IoT Edge Module Deployment

8/17/2018

Introduction / Summary

Deployment and Management of the IoT Edge module is one of key usages of the IoT Edge. You can deploy the IoT Edge module to a single edge node, or at the scale.

We will briefly explain the mechanics and methodologies of the IoT Edge module Deployment.

Azure IoT Edge devices follow a device lifecycle that is similar to other types of IoT devices:

- IoT Edge devices are provisioned, which involves imaging a device with an OS and installing the IoT Edge runtime.
- The devices are configured to run IoT Edge modules, and then monitored for health.
- Finally, devices may be retired when they are replaced or become obsolete.

Goals

After completing this instruction, your will complete :

- Cloning sample code from Git repository
- Building sample code
- Configuring IoT Edge deployment
- Deploying IoT Edge module(s)

Requirements/Prerequisites

- A development environment
  With required tools such as VSCode and Git client
- Azure Resources
  IoT Hub, IoT Edge Device, and Azure Container Registry
- IoT Edge Device With this sample, we will use Raspberry Pi with Raspbian OS and the Azure IoT Edge Runtime configured
- Camera
  The sample uses camera input for AI workload. Any camera that works with OpenCV such as USB Webcam or Raspberry Pi Camera
- SenseHat (Optional)
  The sample uses Pi Hat to display AI analysis result to Pi Hat. With this Article, we will remove this module.

Deployment Methods

Azure IoT Edge provides two ways to configure the modules to run on IoT Edge devices

- A single device deployment
  Deploy to one device
- Managing large fleets of IoT Edge Devices
  Deploy to multiple devices based on query

Both methods are available in the Azure Portal and programmatically, including AZ CLI command line tool.

Deployment Manifest

The IoT Edge runtime installs and executes IoT Edge modules on target IoT Edge devices by configuring an IoT Edge Deployment Manifest.
The deployment manifest contains :

- A list of modules ("modules", "image")
- Initialization parameters ("createOptions")
- Container Registry Access credential ("registryCredentials")
- Message Route information

Sample Deployment Manifest

```
{
    "modulesContent": {
        "$edgeAgent": {
            "properties.desired": {
                "schemaVersion": "1.0",
                "runtime": {
                    "type": "docker",
                    "settings": {
                        "loggingOptions": "",
                        "minDockerVersion": "v1.25"
                        "registryCredentials": {
                          "registryName": {
                            "username": "mysampleacr",
                            "password": "***********************",
                            "address": "mysampleacr.azurecr.io"
                        }
                    }
                },
                "systemModules": {
                    "edgeAgent": {
                        "type": "docker",
                        "settings": {
                            "image": "mcr.microsoft.com/azureiotedge-agent:1.0",
                            "createOptions": "{}"
                        }
                    },
                    "edgeHub": {
                        "type": "docker",
                        "settings": {
                            "image": "mcr.microsoft.com/azureiotedge-hub:1.0",
                            "createOptions": "{\n  \"HostConfig\": {\n    \"PortBindings\": {\n      \"8883/tcp\": [\
                        },
                        "status": "running",
                        "restartPolicy": "always"
                    }
                },
                "modules": {
                    "tempSensor": {
                        "type": "docker",
                        "settings": {
                            "image": "mcr.microsoft.com/azureiotedge-simulated-temperature-sensor:1.0",
                            "createOptions": "{}"
                        },
                        "status": "running",
                        "restartPolicy": "always",
                        "version": "1.0"
                    }
                }
            }
        },
        "$edgeHub": {
            "properties.desired": {
                "schemaVersion": "1.0",
                "routes": {
                    "route": "FROM /messages/* INTO $upstream"
                },
                "storeAndForwardConfiguration": {
                    "timeToLiveSecs": 7200
                }
            }
        }
    }
}
```

Step 1 : Clone Sample Code from Git Repository

In your development machine, open CMD console or Powershell console, then run the following command

```
git clone https://github.com/Azure-Samples/Custom-vision-service-iot-edge-raspberry-pi.git <your destination foler>
```

Example : Clone to **C:\Git\Custom-vision-service-iot-edge-raspberry-pi** folder.

```
git clone https://github.com/Azure-Samples/Custom-vision-service-iot-edge-raspberry-pi.git C:\Git\Custom-vision-servi
```

Step 2 : Open the sample with Visual Studio Code

In your console, navigate to **C:\Git\Custom-vision-service-iot-edge-raspberry-pi** then type "code ."

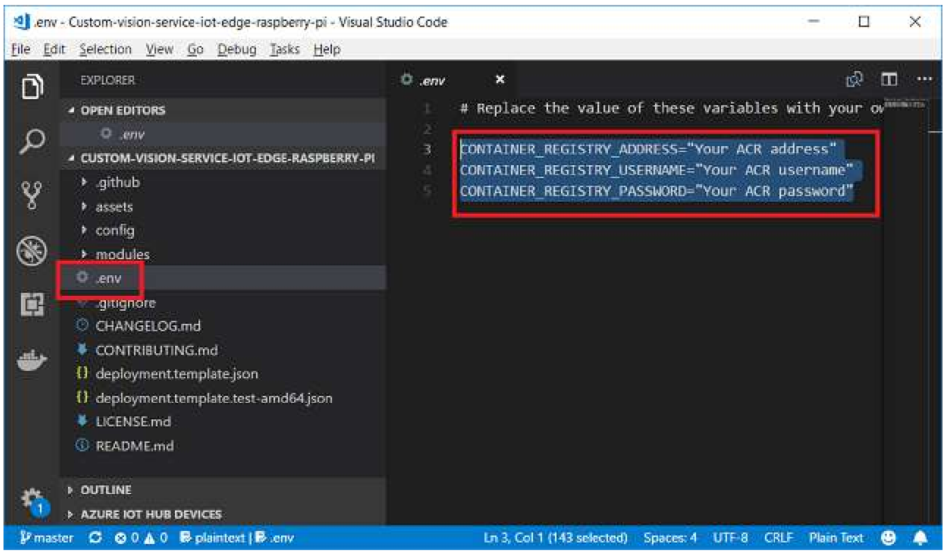
or

Launch VSCode then

- **[File]** -> **[Open Folder... [CTRL+K, CTRL+O]]**
- Navigate to C:\Git\Custom-vision-service-iot-edge-raspberry-pi

Step 3 : Set Azure Container Registry credential

- In VSCode, open **[.env]** file from left page
- Add ACR credential to the .env file
- Save and close the .env file



```
# Replace the value of these variables with your ow

CONTAINER_REGISTRY_ADDRESS="Your ACR address"
CONTAINER_REGISTRY_USERNAME="Your ACR username"
CONTAINER_REGISTRY_PASSWORD="Your ACR password"
```

.env - Custom-vision-service-iot-edge-raspberry-pi - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
.env
CUSTOM-VISION-SERVICE-IOT-EDGE-RASPBERRY-PI
.github
assets
config
modules
.env
.gitignore
CHANGELOG.md
CONTRIBUTING.md
deployment.template.json
deployment.template.test-amd64.json
LICENSE.md
README.md
OUTLINE
AZURE IOT HUB DEVICES
master
0 0
plaintext | .env
Ln 3, Col 1 (143 selected)
Spaces: 4
UTF-8
CRLF
Plain Text

Example :

```
CONTAINER_REGISTRY_ADDRESS="mysampleacr.azurecr.io"
CONTAINER_REGISTRY_USERNAME="mysampleacr"
CONTAINER_REGISTRY_PASSWORD="dfidaupre*7ur98eaed"
```

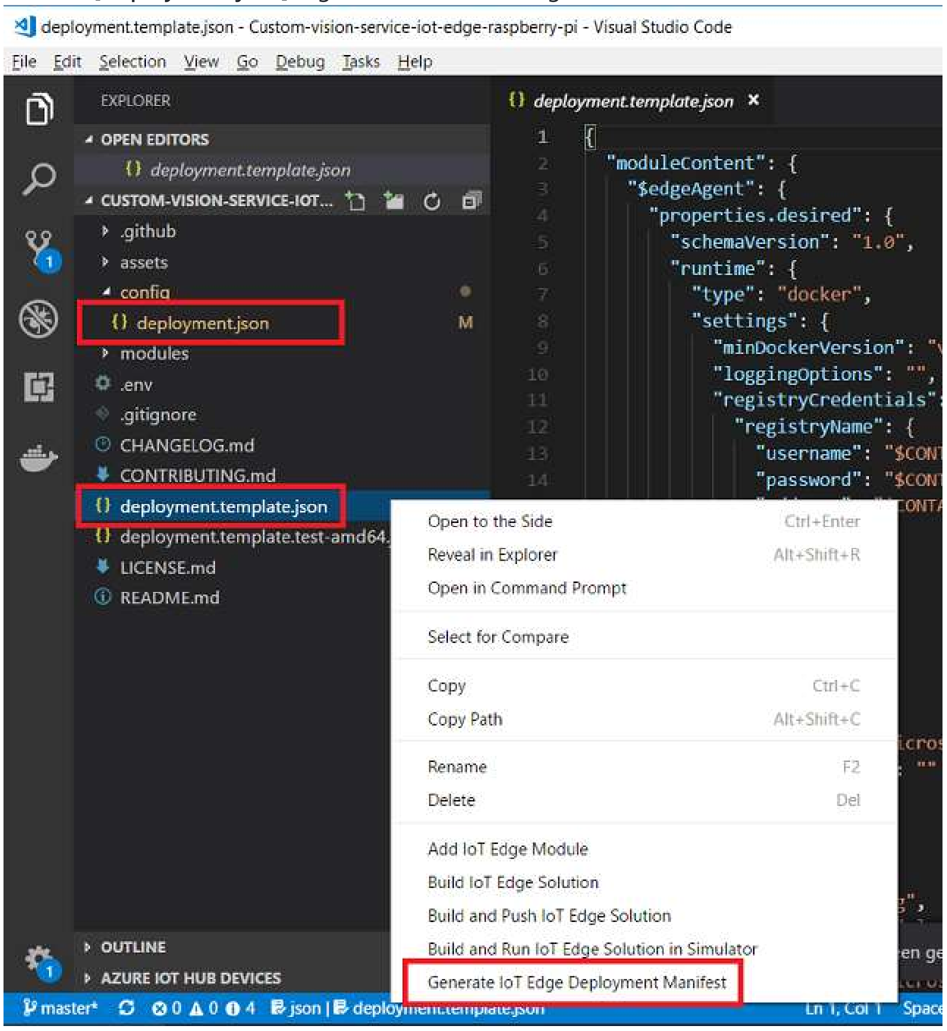
> [!TIP] To retrive ACR Credential, please refer to Setup Azure Resources

Step 4 : Generate Deployment Manifest

VSCode can generate deployment manifest file based on the tamplete.

- Right click **[deployment.template.json]** in left pane
- Select **[Generate IoT Edge Deployment Manifest]**

- Confirm **[Deployment.json]** is generated under config folder


deployment.template.json - Custom-vision-service-iot-edge-raspberry-pi - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
deployment.template.json
CUSTOM-VISION-SERVICE-IOT...
.github
assets
config
deployment.json
modules
.env
.gitignore
CHANGELOG.md
CONTRIBUTING.md
deployment.template.json
deployment.template.test-amd64
LICENSE.md
README.md
OUTLINE
AZURE IOT HUB DEVICES
Open to the Side Ctrl+Enter
Reveal in Explorer Alt+Shift+R
Open in Command Prompt
Select for Compare
Copy Ctrl+C
Copy Path Alt+Shift+C
Rename F2
Delete Del
Add IoT Edge Module
Build IoT Edge Solution
Build and Push IoT Edge Solution
Build and Run IoT Edge Solution in Simulator
Generate IoT Edge Deployment Manifest
master*
Ln 1, Col 1

```
{
  "moduleContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.0",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion": "v
            "loggingOptions": "",
            "registryCredentials":
              "registryName": {
                "username": "$CONT
                "password": "$CONT
```

Step 5 : Login to ACR

In order to publish container to ACR, you need to login to ACR.

- Open Terminal Window in VSCode
  [View] -> **[Integrated Terminal]**
- In the terminal window, type following command to login

```
docker login --password <ACR Password> --username <ACR User Name> <ACR FQDN>
```

Example:

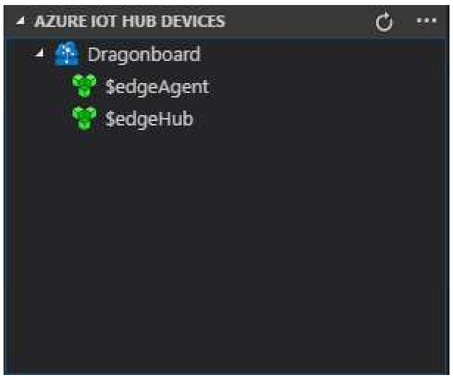
```
PS C:\> docker login --password iYJfdc55JUTHgJX4dspk/dmQT8=VCoWW --username mysampleacr mysampleacr.azurecr.io
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS C:\>
```

> [!TIP]
> Please make sure Docker is running in your Development Machine.

Step 6 : Connect to IoT Hub in VSCode

Sign in and connect to your IoT Hub.

> [!TIP]
> Make sure you can see your IoT Edge devices in VSCode

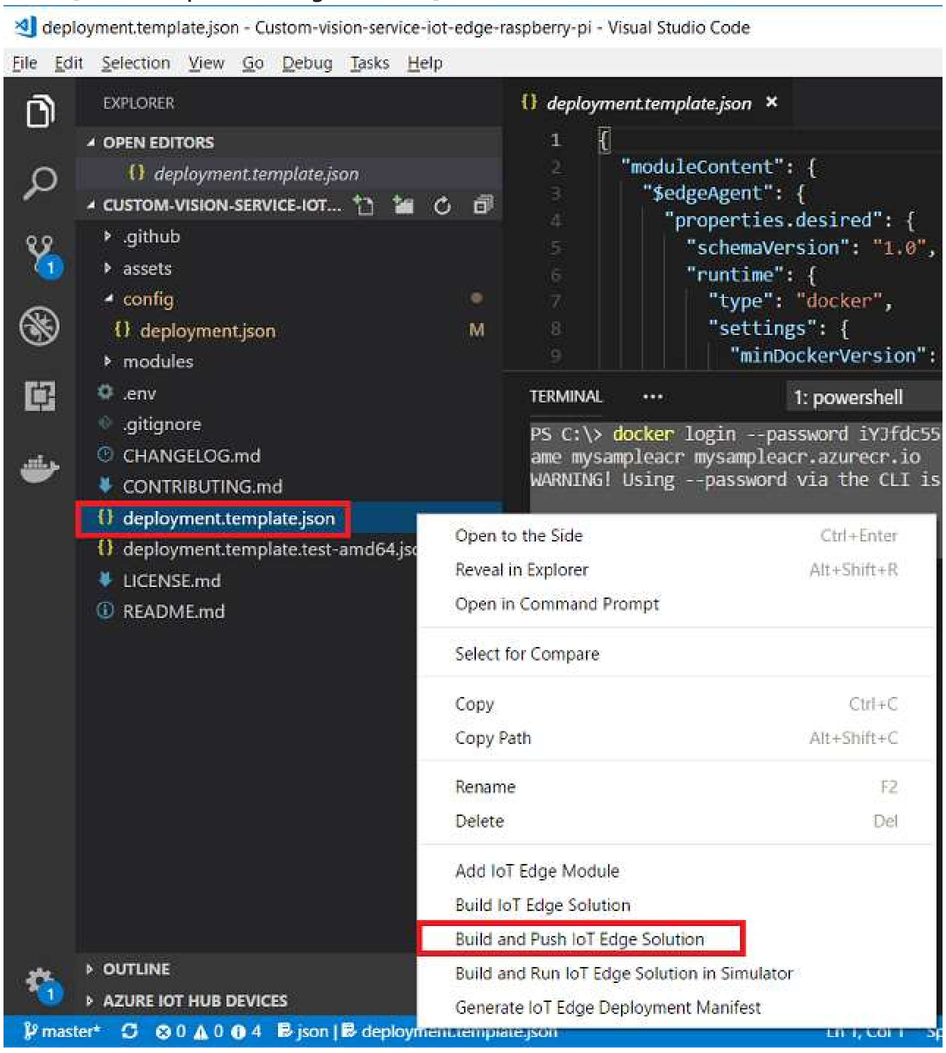


Please refer to this for detailed instruction.

Step 7 : Build and Push container to ACR

You can now compile and upload the Azure IoT Edge module containers to your ACR

- Right click **[deployment.template.json]** in left pane
- Select **[Build and push IoT Edge Solution]**


deployment.template.json - Custom-vision-service-iot-edge-raspberry-pi - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
deployment.template.json
CUSTOM-VISION-SERVICE-IOT...
.github
assets
config
deployment.json M
modules
.env
.gitignore
CHANGELOG.md
CONTRIBUTING.md
deployment.template.json
deployment.template.test-amd64.js
LICENSE.md
README.md
OUTLINE
AZURE IOT HUB DEVICES
TERMINAL
1: powershell
Open to the Side Ctrl+Enter
Reveal in Explorer Alt+Shift+R
Open in Command Prompt
Select for Compare
Copy Ctrl+C
Copy Path Alt+Shift+C
Rename F2
Delete Del
Add IoT Edge Module
Build IoT Edge Solution
Build and Push IoT Edge Solution
Build and Run IoT Edge Solution in Simulator
Generate IoT Edge Deployment Manifest
master* ⊗0 ⚠0 ❶4 json deployment.template.json

```
{
  "moduleContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.0",
        "runtime": {
          "type": "docker",
          "settings": {
            "minDockerVersion":
```

```
PS C:\> docker login --password iYJfdc55
ame mysampleacr mysampleacr.azurecr.io
WARNING! Using --password via the CLI is
```

> [!WARNING]
> The initial build process can take hours depending on your development machine's performance.

> [!TIP]
> For the HOL, sample is pre-built to save time.

Step 8 : Deploy to the IoT Edge Device

Once build and push operation finished, you can deploy the solution to your IoT Edge device.

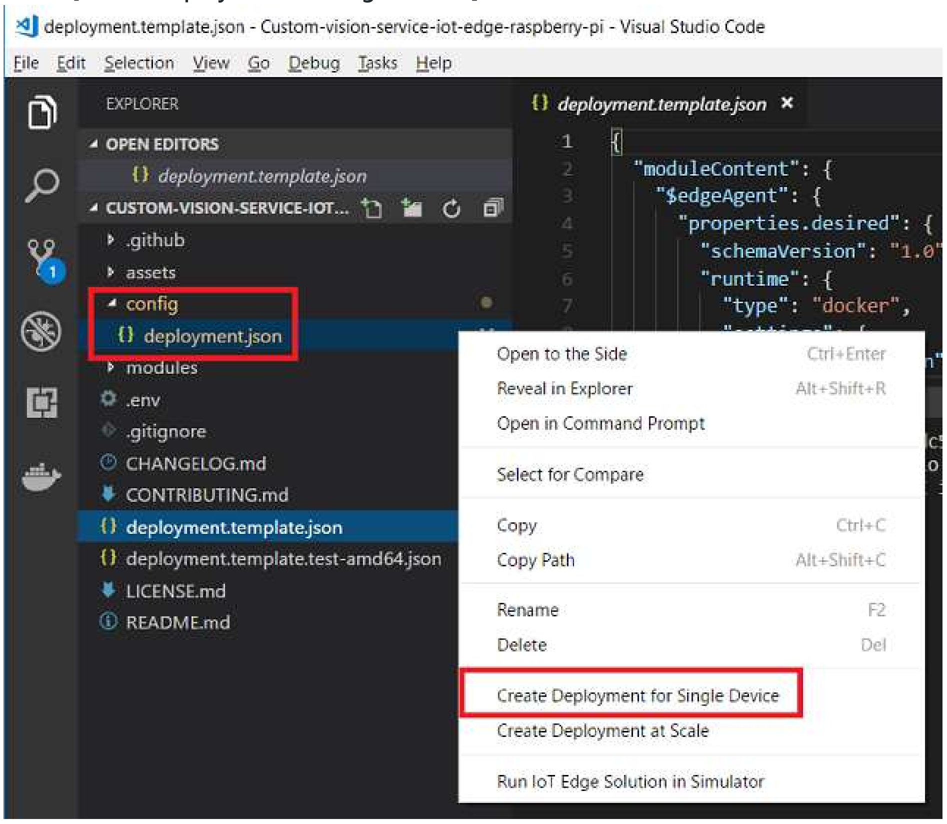
There are two ways to deploy

1. Deploy to a single device
   This will deploy to one device.

2. Deploy to multiple device With this option, you can deploy to multiple devices based on pre-configured conditions such as tag.

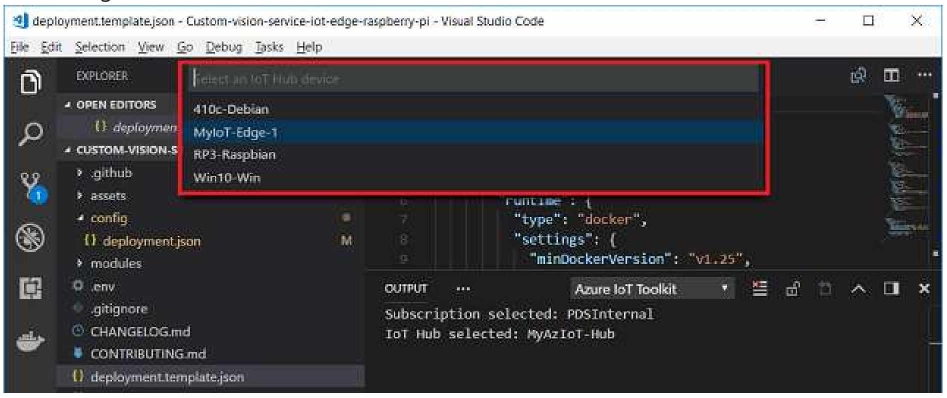
> [!NOTE] For the HOL, please select Single Device Deployment

- Expand **[config]** folder
- Right click on **[deployment.json]** file
- Select **[Create Deployment for Single Device]**


deployment.template.json - Custom-vision-service-iot-edge-raspberry-pi - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
deployment.template.json
CUSTOM-VISION-SERVICE-IOT...
.github
assets
config
deployment.json
modules
.env
.gitignore
CHANGELOG.md
CONTRIBUTING.md
deployment.template.json
deployment.template.test-amd64.json
LICENSE.md
README.md
Open to the Side Ctrl+Enter
Reveal in Explorer Alt+Shift+R
Open in Command Prompt
Select for Compare
Copy Ctrl+C
Copy Path Alt+Shift+C
Rename F2
Delete Del
Create Deployment for Single Device
Create Deployment at Scale
Run IoT Edge Solution in Simulator

```
{
  "moduleContent": {
    "$edgeAgent": {
      "properties.desired": {
        "schemaVersion": "1.0"
        "runtime": {
          "type": "docker",
```

- Select target device


deployment.template.json - Custom-vision-service-iot-edge-raspberry-pi - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
Select an IoT Hub device
410c-Debian
MyIoT-Edge-1
RP3-Raspbian
Win10-Win
OUTPUT
Azure IoT Toolkit

```
"runtime": {
  "type": "docker",
  "settings": {
    "minDockerVersion": "v1.25",
```

```
Subscription selected: PDSInternal
IoT Hub selected: MyAzIoT-Hub
```

Step 9 : Verify successful deployment

Once modules are deployed to your IoT Edge device, confirm the state of modules.

- Check status of modules

  ```
  # Check status of modules/containers
  # get module names
  sudo docker ps

  and/or

  sudo iotedge list
  ```

- Check logs from modules

  ```
  # Check logs from modules
  sudo docker logs -f <Module Name>

  and/or

  sudo iotedge logs <Module Name>
  ```

- Check telemetry data Start D2C message monitoring

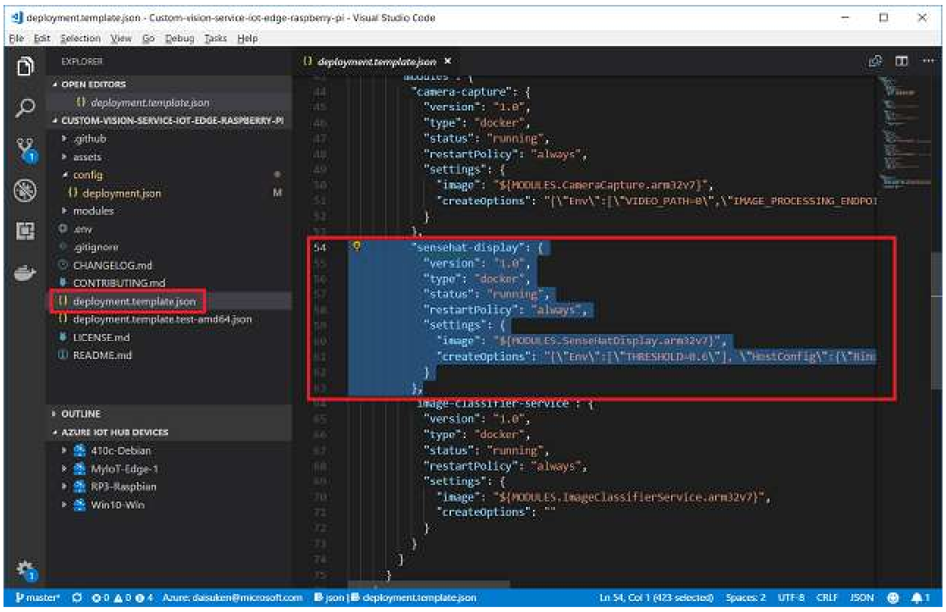
  > [!TIP]
  > imageClassificationService sends following telemetry data
  > Results: [{'Probability': 9.749999662744813e-06, 'Tag': 'Apple'}, {'Probability': 4.300000000512227e-06, 'Tag':'Banana'}] Predicting image Image size 256 x 256 crop_center: 256 x 256 to 227 x 227

Appendix

Removing SenseHat module

If you do not have SenseHat, please remove SenseHat module from the solution.

1. Open **[deployment.template.json]**
2. Remove lines for SenseHatModule (line 54~63)



```
"camera-capture": {
  "version": "1.0",
  "type": "docker",
  "status": "running",
  "restartPolicy": "always",
  "settings": {
    "image": "${MODULES.CameraCapture.arm32v7}",
    "createOptions": "{\"Env\":[\"VIDEO_PATH=0\",\"IMAGE_PROCESSING_ENDPOI
  }
},
"sensehat-display": {
  "version": "1.0",
  "type": "docker",
  "status": "running",
  "restartPolicy": "always",
  "settings": {
    "image": "${MODULES.SenseHatDisplay.arm32v7}",
    "createOptions": "{\"Env\":[\"THRESHOLD=0.6\"], \"HostConfig\":{\"Bin
  }
},
"image-classifier-service": {
  "version": "1.0",
  "type": "docker",
  "status": "running",
  "restartPolicy": "always",
  "settings": {
    "image": "${MODULES.ImageClassifierService.arm32v7}",
    "createOptions": ""
  }
}
```

Additional Resources

- [Custom Vision + Azure IoT Edge on a Raspberry Pi 3]()