KEL
KANEMATSU ELECTRONICS LTD.

2020年度新技術ワーキンググループ

Cチーム　エッジコンピューティング技術検証結果報告書

© 2021 KANEMATSU ELECTRONICS LTD.

アジェンダ

1．技術概要

1-1　エッジコンピューティングに関する5W1H

1-2　Azure IoT Edge概要

2．検証結果報告

2-1　検証概要

2-2　検証①結果報告

2-3　検証②結果報告

2-4　所感・Azure IoT Edge評価

3．参考

© 2021 KANEMATSU ELECTRONICS LTD.

1．技術概要

1-1　エッジコンピューティングに関する5W1H

■Whatーエッジコンピューティングとは何か（概要）

エッジコンピューティング＝データ処理装置をエンドユーザーの近くに分散配置し、**データの一次処理**を行うこと

例：NEC－Neoface

■Whenーエッジコンピューティングが登場したのはいつか（背景）

IoTデバイス登場による**データ量の増加**、AIやクラウドの登場による大量で**複雑な情報処理**

⇒待ち時間が長く、リアルタイムでの処理が実現できないという問題点

⇒クラウドにデータを転送する前にユーザの近くで一次処理を行う

■Whyーなぜエッジコンピューティングを使うか（メリット）

通信コストの削減、**リアルタイム処理、セキュリティの向上**

■Whereーエッジコンピューティングはどこに応用されるか（応用分野）

車・ドローン・農業機器の**自動運転**、設備の**異常検知**、**気象予測**などの

リアルタイムの処理と高い安全性を求める業界で広く応用される

■Whoーエッジコンピューティングサービスは誰が提供できるか（サービス提供者）

Microsoft、Amazon、Oracleなど

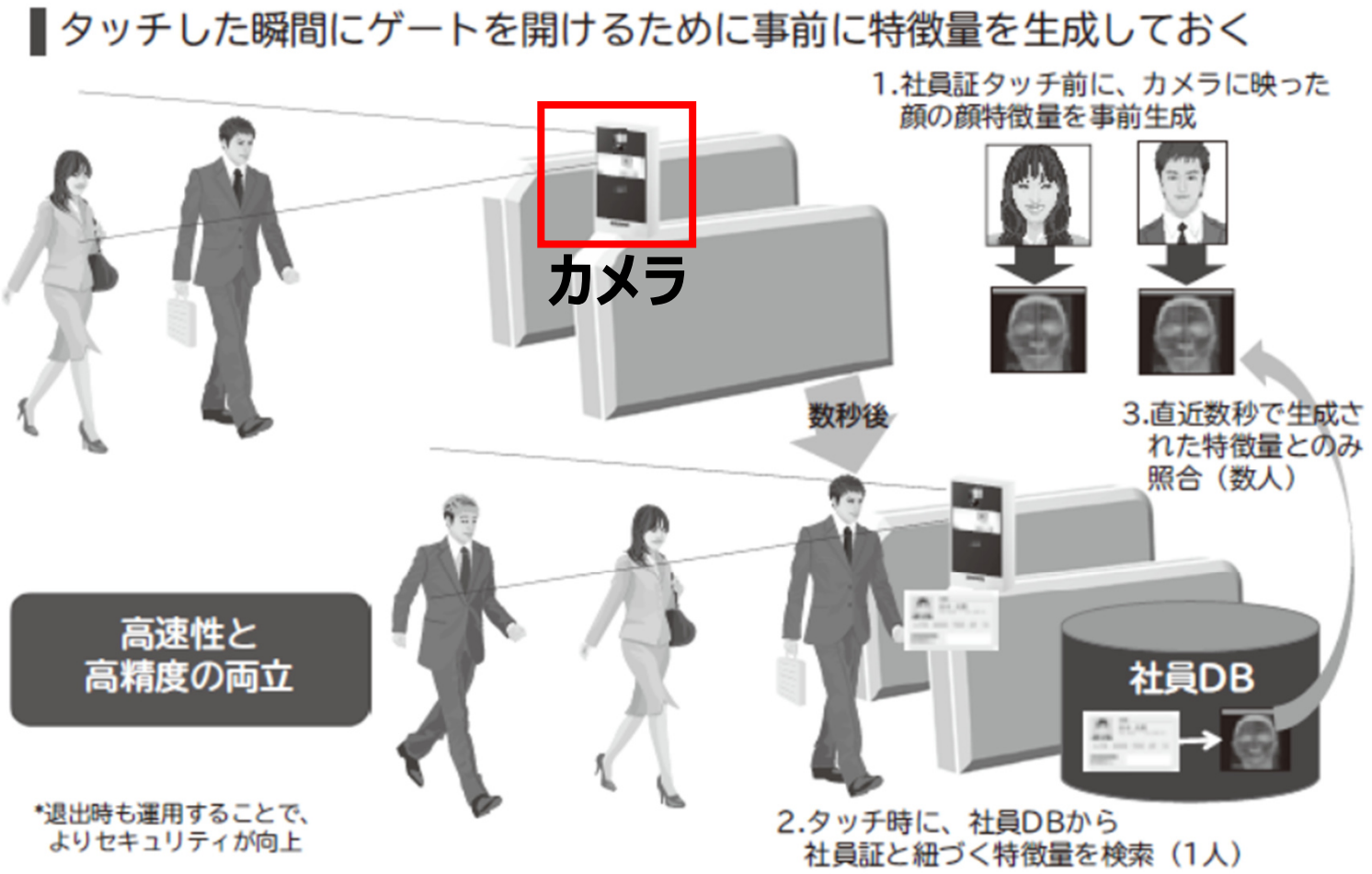
特に近年**Microsoft**の方は成長率が高まっている

© 2021 KANEMATSU ELECTRONICS LTD.

1．技術概要

NEC（Neoface）

社員はカメラの前で立ち止まることなく歩きながらの顔認証が可能となり、安全かつ迅速な入退管理を実現



© 2021 KANEMATSU ELECTRONICS LTD.

1．技術概要

1-1　エッジコンピューティングに関する5W1H

■Whatーエッジコンピューティングとは何か（概要）

エッジコンピューティング＝データ処理装置をエンドユーザーの近くに分散配置し、**データの一次処理**を行うこと

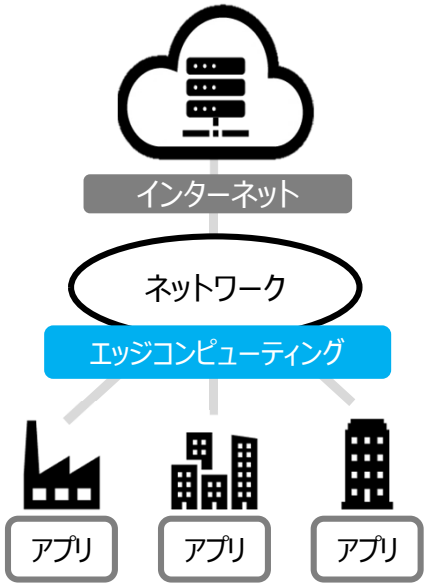
例：NEC－Neoface

■Whenーエッジコンピューティング登場したのはいつか（背景）

IoTデバイス登場による**データ量の増加**、AIやクラウドの登場による大量で**複雑な情報処理**

⇒待ち時間が長く、リアルタイムでの処理が実現できないという問題点

⇒クラウドにデータを転送する前にユーザの近くで一次処理を行う



■Whyーなぜエッジコンピューティングを使うか（メリット）

通信コストの削減、**リアルタイム処理、セキュリティの向上**

■Whereーエッジコンピューティングはどこに応用されるか（応用分野）

車・ドローン・農業機器の**自動運転**、設備の**異常検知**、**気象予測**などの

リアルタイムの処理と高い安全性を求める業界で広く応用される

■Whoーエッジコンピューティングサービスは誰が提供できるか（サービス提供者）

Microsoft、Amazon、Oracleなど

特に、近年、**Microsoft**の方は成長率が高まっている

© 2021 KANEMATSU ELECTRONICS LTD.

1-1　エッジコンピューティングに関する5W1H

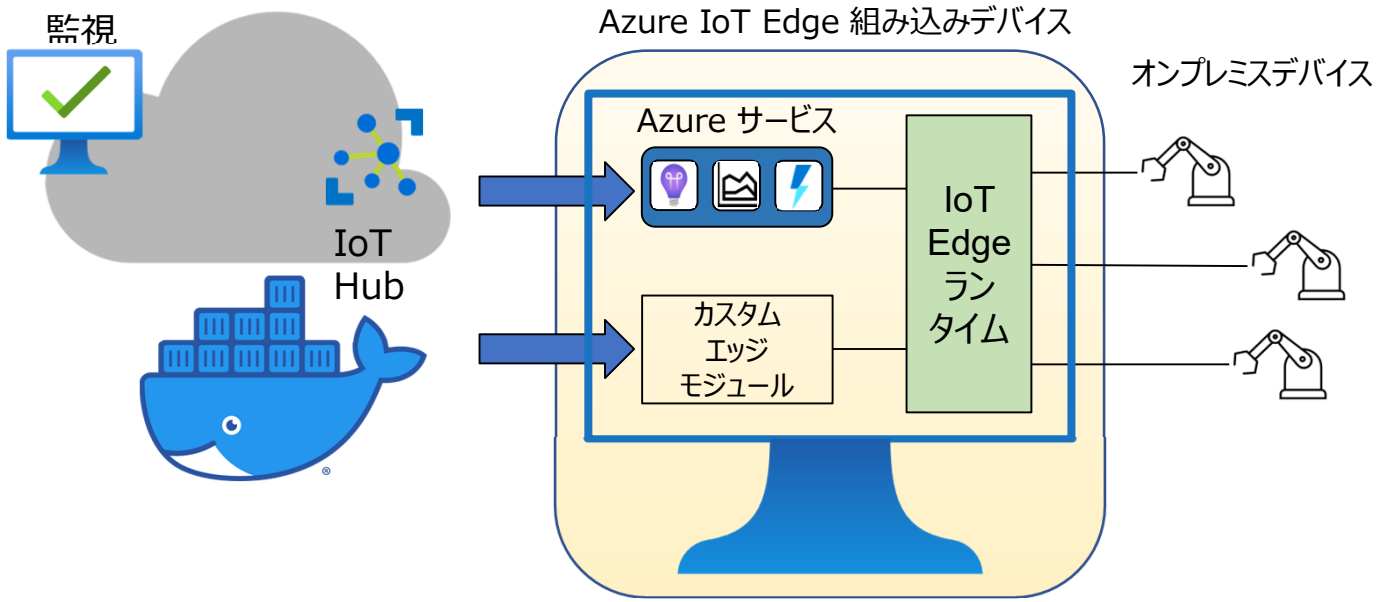
How－エッジコンピューティングはどのように動くか（仕組み）

クラウド上で行っていたAzureサービスやDocker Hub上のサードパーティーのサービス、または自身で作成したカスタムエッジモジュールをIoTデバイス上で動かす。

Azure IoT Edgeを構成する主なコンポーネントは以下の2つである

1.IoT Edge クラウドインターフェース

2.Azure IoT Edge ランタイム

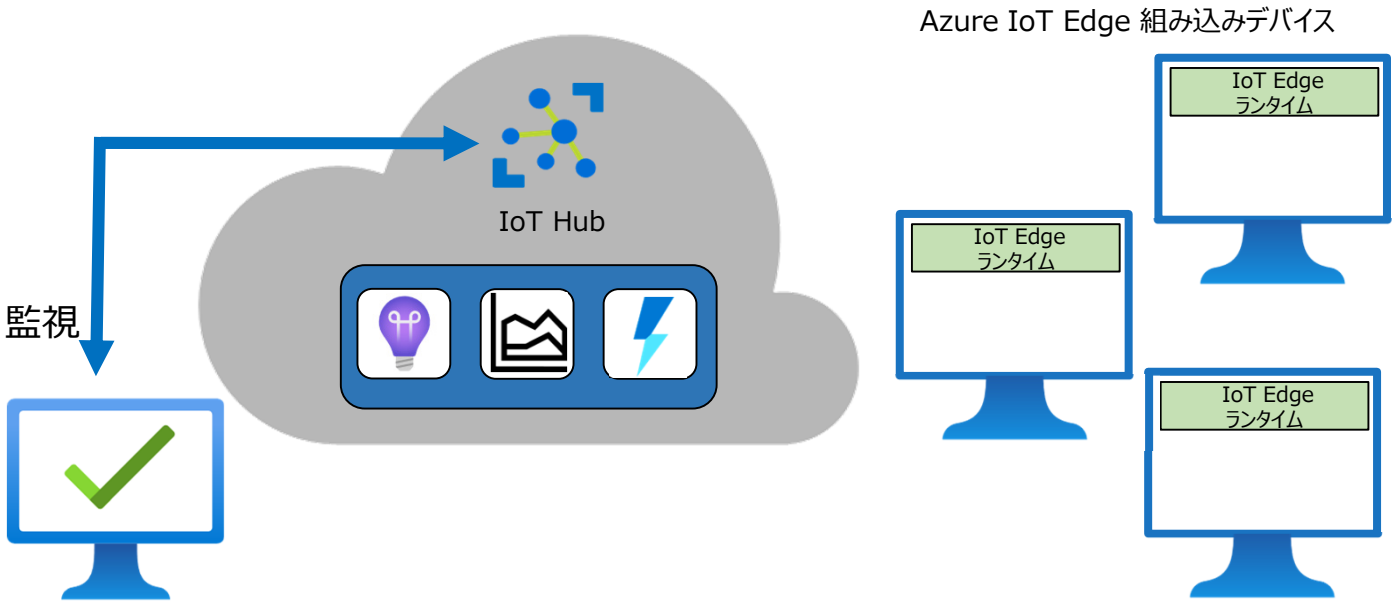


© 2021 KANEMATSU ELECTRONICS LTD.

1-2 Azure IoT Edge概要

・IoT Edge クラウドインターフェースの機能

- クラウドからワークロードを構成・管理し、IoT Edgeデバイス上に配置する
- IoT Edgeデバイス上のワークロードが正常に動くかどうか一元的に監視し、
  地理的に分散している可能性のある様々なデバイスを広範囲に管理できる
  （ワークロード～モジュールを組み合わせて、機能的なソフトウェアにしたもの）

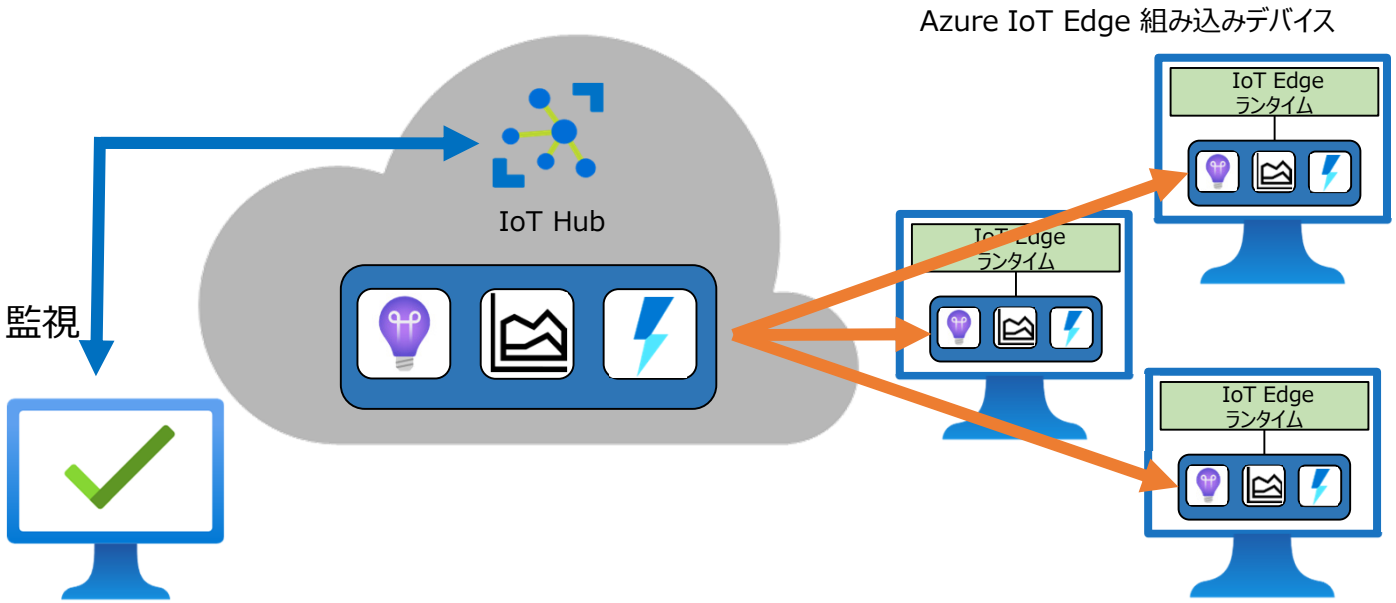


© 2021 KANEMATSU ELECTRONICS LTD.

1-2 Azure IoT Edge概要

・IoT Edge クラウドインターフェースの機能

- クラウドからワークロードを構成・管理し、IoT Edgeデバイス上に配置する
- IoT Edgeデバイス上のワークロードが正常に動くかどうか一元的に監視し、
  地理的に分散している可能性のある様々なデバイスを広範囲に管理できる
  （ワークロード～モジュールを組み合わせて、機能的なソフトウェアにしたもの）

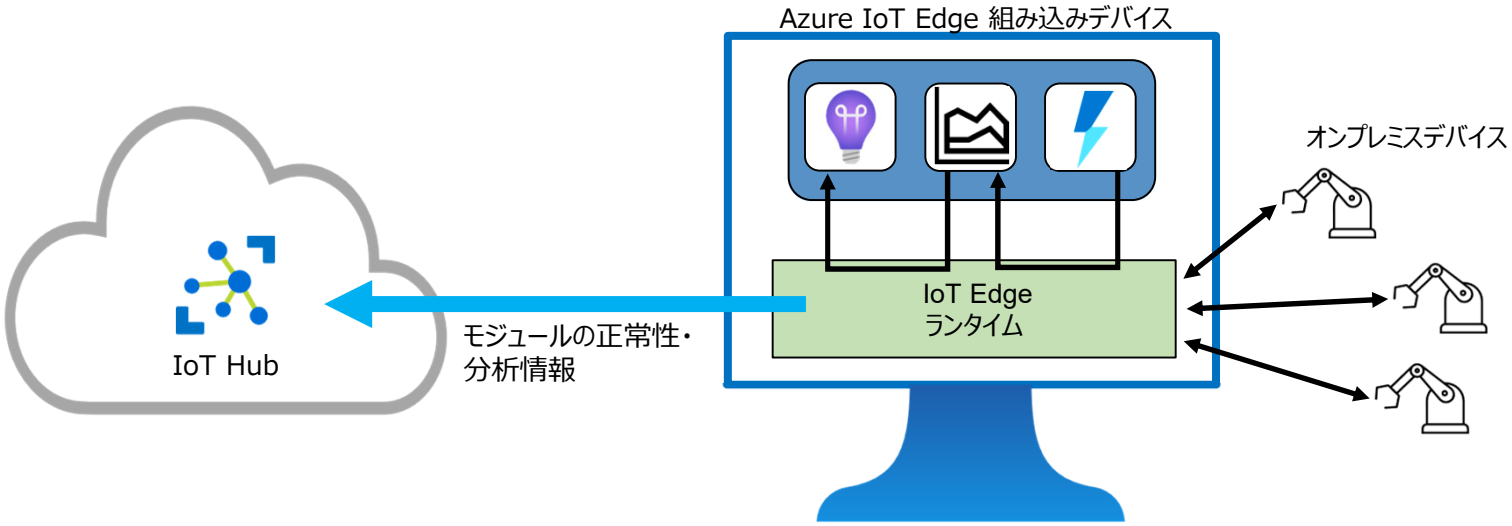


© 2021 KANEMATSU ELECTRONICS LTD.

1-2 Azure IoT Edge概要

・Azure IoT Edgeランタイムの機能

- Azureが提供する標準モジュールやDocker Hub上のカスタムモジュールを IoT Edgeデバイス上にインストール・更新する
- 各デバイスに配置されたモジュールの管理をする
- デバイス間、モジュール間、デバイスとクラウド間の通信を管理する
- モジュールの正常性をクラウドにレポートしてリモート監視を可能にする



© 2021 KANEMATSU ELECTRONICS LTD.

2．検証結果報告

2-1　検証概要

2-2　検証①結果報告

2-2-1　検証①概要

2-2-2 検証①結果

2-2-3 検証①失敗要因

2-2-4 所感

2-3　検証②結果報告

2-3-1 検証②概要

2-3-2 検証②結果

2-3-3 検証②失敗要因

2-3-4 所感

2-4　所感･Azure IoT Edge総合評価

© 2021 KANEMATSU ELECTRONICS LTD.

2．検証結果報告

2-1　検証概要

・目的

KELでAzure IoT Edgeを活かすための知見を得る

・期間

2020/10/6～2021/2/28

・内容

検証①：Raspberry Piに接続したカメラにて、撮影した画像をAzureに転送する

検証②：Azureの画像分類サービス・Custom Visionにて画像分類器を作成し、

エッジデバイスにデプロイ及び画像分類を行う

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証概要

・検証の目標

① Azure IoT Edgeの仕組みを理解する

② Azure IoT Edgeに必要なコンポーネントの設定ができる

③ Raspberry Piのセットアップができる

・検証内容

① Azure IoT Edgeを使用してRaspberry Piに画像転送スクリプトをデプロイする

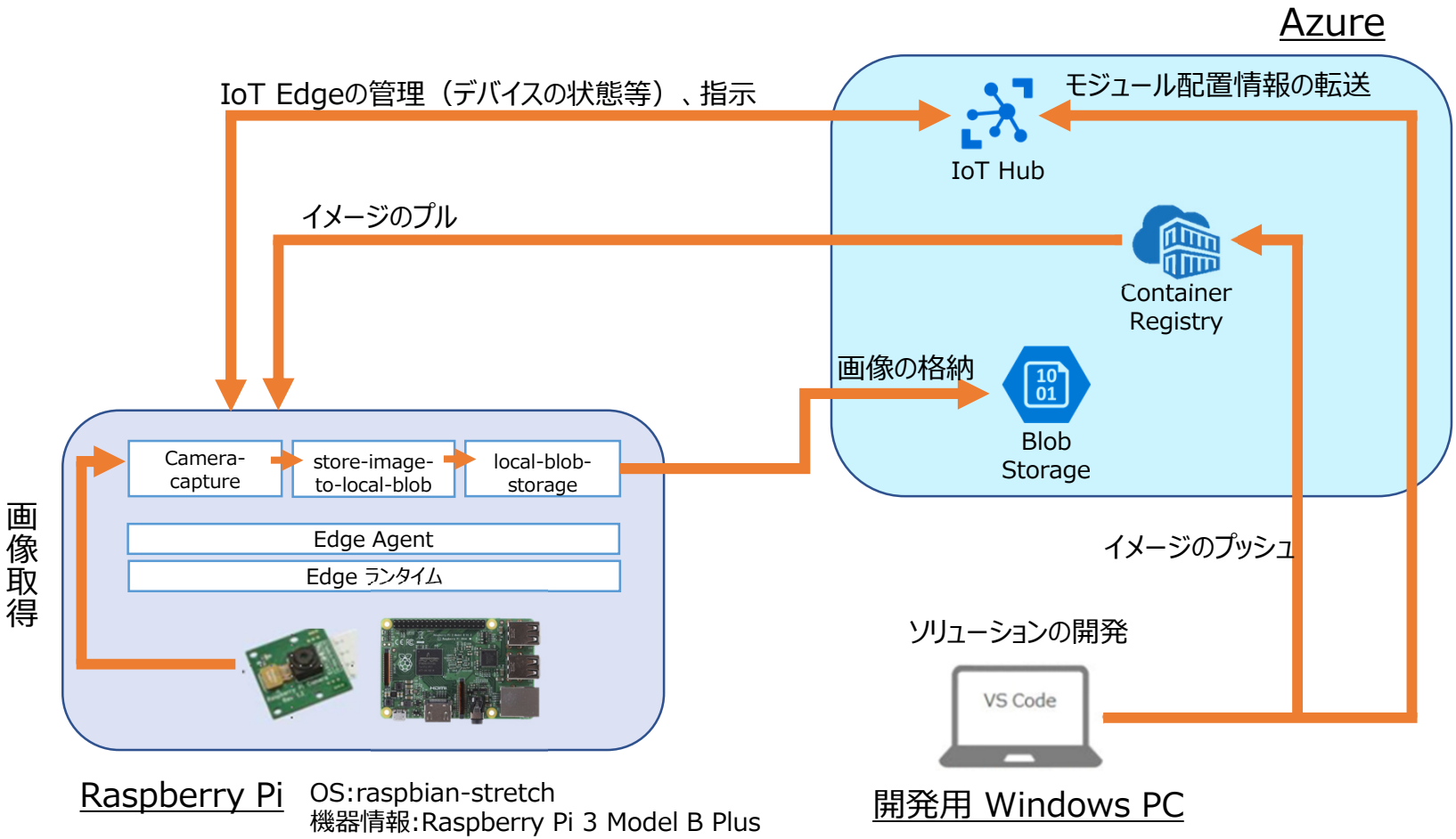
② 接続したカメラで撮影した画像をAzure上へ転送する

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

検証環境図

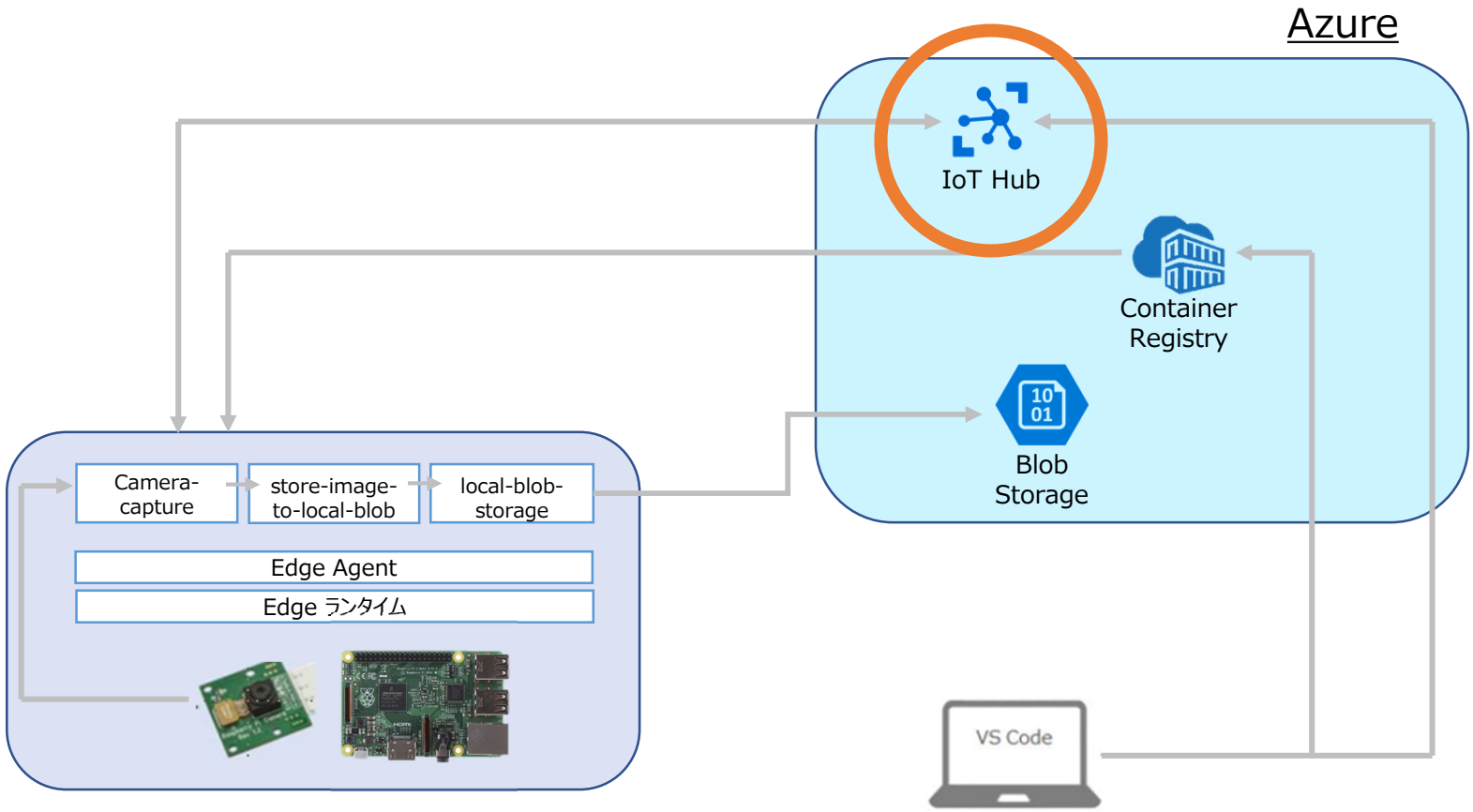


© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

1）IoT Hub の作成



Raspberry Pi　OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

開発用 Windows PC

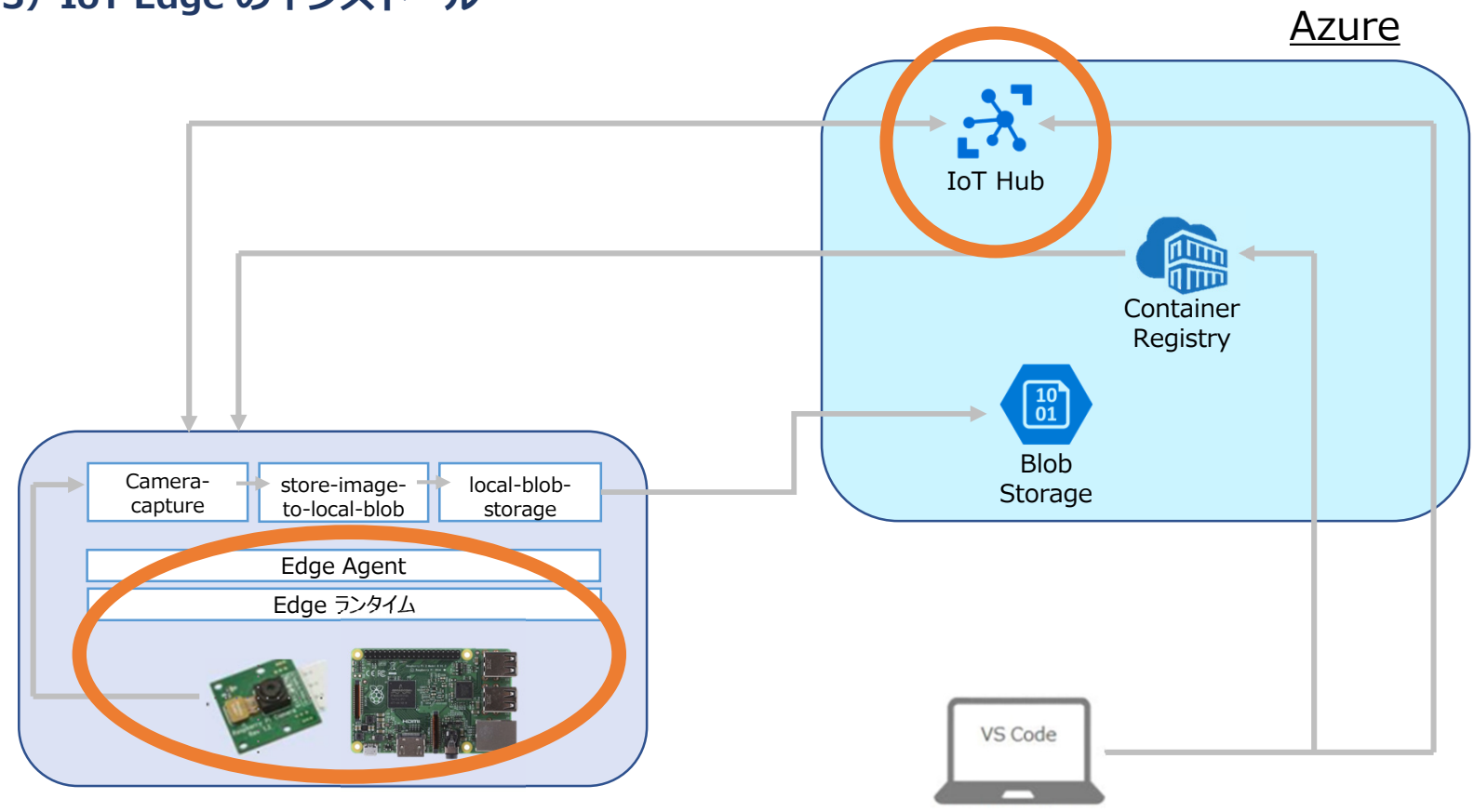
© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

2）IoT Edge デバイスの登録

3）IoT Edge のインストール



Raspberry Pi　OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

開発用 Windows PC

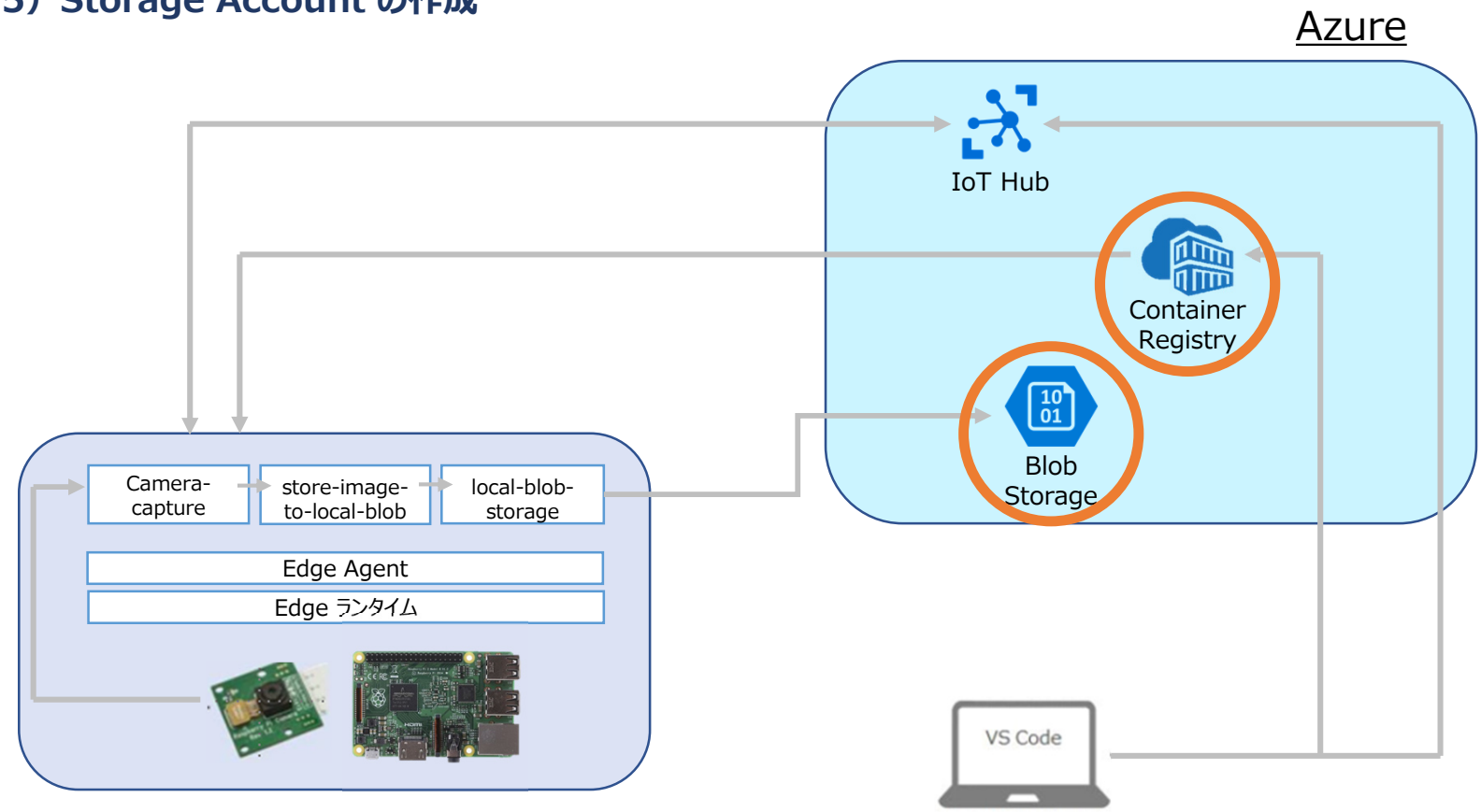
© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

4）Azure Container Registry の作成

5）Storage Account の作成



Raspberry Pi　OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

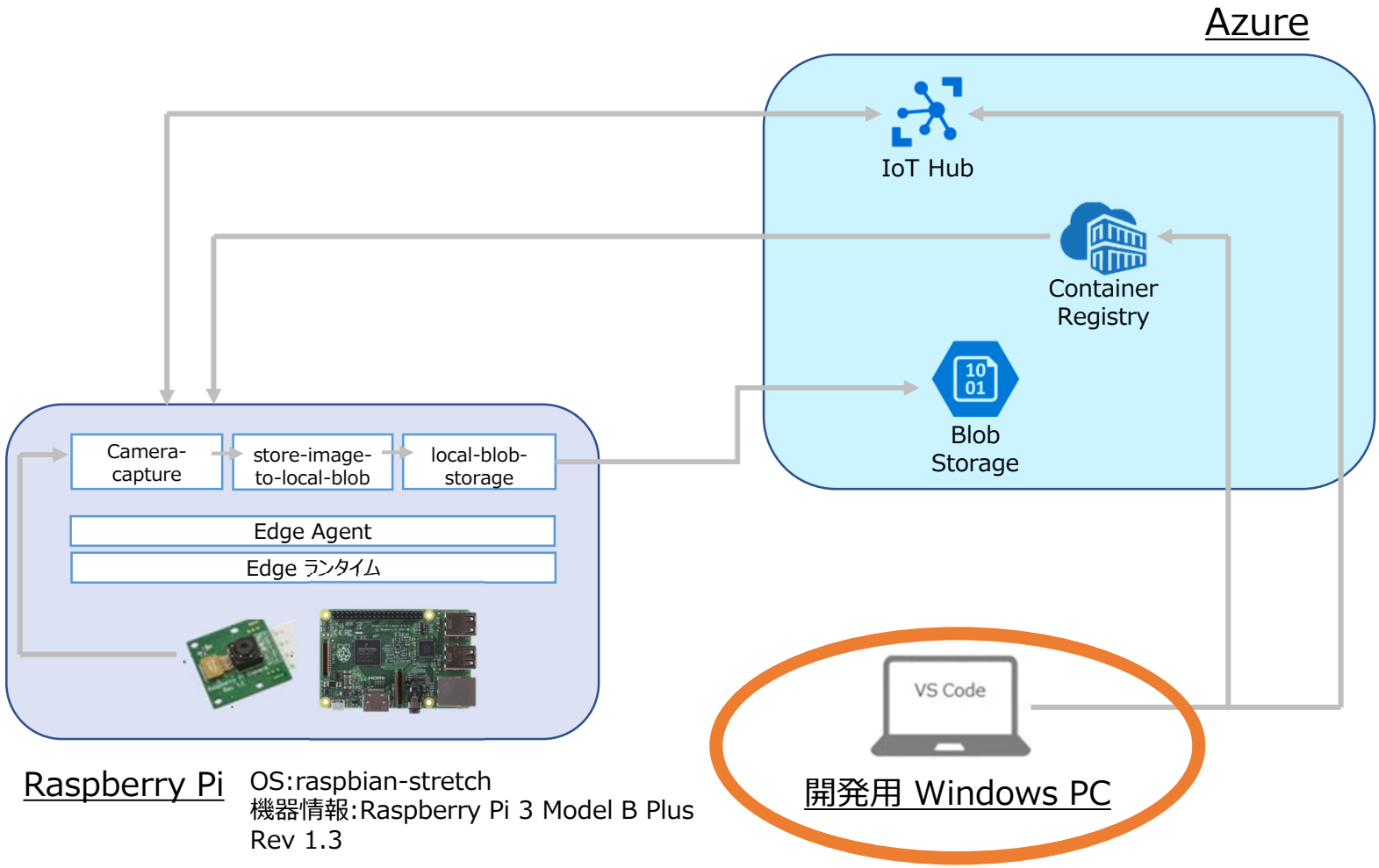
開発用 Windows PC

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

6）Visual Studio Code で IoT Edge ソリューション を作成



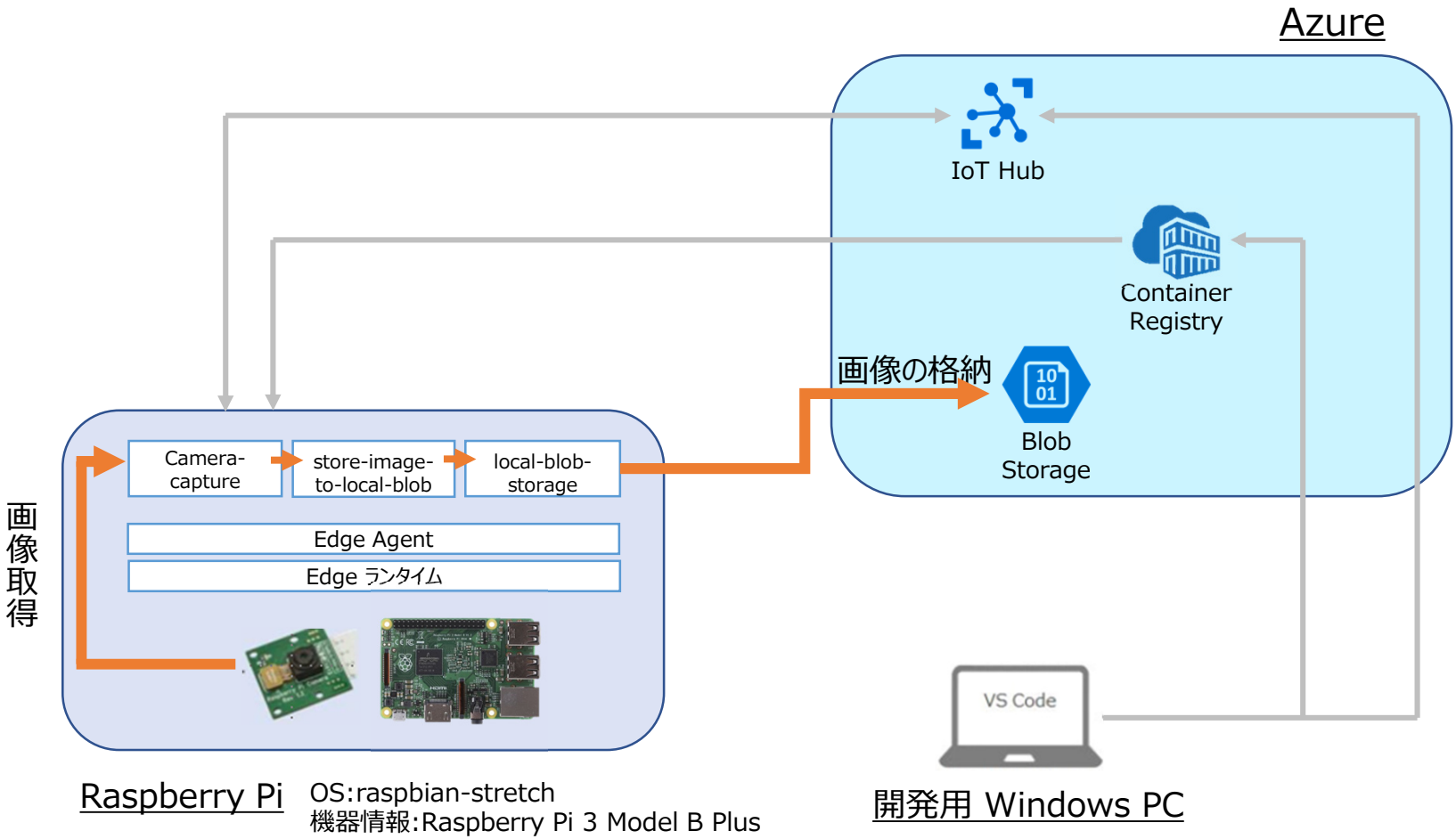
Raspberry Pi　OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

7）モジュールの追加(Blob on Edge モジュール、Camera-capture モジュール)



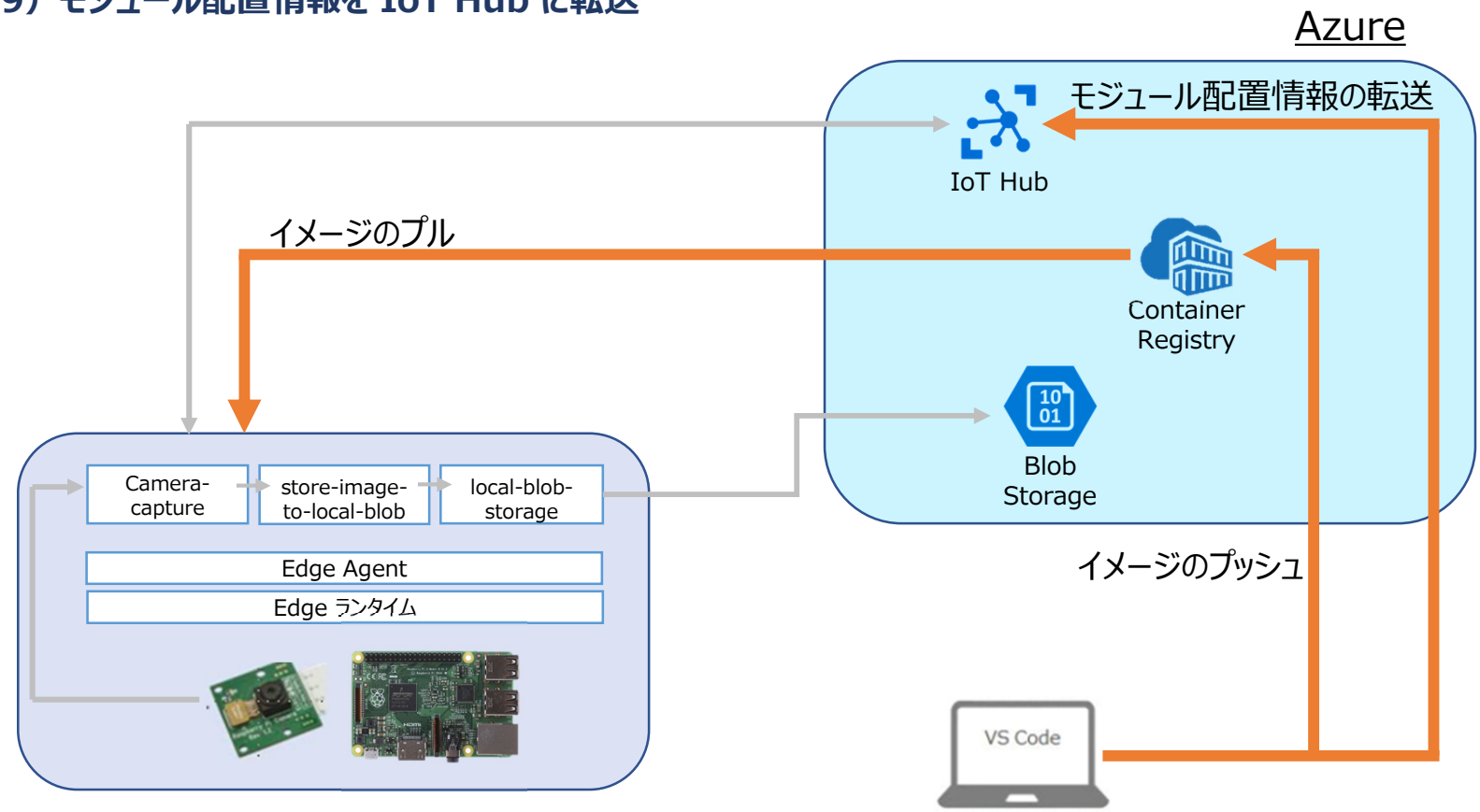
© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-1 検証①概要

8）ソリューションのビルドと Azure Container Registry への プッシュ

9）モジュール配置情報を IoT Hub に転送



Raspberry Pi　OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

開発用 Windows PC

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-2 検証結果

①Raspberry Piに画像転送プログラムをデプロイし、正常に起動させる →失敗

〇 Raspberry Piにプログラムをデプロイすることには成功

× ランタイムが正常に機能せず、画像を送信できなかった

②Azure IoT Edgeを使用してテキストファイルをストレージにアップロードする →成功

〇 作成したコンテナから正しくテキストファイルを参照できる

③同様に画像ファイルをストレージにアップロードする →失敗

× エラーメッセージ等表示されず、原因不明。

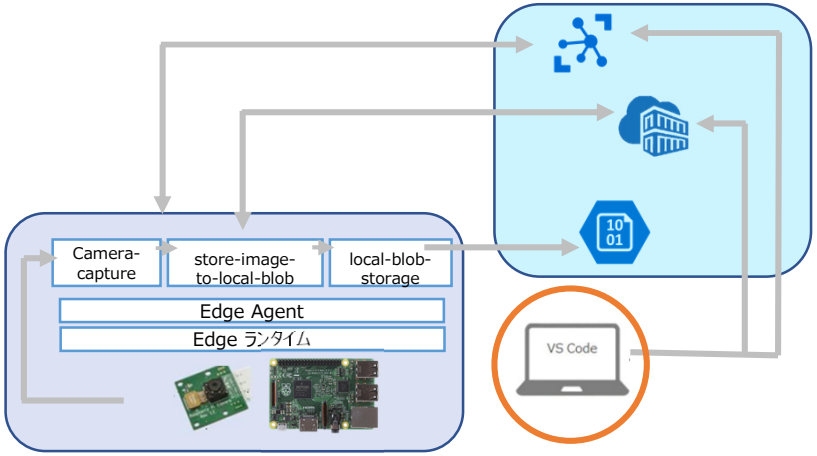
© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-3 検証失敗要因

・Microsoft社提供のモジュールの動作不良

・動作不良を解決するための自身らの知識不足



エラー内容①：cannot import error

・エラーが発報された原因

環境構築時にインストールしたpythonのazure-storage-blobのバージョンは12.6.0だったが、
チュートリアルで参照していたazure-storage-blobのバージョンが2.1であったため

→main.py内でBlockBlobService等モジュールをインポートできなかった

・エラーのトラブルシューティング結果

import先のazure-storage-blob内にBlockBlobServiceが存在しないことが原因と推測

→①azure-blob-storageのバージョンを12.6.0から2.1に下げる

→エラー解消せず

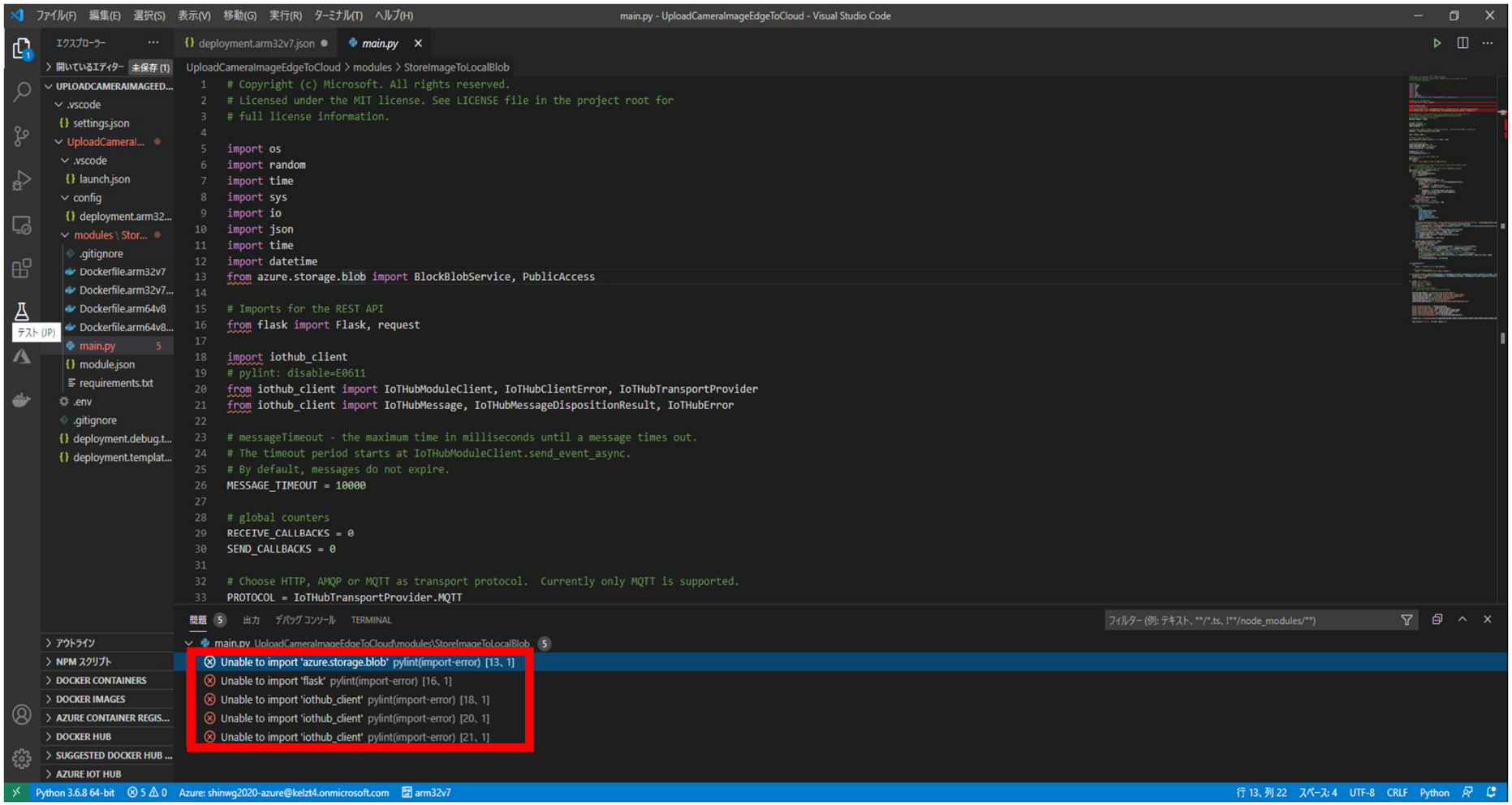
→②Microsoftドキュメントを参考にしてmain.pyのスクリプトを
azure-blob-storage ver12.6.0に対応するように丸ごと変更

→エラーが解消し、テキストファイルのアップロードに成功。

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

【修正前】azure-blob-storage ver2.1に対応するスクリプト



© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

【修正後】main.pyの中身 (1/4)

```
#モジュールをインポートする
import os, uuid, traceback
from azure.storage.blob import BlobServiceClient, BlobClient, ContainerClient,
__version__


try:
    print("Azure Blob storage v" + __version__ + " - Python quickstart sample")

#接続文字列を取得する
    connect_str
='DefaultEndpointsProtocol=https;AccountName=swgc20storage3;AccountKey=lTpuBsk
KzoOZ3zqyQc6ELkvAocFEIolW5/XRiBggeGWgiW6DC46YXrDGxGqaZZfzlCRIf23DXShLez4
+rTQItg==;EndpointSuffix=core.windows.net'

#コンテナを作成する
    blob_service_client = BlobServiceClient.from_connection_string(connect_str)
    container_name = "quickstart" + str(uuid.uuid4())
    container_client = blob_service_client.create_container(container_name)
```


© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

【修正後】main.pyの中身 (2/4)

```
#コンテナにテキストファイルをアップロードする
    local_path = “./data“
    local_file_name = ”quickstart” + str(uuid.uuid4()) + “.txt“
    local_file_name = ”test.text“
    upload_file_path = os.path.join(local_path, local_file_name)

    # Write text to the file
    file = open(upload_file_path, ‘w’)
    file.write(”Hello, World!”)
    file.close()

    # Create a blob client using the local file name as the name for the blob
    blob_client = blob_service_client.get_blob_client(container=container_name,
blob=local_file_name)
    print(“¥nUploading to Azure Storage as blob:¥n¥t” + local_file_name)

    # Upload the created file
    with open(upload_file_path, “rb”) as data:
        blob_client.upload_blob(data)
        print("¥nListing blobs...")
```


© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

【修正後】main.pyの中身 (3/4)

```
#コンテナに画像ファイルをアップロードする
    local_pic_name = “testpic.png“
    upload_pic_path = os.path.join(local_path, local_pic_name)
    print(upload_pic_path)
    pic_blob_client = blob_service_client.get_blob_client(container=container_name,
blob=local_pic_name)
    print(”¥nUploading to Azure Storage as blob:¥n¥t” + local_pic_name)

    #Upload the created file
    with open(upload_pic_path, “rb”) as data:
        pic_blob_client.upload_blob(data, blob_type="BlockBlob")
```


© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

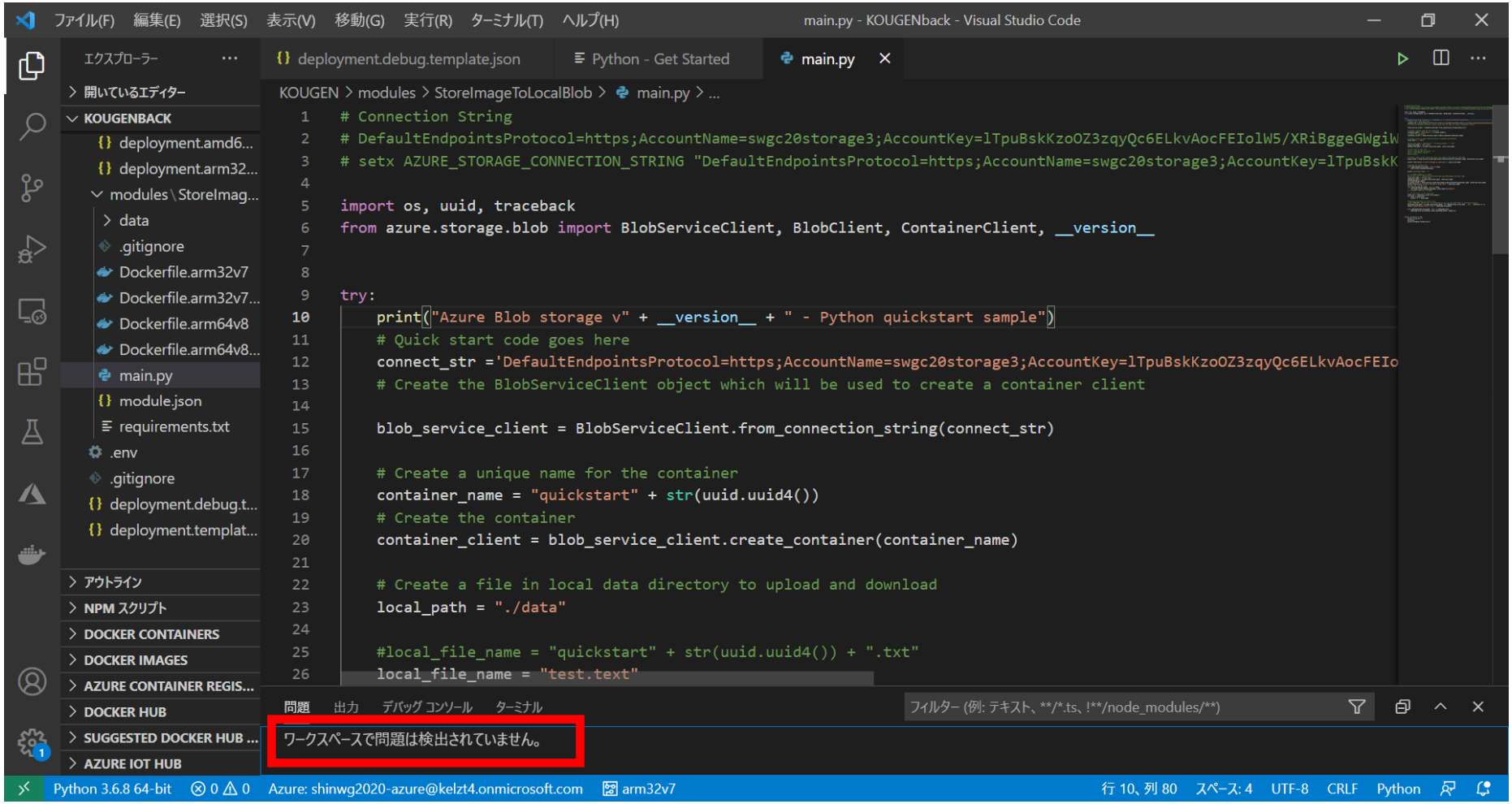
【修正後】main.pyの中身 (4/4)

```
#コンテナ内のBLOBを一覧表示する
    blob_list = container_client.list_blobs()
        for blob in blob_list:
        print(“¥t” + blob.name)

#BLOBをダウンロードする
    download_file_path =
os.path.join(local_path,str.replace(local_file_name ,‘.txt‘,     ’DOWNLOAD.txt’))
    print(“¥nDownloading blob to ¥n¥t” + download_file_path)
    with open(download_file_path, "wb") as
download_file:download_file.write(blob_client.download_blob().readall())

#例外処理
    except Exception as ex:
        print(‘Exception:’)
        print(ex)
        print(traceback.format_exc())
```


© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

【修正後】azure-blob-storage ver12.6.0に対応するスクリプト



© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

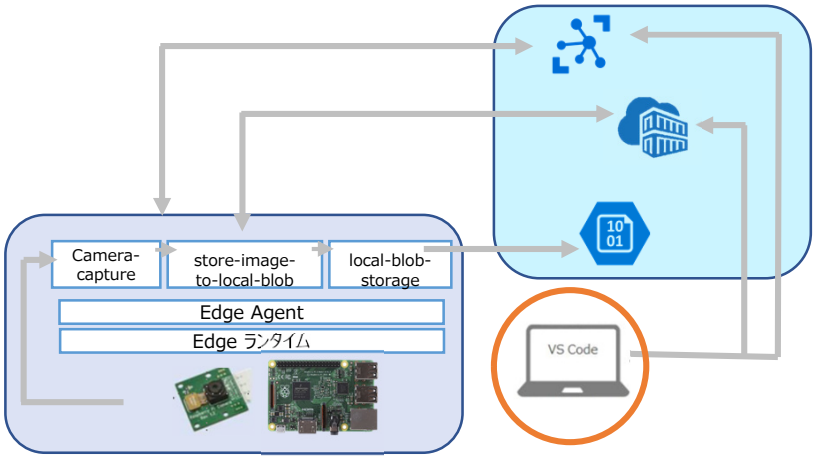
【修正後】ストレージにテキストファイルのアップロード成功

© 2021 KANEMATSU ELECTRONICS LTD.

2-2 検証①結果報告

2-2-3 検証失敗要因

エラー内容② : テキストファイルのアップロードに失敗



・エラーが発報された原因

コンテナレジストリーにプッシュするモジュールのタグを
更新していなかったため

・エラーのトラブルシューティング結果

開発環境のVisual Studio該当箇所でタグを0.0.1から0.0.2に置き換える
→更新したタグで再プッシュ

→コンテナレジストリーの内容が更新され、最新のモジュールがアップロードされていることを確認

© 2021 KANEMATSU ELECTRONICS LTD.

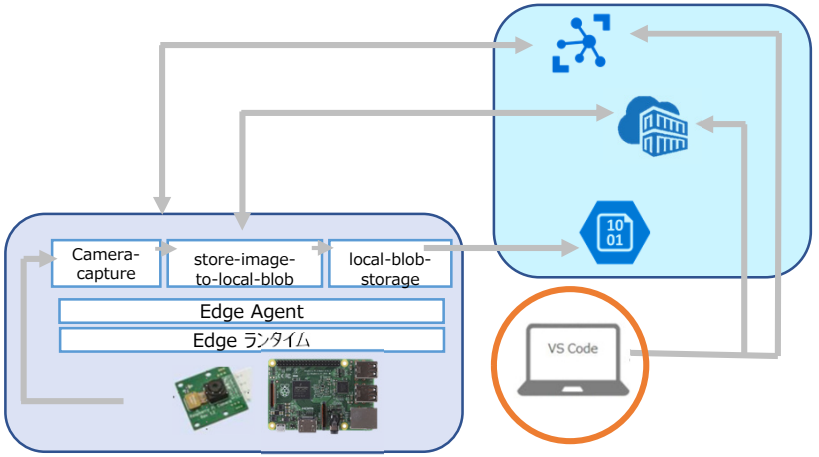
2-2 検証①結果報告

2-2-3 検証失敗要因

エラー内容③：画像ファイルのアップロードに失敗

・エラーが発報された原因

Microsoft提供のモジュールの動作不良



・エラーのトラブルシューティング結果

画像データのサイズが大きく、アップロードができないという仮説を立てたが、
画像データのサイズを20KBにしてもアップロードはできなかった。 →失敗

エラーメッセージ等が表示されず、有効なトラブルシューティングを実施できなかった
→検証①の終了

© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

・検証の目標

① エッジデバイスとAzureのAIサービスの連携に必要な設定ができる

② AIを用いた画像認識の知見を得る

・検証内容

① 画像分類サービスCustom Visionで画像分類器を作成する

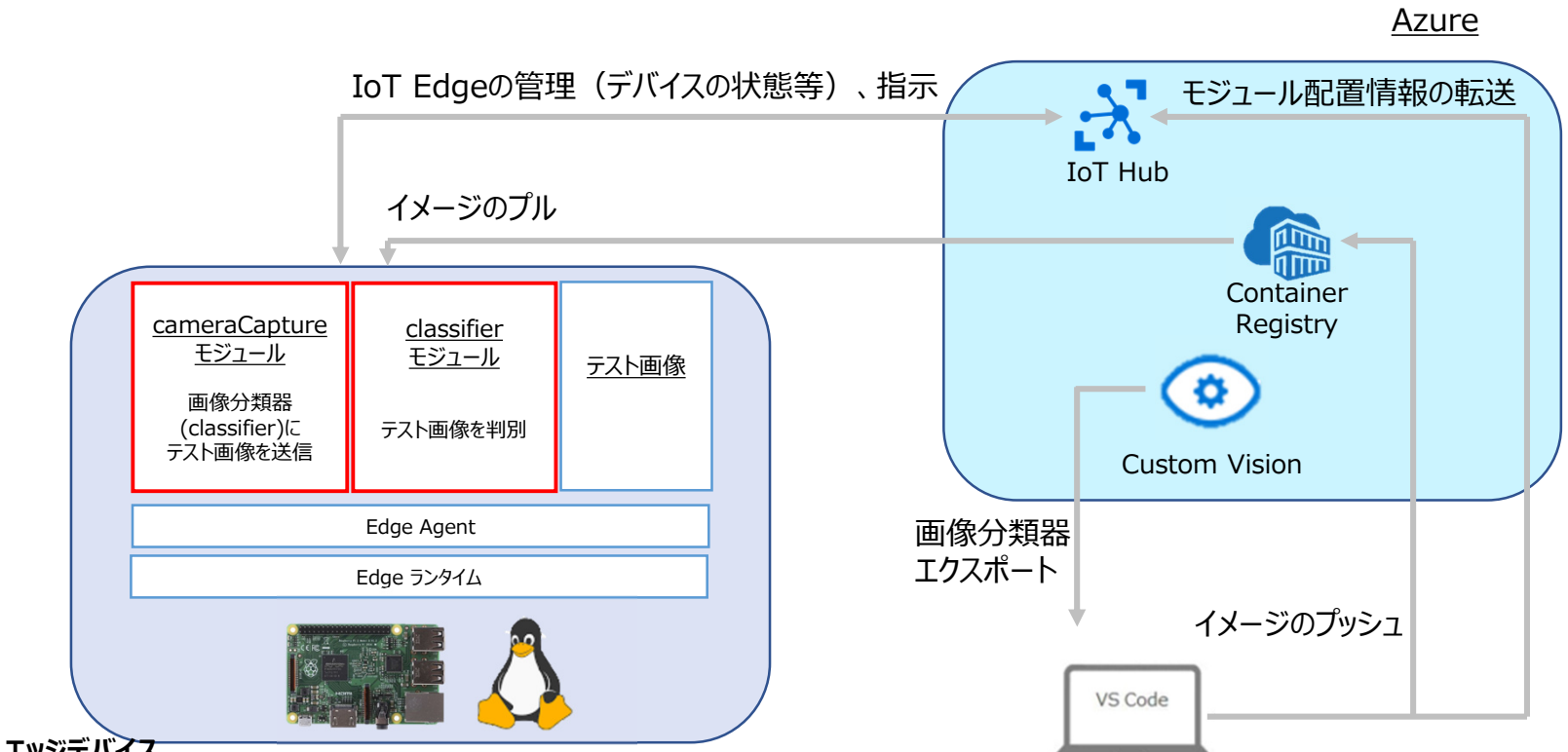
② 画像分類器をエッジデバイス（Raspberry Pi・Linux仮想マシン）にデプロイし、画像分類結果を取得する

© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

検証環境概要



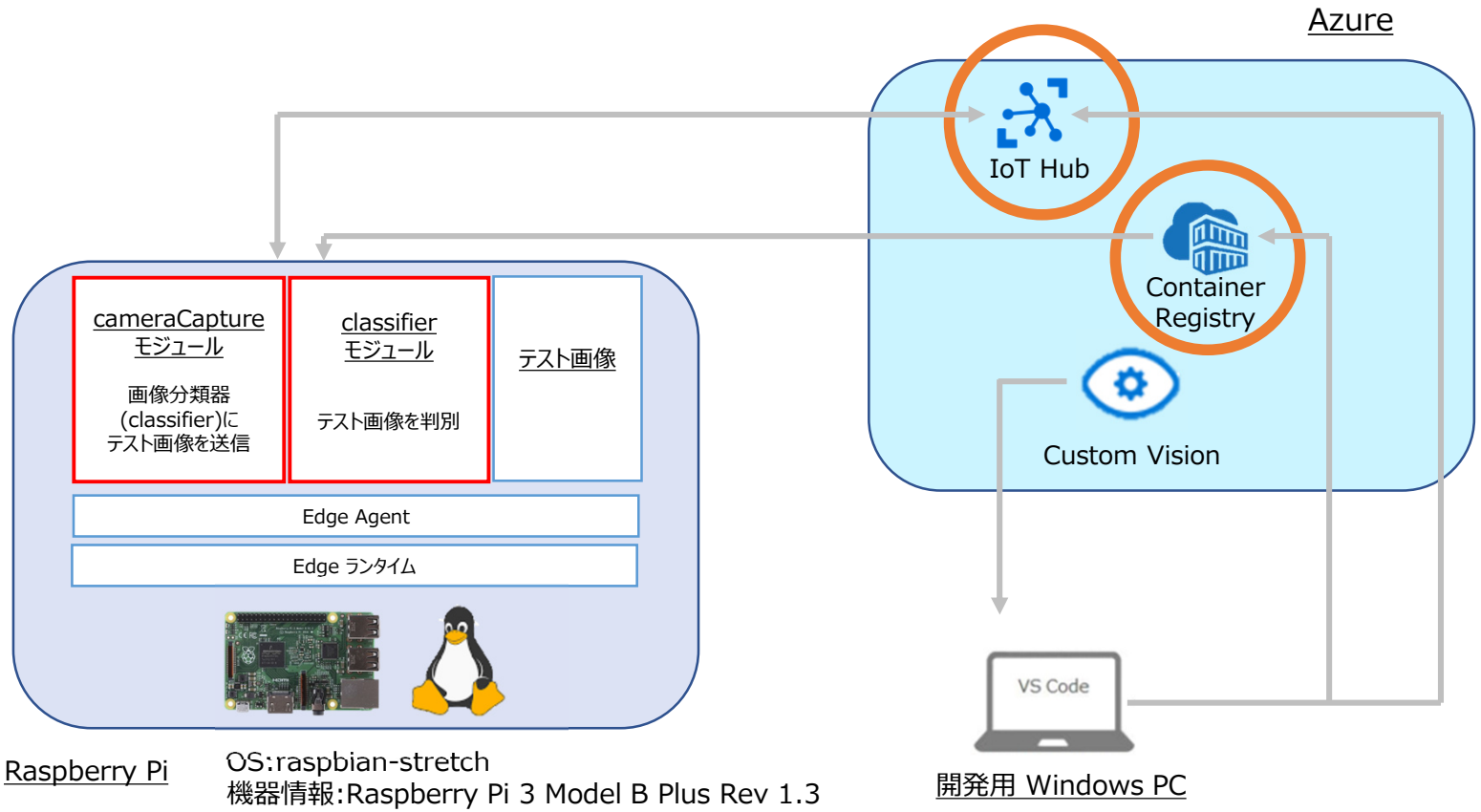
© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

1）IoT Hub の作成

2）Azure Container Registry の作成



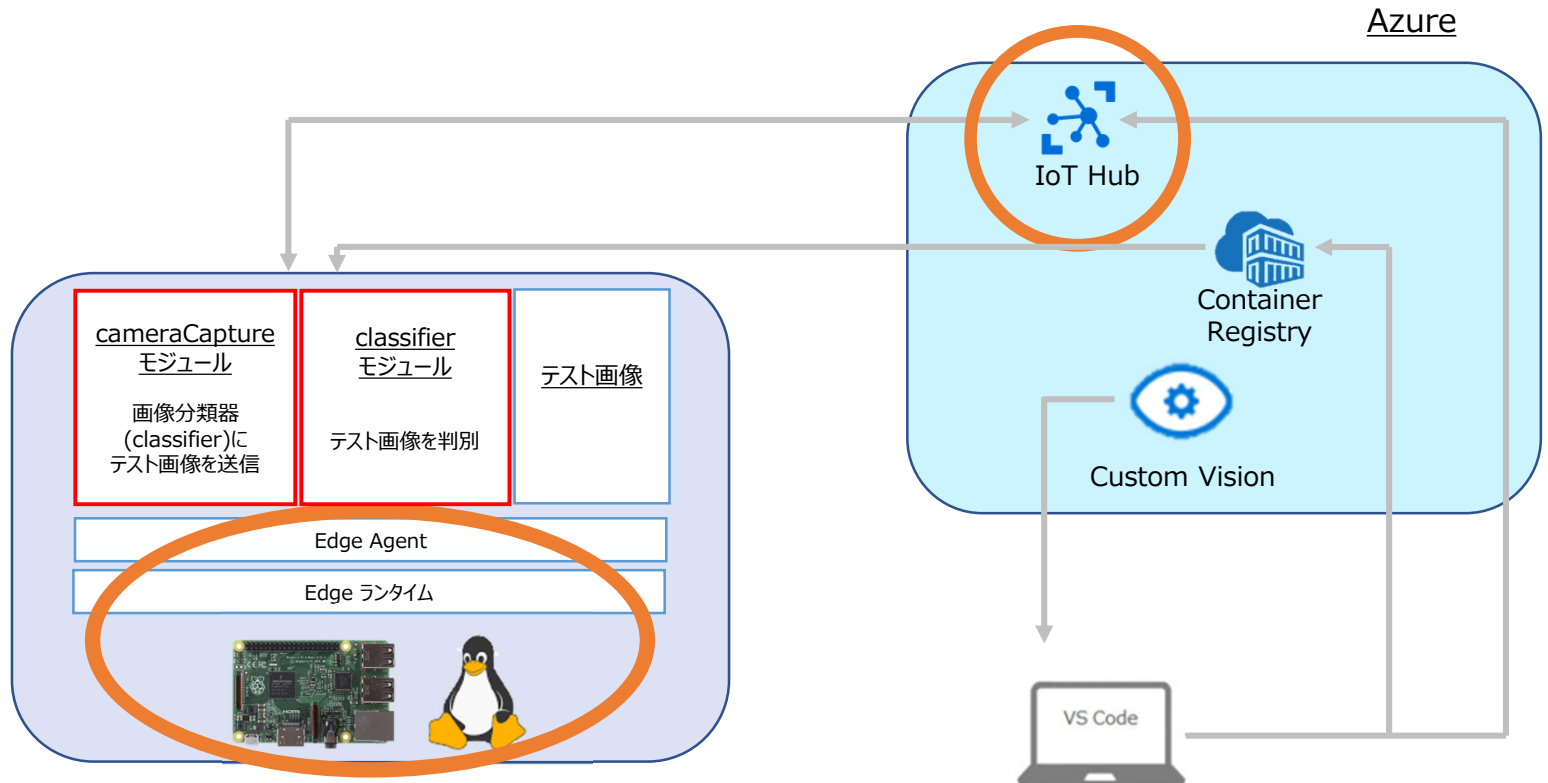
© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

3）IoT Edge デバイスの登録

4）IoT Edge のインストール



Raspberry Pi
OS:raspbian-stretch
機器情報:Raspberry Pi 3 Model B Plus Rev 1.3

Linux
OS:ubuntu18.04

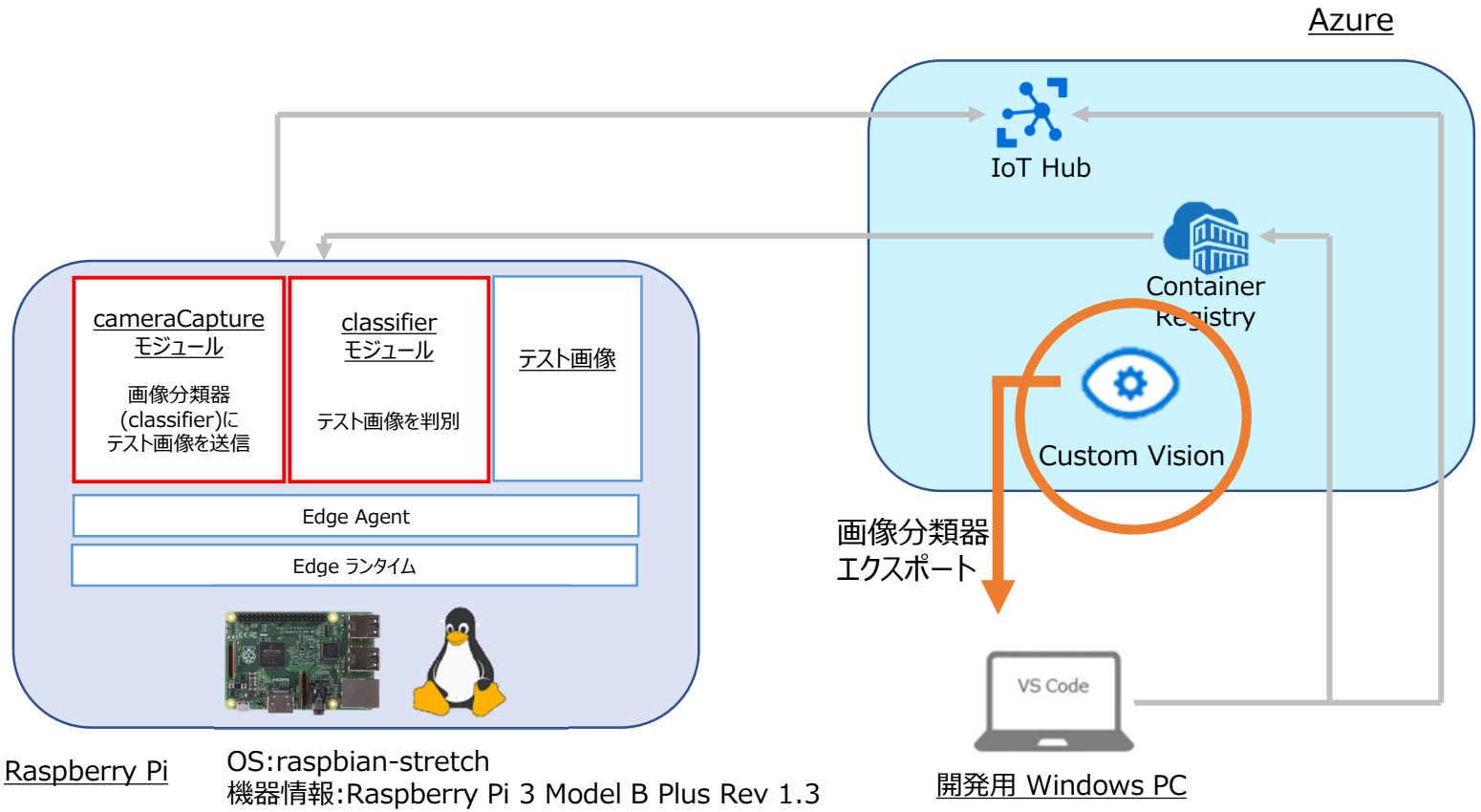
開発用 Windows PC

© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

5）画像分類器構築･エクスポート



© 2021 KANEMATSU ELECTRONICS LTD.

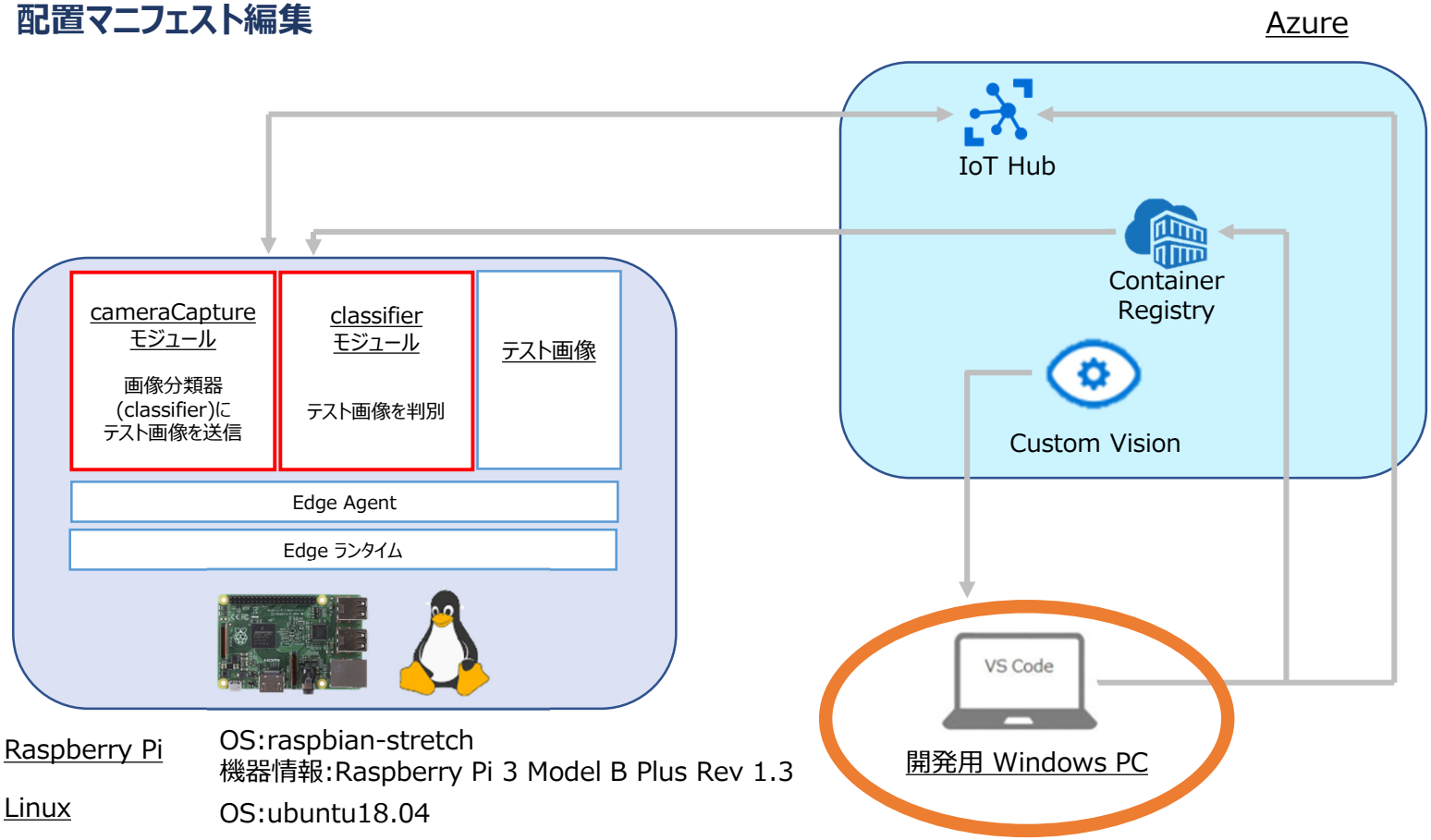
2-3 検証②結果報告

2-3-1 検証②概要

6）ソリューション作成

7）classifier・cameraCaptureモジュール追加

8）配置マニフェスト編集

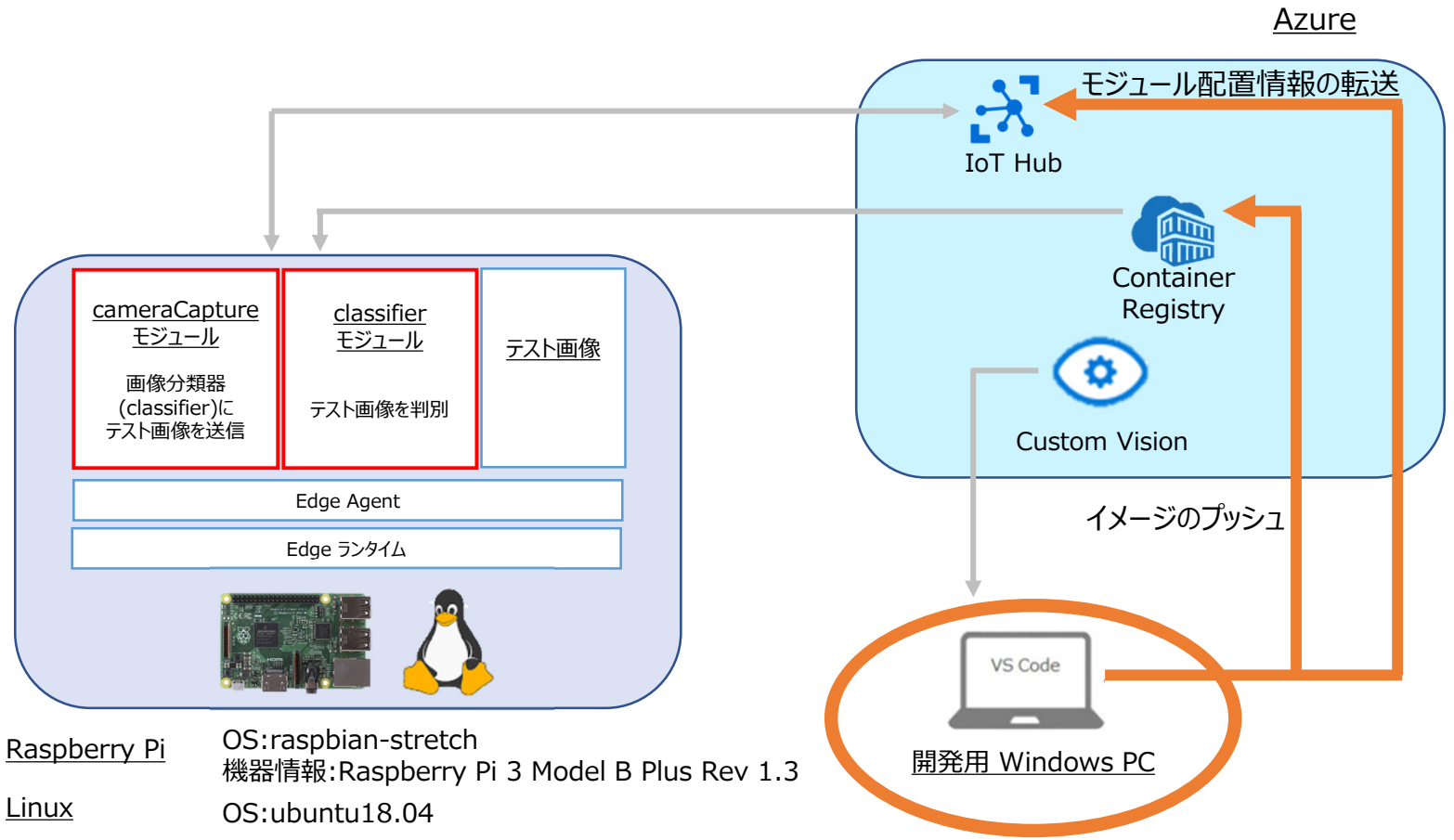


© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

9）ソリューションのビルド･プッシュ



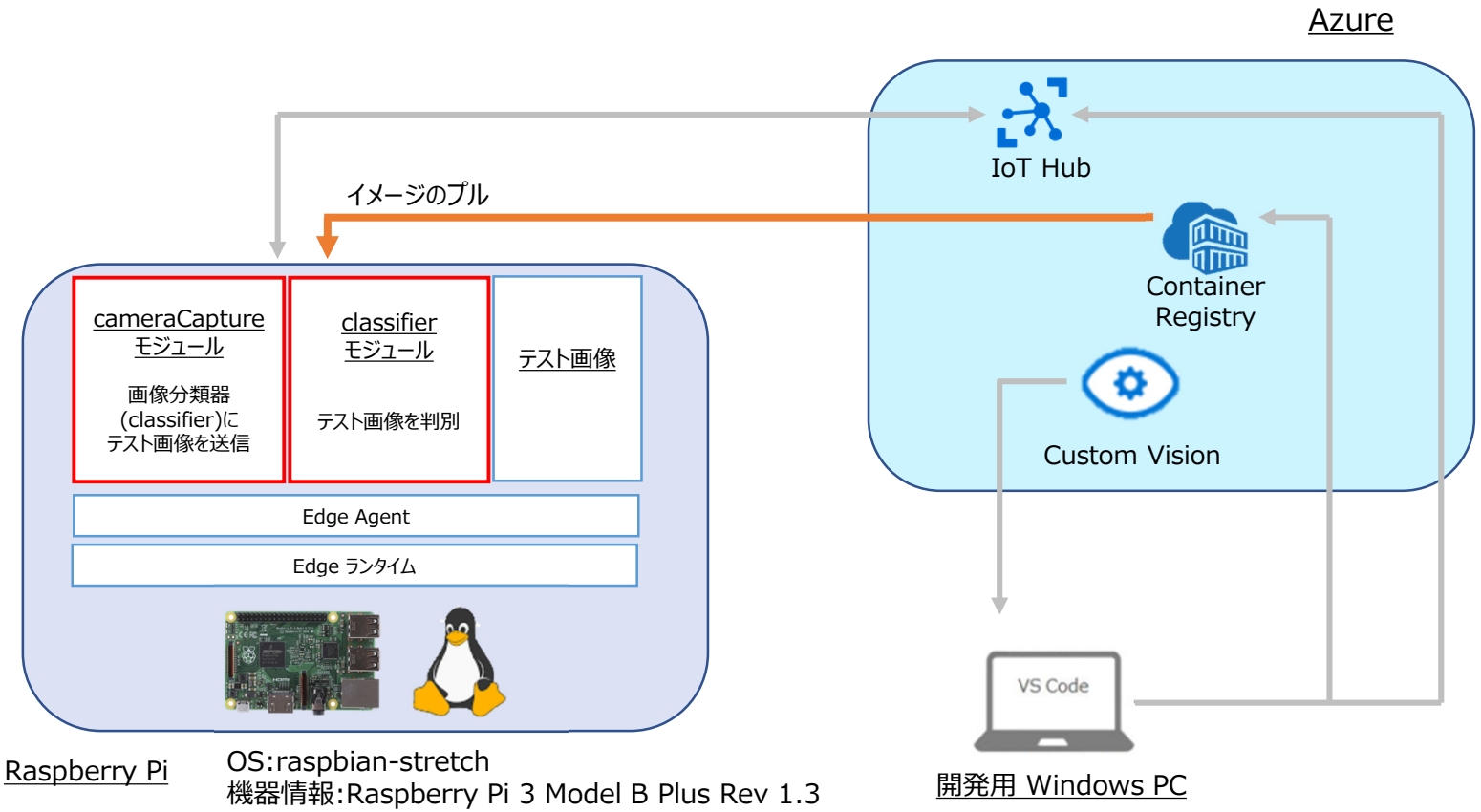
© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-1 検証②概要

10）モジュールをデバイスへデプロイ

11）分類結果を表示する



© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-2 検証結果

①Azure Custom Visionで画像分類器を作成し、エクスポートする →成功

②モジュールのデプロイ・起動 →一部成功

×１つのエッジデバイス上で全てのモジュールを正常に起動させられなかった

〇RaspberryPi上でシミュレーションモジュール(cameraCapture)のデプロイに成功

〇Linux仮想マシン上でカスタムモジュール(classifier)のデプロイに成功

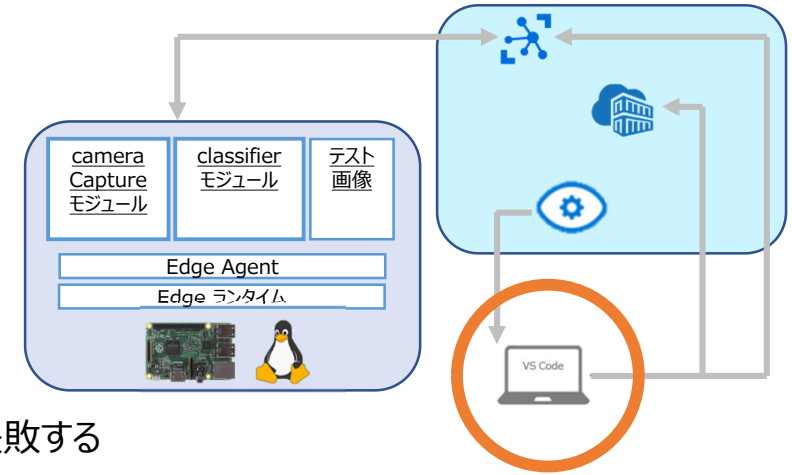
③エッジデバイス上で画像分類結果の取得に成功する →失敗

×cameraCaptureとclassifier間で通信できず、画像分類結果を取得できなかった

© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-2 検証失敗要因



Custom Visionで提供しているファイル情報が古かった

Dockerの理解不足（アーキテクチャ・コンテナ間通信）

エラー内容①：ERROR RUN pip install

Pythonパッケージをインストールできず、ソリューションのビルドに失敗する

・エラーが発報された原因

Custom Visionからダウンロードした画像分類器のDockerfileの情報が古く、

Pythonに参照させるURLが最新のものではなかった

・エラーのトラブルシューティング結果

Dockerfile内のパッケージ参照先URLを修正

→外部ライブラリからパッケージをダウンロードすることに成功し、ビルド作業を続行可能となった

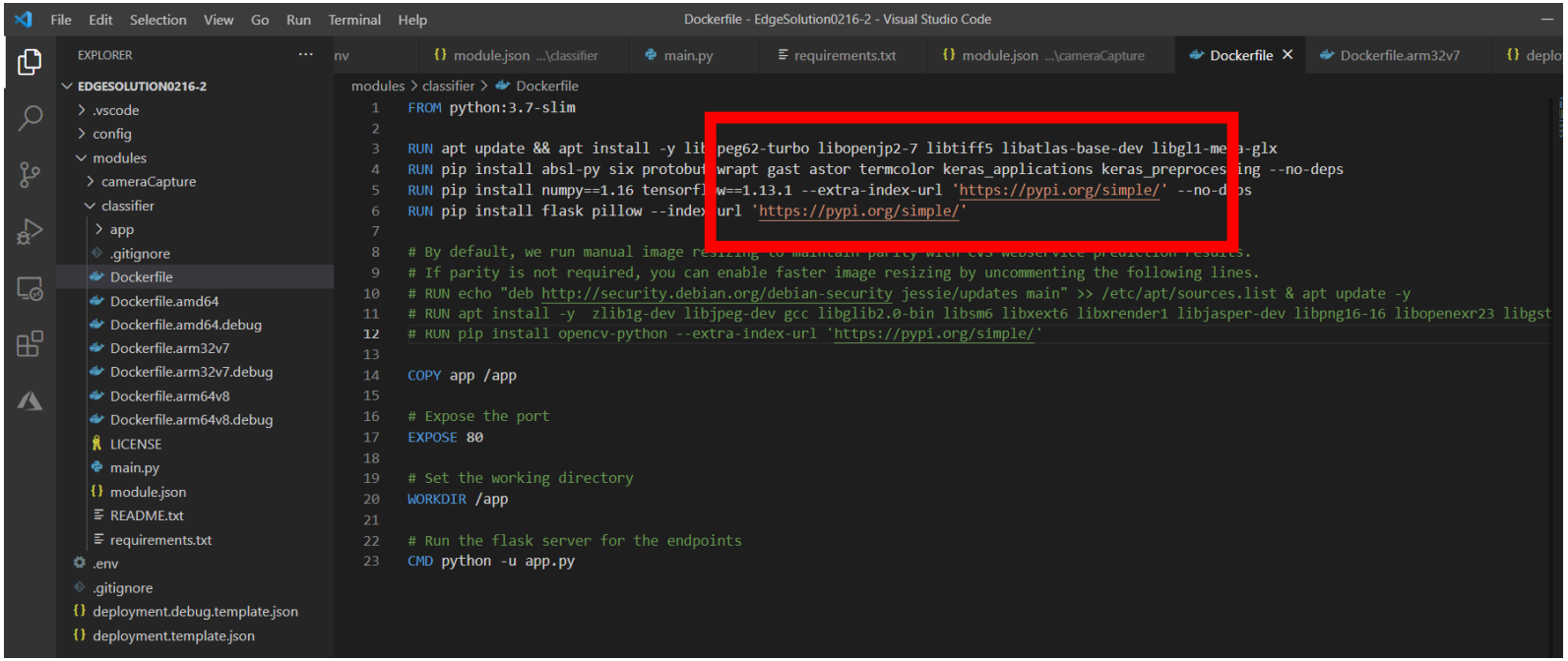
© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

Pythonライブラリの参照先URLを以下のように修正

（修正前）https://www.piwheels.org/simple

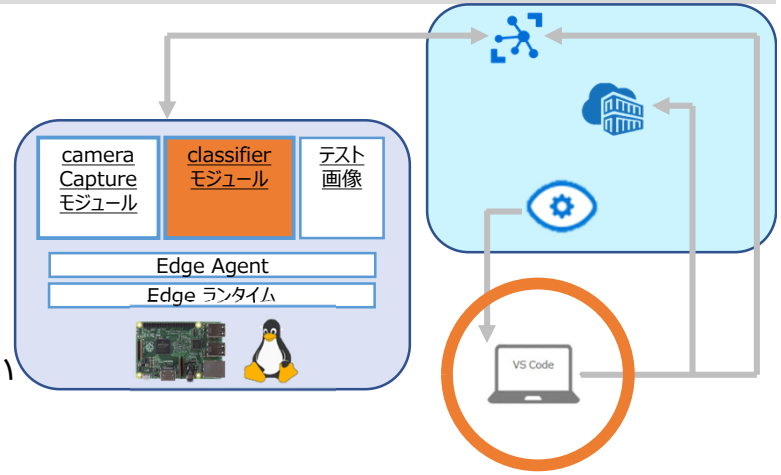
（修正後）https://pypi.org/simple



© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

2-3-2 検証失敗要因



エラー内容②：exec format error

Raspberry Piに対してデプロイしたclassifierモジュールが起動しない

・エラーが発報された原因

開発環境と検証環境の違い（OS・CPUのアーキテクチャ）

※armに設定しなければならなかったがamd64になっていた

※アーキテクチャ・・・CPUで使用される命令体系や内部構造

・エラーのトラブルシューティング結果

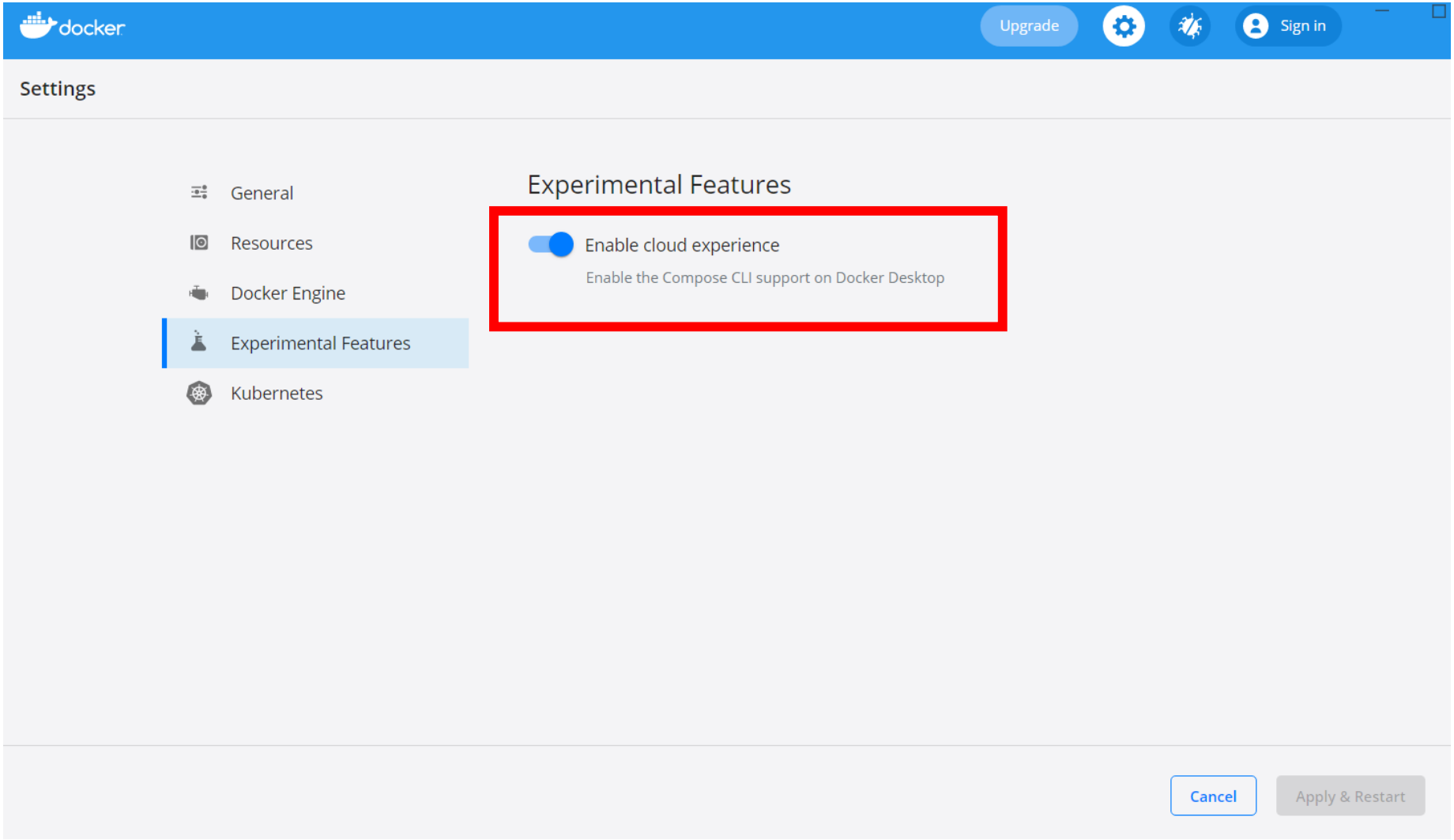
Docker試験機能のマルチCPUアーキテクチャ機能を有効化し、一旦はarmイメージのビルドに成功

※その後再びarmイメージのビルドができなくなった。 原因としてDockerのアップデートが考えられるが、原因特定には至らず

© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

DockerのExperimental Featuresを有効化する



© 2021 KANEMATSU ELECTRONICS LTD.

2-3 検証②結果報告

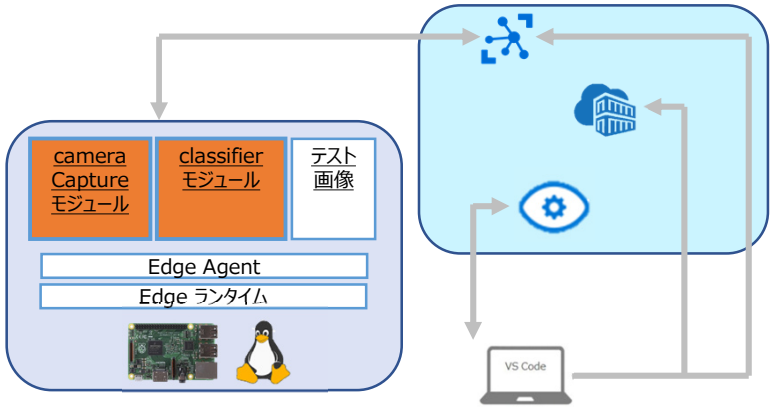
2-3-2 検証失敗要因

エラー内容③：ConnectionRefusedError: not authorized等の接続拒否エラー

classifierモジュールとcameraCaptureモジュール間の通信に失敗し、画像分類結果を取得できない

・エラーが発報された原因

コンテナ間通信設定の不備



・エラーの原因調査

cameraCaptureの画像転送先設定をclassifierのコンテナ名でなくIPアドレスに変更

→全てのモジュールがrunning状態となるが、画像分類結果は取得できない

Microsoft社のチュートリアルではcameraCaptureが自動的にコンテナ間通信を行えないことが判明

© 2021 KANEMATSU ELECTRONICS LTD.

2-4 所感・Azure IoT Edge評価

■Good Point

○ インターネット上のドキュメントが豊富

・検証では日本語のサイト（MicrosoftのチュートリアルやQiitaなど）を参照して作業を進められた

・海外サイトには検証と似た状況におけるエラーについての投稿が多く、エラーの解消に結びつくことが多かった

○ Azure PortalのUIは分かりやすく、操作しやすい

・Edge Hub・Agentの起動や、画像分類器の作成・エクスポートをスムーズに行えたなど、

検証環境のセットアップは容易

・検証中、Azure上からモジュールの状態を容易に把握することができた

© 2021 KANEMATSU ELECTRONICS LTD.

2-4 所感・Azure IoT Edge評価

■Bad Point

△ エッジコンピューティング技術ではエラーの原因特定が困難

・エッジ、クラウド、Docker、プログラミング言語など必要な知識が多岐にわたり、理解やトラブルシューティングに時間を要する

・原因の切り分けが難しく、情報収集時にはエラーコードの検索が主となったが、

必ずしもエラー解消には結び付かなかった。

調査を進めていくなかで、エラーの発生原因とは無関係であることが判明するというケースが多かった

△ トラブルシューティングの際に参照できる情報源がインターネット上のものに限られた

・社内でAzure IoT Edgeに詳しい先輩社員がいなかった

・技術知識の限られる若手社員では仮説を立てることは困難

© 2021 KANEMATSU ELECTRONICS LTD.

2-4 所感・Azure IoT Edge評価

■考察

・エッジコンピューティングは高速・大量のデータ処理が求められる環境や

店舗・工場・物流や公共施設など**多様な環境での活用が今後期待される**

・IoTの普及・デバイスの高性能化・5Gサービスの開始などの社会的な背景から

今後の市場拡大や普及が見込まれる

・検証目標すべてを達成することはできなかったことから、

本検証では技術評価を行うのに十分な知見を得られたとは言い難い

・学習コストが高く、**発展途上の技術である**

© 2021 KANEMATSU ELECTRONICS LTD.

2-4 所感･Azure IoT Edge評価

■考察

検証を行う場合は、以下2点を考慮する必要がある

① トラブルシューティングに時間を要する可能性が高い

プログラミング言語･Docker･クラウドなど、関連する技術分野が多岐にわたるため、幅広い技術知識が必要

→**検証の際にはKEL単体ではなく、他社（Microsoft社･アプリ会社など）と連携して進めることが望ましい**

②Raspberry Pi以外の商用エッジデバイスを対象とした検証や、

AWSなど他製品での検証についても検討の余地がある

© 2021 KANEMATSU ELECTRONICS LTD.

3．参考

・Qiita：[Azure] Azure IoT Edge を使って取得した画像を Azure に送ってみる（検証①）

https://qiita.com/Yoshifumi/items/6c7722d120c51293944e

・Microsoft：Linux のデバイス用の IoT Edge モジュールを開発する（検証①・②）

https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-develop-for-linux?view=iotedge-2018-06

・Microsoft：Custom Vision Service を使用してエッジで画像の分類を実行する（検証②）

https://docs.microsoft.com/ja-jp/azure/iot-edge/tutorial-deploy-custom-vision?view=iotedge-2018-06#build-and-push-your-iot-edge-solution

© 2021 KANEMATSU ELECTRONICS LTD.

KEL
KANEMATSU ELECTRONICS LTD.

© 2021 KANEMATSU ELECTRONICS LTD.