Overview – Image Capture, Tagging and Deployment for AI Vision Devkit

There are a variety of way to deploy the Image Capture module for use with the Workplace Safety demo. The approach used here creates a 2nd IoT Edge deployment that will only provide the functionality to capture images and upload them to blob storage. After this is done, the camera will need to be changed to use the workplace safety edge deployment.

Authors: Teo De Las Heras and Mike Iem

Build the IoT Edge Deployment

1. Clone the Vision AI Developer Kit via a cmd line:
   C:\Users\billg\source\repos>**git clone https://github.com/microsoft/vision-ai-developer-kit.git**
2. Launch Visual Studio Code, and select:
   File > Open Workspace. Chose the camera-tagging directory as workspace root.
3. Create a .env file with the values for your container registry. This should be the same values from the demo
   CONTAINER_REGISTRY_NAME=**<Your_Acr_Uri>**
   CONTAINER_REGISTRY_USERNAME=**<Your_Acr_UserName>**
   CONTAINER_REGISTRY_PASSWORD=**<Your_Acr_Password>**
4. Sign in to your Azure Container Registry by entering the following command in the Visual Studio Code integrated terminal (replace <REGISTRY_USER_NAME>, <REGISTRY_PASSWORD>, and <REGISTRY_NAME> to your container registry values set in the .env file).

   docker login -u <REGISTRY_USER_NAME> -p <REGISTRY_PASSWORD> <REGISTRY_NAME>.azurecr.io

5. Set target architecture. Open the command palette: View -> Command Palette (Ctrl+Shift+P). Search for **Azure IoT Edge: Set Default Target Platform for Edge Solution**, Select **arm32v7**
6. Add an edge module: . Open the command palette: View -> Command Palette (Ctrl+Shift+P). Search for **azure iot edge: Add IoT Edge Module**
7. Under **Select Module Template** choose **Module from Azure Marketplace**.
8. Search for *AI Vision Dev Kit Get Started Module*. Accept the defaults and click **Import**
9. Under the VSCode explorer on the right hand side, click on the file deployment.peabody.template.json and change the value of **ShowVideoOverlay** to false. If the file is missing then make sure the workspace was opened correctly (Step 2 above)
10. Right-click on *deployment.peabody.template.json* select the **Build and Push IoT Edge Solution command** to generate a new deployment.json file in the config folder, build a module image, and push the image to the specified ACR repository
11. Right-click on **config/deployment.peabody.arm32v7.json**, select **Create Deployment for Single Device**, and choose the targeted IoT Edge device to deploy the container image.
12. Verify the deployment is complete by plugging the AI DevKit into a PC via a USB cable and issuing the command:
    C:\>**adb shell docker ps -a**
    Verify that the Camera Tagging and Vision Sample modules are running
13. Note: Give it a coupe minutes as it takes time to download and start

Use Camera Tagging Module

1. With the AI DevKit plugged into a PC, find the IP address of the camera by issuing the command:
   C:\>**adb shell ifconfig wlan0**
2. Browse to website for the Camera Tagging module by entering the IP address:3000 in a browser
   example: https://192.168.1.127:3000
3. Once on the website, click on **Change Camera**
4. In the window that opens, fill in the Camera Name and RTSP Address as follows
   Camera Name: **AIDevKit**

RSTP Address: **rtsp://<your ip address>:8900/live**

5. Click **Add Camera**. Chose the camera that was added from the drop-down list and click **Save**
6. At this point, you should see the live stream of the camera in the browser
7. Click **Capture** on the right-hand side to take pictures to that will be used to train the new model. Use the values for Tag… and Name… as appropriate.
8. For example:

   PersonNoPPE", "PersonHardHat", "PersonSafetyVest" are the tags you need to use for this demo.

   Tag: PersonNoPPE
   Image Name: PersonNoPPE1

   Tag: PersonHardHat
   Image Name: PersonHardHat1

   The more pictures that are taken, the more accurate the model will be. In addition, the camera angle, lighting, distance from the camera, etc… all affect the accuracy of the model. Depending on what you are doing with your Qualcomm "Peabody" camera.

Train the Vision AI Model

Create Vision AI project and upload images

1. Log into https://www.customvision.ai/
2. Create a new project with the values as follows:

   Name: **Workplace Safety**
   Project Types: **Classification**
   Classification Types: **Multiclass**
   Domains: **General (compact)**
   Export Capabilities: **Vision AI Dev Kit**

3. After the project is created, click on the Project Name and then click on the gear icon in the top right to look at the project settings
4. Copy the values for **Key** and **Endpoints**
5. In the Camera Tagging Web App, click on **Upload Settings** on the left
6. Click on **Vision**
7. Enter the values for Endpoint and Training key from above
8. Choose the project that was created earlier
9. Click 'Push to Custom Vision'

Train and Export the Vision AI model

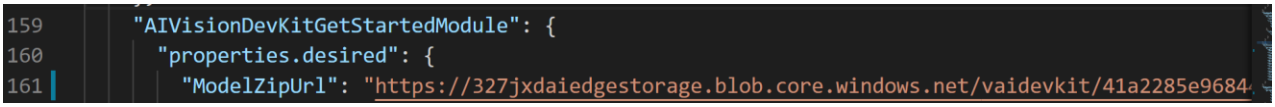
1. Back at the Vision AI site, https://www.customvision.ai/ go into the project created
2. In the Top Right, click on **Train**. In the window that opens, accept the defaults and click **Train**
3. The browser will automatically place you on the **Performance** tab. When the training finishes, **Export** should be available. Click on **Export**
4. At the Choose your platform screen, select VAIDK Vision AI Dev Kit, click **Export** and then click **Download** to download the newly created Vision AI model as a .zip file. Remember the location where this was downloaded
5. Go to the **Azure Portal** and find the **Storage Account** created for the Workplace Health and Safety Demo
6. Click on **Storage Explorer (preview)**
7. In the **Storage Explorer** window, right-click on Blob Containers and choose **Create Blob Container**
   Name: **vaidevkit**

Public access level: Blob (anonymous read access for blobs only)
Click **Create**

8. In the Storage Explorer window click on the vaidevkit container that was created and **upload** the Vision AI Model (zip file)
9. After the upload is complete, click on **Copy URL** and save the URL of the Vision AI model

Update the Vision AI DevKit

1. Using VS Code, open the **WorkplaceHealthAndSafetyDemo** workspace. Note: This was the original Workspace created during the original demo setup. Do not use the workspace created during these instructions.
2. Open the file **deployment.template.json** and find the **ModelZipUrl** property under **AIVisionDevKitGetStartedModule**
3. Update the value from "" to the url from the previous step. The result should look as follows



4. Save the changes, right-click deployment.template.json , and choose **Build and Push IoT Edge Solution**
5. Finally, under config, right-click **deployment.arm32v7.json** and choose the option to **Create Deployment for Single Device**. Choose your IoT Edge device.
6. You have now deployed an updated Vision AI model to your Vision AI DevKit

Cleanup and Useful adb commands for your camera

The camera storage can fill up if you may need to do some cleanup to insure you have enough data space. Here are the most popular Adb commands you can use.

What is adb?

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.

If you haven't done so, you need to install the Platform Toolkit (adb tools)

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

1. Show connected devices
   adb devices
2. Check device firmware version
   adb shell cat /etc/version
3. ADB command line to see what docker containers is running on the camera
   adb shell docker ps -a
4. List out the file system and what each mount is taking (Make sure your data is 90% or less)
   adb shell df

```
C:\ADBTools>adb shell df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root         3047184 2074560    956240  69% /
devtmpfs           882388       4    882384   1% /dev
tmpfs              917716       0    917716   0% /dev/shm
tmpfs              917716   75400    842316   9% /run
tmpfs              917716       0    917716   0% /sys/fs/cgroup
tmpfs              917716    1060    916656   1% /var/volatile
/dev/mmcblk0p64      3952     452      3340  12% /systemrw
/dev/mmcblk0p16     58888      40     57568   1% /cache
/dev/mmcblk0p65   4729848 1957376   2756088  42% /data
/dev/mmcblk0p49      2168      64      1980   4% /persist
/dev/mmcblk0p43     65488     608     64880   1% /bt_firmware
/dev/mmcblk0p27     28144   21620      5872  79% /dsp
/dev/mmcblk0p25    112592   41088     71504  37% /firmware
overlay           4729848 1957376   2756088  42% /etc
overlay           4729848 1957376   2756088  42% /var/lib/iotedge
overlay           4729848 1957376   2756088  42% /data/misc/docker/overlay2/4e762239374fd5c4b69a875420682ca830cca92d05f62a305dbf7a5c3c43c125/merged
shm                 65536       8     65528   1% /data/misc/docker/containers/c0ed299bb12cc8114420742089c55b1c531a5b8cdfb2cadf359e97a3ad8d2204/mounts/shm
overlay           4729848 1957376   2756088  42% /data/misc/docker/overlay2/47eece143286cfea1dd58aeaa000d6acd23c4f7863ab790ab937c8af479b208c/merged
shm                 65536       0     65536   0% /data/misc/docker/containers/e019e9a323da37a147995afc45bb9ad5b13e2760d102a06ce8b991f076eff544/mounts/shm
overlay           4729848 1957376   2756088  42% /data/misc/docker/overlay2/83610fc1ab58ee074a388426c103789451e363fe0cba80798a994701e106d467/merged
shm                 65536       8     65528   1% /data/misc/docker/containers/76524615a1ce315db3d28e8e90ac501c6a6be42d282249f911f556abbc442e48/mounts/shm
overlay           4729848 1957376   2756088  42% /data/misc/docker/overlay2/c4eb9b8da7e3aa65ebd1ef0b23750e3b43024a0e98701f37c9135f3b580d477e/merged
shm                 65536       8     65528   1% /data/misc/docker/containers/af247f074ee78458e8e15b4a3ee8c9b18ca58e3f56f0694dd013674b6d58a4cf/mounts/shm
overlay           4729848 1957376   2756088  42% /data/misc/docker/overlay2/e124f85c18a5be5613f94f3a8223f8c6dfcec07e7ec3c13405131cd93dd3ad43/merged
shm                 65536       0     65536   0% /data/misc/docker/containers/216c3d4873c06a62bb49d02abb2d39dbbda2942cf8b54fc1b5fe6487f3cad658/mounts/shm
```

5. To get space back use this command to prune your files
   adb shell docker system prune -a -f

6. List out what is loaded on the camera
   adb shell docker images -a