# Documentation for the Setup of AI Edge Camera Demo

## Introduction

The Vision AI Dev Kit is a camera developed by Qualcomm which includes its proprietary Neural Processing Engine, the Snapdragon Neural Processing Engine (SNPE). It allows users to develop and showcase use-cases for IOT Edge solutions which are especially useful in environments or situations which require:

1. Quick response time and low latency
2. High volumes of network traffic and cloud processing
3. High resilience despite low internet connectivity

This guide will teach you how to setup a demo which features:

1. Workplace Safety AI

This demo imagines a workspace environment where a workspace is monitored by AI Cameras to detect safety violations. When a user is caught entering the work-zone without a helmet, the camera detects it, his picture is taken, and a notification is sent to a supervisor.

* + 1. Data and AI

This demo demonstrates the camera’s AI inferencing capabilities. It can be used as a tool to drive conversations surrounding AI, and the real-world value it can provide. It also demonstrates the feasibility of video analytics as a solution in areas with low connectivity, or as a measure to reduce bandwidth usage.

* + 1. Power Platform

This demo can also be used to show the ease of use of PowerApps to build front-end applications and interfaces from existing data (using data collected from the camera stored in the CDS entity).

1. “Plug and Play” Module switching

This demo shows the capabilities of IOT Hub in the management and deployment of modules to Edge devices, addressing concerns with the scalability of Edge solutions. Clients will be able to see how a module is deployed at scale to several devices via the azure portal.

## Setting up the Camera

### Downloading and using ADB Shell on your laptop

https://azure.github.io/Vision-AI-DevKit-Pages/docs/Cable/

### Flash New Firmware

https://azure.github.io/Vision-AI-DevKit-Pages/docs/Firmware/

### Connect Device to Azure IOT Hub

<https://azure.github.io/Vision-AI-DevKit-Pages/docs/Run_OOBE/#what-will-you-need>

## Loading the Model and Container Images into IOT Hub and Camera

### Remove any existing Vision AI Dev Kit modules

In order to run a new module on the Vision Ai Dev kit it is important to first remove any existing modules. This will prevent two modules from competing to gain control of the Qualcomm hardware at the same time.

* The instructions for Vision AI Dev Kit it will provision a sample module to be deployed to the device from Azure IoT Hub. The sample project described here will replace the module that came with Vision AI Dev Kit.
* Using the (Azure Portal)[[www.portal.azure.com](http://www.portal.azure.com/)] online locate the Azure IoT Hub that was provisioned in your subscription during the Vision AI Dev Kit setup instructions.
* Select IoT Edge on the left pane, then select the IoT Edge Device tab.
* You should see your Vision AI Dev Kit device. Select the device then select Set Modules at the top of the screen. You can delete the sample module provisioned by Vision AI Dev Kit.
* Remember this screen when you want to configure a new module deployment later.

Source: https://github.com/sseiber/peabody-local-service/blob/master/README.md

### Hardware Requirement

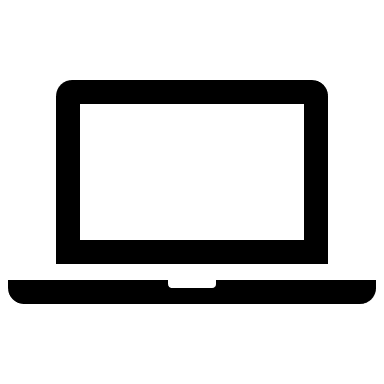
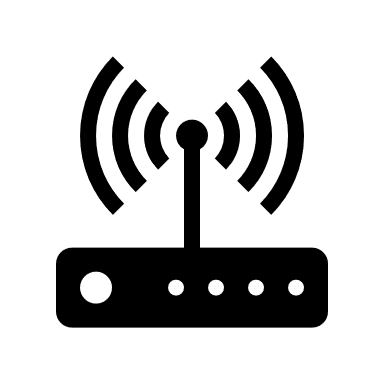
Device:

1. Vision AI Dev Kit
2. Display (support HDMI)
3. Working Machine
4. Smartphone (with screen mirroring)

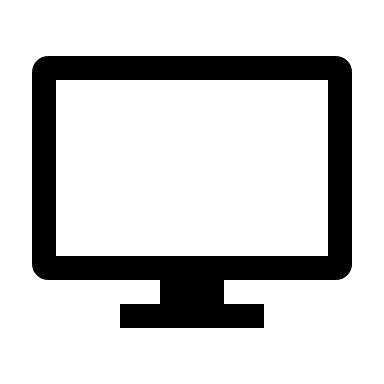
Others:

1. HDMI Cable
2. USB Type C-B Cable
3. Wi-Fi (Internet Access, need the SSID and password)

Device Connections:



Wi-Fi



USB Type C-B Cable

Display with HDMI

### Software Requirement

* Windows 10 1809
* Visual Studio Code and a set of extensions (see the following section for the details)
* Python 3.6
* An active Azure Account

### Setup Visual Studio Code development environment on a local machine

1. Install [**Visual Studio Code**](https://code.visualstudio.com/).
2. Install **Python 3.6**. by Anaconda

The current **Python** is installed by **Anaconda3** (<https://www.anaconda.com/download/> ). You can install the latest version and create a lower version **Python** environment with **conda** as following. Please install the package using default options.

After install the **Anaconda3**, open the **Start Menu\Programs\Anaconda3 (64-bit)\Anaconda Prompt** and run the following commands:

2.1 Create a Python 3.6 runtime environment:

**conda create -n py36 python=3.6 anaconda**

*Press* ***y*** *when it asks Proceed ([y]/n)?*

2.2 Activate the Python 3.6 environment:

(Run following command in current Anaconda Prompt)

**conda activate py36**

or

Activate the Python 3.6 environment by the Start Menu (after the environment is created):

**Start Menu\Programs\Anaconda3 (64-bit)\Anaconda Prompt (py36)**

1. Open **Visual Studio Code** and install following extensions:

* **Azure Machine Learning**

(**Azure Account** extension and the **Microsoft Python** extension will be automatically installed)

* **Azure IoT Hub Toolkit**
* **Azure IoT Edge**
* **Docker extension** (optional)

1. **Install Azure ML SDK** and required packages. (If you choose to use sample code directly, skip to next section: **Deploy a Model Container Image to a Vision AI Dev Kit device in Visual Studio Code**)

|  |
| --- |
| **Note**: When install the Azure Machine Learning SDK or related Python packages, there are some Python packages which depend on specific version of other Python packages. If these(dependency) Python packages version are too high, the installation will fail.  To work around this issue, we create a requirement file which has stricter version defined. |

* 1. Install packages in **the Anaconda Prompt (py36) command prompt**: go to Start Menu\Programs\Anaconda3 (64-bit)\Anaconda Prompt (py36), right click “Anaconda Prompt (py36)”, choose “Run as administrator”. in command prompt:

|  |
| --- |
| Change directory to the folder contains “requirements.txt”, e.g. C:\VisionAIDevKit\, execute the following commands:  **pip install msgpack==0.6.1**  **pip install --ignore-installed PyYAML==4.2b1**  **pip install --upgrade -r requirements.txt** |

1. Restart **Visual Studio Code** in the **Python 3.6** runtime environment.

You may need to close the current **Visual Studio Code**.

In **Anaconda Prompt (py36)** command Prompt, type:

**code**

The new **Visual Studio Code** willinherit theenvironment variable of **Python 3.6** runtime environment

1. Enter [**Python: Select Interpreter**] command in the command palette box to select your Python interpreter (Python 3.6).
2. Enter [**Azure: Sign In**] command in the command palette box to sign in Azure account and select your subscription.
3. Create a new IoT Hub and a new IoT Edge device in VS Code as mentioned in [Create an IoT hub using the Azure IoT Hub Toolkit for Visual Studio Code](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-use-iot-toolkit) and [Register a new Azure IoT Edge device from Visual Studio Code](https://docs.microsoft.com/en-us/azure/iot-edge/how-to-register-device-vscode).

|  |
| --- |
| **Note**: If you have existing IoT Hub and Edge devices (e.g. followed the “Vision AI Dev Kit Setup.doc” to do so, you can use them instead of creating new one |

1. Create a new workspace in VS Code as mentioned in [Get started with Azure Machine Learning for Visual Studio Code](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-vscode-tools).

|  |
| --- |
| **Note**: See this [doc](https://azure.microsoft.com/en-us/global-infrastructure/services/?products=machine-learning-service) for the available regions. |

### Deploy a Model Container Image to a Vision AI Dev Kit device in Visual Studio Code

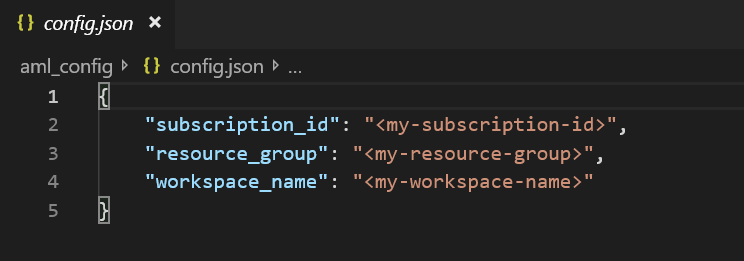
1. Unzip the AIEdgeCam folder provided, or download (or clone) the latest Visual Studio Code sample **AiEdgeCam** from GitHub <https://github.com/mingyi850/AiEdgeCam/>
2. Launch Visual Studio Code, and select [**File** > **Open Folder…**] command to open “**VisionSample**” directory as workspace root.
3. Use [**Python: Select Interpreter**] command in the command palette box or click the current “**Python interpreter**” option on the bottom line to set "**python.pythonPath**" in **.vscode\settings.json**.



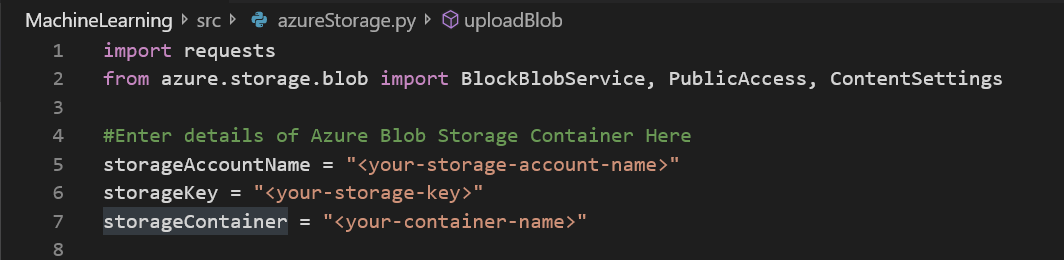
1. **Install Azure ML SDK** and required packages (**Skip this step if you already did step 4 of previous section**)

|  |
| --- |
| Close VS Code and launch VS Code again by “**Run as administrator**”.  Select [**Terminal** > **New Terminal**] command to open a terminal window, change directory to “VisionSample\**MachineLearning**\**scripts**”, and execute the following commands to install required Python packages:  **pip install msgpack==0.6.1**  **pip install --ignore-installed PyYAML==4.2b1**  **pip install --upgrade -r requirements.txt**  **Note**: The above installation steps works for the latest Azure Machine Learning SDK version **v1.0.8** and install Python 3.6.5 by [Anaconda with Python version 3.6.5](https://repo.anaconda.com/archive/Anaconda3-5.2.0-Windows-x86_64.exe) link. If the version of AML SDK, Python, or other packages will be changed in the future, you might have to install or upgrade packages manually. |

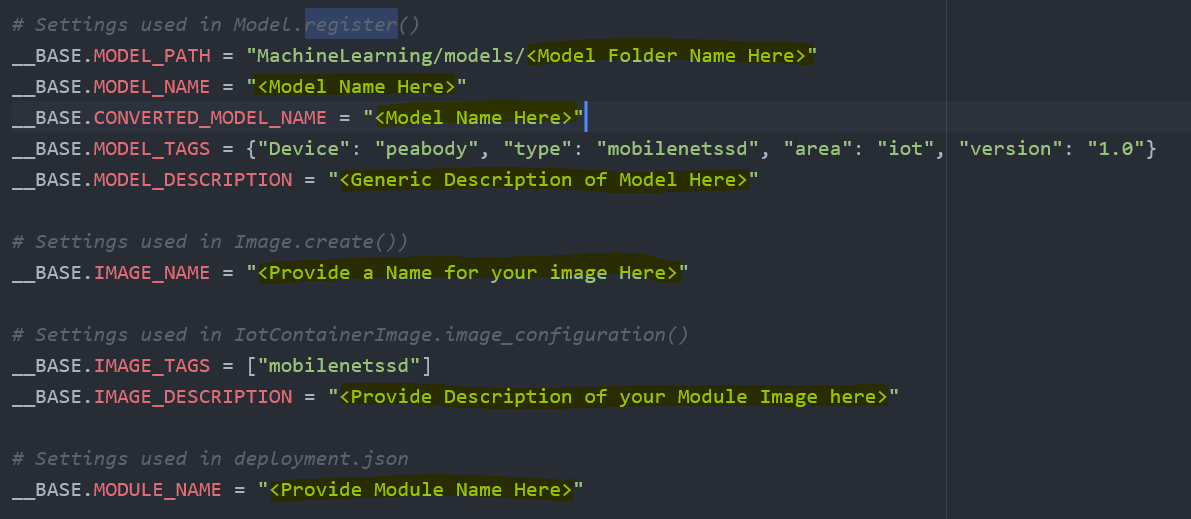
1. Go to aml\_config folder and open the config.json folder. Replace the 3 settings shown below with your AML Workspace Subscription ID, Resource Group and workspace name. This will link the deployment script to your AML Workspace.

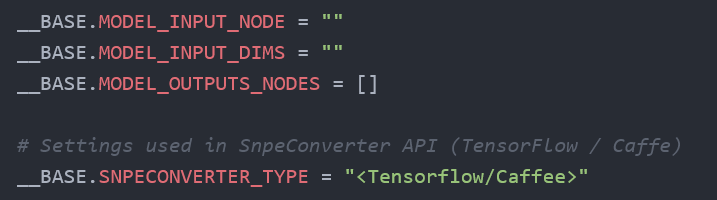


1. Go to the **MachineLearning\src** folder and open **azureStorage.py**. Replace the storageAccountName, storageKey and storageContainer values with the details of your own Azure Storage account and a container which will hold snapshots from the AI-Dev-Kit Camera

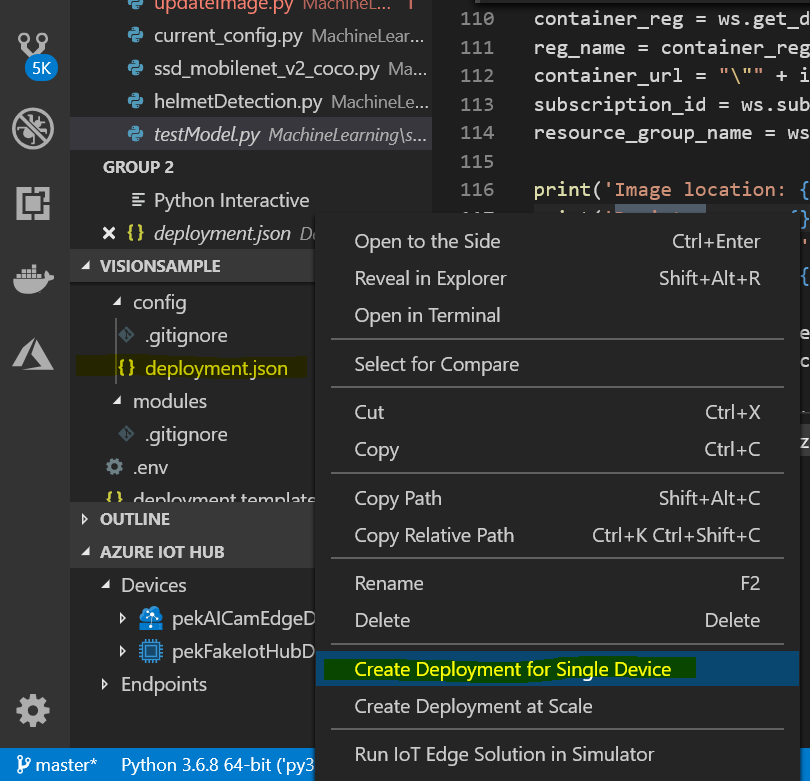


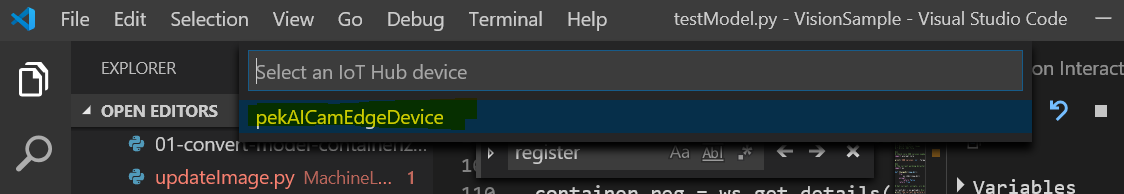
1. There are 3 ways to use the module:
2. Using the default Helmet Detection Module:
3. In **MachineLearning\scripts\model\_configs** folder, copy the contents of **HelmetDetection.py** into **current\_config.py**. This config file will be referenced by all deployment scripts to instruct it to use the correct model.
4. Run testModel.py to register model, create container image, and write settings related to the image to “**.env**” file under “**DeployContainerToAML\modules**” folder.
5. Using a new Custom Vision Model:
6. Upload the exported Custom Vision Model (in DLC format) in the **MachineLearning\models** folder, with its labels.txt file and va\_snpe file.
7. In the **MachineLearning\scripts\model\_configs** folder, use the model\_config\_template.py to generate a new config file, edit the highlighted portions with your own names and descriptions. The MODEL\_PATH parameter should direct to the folder where the model is stored, MODEL\_NAME and CONVERTED\_MODEL\_NAME should both contain the name of the model file (“something.dlc”). DO NOT use the name ‘model.dlc’.  
   All other fields can be filled to your discretion.



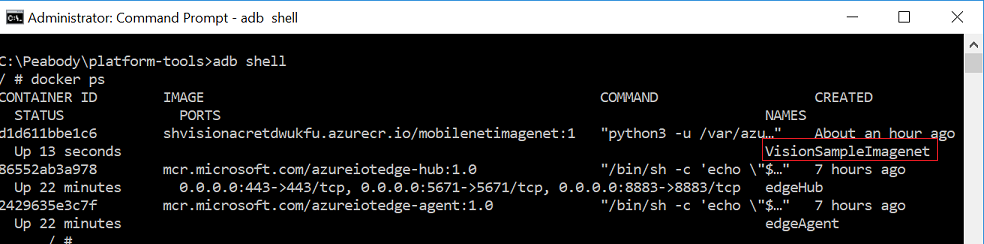
1. Once done, replace the contents of current\_config.py with your new config file. Follow step a)(2) to register model and create an image.
2. Using a new TensorFlow model. **\*Not recommended\***
3. Upload the tensorflow model (.pb) file into the **MachineLearning\models folder**, with its labels.txt file
4. In the **MachineLearning\scripts\model\_configs** folder, use the **model\_config\_template.py** to generate a new config file, edit the highlighted portions with your own names and descriptions. (This is the same as in b)(2) above) . In addition to those portions, you will need to fill out the following fields:  
   

These fields can only be filled out if you have knowledge of the Input Node, Dimensions and Output Nodes of your tensorflow graph. This can possibly be done using TensorBoard (read online for documentation). SNPECONVERTED\_TYPE should be changed to “Tensorflow”.

1. Once done, replace the contents of **current\_config.py** with your new config file. Run “**01-convert-model-containerize.py**” under “**VisionSample\MachineLearning\scripts**” folder and click [**Run Cell**] or [**Run All Cells**] link to register model, convert model, create container image, and write settings related to the image to “**.env**” file.
2. When process is completed, go to DeployContainerFromAML and right click “**deployment.template.json**” and select [**Generate IoT Edge Deployment Manifest**] command to create a new “**deployment.json**” file under “DeployContainerFromAML\**config**” folder.
3. Got to “DeployContainerFromAML\**config**”, right click on **deployment.json** and select “Create Deployment for Single Device”.
4. Select your device from the drop down menu to deploy module to your device.



1. Monitor the deployment status for the AI Vision Kit device by using platform tools commands: [**adb shell**], [**docker ps**] and [**docker logs edgeAgent**] commands.



1. Check object detection results:

* Use platform tools commands [**adb shell** > **iotedge logs** *<module name>*] (e.g. “i**otedge logs <Your-Module-Name>**”) to check module outputs. Alternatively, connect the dev kit to a display using a HDMI cable or wirelessly through RTSP (documented on the AI Dev Kit site) to view bounding boxes and camera inferences

### Maintenance of Camera Module

1. Changing Models

In order to change models and create a new image, go through step 7(a, b or c) in order to register a new model on AML Workspace and its corresponding image.

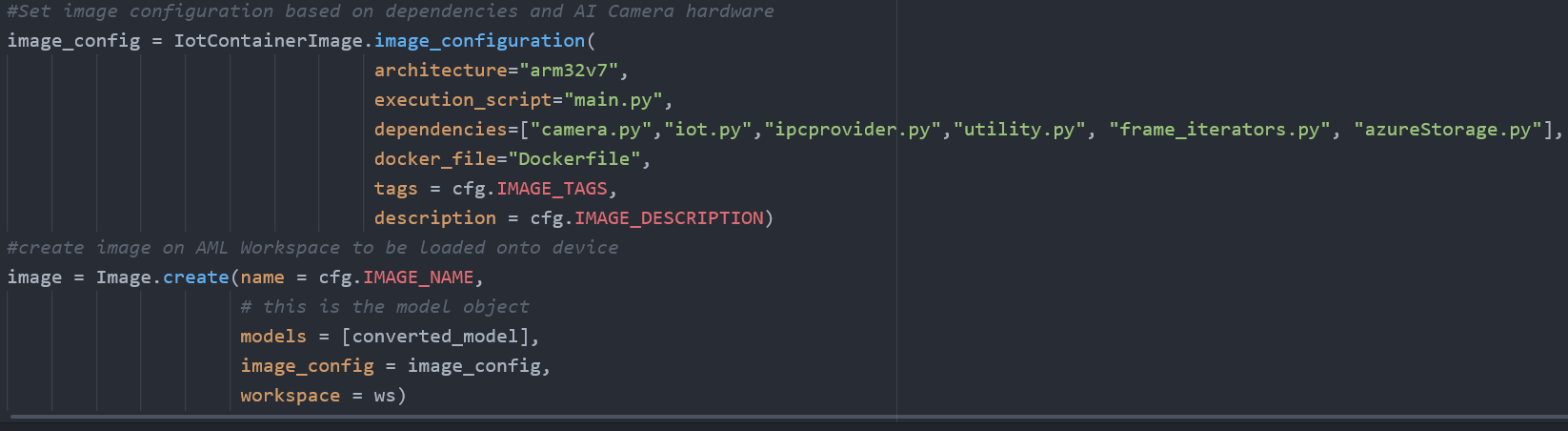
1. Updating Image
2. Updating Image with only modifications to existing scripts:

Go to **MachineLearning\scripts** and run **updateImage.py**. Ensure that **current\_config.py** is directed at the correct model name (on AML).

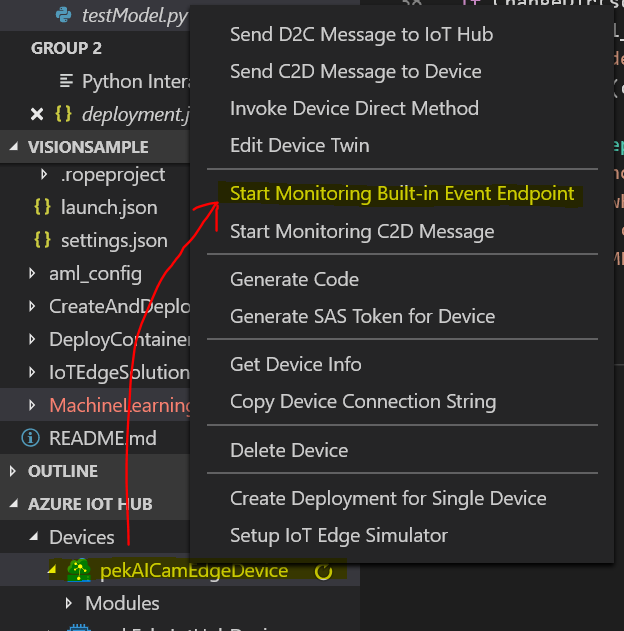
1. Updating Image with the introduction of new scripts and dependencies:

Under **MachineLearning\src** add the new scripts/dependencies.

Go to **MachineLearning\scripts\updateImage.py**, add the name of the new script/dependency to the array “dependencies”.

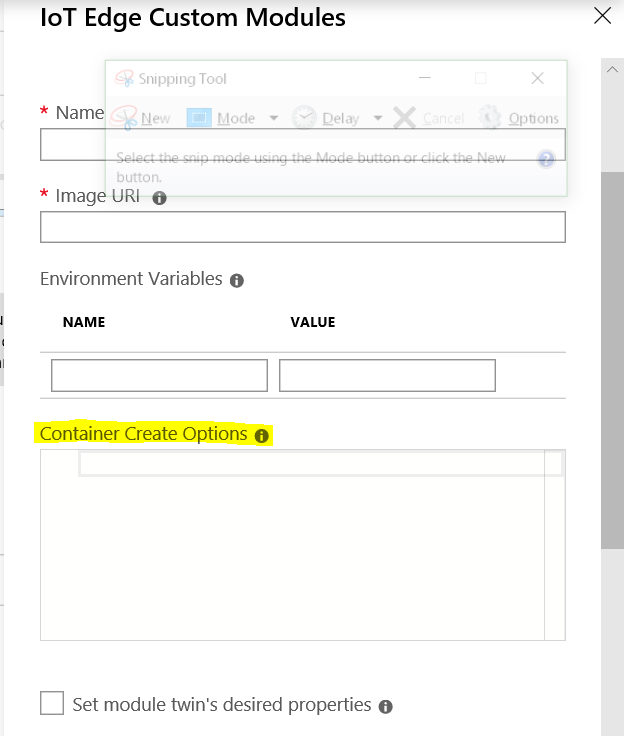


If required, update the Dockerfile located in the MachineLearning\**src** folder. Run the script **updateImage.py** again and follow steps 8 onwards above.

1. Monitoring Messages to IOT Hub:  
   

In VS Code, under the Azure IOT Hub Tab, right click on your device and select “Start Monitoring Built-in Event Endpoint. This will show you a list of messages which have been sent to the IOT Hub.   
**Note: IF you have set up a route which routes these messages into a separate endpoint, you will NOT see the messages appear. This feature only monitors the Default endpoint.**

### Deployment from Portal

In order to perform the switching of modules on-site, 2 different modules should be setup for deployments at scale on the IOT Hub.   
If you have completed the above steps or have 2 models and their respective images registered on Azure Container Registry you can use the following steps to load your images into deployments on IOT Hub.  
In IOT Hub, create 2 separate deployments which reference these images. And create deployments based on these 2 images. Under Step 2, Add Modules, under “Container Create Options”  


Find **container\_create\_options.json** in the AiEdgeCam folder and copy its contents into the box.

In order to switch between the 2 deployments, simply change the priority of the deployments such that the desired deployment is of a higher priority than the current deployment.

## Setting up Service Bus and Routes

### Creating Service Bus and Queue

Create a service bus using the same resource group as your IOT Hub. On the service Bus, create a queue. Filtered messages from the IOT Hub from the Helmet detection module will be routed into the queue. This will be used to trigger the LogicApp.

### Routing of telemetry to service bus

Go back to IOT Hub, under message routing, go to Custom Endpoints and select “Add” to add a new service bus endpoint.   
Reference the service bus and queue created in step 3(a) to add the service bus to the IOT Hub as a custom endpoint.

Go to Routes, add a new route to your custom endpoint, using data Telemetry messages as the data source.

Under “Routing query”, enter :

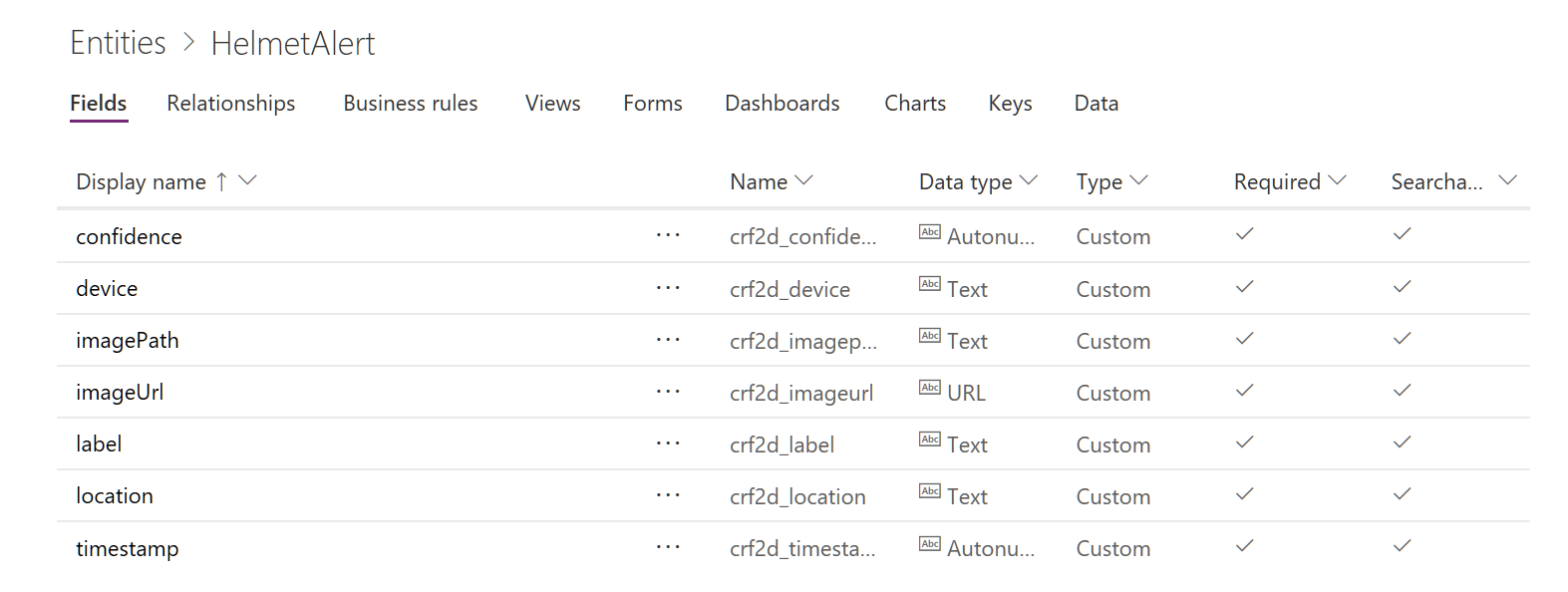
helmetAlertFlag = "true"

The helmetAlertFlag is a application property on the message sent from the HelmetDetection Module.   
It is set to “true” when the camera detects a person without a helmet above a certain confidence level (set to 75% confidence as default).

## Setting Up CDS Entity

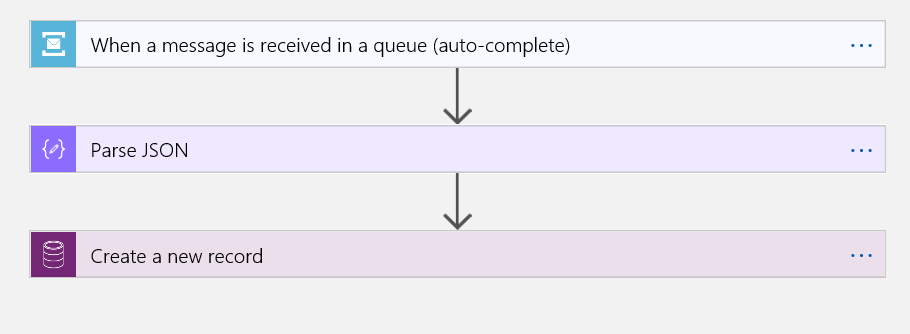
The CDS entity will be used to store data sent from the camera, as well as provide data to the PowerApp.

1. Go to make.powerapps.com
2. Use an existing environment (or create a new one: <https://docs.microsoft.com/en-sg/power-platform/admin/create-environment>)
3. In your new environment, setup a database : <https://docs.microsoft.com/en-sg/power-platform/admin/create-database>
4. Then, under Data->Entities, create an entity with the following fields:

  
These fields should all be required and searchable.

## Setting up Logic-Apps

Create a Logic App with the following schema:

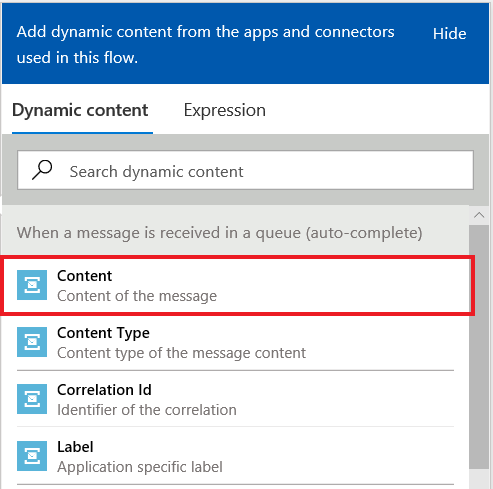


1. **When message is received in a queue:**

Select Queue created in step 3a as the target. LogicApp should check for updates in queue at least every 3s (or higher frequency for higher responsiveness).

1. **Parse JSON:**

Content: From Dynamic Content, select ‘content’. This will load the message body from the message in the queue.



Schema:

Use the following schema by copying it into the box:  
{

"properties": {

"confidence": {

"type": "integer"

},

"device": {

"type": "string"

},

"id": {

"type": "integer"

},

"imagePath": {

"type": "string"

},

"imageURL": {

"type": "string"

},

"label": {

"type": "string"

},

"location": {

"type": "string"

},

"noHelmetAlert": {

"type": "boolean"

},

"timestamp": {

"type": "number"

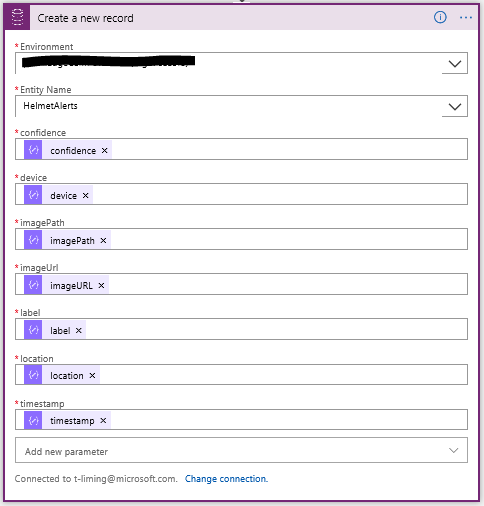
}

},

"type": "object"

}

1. **Create a new Record**



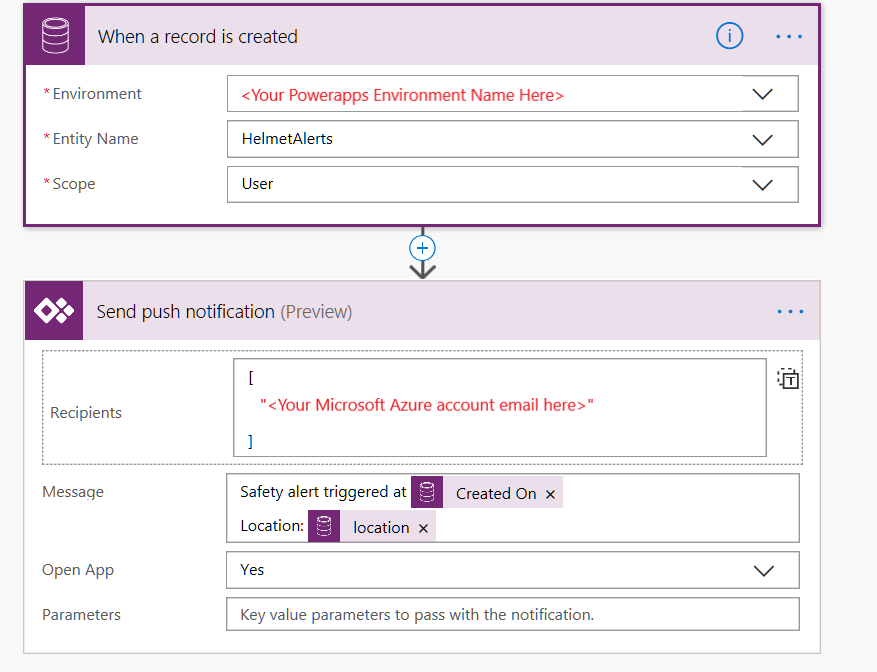
Use the following inputs to setup the creation of a new record, referencing the CDS environment and entity created in step 4.

## Setting Up PowerApps

1. Go to make.powerapps.com. Under the “Apps” tab, select “Import package”. Then, select the .zip file from the AiEdgeCam repository which starts with “AiEdgeCamTemplate…”.
2. Under “Related resources” you should see an option to link your Azure Blob Storage account with the app. This will be used to link the database of snapshots from the camera with the application.
3. Once done, the powerapp should be up and running on your studio. Under “Apps”, you should be able to see the new powerapp, on your dashboard.
4. Open the powerapp in edit mode to verify that it has been imported correctly. You should be able to view the components in the view on the left. (Ensure that the HelmetAlerts entity is populated with at least one entity or it will show an error. IF it is not populated, edit the entity using Excel under Data->Entities->Helmet Alert.
5. Your powerapp should be working now.

## Setting Up Notification Flow

1. Setup a Flow on the Powerapps Portal with the following parameters:



1. When a record is created  
   Create a new connection with your Microsoft CDS account by signing in

Then create the trigger as shown above.

1. Send push notification  
   Create a new connection to your application using your app ID.

Then create the action as shown above.

## Architecture Diagram

Architecture Diagram of the Vision AI Kit Demo 



## Demo Guide:

The demo is designed to be deployed in the MTC with three separate screens.  
  


Suggested screen setup for client demo

There are three aspects of the demo which can be shown using the setup here, with increasing complexity:

* 1. AI-on-the-Edge

Presenter starts of presentation wearing a helmet. He/she then shows output of camera’s video inference and its ability to recognise and detect the position of the helmet on the presenter’s head in real-time. The presenter can then choose to remove helmet, prompting a change in the camera’s output. Demo can be used to drive conversation on AI in general as a solution for general use cases, as well as showcase our AI-Edge Capabilities in action.

* 1. AI in the workplace

Presenter builds on step (1) to present a plausible use case to the customer. Using the PowerApps View, the presenter can then explain how the integration of edge devices with IOT Hub and other Microsoft solutions can present viable business value to the customers. The presenter should allow camera to detect him/herself without a helmet which should then trigger a notification on the PowerApps view. The presenter can then start using the Powerapp to view data collected from the incident. This simulates the view of a “Supervisor”, who can then use the data gathered in the Power-app to take preventive or corrective action.

Conversation around this can be extended to other forms of security, or video monitoring.

* 1. Deployment-at-Scale

The deployment-at-Scale demo shows the ease of switching modules deployed on edge devices connected to IOT Hub, to showcase the potential versatility of having edge devices connected to Azure IOT Hub. With all screens used, the presenter goes into Azure Portal on a separate screen (or on the smartphone displaying the PowerApp) and selects the deployment not currently in-use for the Camera. He then switches it’s priority to exceed the current deployment’s priority level. This will trigger a redeployment of the new module.  
  
On the output screen, the camera will stop showing bounding boxes for a brief period of time (10s if the new module has been deployed before), before starting to show the new categories listed in the bounding boxes. This shows that the new module has been successfully deployed.

## Known Issues

### Inability of DevKit to Connect to IOT Hub

The AI-Dev-Kit initially faced difficulties connecting to the IOT Hub through the standard setup process. This could have been due to a firmware issue or an issue with the internet connection provided. To troubleshoot the issue, ensure that the connection string used by the Ai-Dev-Kit is not being used by another device, and that the Wifi connection has been setup properly. If the problem ensues, perform a hard reset and try again.

### Memory management

When a new module is loaded into the Vision AI Dev Kit, old, unused images remain within the device. These images are generally big and take up large amounts of memory. If memory is not managed, modules could fail to load or execute. To remedy this, ensure that the number of images inside the Dev Kit is kept to a minimum. This can be done by logging into the device shell, checking the images currently loaded, and removing the old images, which can be done by running the following commands:

adb shell  
docker images  
docker rmi <image id>

### Capture Image failure

The provided API for Capture Image occasionally fails and subsequently causes the entire module to crash. This happens occasionally but only at the start when the module starts (i.e if the running module has already managed to capture an image, it will not subsequently fail). The cause of this failure is not known but it is suspected to be an issue with multiple modules attempting to connect to the camera endpoint at the same time. If the issue happens, switch out the modules for another one, or re-deploy the current module to the device. Should the issue persist, perform a factory reset on the device and try again.

### EdgeHub Failure

There have been instances where the module runs and appears to send telemetry and logs, but the PowerApps notification does not trigger and no messages are sent to the IOT Hub (or messages are sent in bulk, but only upon initialization of a new module). This is due to a fault with IOT Hub version 1.0 and happens occasionally, especially after factory resets. To fix this, perform a factory reset on the device so it re-downloads the edgeAgent and edgeHub modules. Future iterations of the firmware should allow the device to download newer versions of edgeHub and edgeAgent which should resolve this issue.