]: 2020-04-18T13:39:04Z [INFO] - Creating module edgeAgent...
Lines 1-21/21 (END)
```

```
iotuser@CatsDogsEdgeLinux: ~
iotuser@CatsDogsEdgeLinux: ~$ sudo iotedge list
NAME          STATUS        DESCRIPTION          CONFIG
cameraCapture  running       Up 7 seconds      catsdogsreg.azurecr.io/cameracapture:0.0.1-amd64
edgeAgent      running       Up 2 minutes      mcr.microsoft.com/azureiotedge-agent:1.0
classifier     running       Up 23 seconds     catsdogsreg.azurecr.io/classifier:0.0.1-amd64
edgeHub        running       Up 13 seconds     mcr.microsoft.com/azureiotedge-hub:1.0
iotuser@CatsDogsEdgeLinux: ~$
```

Edge デバイスで実行 (4/4)

■cameraCapture モジュールのログを確認

```
sudo iotedge logs cameraCapture -f
```



```
iotuser@CatsDogsEdgeLinux: ~
File "/usr/local/lib/python3.7/site-packages/paho/mqtt/client.py", line 3522, in _create_socket_connection
    return socket.create_connection(addr, source_address=source, timeout=self._keepalive)
File "/usr/local/lib/python3.7/socket.py", line 728, in create_connection
    raise err
File "/usr/local/lib/python3.7/socket.py", line 716, in create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

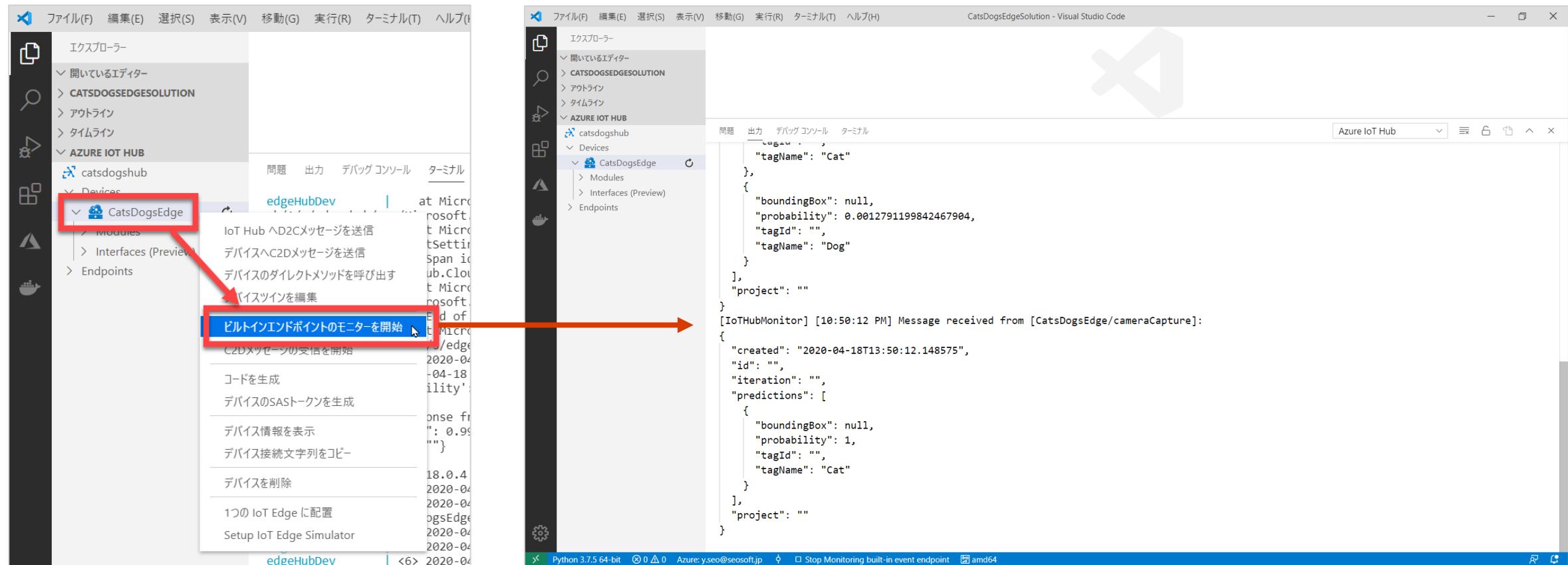
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/site-packages/azure/iot/device/common/pipeline/pipeline_stages_mqtt.py", line 115, in _run_op
    self.transport.connect(password=self.sas_token)
  File "/usr/local/lib/python3.7/site-packages/azure/iot/device/common/mqtt_transport.py", line 345, in connect
    raise exceptions.ConnectionFailedError(cause=e)
azure.iot.device.common.transport_exceptions.ConnectionFailedError: None caused by [Errno 111] Connection refused

ConnectOperation: completing with error None caused by [Errno 111] Connection refused
RetryStage(ConnectOperation): Op needs retry with interval 20 because of None caused by [Errno 111] Connection refused.
Setting timer.
imagePath: test_image_0.jpg
Total images sent: 1
Response from classification service: (200) {"created": "2020-04-18T13:41:32,344618", "id": "", "iteration": "", "predictions": [{"boundingBox": null, "probability": 1.3070000022707973e-05, "tagId": "", "tagName": "Cat"}, {"boundingBox": null, "probability": 0.9999868869781494, "tagId": "", "tagName": "Dog"}], "project": ""}

imagePath: test_image_1.jpg
Total images sent: 2
```

VSCode でもモニタリング可能

■Edge デバイスで
“ビルトインエンドポイントのモニターを開始”を選択

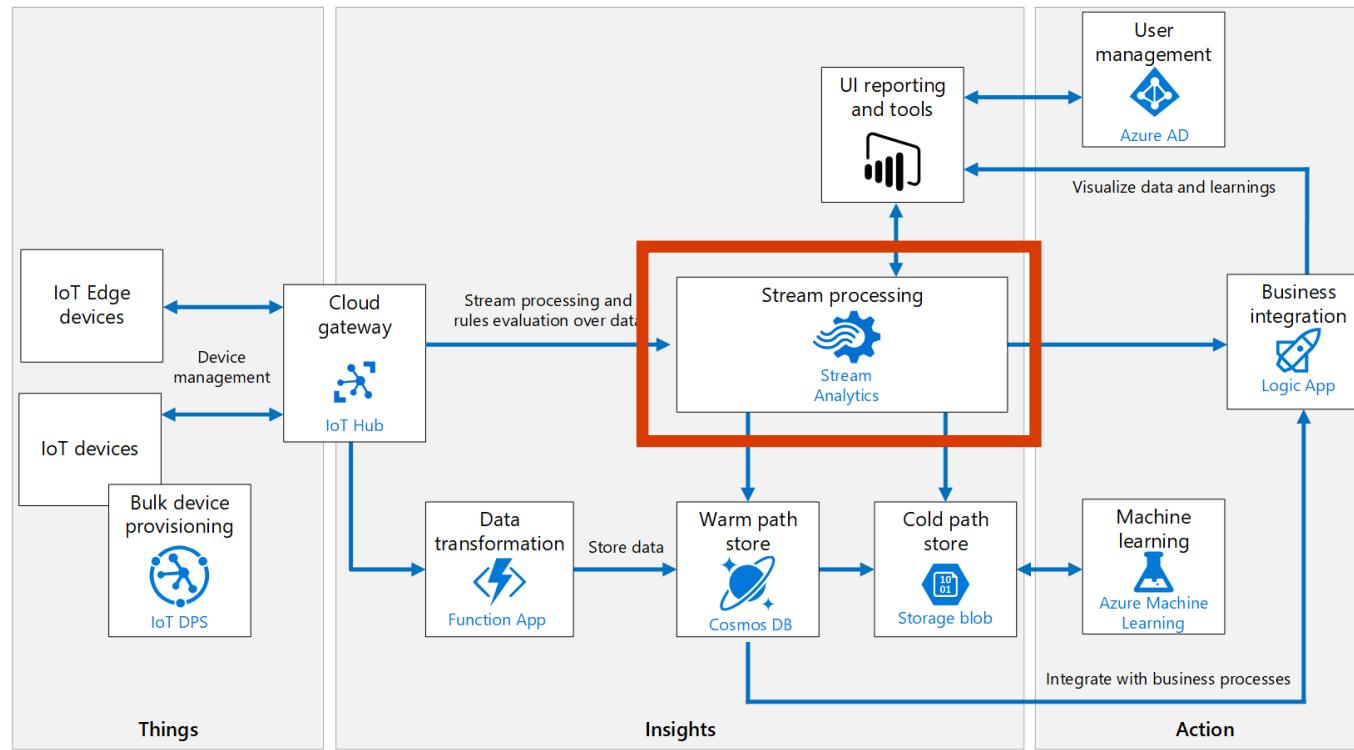


その他の参考情報

参考) Stream Analytics

■ IoT Hub からのストリーミング入力を
クラウドの各サービスに配信するサービス

- ・それぞれに適切なフォーマットに整形・集計することも可能



参考) IoT Edge をゲートウェイとして利用

■<https://docs.microsoft.com/ja-jp/azure/iot-edge/iot-edge-as-gateway>

The screenshot shows a Microsoft Azure documentation page for "IoT Edge デバイスをゲートウェイとして使用する方法". The page is in Japanese. The left sidebar contains a navigation menu for IoT Edge documents, including sections like "クイックスタート", "概念", and "パターン". The main content area features the title "IoT Edge デバイスをゲートウェイとして使用する方法" and a summary of the article's purpose. The right sidebar includes a poll question and a section for related articles.

Microsoft Azure

概要 ソリューション 製品 ドキュメント 價格 トレーニング Marketplace パートナー サポート ブログ その他 無料アカウント

Azure / モノのインターネット (IoT) / IoT Edge

タイトルでフィルター

Azure IoT Edge のドキュメント

概要

クイックスタート

Linux デバイスへのコードのデプロイ
Windows デバイスへのコードのデプロイ

チュートリアル

概念

IoT Edge ランタイム
IoT Edge のモジュール
開発
デプロイ
オフライン機能

ゲートウェイとしての IoT Edge デバイス

Security

PDF をダウンロード

IoT Edge デバイスをゲートウェイとして使用する方法

2019/02/25 • ● ●

IoT Edge ソリューションのゲートウェイは、デバイス接続とエッジ分析を IoT デバイスに提供します。ゲートウェイ以外ではこれらの機能は提供されません。Azure IoT Edge を使うと、目的が接続、ID、エッジ分析のいずれに関係していても、IoT ゲートウェイのあらゆるニーズを満たすことができます。この記事のゲートウェイパターンでは、ダウンストリーム デバイスの接続の特性とデバイスの ID のみが示されており、ゲートウェイでのデバイスデータの処理方法は示されていません。

パターン

ゲートウェイとしての IoT Edge デバイスの使用には、透過、プロトコル変換、ID 変換の 3 つのパターンがあります。

- 透過 – IoT Hub に理論的に接続できるデバイスは、代わりにゲートウェイ デバイスに接続できます。このダウンストリーム デバイスには独自の IoT Hub ID があり、MQTT、AMQP、HTTP のいずれかのプロトコルを使っています。ゲートウェイは、デバイスと IoT Hub の間の通信を単純に受け渡します。デバイスも、IoT Hub を介してそれと対話しているユーザーも、ゲートウェイが通信を仲介していることを認識しません。このように認識されないことは、ゲートウェイが "透過的" と見なされることを意味します。透過的なゲートウェイとして IoT Edge デバイスを使う方法については、「[透過的なゲートウェイの作成](#)」を

参考) 多数のデバイスにデプロイ

■<https://docs.microsoft.com/ja-jp/azure/iot-edge/module-deployment-monitoring>

The screenshot shows a Microsoft Azure documentation page for IoT Edge. The URL is <https://docs.microsoft.com/ja-jp/azure/iot-edge/module-deployment-monitoring>. The page title is "1台のデバイスまたは多数のデバイスを対象とした IoT Edge 自動デプロイについて". The left sidebar has a navigation tree under "Azure IoT Edge のドキュメント" with sections like "概要", "クイックスタート", "チュートリアル", "概念", "開発", "デプロイ", "配置マニフェスト", "自動デプロイ", "オフライン機能", and "ゲートウェイとしての IoT Edge デバイス". The main content area discusses automatic deployment methods for IoT Edge devices, mentioning single-device and multi-device deployment, and how to define modules for specific devices or a fleet. It also covers configuration manifest creation and deployment to IoT Hub services. A sidebar on the right provides feedback options ("このページはお役に立ちましたか?") and a sidebar menu for "デプロイ" topics.

最後まで聞いてくれてありがとう

- <https://github.com/seosoft>
- <https://yseosoft.wordpress.com/>
- <https://twitter.com/seosoft>
- <https://www.facebook.com/seosoft>

