

Guide to deploying IoT-Sistem-Za-Kontrolu-Ulaza solution

Stokić Dušan

September 2021.

Contents

1	Introduction	3
2	Raspberry Pi device setup	3
2.1	Installing Raspberry Pi OS	3
2.2	Raspberry Pi quick start	5
3	Create Azure resources	6
3.1	Create an IoT hub	6
3.1.1	Register an IoT Edge device in IoT Hub	7
3.2	Create a Storage Account for Table Storage	9
3.3	Create a Stream Analytics job to Save Data in Table Storage	9
3.3.1	Configure job input	10
3.3.2	Configure job output	10
3.3.3	Define the transformation query	11
3.4	Create a Container Registry	12

4	IoT Edge on Raspberry Pi	13
4.1	SSH into your Raspberry Pi	13
4.2	Install IoT Edge runtime on Raspberry Pi	14
4.3	Provision the device with its cloud identity	15
5	Connect the sensors and actuators to your Device	18
6	Update the AI model	21
6.1	Choose training images	22
6.2	Upload and tag images	22
6.3	Train and export the classifier	23
7	Build and deploy your solution	23
7.1	Prepare the development environment	23
7.2	Deploying the solution	25
8	Provide Feedback	26

1 Introduction

This is a detailed guide on deploying the solution in the IoT-Sistem-Za-Kontrolu-Ulaza repository¹.

In this project you will learn how to use Microsoft Azure IoT Edge on a Raspberry Pi Model 3 B as an edge device with sensors attached to it. This project can be used as a base for larger Azure IoT Edge projects or as a Proof of Concept for Azure IoT Edge with real sensors.

This guide will cover setting up Azure services, installing the latest Raspian OS on the Raspberry Pi device and configuring the device as a IoT Edge device, connecting sensors and actuators and, finally, building and deploying the solution on your Edge device.

We are using a fresh installation of Raspberry Pi OS as the operating system but it might work with a not so fresh installation of Raspberry Pi. You can follow the Raspberry Pi OS installation guide in the next section.

2 Raspberry Pi device setup

This section will cover installing Raspberry pi OS (previous Raspbian) on your Raspberry Pi. After completing this section you will have a Raspberry Pi device up and running.

2.1 Installing Raspberry Pi OS

To get started with your Raspberry Pi computer you'll need A computer monitor, or television and a computer keyboard and mouse. Most monitors should work as a display for the Raspberry Pi, but for best results, you should use a display with HDMI input. You'll also need a appropriate display cable, to connect your monitor to your Raspberry Pi.

Finally you'll need an SD card. It is recommend to use a micro SD card with a minimum of 8GB, and to use an Imager to install an operating system onto it.

¹<https://github.com/StokicDusan/IoT-Sistem-Za-Kontrolu-Ulaza>

Raspberry Pi recommend the use of Raspberry Pi Imager² to install an operating system on your SD card. You will need another computer with an SD card reader to install the image. After installing the imager:

- Connect an SD card reader with the SD card inside.
- Open Raspberry Pi Imager and choose the recommended Raspberry Pi OS (32-bit) from the list presented.
- Choose the SD card you wish to write your image to.

After placing the SD card in your Raspberry Pi device, start it and begin the setup. Upon booting the device, you will be met with the welcome screen like in figure 1 and start the setup.

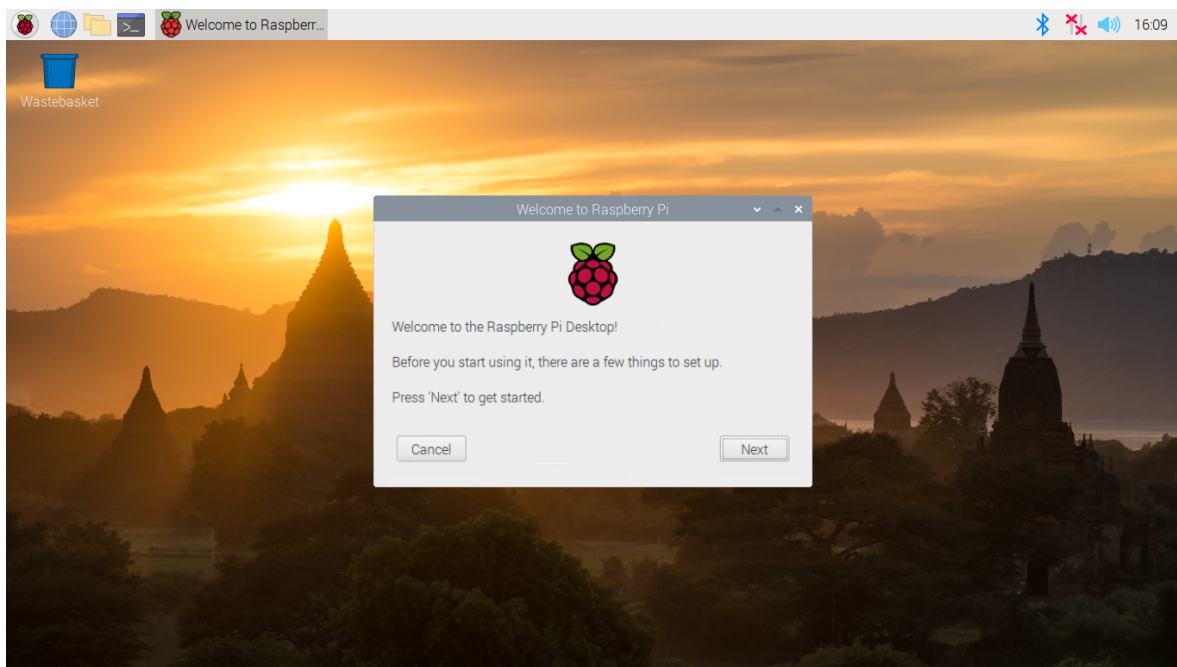


Figure 1: Welcome screen after booting your Raspberry pi device for the first time

²<https://www.raspberrypi.org/software/>

2.2 Raspberry Pi quick start

Now we have all the hardware and software in place and ready to start. Booting your Raspberry pi device for the first time, you can go through initial setup of the device. Connect your device to a Wi-Fi network and be sure to change the default password for your device. It is also recommended to update your software.

After completing the setup open the terminal and type:

```
$ sudo raspi-setup
```

This command should open the software configuration tool like in figure 2.

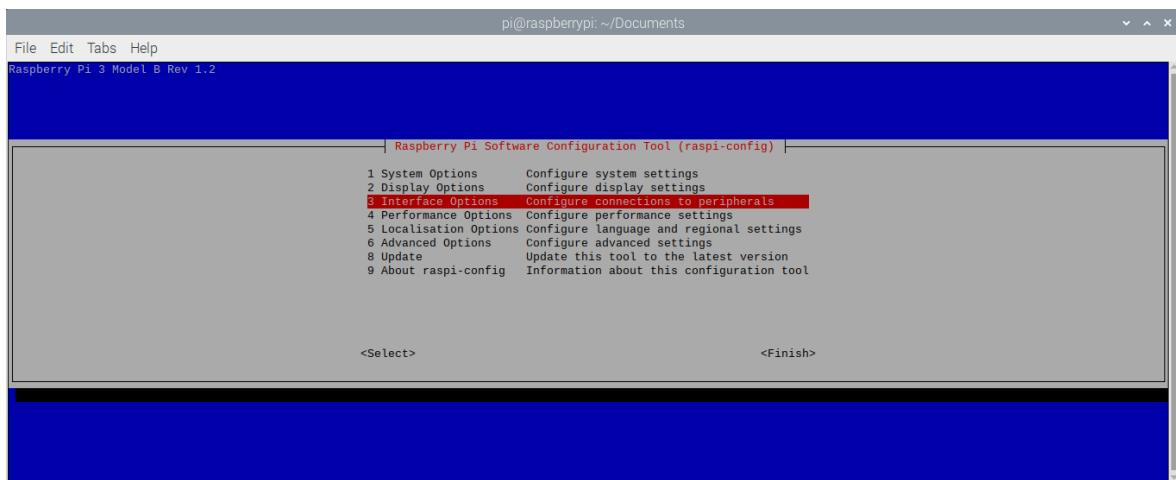


Figure 2: Raspberry pi software configuration tool

From here you can enable the camera interface by navigating through **Interface Options > Camera** and choose **Enable**. Do the same for **SSH** and **I2C**.

From now on you can access the Raspberry Pi remotely. I am using Putty³ as the SSH utility to connect to the Raspberry Pi shell. Returning back to the terminal you can issue the

```
$ ifconfig
```

command to see the IP address of your Raspberry Pi. You will connect to this IP address in Putty to connect to the Raspberry Pi (if you have not changed the default password you can use pi as the login name and raspberry as the password.)

³<https://www.putty.org/>

3 Create Azure resources

IoT Edge runtime, we will be installing on Raspberry Pi in the next section, will communicate to an IoT Hub in Azure. This IoT Hub needs to be created and initialized in Azure platform as well as other resources needed for this solution. It is possible to create these resources on the Azure portal or through command line tool package, Azure CLI, provided by Microsoft. Here we will use the Azure portal.

3.1 Create an IoT hub

This section describes how to create an IoT Hub using the Azure portal:

1. Sign in to the Azure portal
2. From the Azure homepage, select the + **Create a resource** button, and then enter *IoT Hub* in the **Search the Marketplace** field,
3. Select **IoT Hub** from the search results, and then select **Create**.
4. On the **Basics** tab, complete the fields as follows:
 - **Resource Group:** Select a resource group or create a new one you will use for this project. To create a new one, select **Create new** and fill in the name you want to use. To use an existing resource group, select that resource group.
 - **Region:** Select the region in which you want your hub to be located. Select the location closest to you.
 - **IoT Hub Name:** Enter a name for your hub. This name must be globally unique, with a length between 3 and 50 alphanumeric characters. The name can also include the dash ('-') character.
(Because the IoT hub will be publicly discoverable as a DNS endpoint, be sure to avoid entering any sensitive or personally identifiable information when you name it.)
5. Select **Next: Networking** to continue creating your hub.

Choose the endpoints that devices can use to connect to your IoT Hub. You can select the default setting **Public endpoint (all networks)**, or choose **Public endpoint (selected IP ranges)**, or **Private endpoint**.

Accept the default setting for this project.

6. Select **Next: Management** to continue creating your hub.

- **Pricing and scale tier:** Your selected tier. You can choose from several tiers, depending on how many features you want and how many messages you send through your solution per day. The IoT Edge feature we need for this project is included in the Standard tier. For this project you can use the Free Standard Tier.
- **IoT Hub units:** For this project we need only one unit.
- **Defender for IoT:** Turn this on to add an extra layer of threat protection to IoT and your devices. This option is not available for hubs in the free tier. If you've chosen one of the paid tiers, can leave this option off.

7. Select **Next: Tags** to continue to the next screen.

Tags are name/value pairs. You can assign the same tag to multiple resources and resource groups to categorize resources and consolidate billing. In this project, you won't be adding any tags.

8. Select **Next: Review + create** to review your choices. You see something similar to this screen, but with the values you selected when creating the hub.

9. Select **Create** to start the deployment of your new hub. Your deployment will be in progress a few minutes while the hub is being created. Once the deployment is complete, click **Go to resource** to open the new hub.

3.1.1 Register an IoT Edge device in IoT Hub

This subsection provides the steps to register a new IoT Edge device in your IoT Hub.

In your IoT hub in the Azure portal, IoT Edge devices are created and managed separately from IoT devices that are not edge enabled.

- Start by navigating to your IoT Hub.
- In the left pane, select **IoT Edge** from the menu, then select **Add an IoT Edge device**.
- On the **Create a device** page, provide the following information:
 - Create a descriptive device ID.
 - Select **Symmetric key** as the authentication type.

- Use the default settings to auto-generate authentication keys and connect the new device to your hub.
- Select Save.

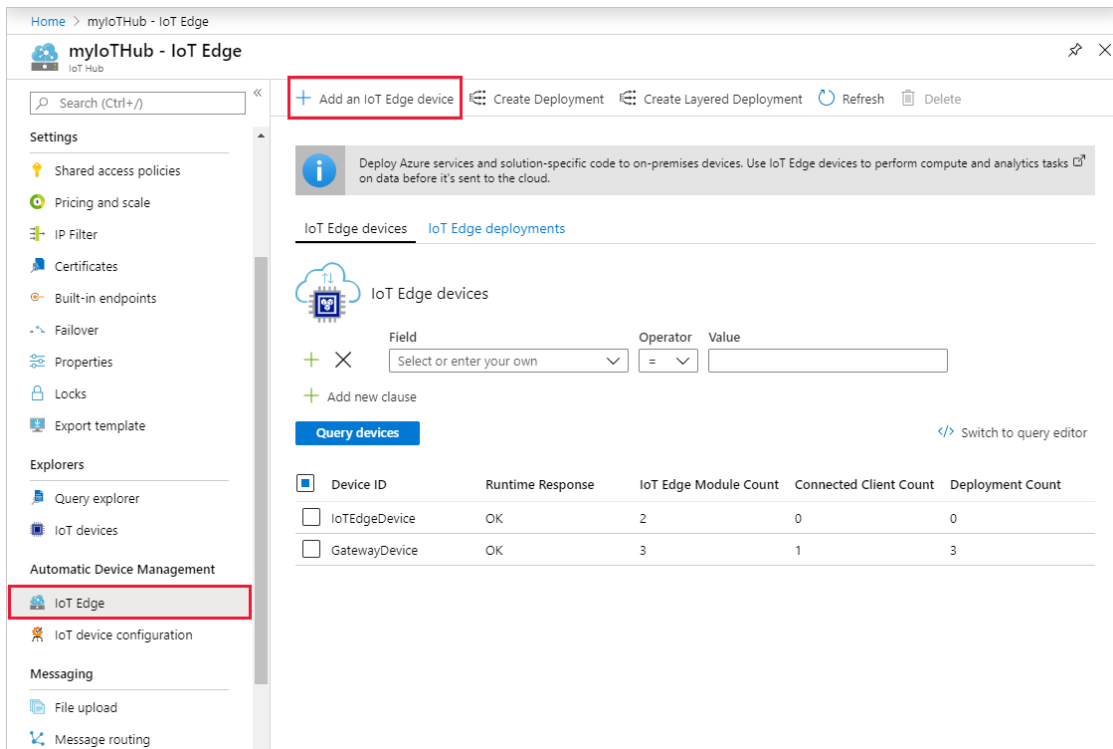


Figure 3: Add IoT Edge Device

Now that you have a device registered in IoT Hub, retrieve the *Primary Connection String* which you will use to complete installation and provisioning of the IoT Edge runtime.

Devices that authenticate with symmetric keys have their connection strings available to copy in the portal:

- From the **IoT Edge** page in the portal, click on the device ID from the list of IoT Edge devices.
- Copy the value of **Primary Connection String**.

3.2 Create a Storage Account for Table Storage

An Azure storage account contains all of your Azure Storage data objects: blobs, file shares, queues, tables, and disks. The storage account provides a unique namespace for your Azure Storage data that's accessible from anywhere in the world over HTTP or HTTPS. Data in your storage account is durable and highly available, secure, and massively scalable.

We will use our Storage Account to store data sent from the Edge device.

The following steps describe how to create a Storage Account for Table Storage using the Azure portal:

- From the upper left-hand corner of the Azure portal, select **Create a resource** > **Storage** > **Storage account**.
- Choose **Classic** for the deployment model and click on **Create**.
- Enter the name of your choice for the account name, Standard for the type, choose your subscription, select the resource group you created earlier. Choose the same location and resource group as the IoT Hub you created. Then click on **Review** + **Create** to create the account.

Once the account is created, find it in the **resources blade**, go to **Settings**, **Keys**, and write down the *primary connection string*.

3.3 Create a Stream Analytics job to Save Data in Table Storage

Stream Analytics is an Azure IoT service that streams and analyzes data in the cloud. We'll use it to process data coming from your device.

- Select **Create a resource** in the upper left-hand corner of the Azure portal.
- Select **Analytics** > **Stream Analytics job** from the results list.
- Fill out the Stream Analytics job page. Enter a name for the job, a preferred region, then choose your subscription. At this stage you are also offered to create a new or to use an existing resource group. Choose the resource group you created earlier.

3.3.1 Configure job input

In this section, you will configure an IoT Hub device input to the Stream Analytics job. Use the IoT Hub you created in the previous section.

- Navigate to your Stream Analytics job.
- Select **Inputs > Add Stream input > IoT Hub**.
- Fill out the **IoT Hub** page with the following values:
 - **Input Alias**: name your input alias (in my example i'm using "OdlukaInput")
 - **Source Type**: Data Stream
 - **Source**: IoT Hub
 - **Subscription**: Choose your subscription
 - **IoT Hub**: use the name for the IoT Hub you create before
 - **Shared Access Policy Name**: Shared access policies: iothubowner
 - **Shared Access Policy Key**: The iothubowner primary key can be found in your IoT Hub **Settings > Shared access policies**
 - IoT Hub Consumer Group: leave it to the default value
 - Event serialization format: JSON
 - Encoding: UTF-8

3.3.2 Configure job output

Navigate to the Stream Analytics job that you created earlier. Click on the **Outputs** tile and in the **Outputs blade**, click on **Add > Table storage**. Enter the following settings then click on **Create**:

- **Output Alias**: OdlukaTable
- **Sink**: Table Storage
- **Storage account**: The storage you made earlier
- **Storage account key**: The primary key from the storage account you made earlier (*can be found in Settings > Keys > Primary Access Key*)

- **Table Name:** TableOdluka1 (You can choose what ever name you want - If the table doesn't already exist, Local Storage will create it)
- **Partition Key:** deviceId
- **Row Key:** created
- **Batch size:** 1

3.3.3 Define the transformation query

Navigate to the Stream Analytics job that you created earlier. Click on the **Query tile** (next to the Inputs tile). In the Query settings blade, type in the below query and click **Save**:

Listing 1: Stream Analytics Job Query

```
SELECT
    prediction.ArrayValue.tagName AS Name,
    prediction.ArrayValue.probability AS Probability,
    OdlukaInput.created,
    OdlukaInput.IoTHub.ConnectionDeviceId AS DeviceID
INTO
    OdlukaTable
FROM
    OdlukaInput TIMESTAMP BY OdlukaInput.created
CROSS APPLY
    GetArrayElements(OdlukaInput.prediction) AS prediction
WHERE
    prediction.ArrayValue.probability > 0.8
```

Here, my input alias is "OdlukaInput" and my output alias is "OdlukaTable." If you've chosen other input and output aliases, use those names in the query instead.

Back in the Stream Analytics blade, start the job by clicking on the **Start** button at the top.

Note: You are charged for this resource for as long as the job is running so be sure to *Stop* every Stream Analytics Job when you're not using it!

3.4 Create a Container Registry

Azure Container Registry is a private registry service for building, storing, and managing container images and related artifacts. In this project, you create an Azure container registry instance with the Azure portal. Later we will push container images we need for our project into the registry, and finally pull and run the image from your registry onto the device.

Follow the next steps to create a container registry:

- Select **Create a resource** in the upper left-hand corner of the Azure portal.
- Select **Containers > Container Registry**.
- On the **Basics** tab, complete the fields as follows:
 - **Subscription:** Select your subscription. If you have multiple, select the one you've been using for other resources.
 - **Resource Group:** Choose the resource group you created earlier.
 - **Location:** Select the location closest to you or the one you used for other resources like your IoT Hub.
 - **Registry name:** You can choose whatever name you want. The registry name must be unique within Azure, and contain 5-50 alphanumeric characters.
 - **SKU:** Select 'Basic'.

Accept default values for the remaining settings. Then select **Review + create**. After reviewing the settings, select **Create**.

When the **Deployment succeeded** message appears, select the container registry in the portal.

Take note of the registry name and the value of the **Login server**, which is a fully qualified name ending with *azurecr.io* in the Azure cloud. You will use these values in the later steps when you push and pull the images from the container.

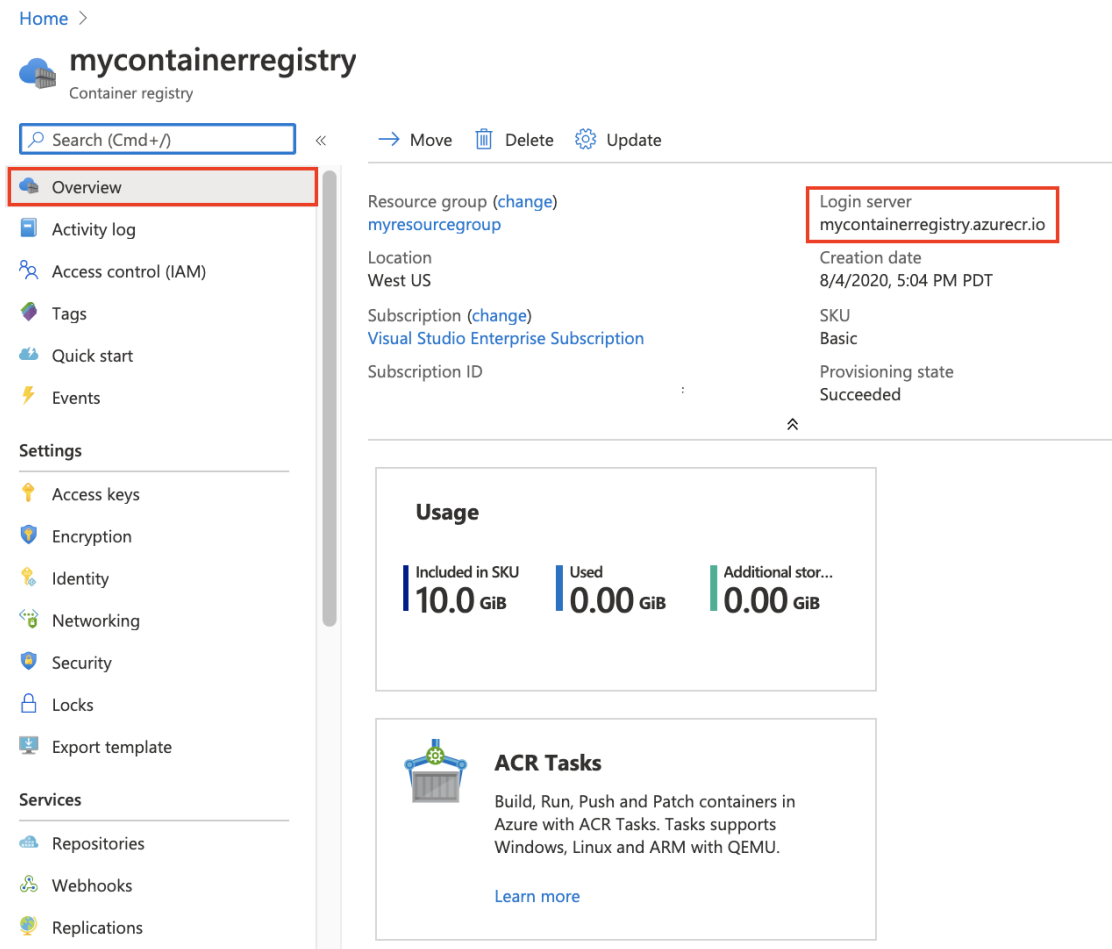


Figure 4: Container Registry Overview

4 IoT Edge on Raspberry Pi

This section will cover the installation of Azure IoT Edge runtime on your Raspberry Pi device as well as connecting the Edge device to the IoT Hub in Azure.

4.1 SSH into your Raspberry Pi

Now that you have enabled SSH and found out your IP address you can go ahead and SSH into your Raspberry Pi from any other computer. You'll also need the username and the password for the Raspberry Pi.

Default Username and Password are:

- username: pi
- password: raspberry

If you have changed the default password then use the new password instead of the above.

On a successful login you'll be presented with the terminal of your Raspberry Pi. Now you can run any commands on your Raspberry Pi through this terminal remotely (within the current network) without having to access your Raspberry Pi physically.

4.2 Install IoT Edge runtime on Raspberry Pi

The Azure IoT Edge runtime is what turns a device into an IoT Edge device. Once a device is configured with the IoT Edge runtime, you can start deploying business logic to it from the cloud. This section lists the steps to install the Azure IoT Edge runtime on your Raspberry Pi device.

Enter the following command into the Raspberry pi terminal you have opened:

```
curl https://packages.microsoft.com/config/debian/stretch\
/multiarch/prod.list > ./microsoft-prod.list

sudo cp ./microsoft-prod.list /etc/apt/sources.list.d/

curl https://packages.microsoft.com/keys/microsoft.asc | \
gpg --dearmor > microsoft.gpg

sudo cp ./microsoft.gpg /etc/apt/trusted.gpg.d/
```

Now you have prepared your device to access the Microsoft installation packages.

With the following commands you will have a container engine needed for Azure IoT Edge runtime:

```
sudo apt-get update
sudo apt-get install moby-engine
```

Now we have all we need to install IoT Edge on your device:

```
sudo apt-get install iotedge=1.1* libiothsm-std=1.1*
```

The IoT Edge security daemon provides and maintains security standards on the IoT Edge device. The daemon starts on every boot and bootstraps the device by starting the rest of the IoT Edge runtime.

4.3 Provision the device with its cloud identity

Now that the container engine and the IoT Edge runtime are installed on your device, you're ready for the next step, which is to set up the device with its cloud identity and authentication information.

On the IoT Edge device, open the configuration file:

```
sudo nano /etc/iotedge/config.yaml
```

Find the provisioning configurations of the file and uncomment the **Manual provisioning configuration using a connection string** section, if it isn't already uncommented.

```
# Manual provisioning configuration using a connection string
provisioning:
  source: "manual"
  device_connection_string: "<ADD DEVICE CONNECTION STRING HERE>"
```

Update the value of **device_connection_string** with the primary connection string from your IoT Edge device in the cloud. Make sure that any other provisioning sections are commented out. Make sure the **provisioning:** line has no preceding whitespace and that nested items are indented by two spaces. (*To paste clipboard contents into Nano Shift+Right Click or press Shift+Insert.*)

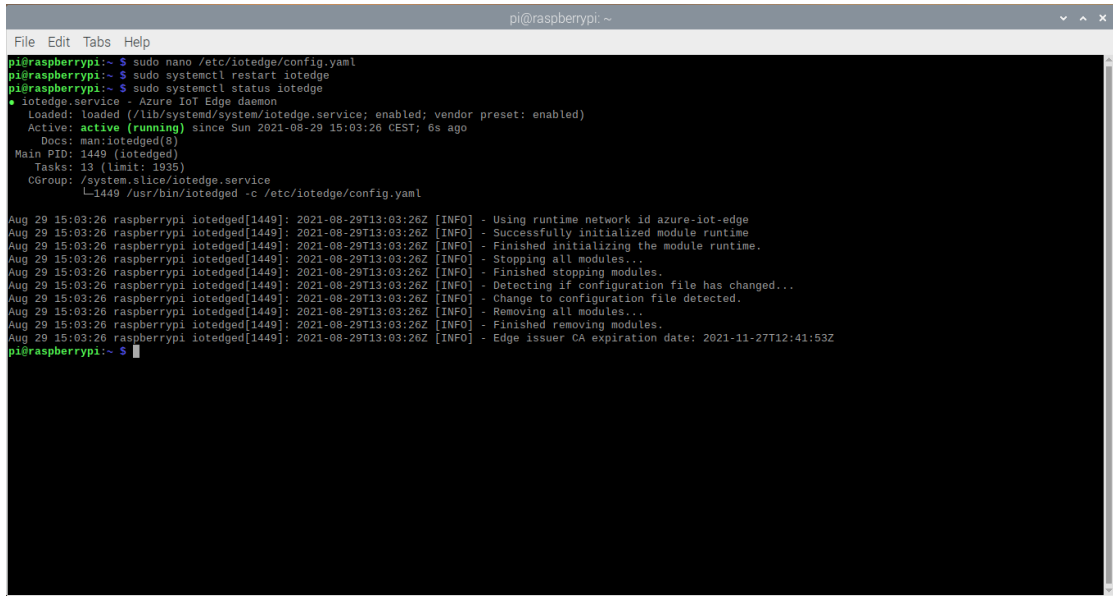
Save and close the file. (*CTRL + X, Y, Enter*)

After entering the provisioning information in the configuration file, restart the daemon:

```
sudo systemctl restart iotedge
```

Verify that the runtime was successfully installed and configured on your IoT Edge device with:

```
sudo systemctl status iotedge
```



```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo nano /etc/iotedge/config.yaml  
pi@raspberrypi:~$ sudo systemctl restart iotedge  
pi@raspberrypi:~$ 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 Sun 2021-08-29 15:03:26 CEST; 6s ago  
     Docs: man:iotedged(8)  
    Main PID: 1449 (iotedged)  
      Tasks: 13 (limit: 1935)  
    CGroup: /system.slice/iotedge.service  
            └─1449 /usr/bin/iotedged -c /etc/iotedge/config.yaml  
  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Using runtime network id azure-iot-edge  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Successfully initialized module runtime  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Finished initializing the module runtime.  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Stopping all modules...  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Finished stopping modules.  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Detecting if configuration file has changed...  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Change to configuration file detected.  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Removing all modules...  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Finished removing modules.  
Aug 29 15:03:26 raspberrypi iotedged[1449]: 2021-08-29T13:03:26Z [INFO] - Edge issuer CA expiration date: 2021-11-27T12:41:53Z  
pi@raspberrypi:~$
```

Figure 5: systemctl status iotedge

Use the check tool to verify configuration and connection status of the device. You should get something like in figure 6.

```
sudo iotedge check
```

View all the modules running on your IoT Edge device with the list tool. When the service starts for the first time, you should only see the **edgeAgent** module running. The edgeAgent module runs by default and helps to install and start any additional modules that you deploy to your device.

```
sudo iotedge list
```

When you create a new IoT Edge device, it will display the status code 417 – The device's deployment configuration is not set in the Azure portal. This status is normal, and means that the device is ready to receive a module deployment.


```
pi@Rpi-etf-projekat: ~  
pi@Rpi-etf-projekat:~ $ sudo iotedge check  
Configuration checks  
-----  
✓ config.yaml is well-formed - OK  
✓ config.yaml has well-formed connection string - OK  
✓ container engine is installed and functional - OK  
✓ config.yaml has correct hostname - OK  
✓ config.yaml has correct URIs for daemon mgmt endpoint - OK  
✓ latest security daemon - OK  
✓ host time is close to real time - OK  
✓ container time is close to host time - OK  
!! DNS server - Warning  
    Container engine is not configured with DNS server setting, which may impact connectivity to IoT Hub.  
    Please see https://aka.ms/iotedge-prod-checklist-dns for best practices.  
    You can ignore this warning if you are setting DNS server per module in the Edge deployment.  
!! production readiness: certificates - Warning  
    The Edge device is using self-signed automatically-generated development certificates.  
    They will expire in 89 days (at 2020-11-30 20:13:58 UTC) causing module-to-module and downstream device  
    communication to fail on an active deployment.  
    After the certs have expired, restarting the IoT Edge daemon will trigger it to generate new developmen  
t certs.  
    Please consider using production certificates instead. See https://aka.ms/iotedge-prod-checklist-certs  
for best practices.  
✓ production readiness: container engine - OK  
!! production readiness: logs policy - Warning  
    Container engine is not configured to rotate module logs which may cause it run out of disk space.  
    Please see https://aka.ms/iotedge-prod-checklist-logs for best practices.  
    You can ignore this warning if you are setting log policy per module in the Edge deployment.  
!! production readiness: Edge Agent's storage directory is persisted on the host filesystem - Warning  
    The edgeAgent module is not configured to persist its /tmp/edgeAgent directory on the host filesystem.  
    Data might be lost if the module is deleted or updated.  
    Please see https://aka.ms/iotedge-storage-host for best practices.  
* production readiness: Edge Hub's storage directory is persisted on the host filesystem - Error  
    Could not check current state of edgeHub container  
  
Connectivity checks  
-----  
✓ host can connect to and perform TLS handshake with IoT Hub AMQP port - OK  
✓ host can connect to and perform TLS handshake with IoT Hub HTTPS / WebSockets port - OK  
✓ host can connect to and perform TLS handshake with IoT Hub MQTT port - OK  
✓ container on the default network can connect to IoT Hub AMQP port - OK  
✓ container on the default network can connect to IoT Hub HTTPS / WebSockets port - OK  
✓ container on the default network can connect to IoT Hub MQTT port - OK  
✓ container on the IoT Edge module network can connect to IoT Hub AMQP port - OK  
✓ container on the IoT Edge module network can connect to IoT Hub HTTPS / WebSockets port - OK  
✓ container on the IoT Edge module network can connect to IoT Hub MQTT port - OK  
  
18 check(s) succeeded.  
4 check(s) raised warnings. Re-run with --verbose for more details.  
1 check(s) raised errors. Re-run with --verbose for more details.  
  
pi@Rpi-etf-projekat:~ $ sudo iotedge list  
NAME          STATUS      DESCRIPTION          CONFIG  
edgeAgent     running     Up 34 minutes        mcr.microsoft.com/azureiotedge-agent:1.0  
pi@Rpi-etf-projekat:~ $
```

Figure 6: sudo iotedge check

5 Connect the sensors and actuators to your Device

To run this solution we need the following hardware:

- Raspberry Pi 3 with Azure IoT Edge set up,
- USB Camera connected on a Raspberry Pi 3 USB port (any),
- Electronics Components: Most of what you needed for this solution can be found in the SparkFun Inventor's Kit which includes:
 - Solderless Breadboard,
 - Ultrasonic Distance Sensor,
 - Relay module,
 - Jumper Wires,
 - Red, Green and Yellow LED,
 - 3x 220 Ω , 1x 1k Ω and 1x 2k Ω resistor.

Using the scheme seen in the figure 7 as a reference and the table below, connect all the hardware on the Solderless Breadboard and Raspberry pi device.

Note: Column on the left and middle corresponds to where the connection needs to be made and the column on the Right shows the the connector for that connection. On the image, the Breadboard pines are labeled alphabetically from bottom to top and numerically from left to right (starting from 1). On the Raspberry pi, table 1 is showing the lables for the physical pins, not their GPIO labels.

The short names used in Table 1 have next meanings:

- USS — HC-SR04 ultrasonic distance sensor
- board — Pins on the solderless breadboard
- Relay — Relay module
- Rpi pin — Physical pins on Raspberry Pi

Table 1: Wiring scheme

Start	End	Connector
USS(GND)	board(-)	Female/Male Jumper Wire
USS(Echo)	board(A2)	Female/Male Jumper Wire
USS(Trig)	board(A3)	Female/Male Jumper Wire
USS(Vcc)	board(+)	Female/Male Jumper Wire
board(E2)	board(G2)	$1k\Omega$ resistor
board(J2)	board(J6)	$2k\Omega$ resistor
board(F6)	board(-)	Male/Male Jumper Wire
board(I21)	board(I22)	Yellow LED (-/+)
board(I24)	board(I25)	Green LED (-/+)
board(I27)	board(I28)	Red LED (-/+)
board(G21)	board(E21)	330Ω resistor
board(G24)	board(E24)	330Ω resistor
board(G27)	board(E27)	330Ω resistor
board(A21)	board(-)	Male/Male Jumper Wire
board(A24)	board(-)	Male/Male Jumper Wire
board(A27)	board(-)	Male/Male Jumper Wire
Relay(+)	board(+)	Female/Male Jumper Wire
Relay(-)	board(-)	Female/Male Jumper Wire
Relay(s)	Rpi pin(18)	Female/Female Jumper Wire
Relay(NO)	board(J25)	Male/Male Jumper Wire
Relay(C)	Rpi pin(16)	Male/Male Jumper Wire
Relay(NC)	board(J28)	Male/Male Jumper Wire
Rpi pin(22)	board(J22)	Female/Male Jumper Wire
Rpi pin(14)	board(-)	Female/Male Jumper Wire
Rpi pin(13)	board(C3)	Female/Male Jumper Wire
Rpi pin(11)	board(H2)	Female/Male Jumper Wire
Rpi pin(4)	board(+)	Female/Male Jumper Wire

Connector "Yellow LED (-/+)" means that the **Start** column shows where to connect the cathode and the **End** where to connect the anode (the same applies for all three LED).

For a better understanding consult the scheme in Figure 7 shown below.

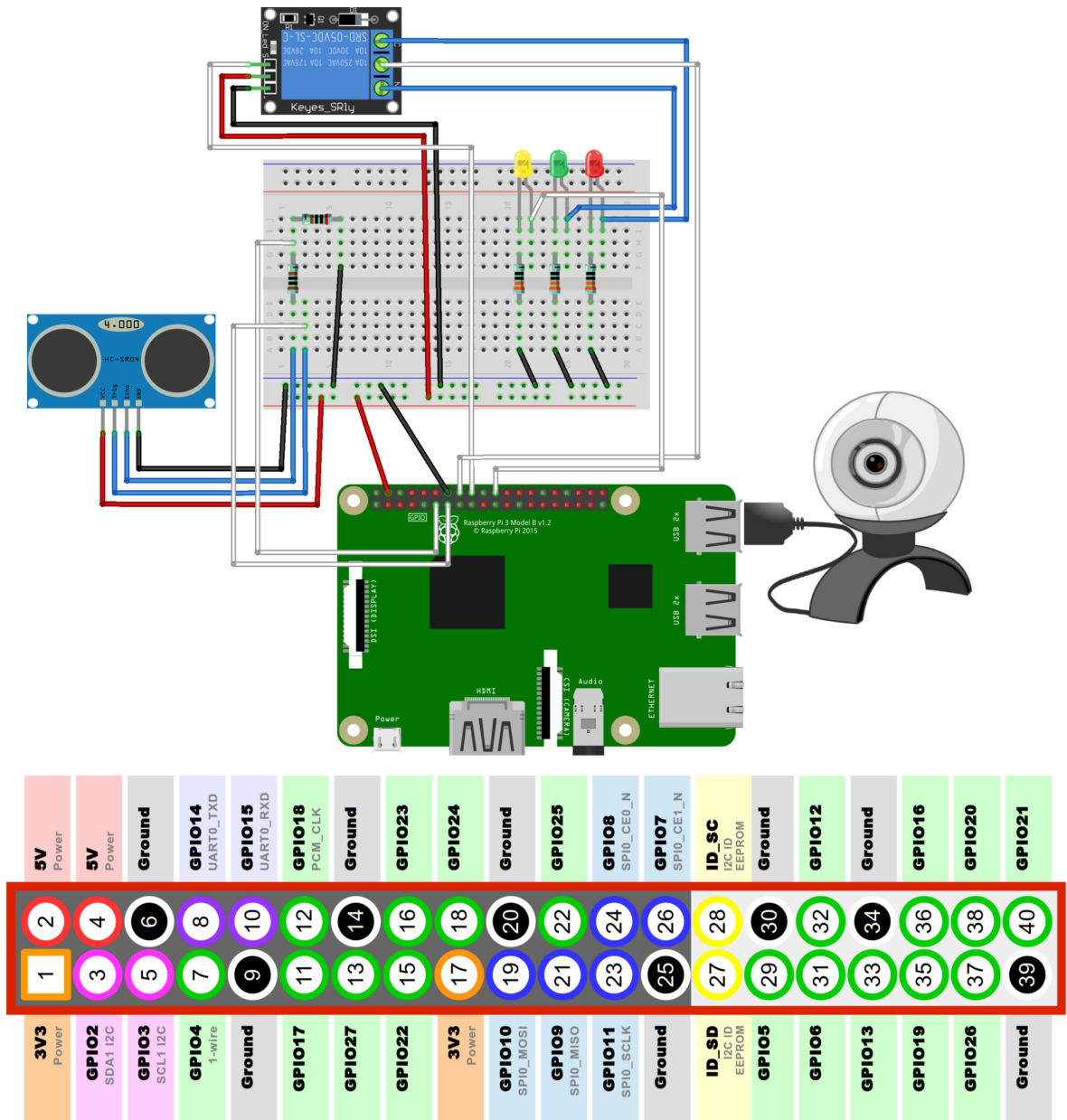


Figure 7: Hardware connected on Raspberry pi

6 Update the AI model

Before you deploy your solution, we could update the AI model so that it recognizes your face (instead of mine). If you wish to do so, in your web browser, navigate to the Custom Vision web page <https://www.customvision.ai/> and sign in with your Azure credentials. the welcome page is shown in figure 8. Follow the next steps:

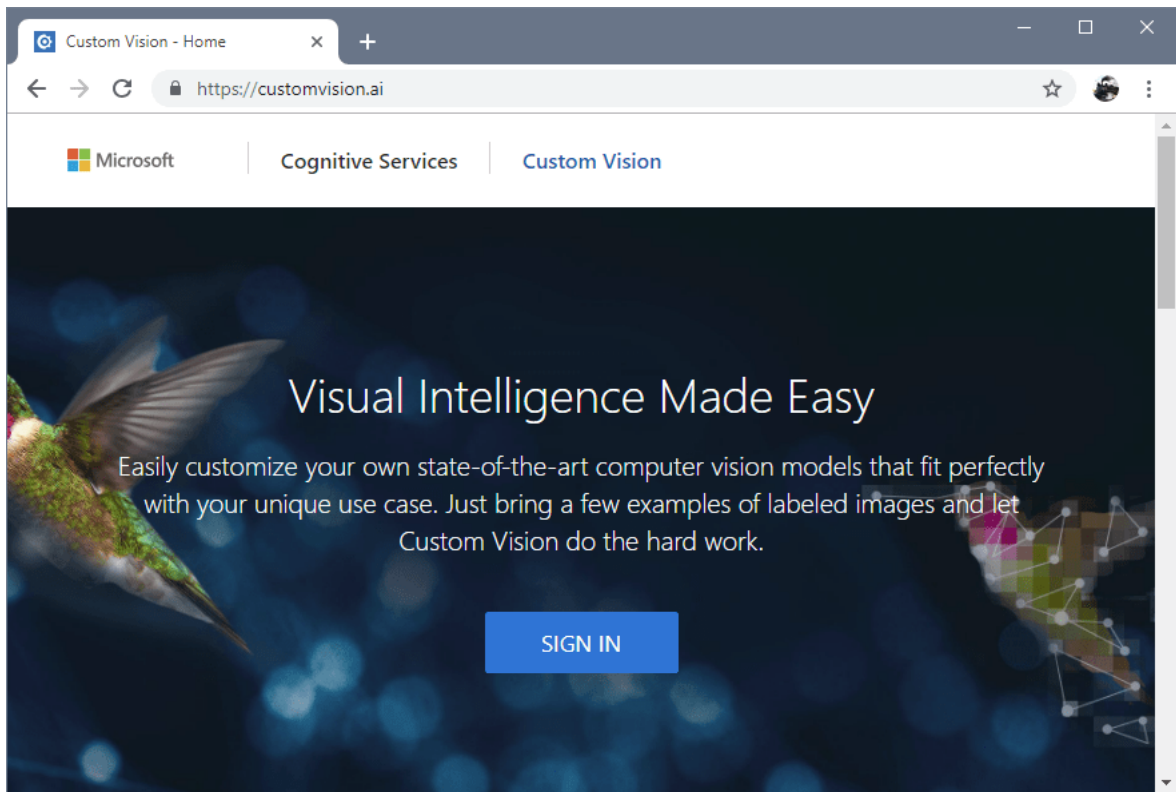


Figure 8: Custom vision web page

1. To create your first project, select **New Project**. The Create new project dialog box will appear.
2. Enter a name and a description for the project. Then select a Resource Group. If your signed-in account is associated with an Azure account, the Resource Group dropdown will display your Azure Resource Groups you created earlier.
3. Select **Classification** under Project Types. Then, under Classification Types, choose **Multiclass**. You'll be able to change the classification type later if you want to.

4. For Domains choose **General(compact)**
5. Export capabilities needs to be set to **Basic Platforms**
6. Finally, select **Create project**.

6.1 Choose training images

As a prerequisite you need a set of images with which to train your classifier, in our case, around 30 will suffice although more is better. It is recommended to use different lighting, background and camera angle in order to train your model effectively.

Additionally, make sure all of your training images meet the following criteria:

- .jpg, .png, .bmp, or .gif format
- no greater than 6MB in size (4MB for prediction images)
- no less than 256 pixels on the shortest edge; any images shorter than this will be automatically scaled up by the Custom Vision Service

6.2 Upload and tag images

In this subsection, you'll upload and manually tag images to help train the classifier.

1. To add images, select **Add images** and then select **Browse local files**. Select **Open** to move to tagging. Your tag selection will be applied to the entire group of images you've selected to upload, so it's easier to upload images in separate groups according to their applied tags. You can also change the tags for individual images after they've been uploaded.
2. To create a tag, enter text in the **My Tags** field and press Enter. If the tag already exists, it will appear in a dropdown menu. To finish uploading the images, use the **Upload [number] files** button.
3. Select **Done** once the images have been uploaded.

To upload another set of images, return to the top of this subsection and repeat the steps.

6.3 Train and export the classifier

To train the classifier, select the **Train** button. The classifier uses all of the current images to create a model that identifies the visual qualities of each tag.

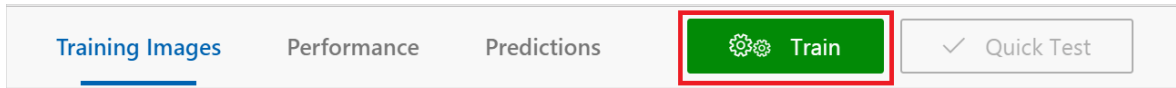


Figure 9: Train your classifier

The training process should only take a few minutes. During this time, information about the training process is displayed in the Performance tab.

In the Performance tab select the **Export** button and for the platform choose DockerFile and ARM(Raspberry pi) version.

After successfully downloading your classifier, you should have a .zip file of your classifier in local files.

7 Build and deploy your solution

Before we start our solution, we need to edit the .env file and build and push modules to our container registry.

7.1 Prepare the development environment

You need the following dev tools to do IoT Edge development in general, to create this solution and to edit it:

- Install Docker for your machine to build the module images
- Visual Studio Code: Development environment.
- Azure IoT Edge Extension for VS Code An extension that connects to your IoT Hub and lets you manage your IoT Devices and IoT Edge Devices right from VS Code. I recommend installing Azure IoT Tools Extension (*vscode.azure-iot-tools*) which should have all you need for this solution. Once installed, connect it to your IoT Hub:

1. In Visual Studio Code, open the **Explorer** view.
2. At the bottom of the Explorer, expand the **Azure IoT Hub** section.

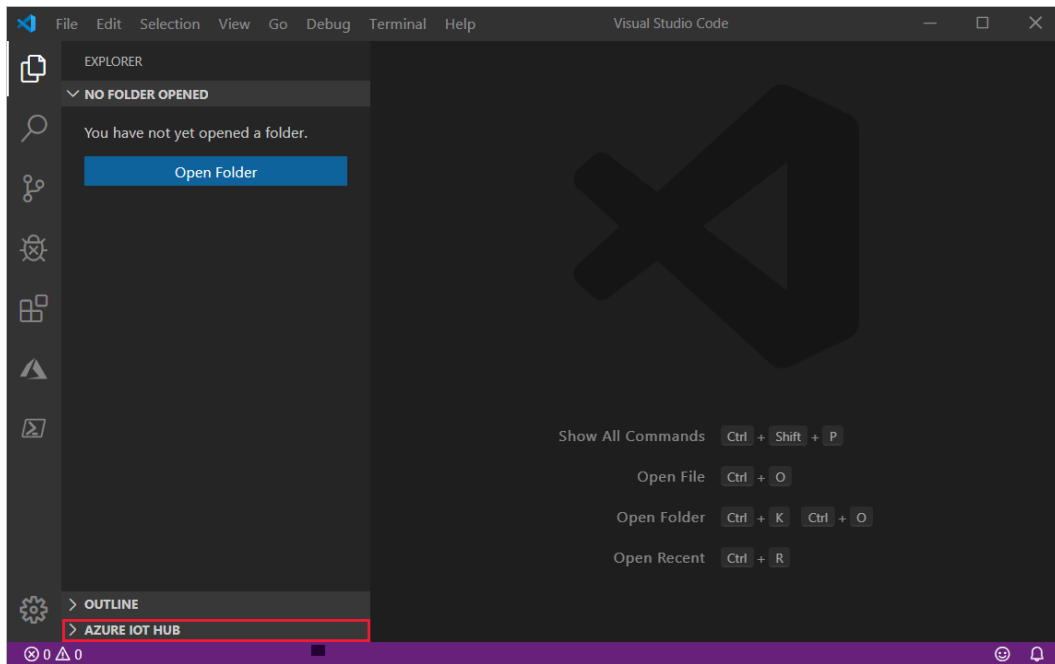


Figure 10: Azure IoT Hub section at the bottom of the Explorer

3. Click on the ... in the **Azure IoT Hub** section header. If you don't see the ellipsis, hover over the header.
4. Choose **Select IoT Hub**.
5. If you are not signed in to your Azure account, follow the prompts to do so.
6. Select your Azure subscription.
7. Select your IoT hub.

7.2 Deploying the solution

Now, assuming you've cloned (or downloaded) this repository, open the folder in VS Code. Update the `.env` file with the values for your container registry (found in **Settings** > **Access keys** on your registry's page in Azure Portal and the overview).

If you've created your own custom vision AI model, open the .zip file downloaded earlier and replace the `modules/ImageClassifierService/app/model.pb` and `modules/ImageClassifierService/app/labels.txt` files with your own.

Now we need to login to the container registry using the following command in VS Code Terminal. You need to replace the user name and registry url with the ones you have copied in previous sections. This command will ask for the password as well.

```
sudo docker login -u <your_container_registry_name> \
<your_container_registry_login_server>
```

Note: if the command doesn't work you might need to install the Docker extension for VS Code

Build the entire solution by right-clicking on the `deployment.template.json` file and select **Build and push IoT Edge Solution**. This should create a config directory with one `deployment.json` file.

Deploy the solution to your device by right-clicking on the `config/deployment.json` file, select **Create Deployment for Single device** and choose your targeted device

The solution should be running on your device and you can start testing and observing your solution!!!

(Be sure to start your Stream Analytics job to capture the messages sent from your device)

Additionally, you can observe how the modules work on your Raspberry Pi device by calling:

```
docker logs <container_name>
```

on Raspberry pi through your Putty connection (or directly).

8 Provide Feedback

If you encounter any bugs or have questions/suggestions, please file an issue in the Issues section of the project.

References

- [1] Microsoft. Azure container registry documentation. <https://docs.microsoft.com/en-us/azure/container-registry/>.
- [2] Microsoft. *Azure IoT Edge documentation*.
- [3] Microsoft. Azure iot hub pricing. <https://azure.microsoft.com/en-us/pricing/details/iot-hub/>.
- [4] Microsoft. Build a classifier with the custom vision website. <https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/getting-started-build-a-classifier>.
- [5] Microsoft. Install or uninstall azure iot edge for linux. <https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge?view=iotedge-2018-06>.
- [6] Microsoft. Register an iot edge device in iot hub. <https://docs.microsoft.com/en-us/azure/iot-edge/how-to-register-device?view=iotedge-2018-06>.
- [7] Microsoft. Understand iot edge automatic deployments. <https://docs.microsoft.com/en-us/azure/iot-edge/module-deployment-monitoring?view=iotedge-2020-11>.
- [8] Raspberry Pi. Setting up your raspberry pi. <https://www.raspberrypi.org/documentation/computers/getting-started.html>.