

# Hands-on-lab

## Create a People Counting solution with Azure Percept DK and Azure Percept Vision

Prepared by Microsoft AEDPLS Customer Success Team

11-10-2021

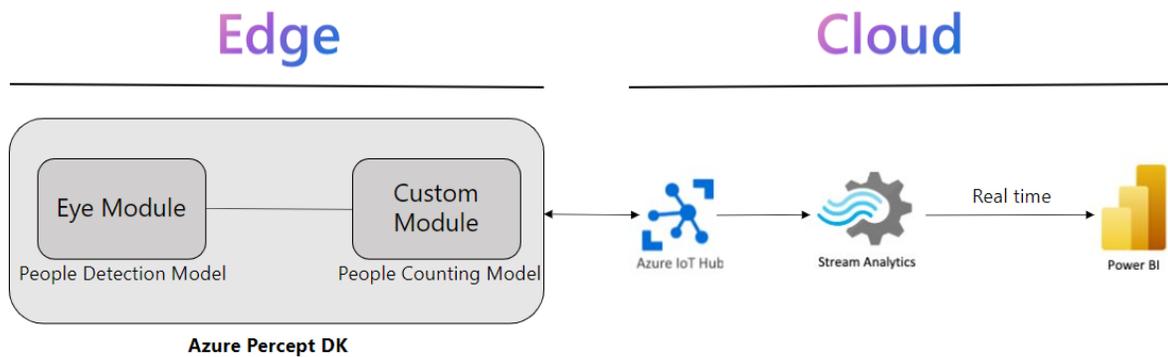
# Table of Contents

Lab Overview .....	3
Solution Architecture.....	3
Prerequisites .....	3
Set up your Azure Percept DK and Vision.....	4
Exercise 1 - Create a Container Registry resource (5 min).....	5
Exercise 2 - Setup for Edge Deployment (20 min).....	8
Ex 2. - Task 1 - Build and push your IoT edge solution (10 min).....	8
Ex 2. - Task 2 - Deploy edge module to device (10 min).....	11
Ex 2. - Task 3 - Deploy edge modules to device (No Docker steps) (10 min) .....	17
Exercise 3 - Add a consumer group to your IoT hub (5 min) .....	26
Exercise 4 – Set up Stream Analytics (25 min).....	27
Ex. 4 - Task 1 – Create a Stream Analytics Job (5 min) .....	27
Ex.4 - Task 2 - Add an input to the Stream Analytics job (5 min) .....	28
Ex. 4 - Task 3 - Add an output to the Stream Analytics job (5 min).....	29
Ex. 4 - Task 4 - Configure the query of the Stream Analytics job (5 min).....	31
Ex. 4 - Task 5 - Run the Stream Analytics job (2 min) .....	32
Exercise 5 – Create and publish a PowerBI report to visualize data (5 min).....	33
Summary .....	35

## Lab Overview

This lab will focus on detecting and counting people using the Azure Percept DK hardware, Azure IoT Hub, Azure Stream Analytics, and Power BI dashboard

## Solution Architecture



## Prerequisites

1. Azure Percept DK
2. Azure Percept Vision
3. [Azure Subscription](#)
4. [Azure Percept DK setup experience](#) - you connected your devkit to a Wi-Fi network, created an IoT Hub, and connected your devkit to the IoT Hub
5. Download and install [VS Code](#)
6. Download and Install [Git](#)
7. Install the IoT Hub Extension in VS Code
8. Install the Azure IoT Tools Extension in VS Code
9. Download and Install [Docker Desktop](#) (Will require PC restart)
10. **(Only for Windows Users)** Install WSL2 by running the following command in Windows PowerShell or Terminal (on MacOS) (Will require a PC restart)

```
wsl --install
```

```
wsl --set-default-version 2
```

## Set up your Azure Percept DK and Vision

1. Power on your Azure Percept DK
2. Connect the camera module to the Azure Percept DK via the USB-C cable
3. Open Command Prompt (on Windows) or Terminal (on MacOS) and execute the command-

git clone <https://github.com/leannhuang/people-counting-with-azure-percept-vision.git>

**Note- If you do not have Docker installed, please proceed to Exercise 2 – Task 3**

# Exercise 1 - Create a Container Registry resource

## (5 min)

1. Login to Azure Portal <https://portal.azure.com/>
2. To create a Container Registry, go to [Create container registry - Microsoft Azure](#)
  - a. Select your Azure Subscription in the **Subscription** drop down box
  - b. Select your preferred resource group from the **Resource group** drop down menu. If you would like to create a new resource group to use with your voice assistant, click **Create new** under the dropdown menu and follow the prompts
  - c. Provide a unique **Registry Name**
  - d. Under **Location**, select the region to deploy resource (We suggest select **West US**)
  - e. **Availability Zones** - disabled
  - f. For **SKU**, select **Standard**
  - g. Keep all other tab as default and click **Review + create** at the bottom of the screen. Once the validation passes, click **Create**. This will create your Container Registry.

[Home](#) > [Container registries](#) >

### Create container registry

Basics Networking Encryption Tags Review + create

Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. [Learn more](#)

**Project details**

Subscription \*

Resource group \*   
[Create new](#)

**Instance details**

Registry name \*  .azurecr.io

Location \*

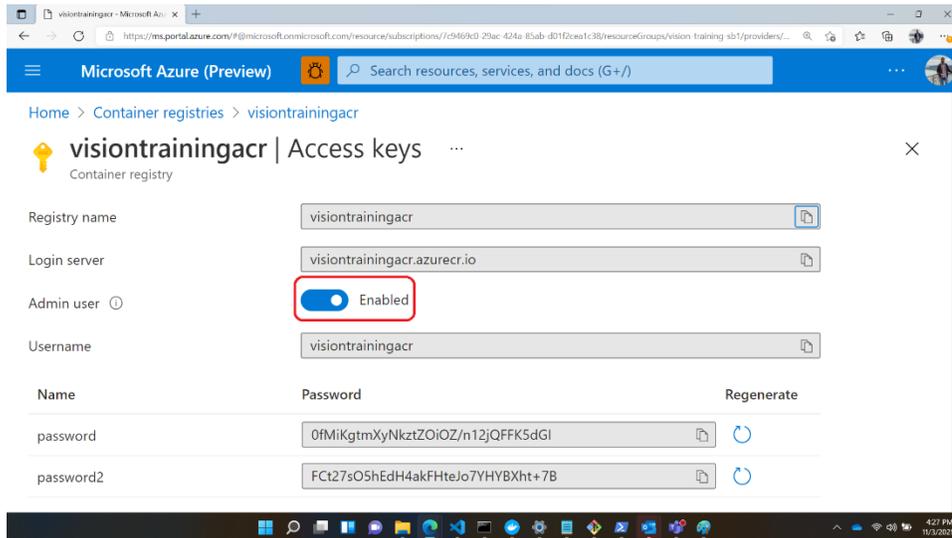
Availability zones  Enabled

Availability zones are enabled on premium registries and in regions that support availability zones. [Learn more](#)

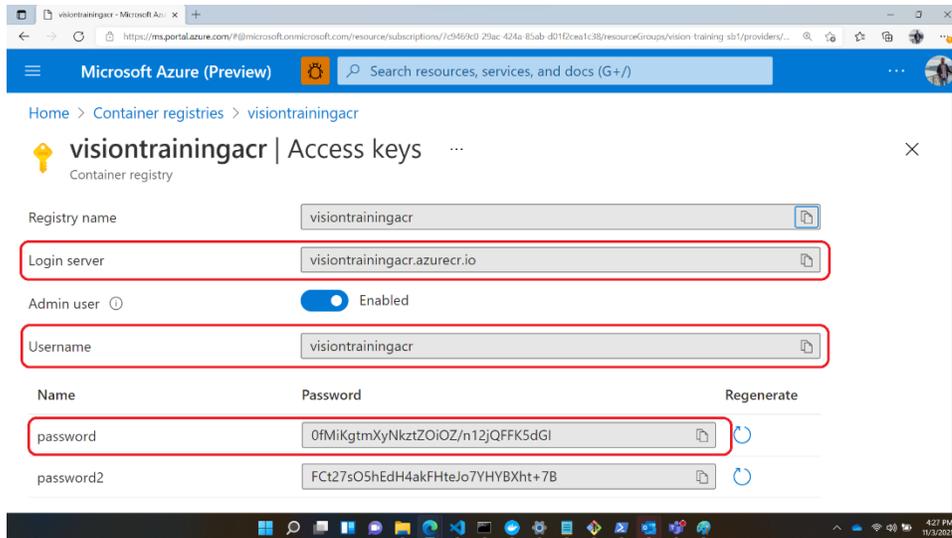
SKU \*

[Review + create](#) [< Previous](#) [Next: Networking >](#)

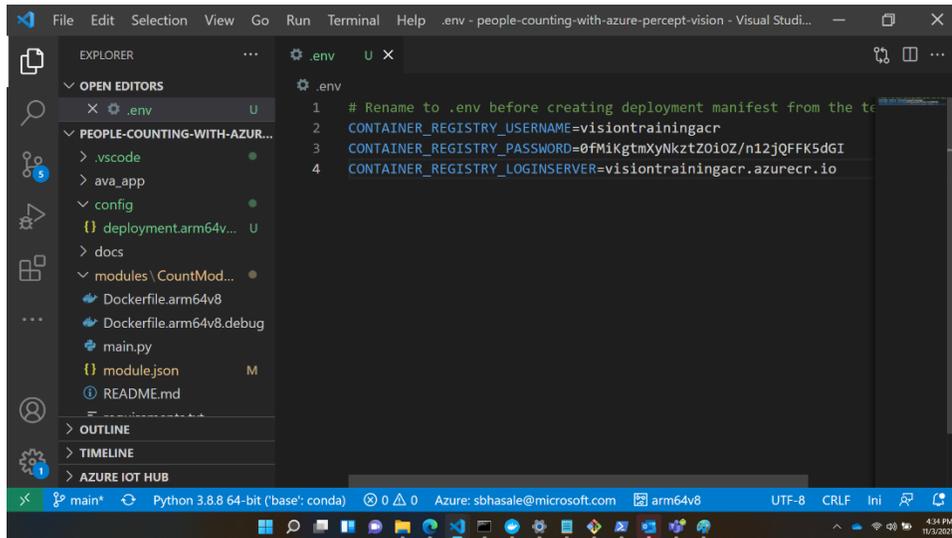
- After successful resource deployment go to your container registry resource. On the left scroll panel select **Access Keys** under **Settings** and enable the **Admin user**



- Make a note of the **Login Server**, **Username**, and **password**



5. Go to the git repository that you cloned in VS Code. Rename the file **envtemplate** to **.env** open the file and fill in the following details-
  - a. CONTAINER\_REGISTRY\_USERNAME=<your container registry Username>
  - b. CONTAINER\_REGISTRY\_PASSWORD=<your container registry Password>
  - c. CONTAINER\_REGISTRY\_LOGINSERVER=<your container registry Login Server>



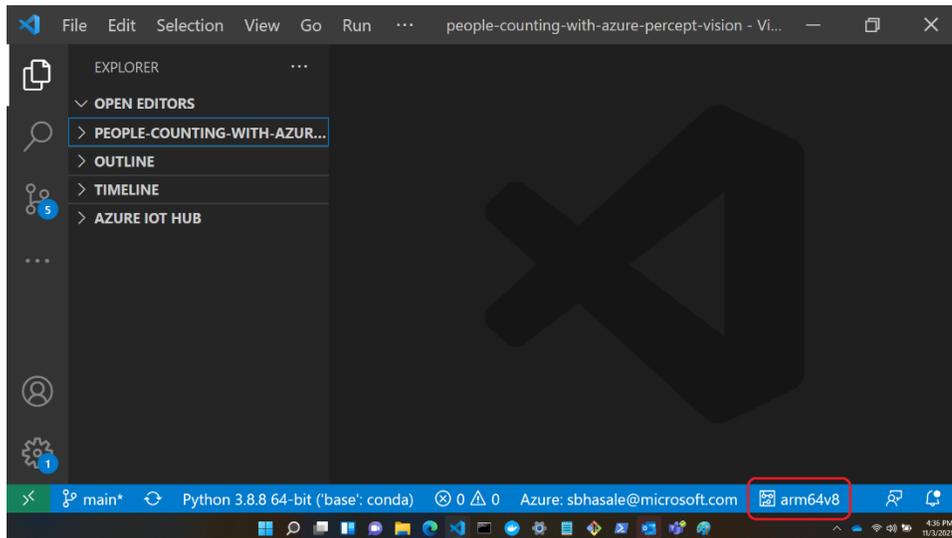
The screenshot shows the Visual Studio Code interface with the Explorer view on the left and the Editor view on the right. The Explorer view shows the file structure of a project named 'PEOPLE-COUNTING-WITH-AZUR...'. The Editor view shows the content of the '.env' file, which has been renamed from 'envtemplate'. The content of the '.env' file is as follows:

```
1 # Rename to .env before creating deployment manifest from the te
2 CONTAINER_REGISTRY_USERNAME=visiontrainingacr
3 CONTAINER_REGISTRY_PASSWORD=0fMiKgtnXyNkztZ0iOZ/n12jQFFK5dGI
4 CONTAINER_REGISTRY_LOGINSERVER=visiontrainingacr.azurecr.io
```

## Exercise 2 - Setup for Edge Deployment (20 min)

### Ex 2. - Task 1 - Build and push your IoT edge solution (10 min)

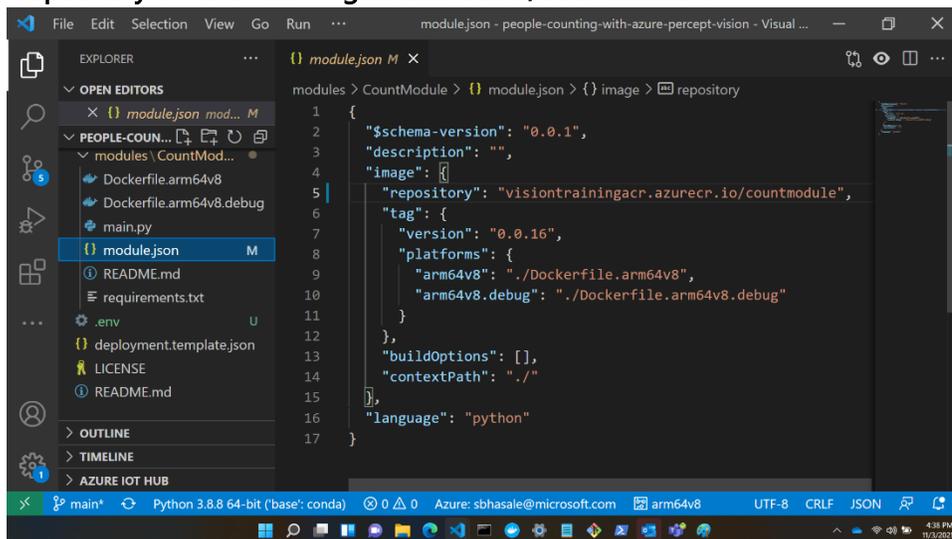
1. Open VS Code, at the bottom of the screen ensure you have **arm64v8** as the **Default Platform for IoT Edge Solution** selected (if not, then please click and select **arm64v8** from the list)



2. Go to `modules/CountModule/` directory and open `module.json`. Fill in your **Container registry address** (same as the **Login server** saved earlier) and followed by a **repository name** (**Note- please make sure your repository name is all lowercase**)

"repository": "<Your container registry login server/repository name>" will change as follows, for example-

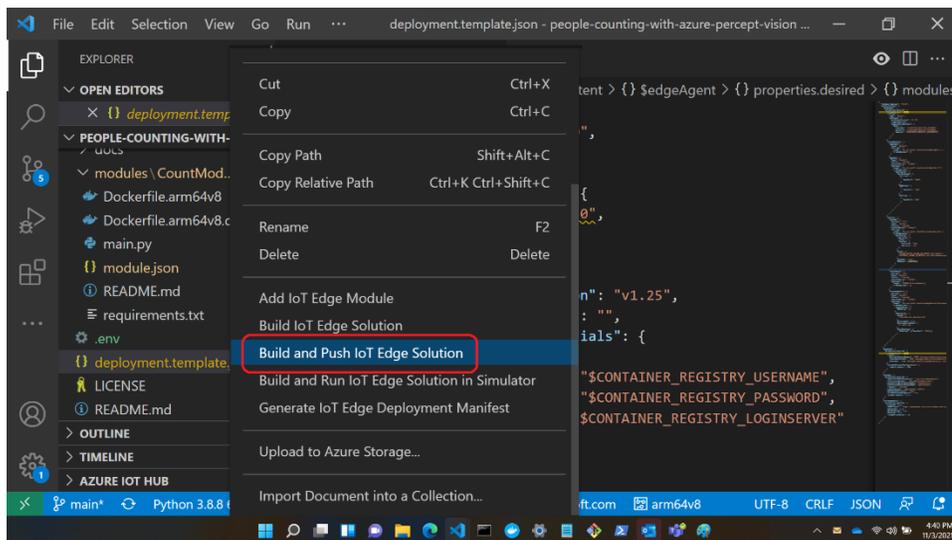
"repository": "visiontrainingacr.azurecr.io/countmodule"



- Now you will build the module image and push it to your container registry. Open Visual Studio Code integrated terminal by selecting **View > Terminal**
- Sign into Docker with the Azure Container registry (ACR) credentials that you saved after creating the registry using below command in terminal-

**docker login -u <ACR username> -p <ACR password> <ACR login server>**

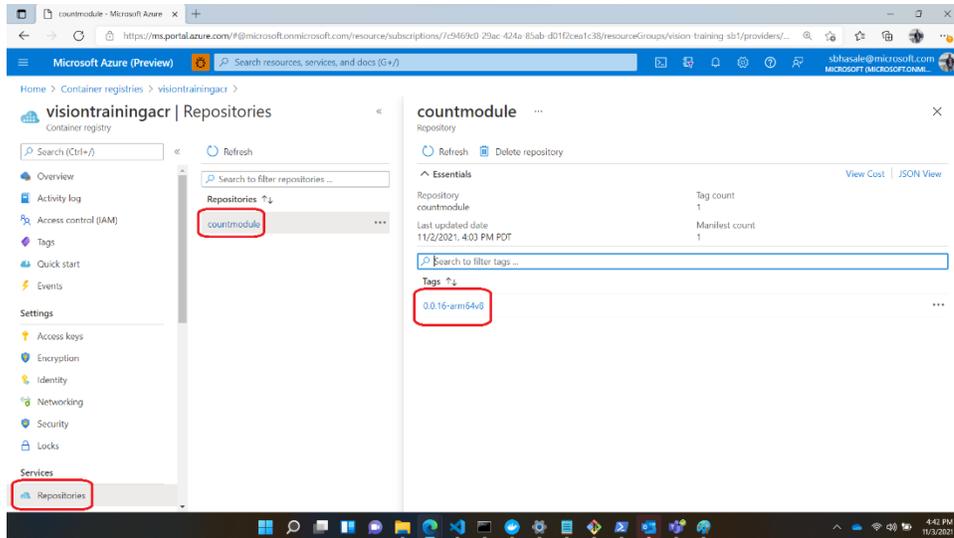
- Visual Studio Code now has access to your container registry. In the next steps you will turn the solution code into a container image. In Visual Studio Code explorer, right click the **deployment.template.json** file and select **Build and Push IoT Edge Solution**



The build and push command starts three operations. First, it creates a new folder in the solution called **config** that holds the full deployment manifest, built out of information in the deployment template and other solution files. Second, it runs **docker build** to build the container image based on the appropriate docker file for your target architecture. Then, it runs **docker push** to push the image repository to your container registry. This process may take several minutes the first time but is faster the next time that you run the commands.

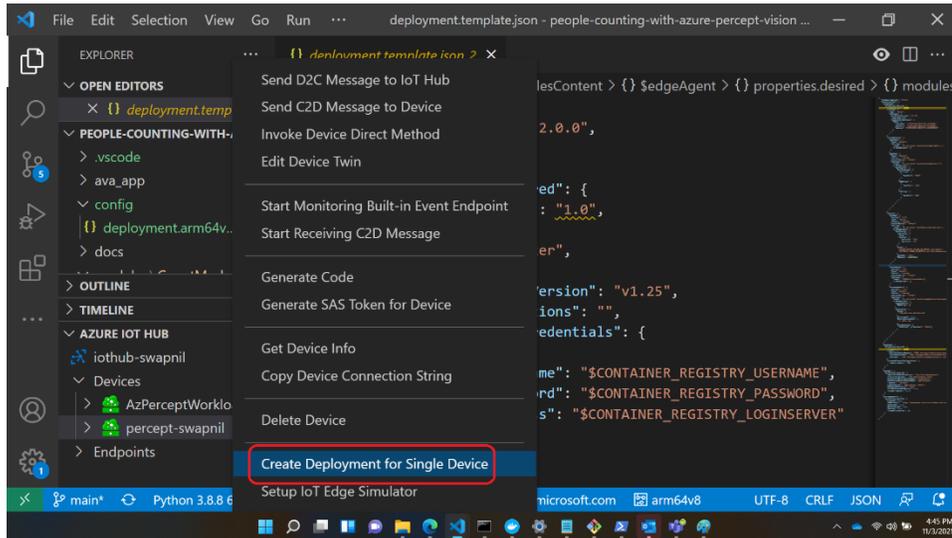
- Open the **deployment.arm64v8.json** file in the newly created **config** folder. The filename reflects the target architecture, so it will be different if you choose a different architecture.
- Notice that the two parameters that had placeholders now are filled in with their proper values. The **registryCredentials** section has your registry username and password pulled from the .env file. The **CountModule** has the full image repository with the **name**, **version**, and **architecture** tag from the **module.json** file.

- To further verify what the build and push command did, go to the Azure portal, and navigate to your container registry. In your container registry, select **Repositories** then **countmodule**



## Ex 2. - Task 2 - Deploy edge module to device (10 min)

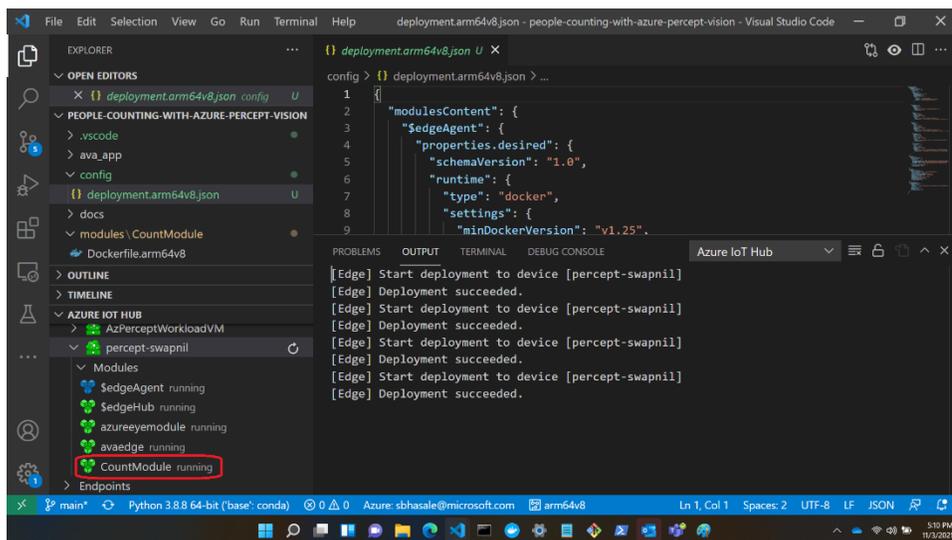
1. In the Visual Studio Code explorer, under the **Azure IoT Hub** section, expand **Devices** to see your list of IoT devices
2. Right-click the IoT Edge device that you want to deploy to, then select **Create Deployment for Single Device**



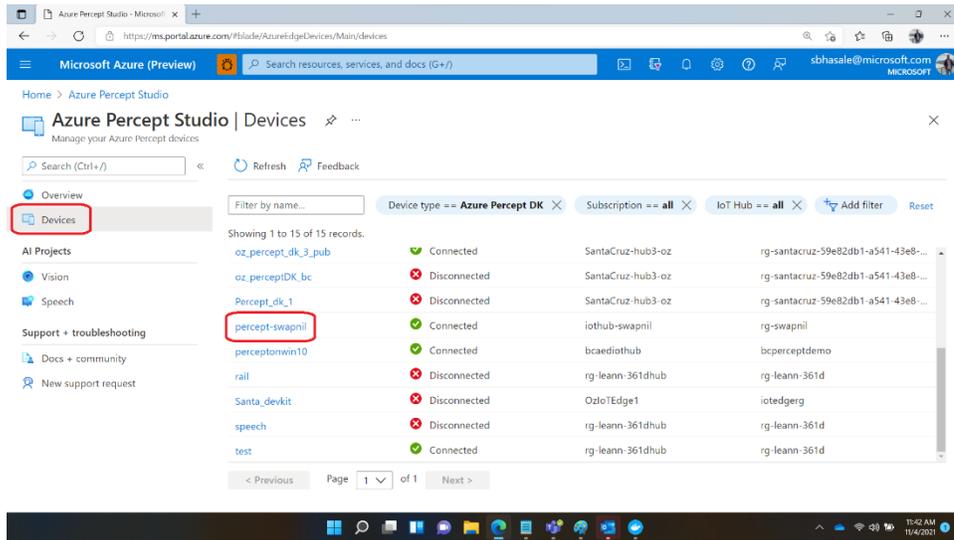
3. In the file explorer, navigate into the **config** folder then select the **deployment.amd64v8.json** file and click **Select Edge Deployment Manifest**.

**Do not use the deployment.template.json file, which does not have the container registry credentials or module image values in it.**

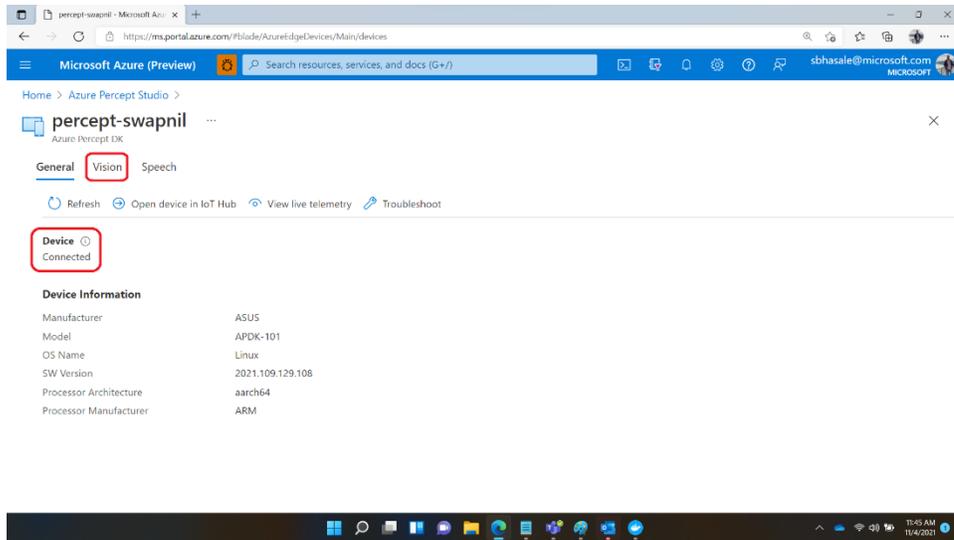
4. Under **OUTLINE** device, expand **Modules** to see a list of deployed and running modules. Click the refresh button. You should see the **CountModule** running on your device.



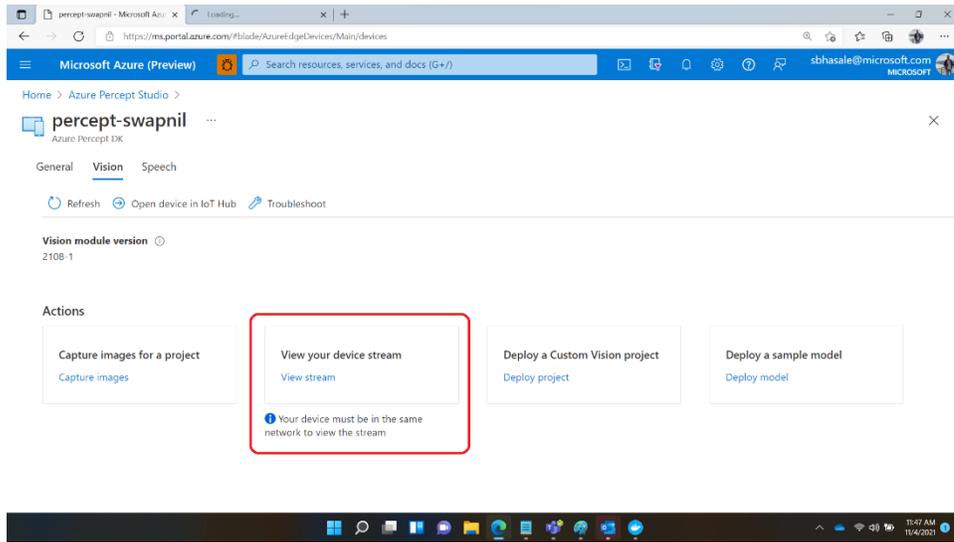
- Go to [Azure Percept Studio](#) and on the left panel, select **Devices**, then select your Azure Percept device



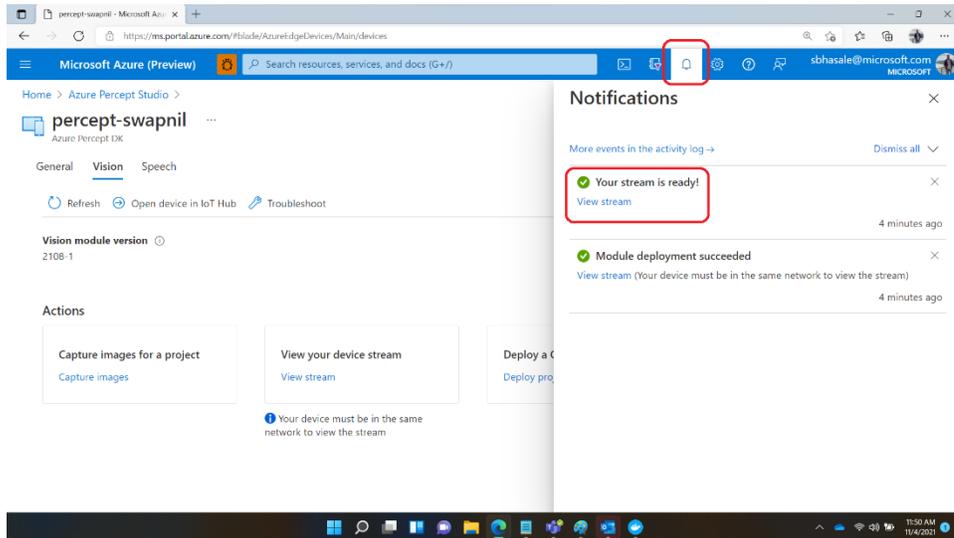
- Ensure that your device is **Connected**. Click on **Vision**



7. Click View your device stream

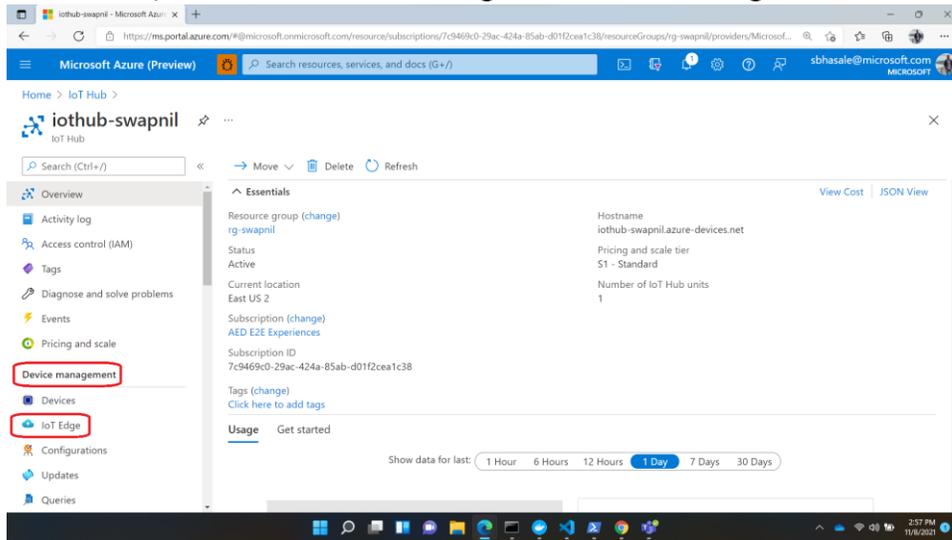


8. The previous step will deploy modules to your device. In the **Notifications** tab click **View Stream**. This will open a new tab in your browser, please verify that you see the video stream. If you point the camera module to a person then you will see the person detection with bounding box

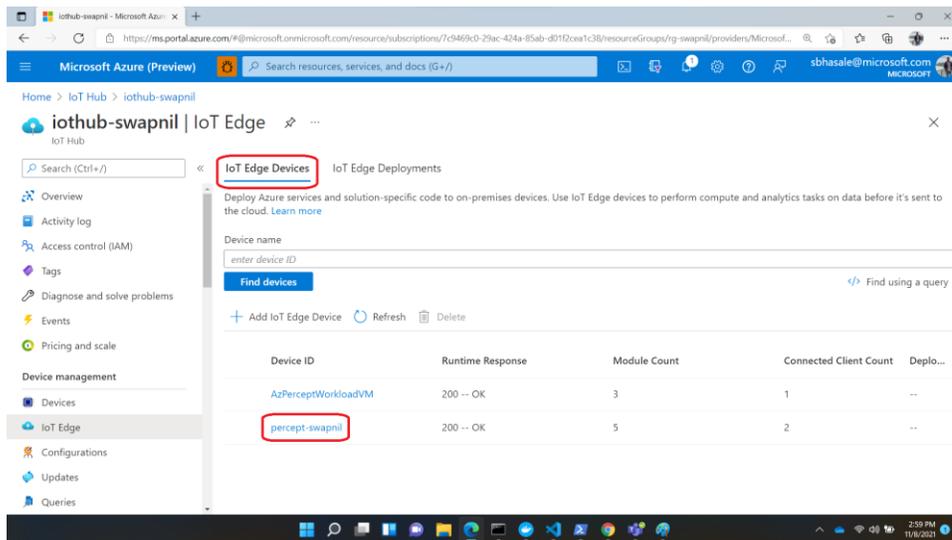


9. After verifying the video stream and bounding boxes, please **close the webstream** browser tab.

10. To ensure the Count Module is setup correctly, in the Azure Portal go to your IoT Hub. On the left panel under **Device management** select **IoT Edge**



11. From the IoT device list click on your Azure Percept DK device



## 12. Scroll down to check if all deployed modules are in **running** status

The screenshot shows the 'Modules' tab in the Azure IoT Hub console for the device 'percept-swapnil'. The table below lists the deployed modules and their runtime status. The 'Runtime Status' column is highlighted with a red box.

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
\$edgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running	0
\$edgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running	0
azureeyemodule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0
avaedge	Module Identity	NA	NA	NA	NA
WebStreamModule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0
CountModule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0

## 13. Click **Troubleshoot**

The screenshot shows the 'Troubleshoot' tab in the Azure IoT Hub console for the device 'percept-swapnil'. The 'Troubleshoot' tab is highlighted with a red box. The console displays various configuration fields for the device, including keys, connection strings, and runtime response.

Device ID: percept-swapnil

Primary Key: [Redacted]

Secondary Key: [Redacted]

Primary Connection String: [Redacted]

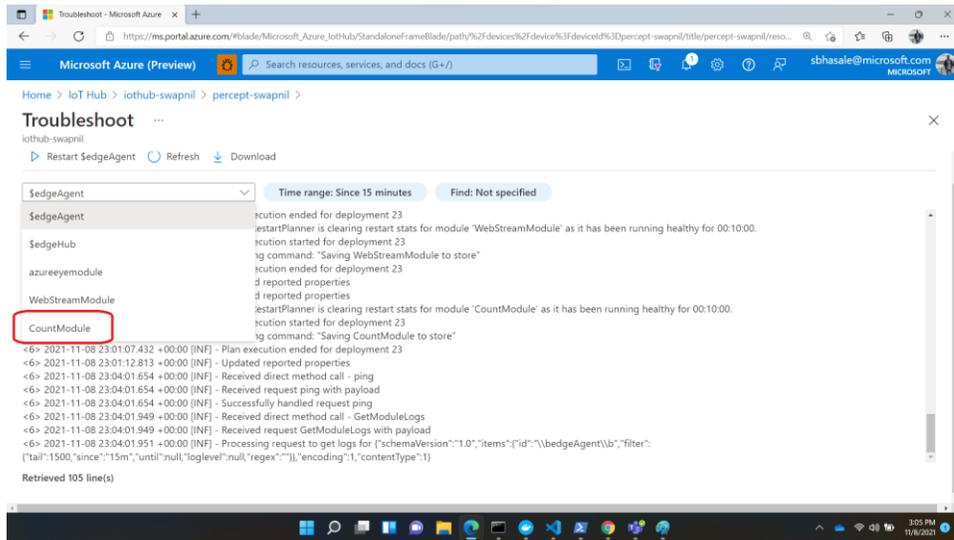
Secondary Connection String: [Redacted]

IoT Edge Runtime Response: 200 -- OK

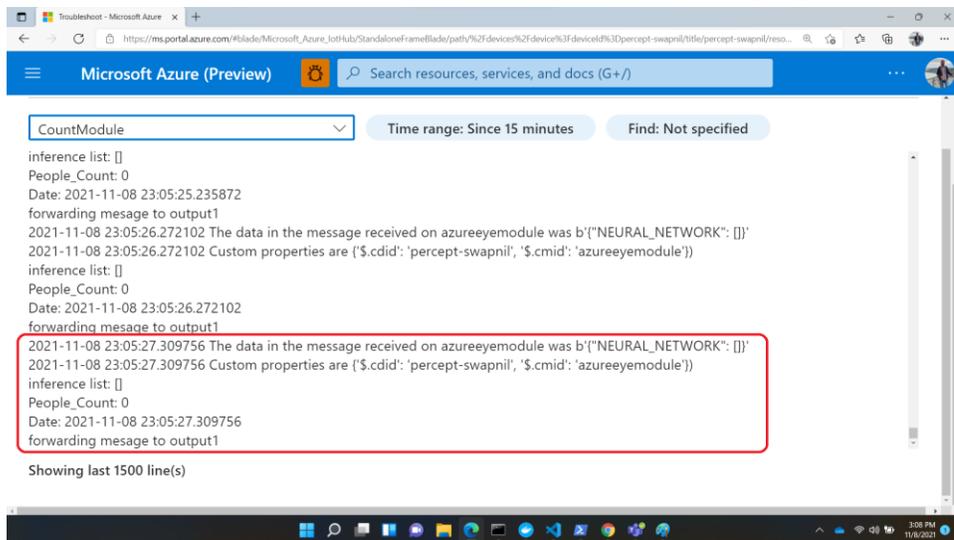
Enable connection to IoT Hub:  Enable  Disable

Parent device: No parent device

#### 14. From the drop-down list select CountModule



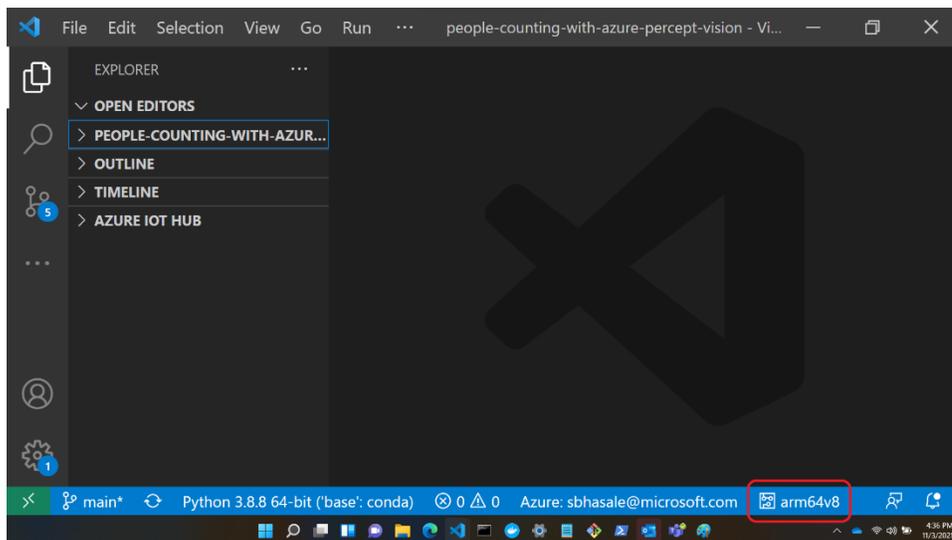
#### 15. Ensure you see People\_Count logs as follows-



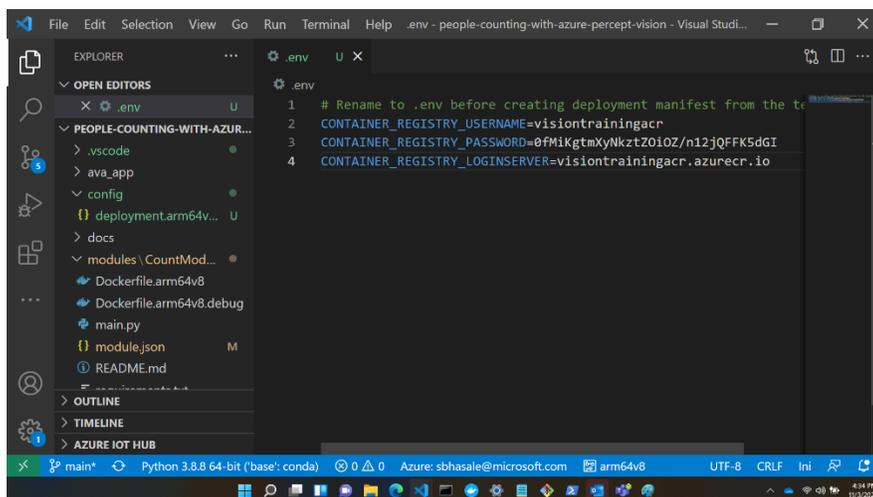
Note- The next task is only for participants who do not have Docker installed. If you have Docker installed, please proceed to Exercise 3.

## Ex 2. - Task 3 - Deploy edge modules to device (No Docker steps) (10 min)

1. Open VS Code, at the bottom of the screen ensure you have **arm64v8** as the **Default Platform for IoT Edge Solution** selected (if not, then please click and select **arm64v8** from the list)



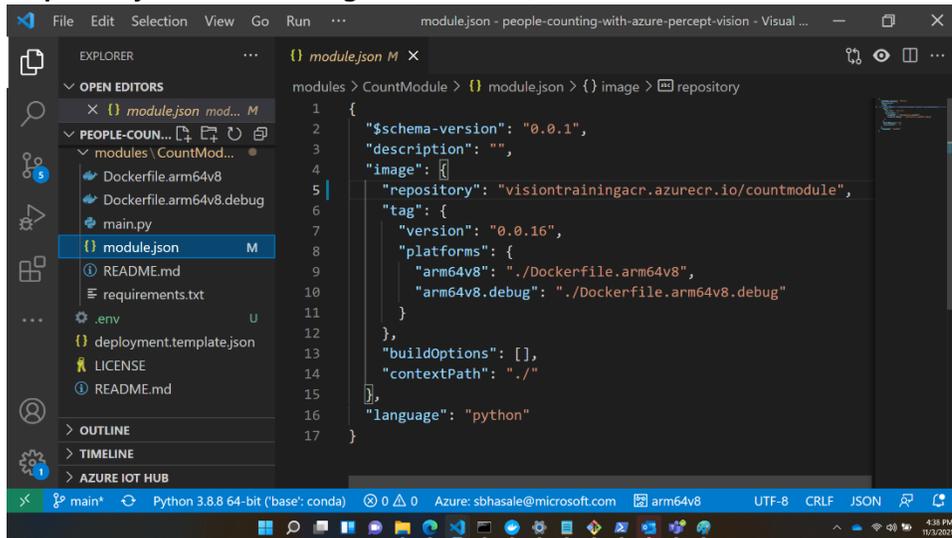
2. Go to the git repository that you cloned in VS Code. Rename the file **envtemplate** to **.env** open the file and fill in the following details
  - a. **CONTAINER\_REGISTRY\_USERNAME=visiontrainingacr**
  - b. **CONTAINER\_REGISTRY\_PASSWORD=0fMiKgtmXyNkztZOiOZ/n12jQFFK5dGI**
  - c. **CONTAINER\_REGISTRY\_LOGINSERVER=visiontrainingacr.azurecr.io**



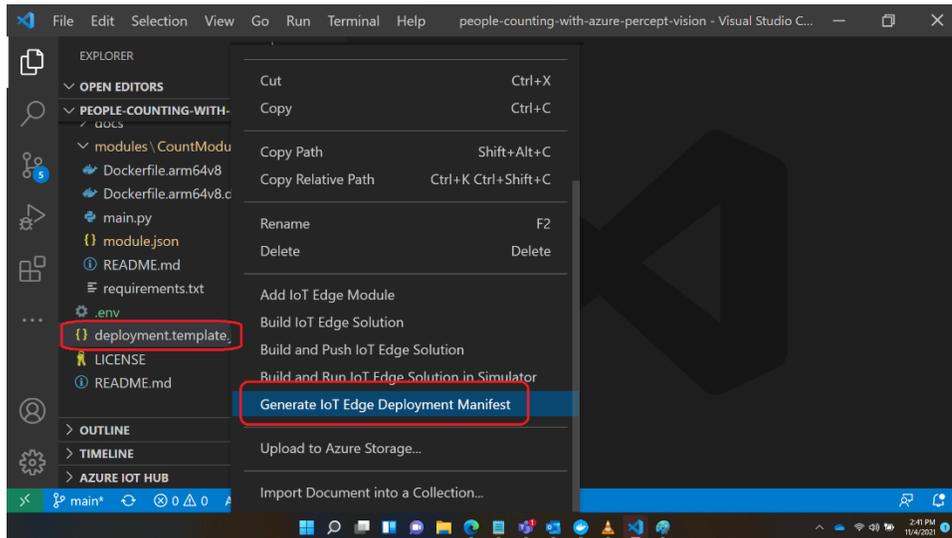
3. In the repository go to `modules/CountModule/` directory and open `module.json`. Fill in your Container registry address followed by a repository name

"repository": "<Your container registry login server/repository name>" will change as follows-

"repository": "visiontrainingacr.azurecr.io/countmodule"



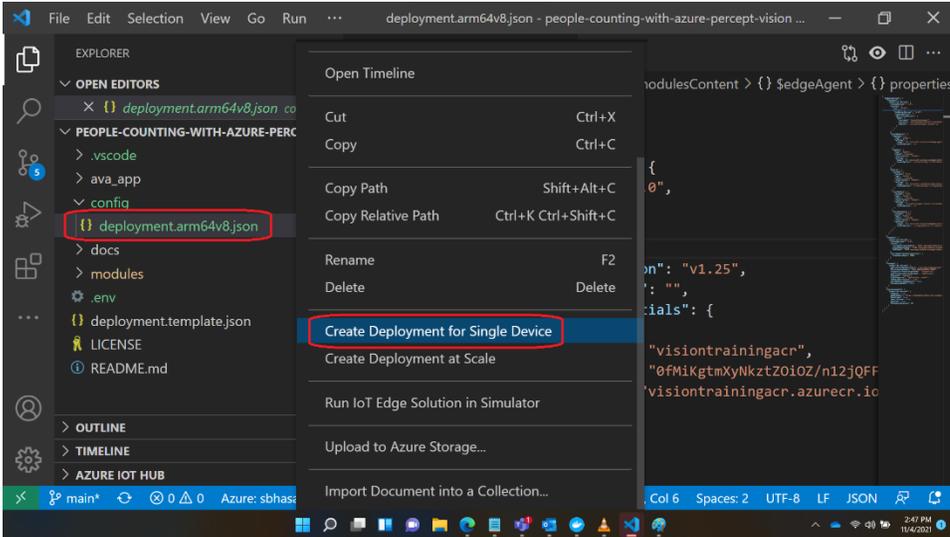
4. Right-click the `deployment.template.json` file and select **Generate IoT Edge Deployment Manifest** option



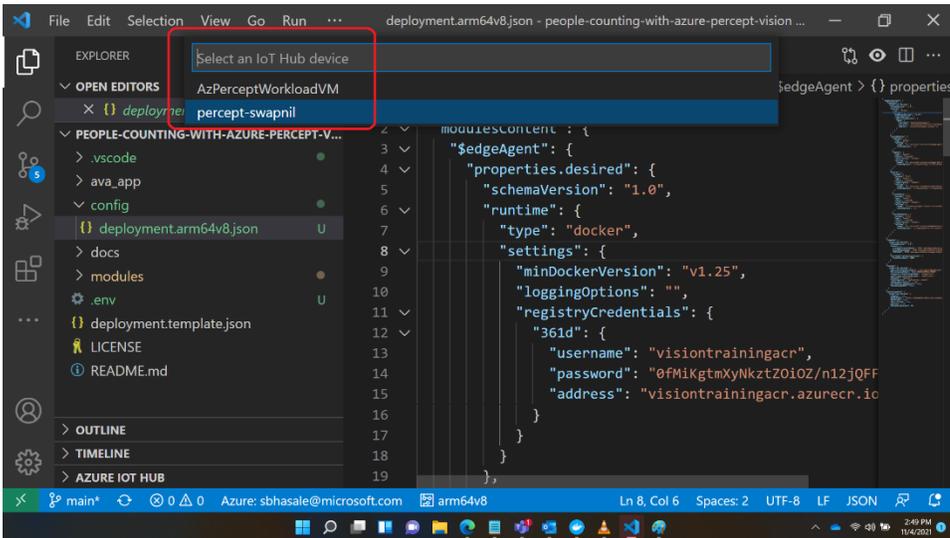
5. The above step will create a `config` folder in your project directory. Within this folder there is a file called `deployment.arm64v8.json`

- Go to the **config** folder and right-click the **deployment.arm64v8.json** file and select **Create Deployment for Single Device**

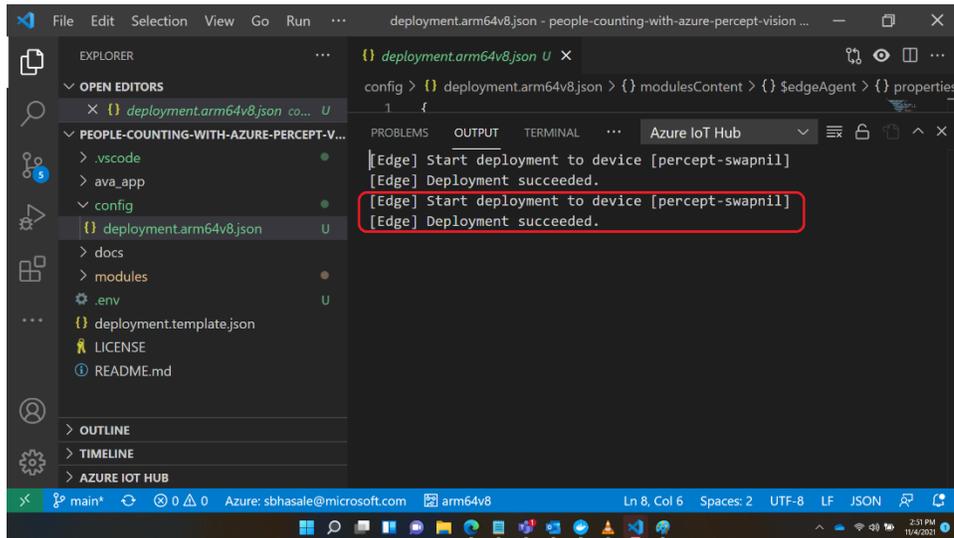
**Do not** use the **deployment.template.json** file, which does not have the container registry credentials or module image values in it.



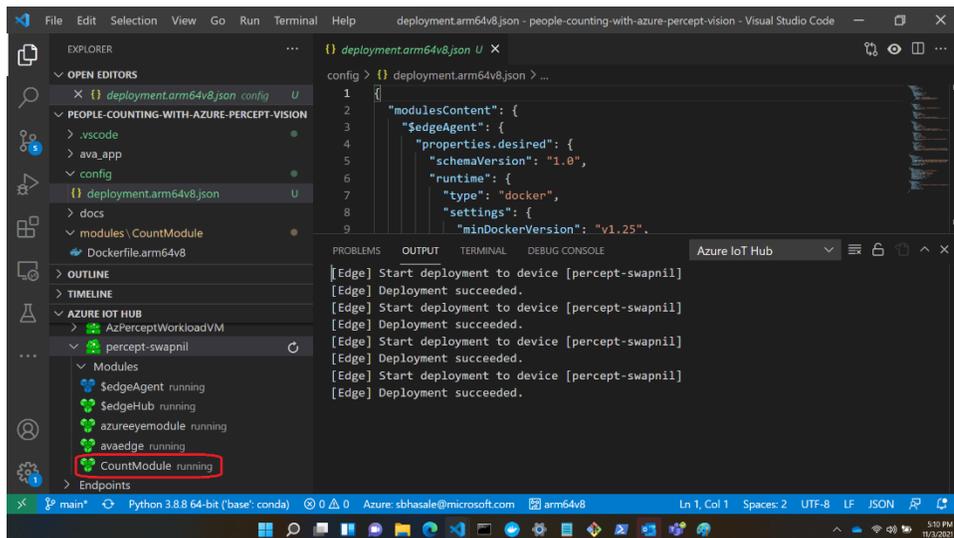
- Select your IoT hub device



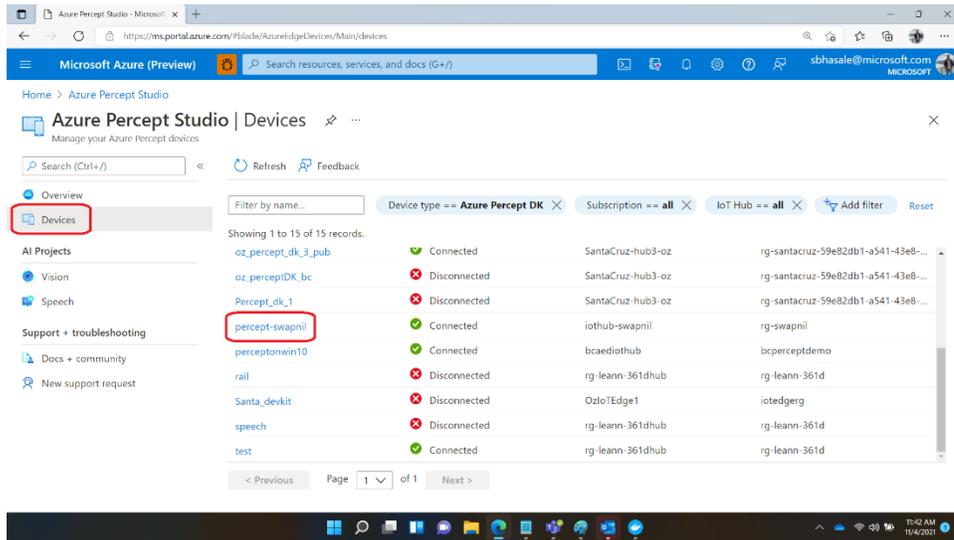
- Once you select your IoT hub device the deployment will start, ensure that the deployment succeeds in the terminal window



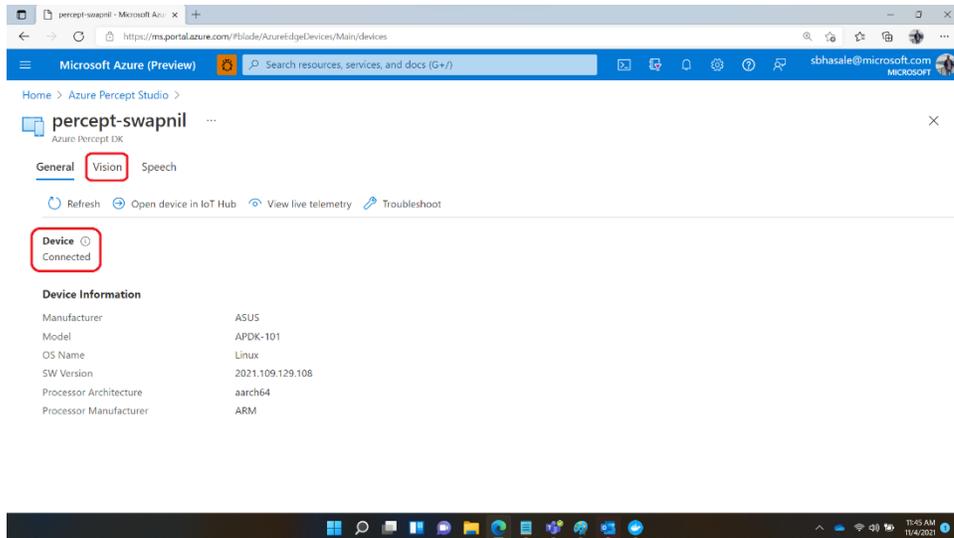
- Under your device, expand **Modules** to see a list of deployed and running modules. Click the refresh button. You should see the **CountModule** running on your device.



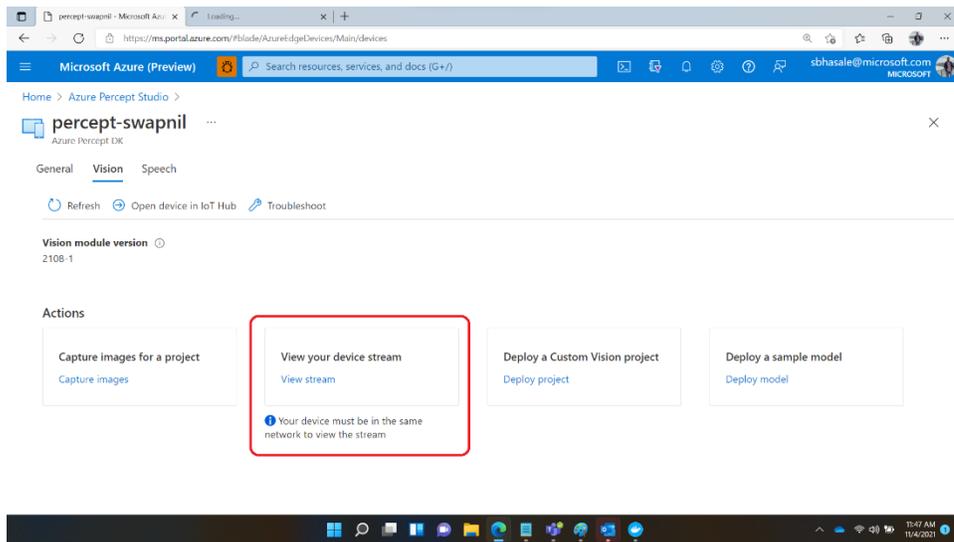
16. Go to [Azure Percept Studio](#) and on the left panel, select **Devices**, then select your Azure Percept device



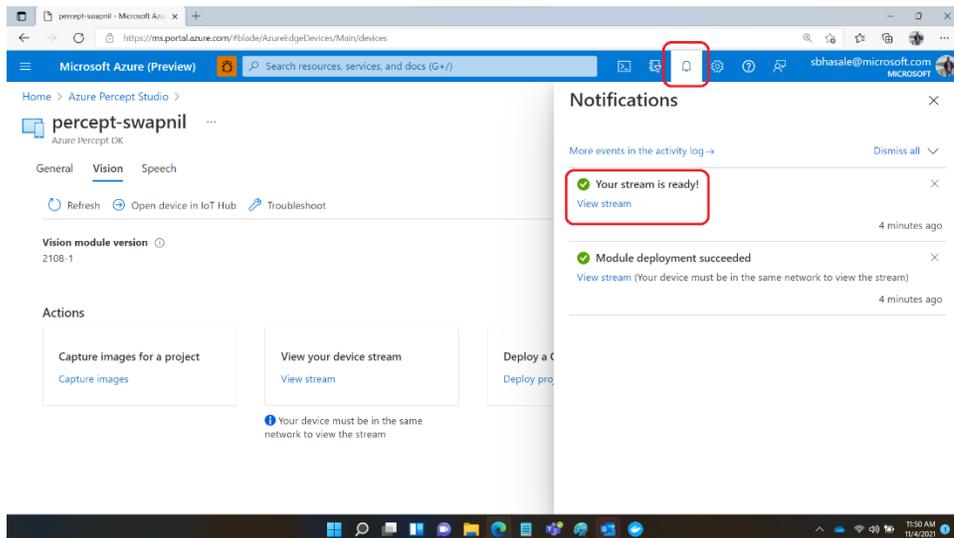
17. Ensure that your device is **Connected**. Click on **Vision**



## 18. Click View your device stream

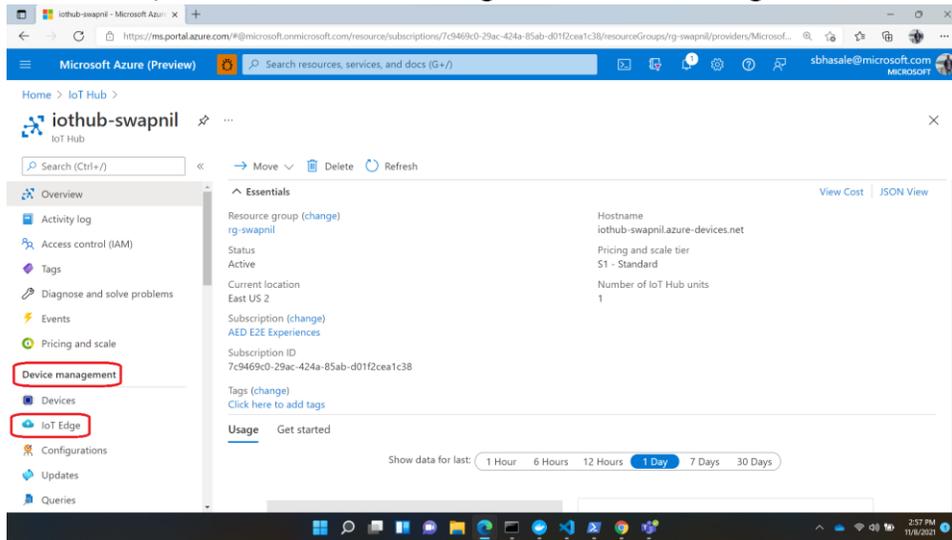


19. The previous step will deploy modules to your device. In the **Notifications** tab click **View Stream**. This will open a new tab in your browser, please verify that you see the video stream. If you point the camera module to a person then you will see the person detection with bounding box

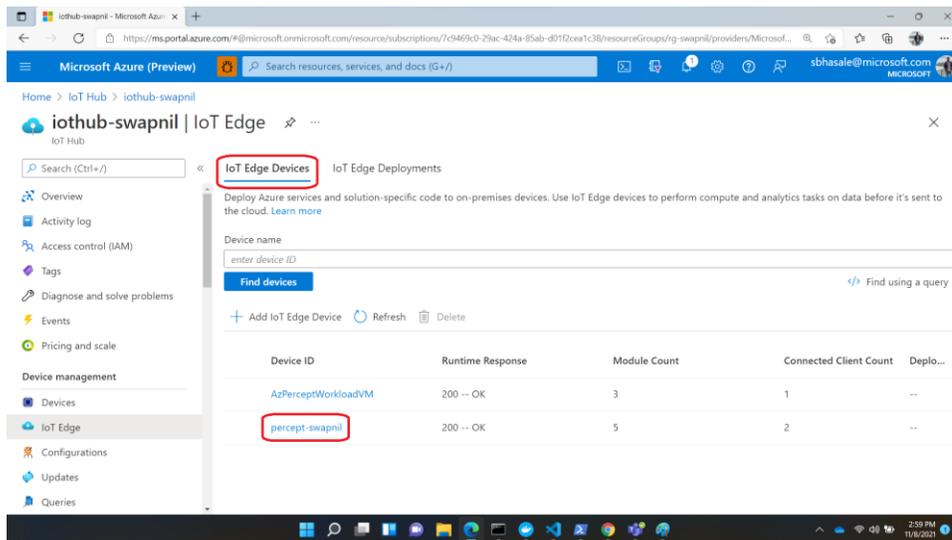


20. After verifying the video stream and bounding boxes, please **close the webstream** browser tab.

21. To ensure the Count Module is setup correctly, in the Azure Portal go to your IoT Hub. On the left panel under **Device management** select **IoT Edge**



22. From the IoT device list click on your Azure Percept DK device



## 23. Scroll down to check if all deployed modules are in **running** status

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
EdgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running	0
EdgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running	0
azureeyemodule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0
avaedge	Module Identity	NA	NA	NA	NA
WebStreamModule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0
CountModule	IoT Edge Custom Module	✓ Yes	✓ Yes	running	0

## 24. Click **Troubleshoot**

Device ID: percept-swapnil

Primary Key: [Redacted]

Secondary Key: [Redacted]

Primary Connection String: [Redacted]

Secondary Connection String: [Redacted]

IoT Edge Runtime Response: 200 -- OK

Enable connection to IoT Hub:  Enable  Disable

Parent device: No parent device

## 25. From the drop-down list select CountModule

Microsoft Azure (Preview) | Search resources, services, and docs (G+)

Home > IoT Hub > iotHub-swapnil > percept-swapnil > Troubleshoot

Restart SedgeAgent Refresh Download

SedgeAgent Time range: Since 15 minutes Find: Not specified

SedgeAgent execution ended for deployment 23  
startPlanner is clearing restart stats for module 'WebStreamModule' as it has been running healthy for 00:10:00.

SedgeHub execution started for deployment 23  
tg command: "Saving WebStreamModule to store"

azureeyemodule execution ended for deployment 23  
d reported properties

WebStreamModule d reported properties

**CountModule** startPlanner is clearing restart stats for module 'CountModule' as it has been running healthy for 00:10:00.  
execution started for deployment 23  
tg command: "Saving CountModule to store"

```
<6> 2021-11-08 23:01:07.432 +00:00 [INF] - Plan execution ended for deployment 23
<6> 2021-11-08 23:01:12.813 +00:00 [INF] - Updated reported properties
<6> 2021-11-08 23:04:01.654 +00:00 [INF] - Received direct method call - ping
<6> 2021-11-08 23:04:01.654 +00:00 [INF] - Received request ping with payload
<6> 2021-11-08 23:04:01.654 +00:00 [INF] - Successfully handled request ping
<6> 2021-11-08 23:04:01.949 +00:00 [INF] - Received direct method call - GetModuleLogs
<6> 2021-11-08 23:04:01.949 +00:00 [INF] - Received request GetModuleLogs with payload
<6> 2021-11-08 23:04:01.951 +00:00 [INF] - Processing request to get logs for ("schemaVersion":"1.0","items":{"id":"\sedgeAgent\"},"filter":{"tail":"1500","since":"15m","until":"null","loglevel":"null","regex":"","encoding":"1","contentType":"1"}
Retrieved 105 line(s)
```

## 26. Ensure you see People\_Count logs as follows-

Microsoft Azure (Preview) | Search resources, services, and docs (G+)

CountModule Time range: Since 15 minutes Find: Not specified

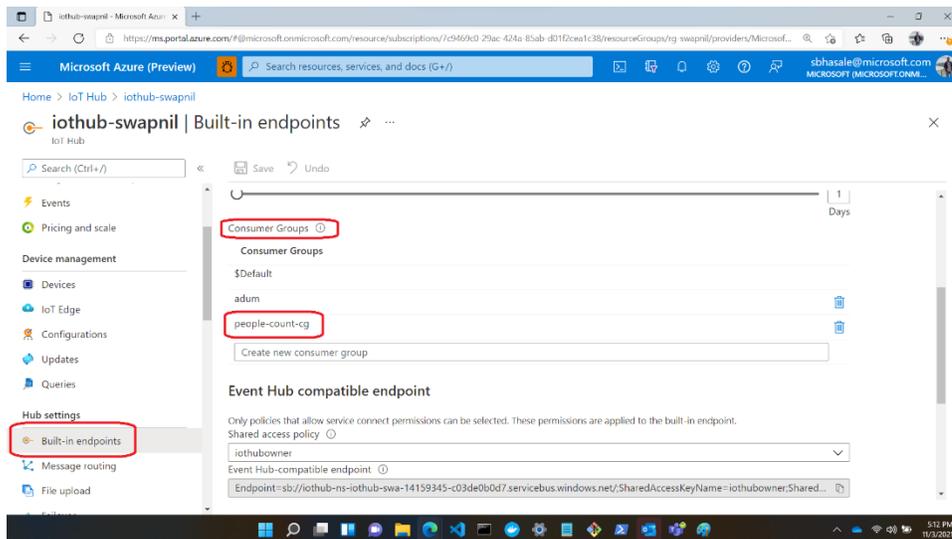
inference list: []  
People\_Count: 0  
Date: 2021-11-08 23:05:25.235872  
forwarding message to output1  
2021-11-08 23:05:26.272102 The data in the message received on azureeyemodule was b'{"NEURAL\_NETWORK": []}'  
2021-11-08 23:05:26.272102 Custom properties are {'\$.cid': 'percept-swapnil', '\$.cmid': 'azureeyemodule'})  
inference list: []  
People\_Count: 0  
Date: 2021-11-08 23:05:26.272102  
forwarding message to output1  
2021-11-08 23:05:27.309756 The data in the message received on azureeyemodule was b'{"NEURAL\_NETWORK": []}'  
2021-11-08 23:05:27.309756 Custom properties are {'\$.cid': 'percept-swapnil', '\$.cmid': 'azureeyemodule'})  
inference list: []  
People\_Count: 0  
Date: 2021-11-08 23:05:27.309756  
forwarding message to output1

Showing last 1500 line(s)

## Exercise 3 - Add a consumer group to your IoT hub (5 min)

Consumer groups provide independent views into the event stream that enable apps and Azure services to independently consume data.

1. In the [Azure portal](#), go to your IoT hub
2. On the left pane, select **Hub settings** > **Built-in endpoints**. Enter a name for your new consumer group in the text box under **Consumer groups**



3. Click anywhere outside the text box to save the consumer group

## Exercise 4 – Set up Stream Analytics (25 min)

### Ex. 4 - Task 1 – Create a Stream Analytics Job (5 min)

1. Go to [New Stream Analytics job - Microsoft Azure](#)
2. Enter the following information for the job -
  - a. **Job name** - The name of the job. The name must be globally unique.
  - b. **Resource group** - Use the same resource group that your IoT hub uses.
  - c. **Location** - Use the same location as your resource group.

The screenshot shows the 'New Stream Analytics job' page in the Microsoft Azure portal. The page title is 'New Stream Analytics job'. Below the title is a blue bar with the text 'Microsoft Azure (Preview)' and a search bar. The breadcrumb navigation is 'Home > Stream Analytics jobs >'. The main heading is 'New Stream Analytics job'. Below this is a blue box with an information icon and the text: 'This will create a new Stream Analytics job. You will be charged according to Azure Stream Analytics bill'. The form contains the following fields:

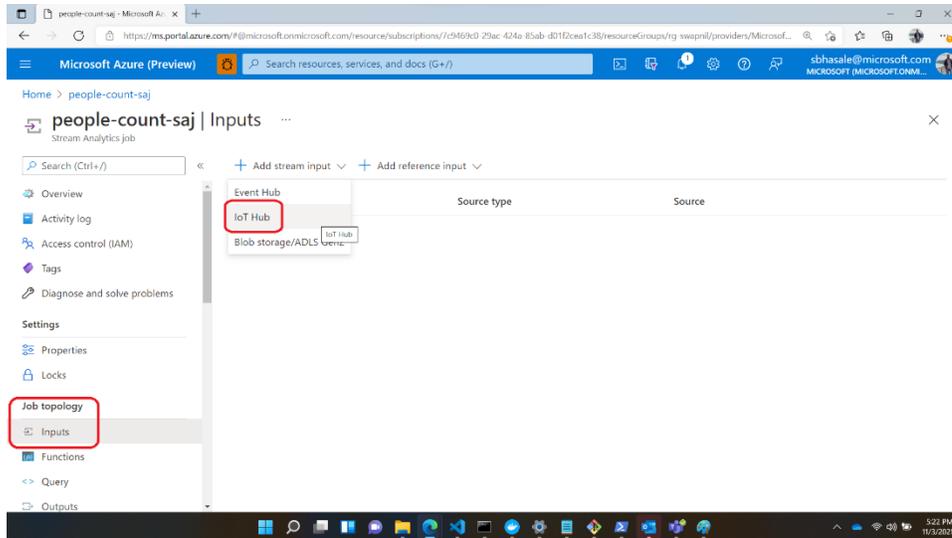
- Job name \***: A text input field containing 'people-count-saj' with a green checkmark on the right.
- Subscription \***: A dropdown menu showing 'AED E2E Experiences'.
- Resource group \***: A dropdown menu showing 'vision-training-sb1' with a 'Create new' link below it.
- Location \***: A dropdown menu showing 'West US'.
- Hosting environment**: Two radio buttons, 'Cloud' (selected) and 'Edge'.
- Streaming units (1 to 192)**: A slider control with a value of 3.
- Secure all private data assets needed by this job in my Storage account.**

At the bottom of the form is a blue 'Create' button.

3. Click **Create**

## Ex.4 - Task 2 - Add an input to the Stream Analytics job (5 min)

1. Open the previously created Stream Analytics job. Under **Job topology**, select **Inputs**
2. In the **Inputs** pane, select **Add stream input**, then select **IoT Hub** from the drop-down list.



3. On the new input pane, enter the following information -
  - a. **Input alias** - Enter a unique alias for the input
  - b. **Select IoT Hub from your subscription** - Select this radio button
  - c. **Subscription** - Select the Azure subscription you are using for this lab
  - d. **IoT Hub** - Select the IoT Hub you are using for this lab
  - e. **Consumer group** - Select the consumer group you created previously
  - f. **Shared access policy name** - Select the name of the shared access policy you want the Stream Analytics job to use for your IoT hub. For this lab, you can select **service**
  - g. **Shared access policy key** - This field is auto filled based on your selection for the shared access policy name
  - h. **Endpoint** - Select Messaging

Leave all other fields as default

**IoT Hub** ✕

New input

Input alias \*  
 ✓

Provide IoT Hub settings manually  
 Select IoT Hub from your subscriptions

Subscription  
 ▾

IoT Hub \* ⓘ  
 ▾

Consumer group \* ⓘ  
 ▾

Shared access policy name \* ⓘ  
 ▾

Shared access policy key ⓘ

Endpoint ⓘ  
 ▾

Partition key ⓘ

**Save**

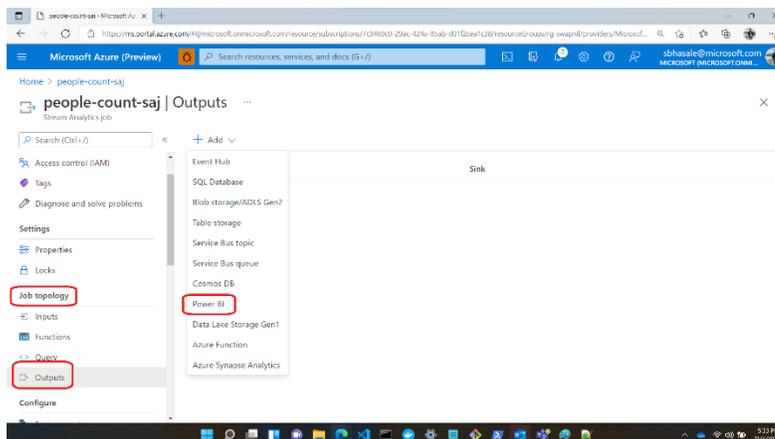
4. Click **Save**

Ex. 4 - Task 3 - Add an output to the Stream Analytics job (5 min)

Create a Group Workspace, take the following steps to create one-

- a. In a new Web Browser tab open [Power BI](#)
- b. On the left panel click on **Workspaces > Create a workspace**
- c. Give your workspace a name and description (optional) and click **Save**
- d. Go back to the Azure Portal and go to the Stream Analytics job

1. Under **Job topology**, select **Outputs**
2. In the **Outputs** pane, select **Add**, and then select **Power BI** from the drop-down list



3. Enter the following information -
  - a. **Output alias** - A unique alias for the output
  - b. **Group workspace** - Select your target group workspace.
  - c. **Dataset name** - Enter a dataset name
  - d. **Table name** - Enter a table name
  - e. **Authentication mode** - Leave at the default

**Power BI** ×

New output

Output alias \*  
people-count-output-sb ✓

Provide Group workspace settings manually  
 Select Group workspace from your subscriptions

Group workspace \*  
Azure-Percept-Reference-Solutions ✓

Authentication mode  
User token ✓

Dataset name \* ⓘ  
peoplecountdataset ✓

Table name \*  
peoplecounttable ✓

Currently authorized as Swapnil Sunilkumar Bhasale  
(sbhasale@microsoft.com)

**Authorize connection**  
You'll need to authorize with Power BI to configure your output settings.

Authorize

Save

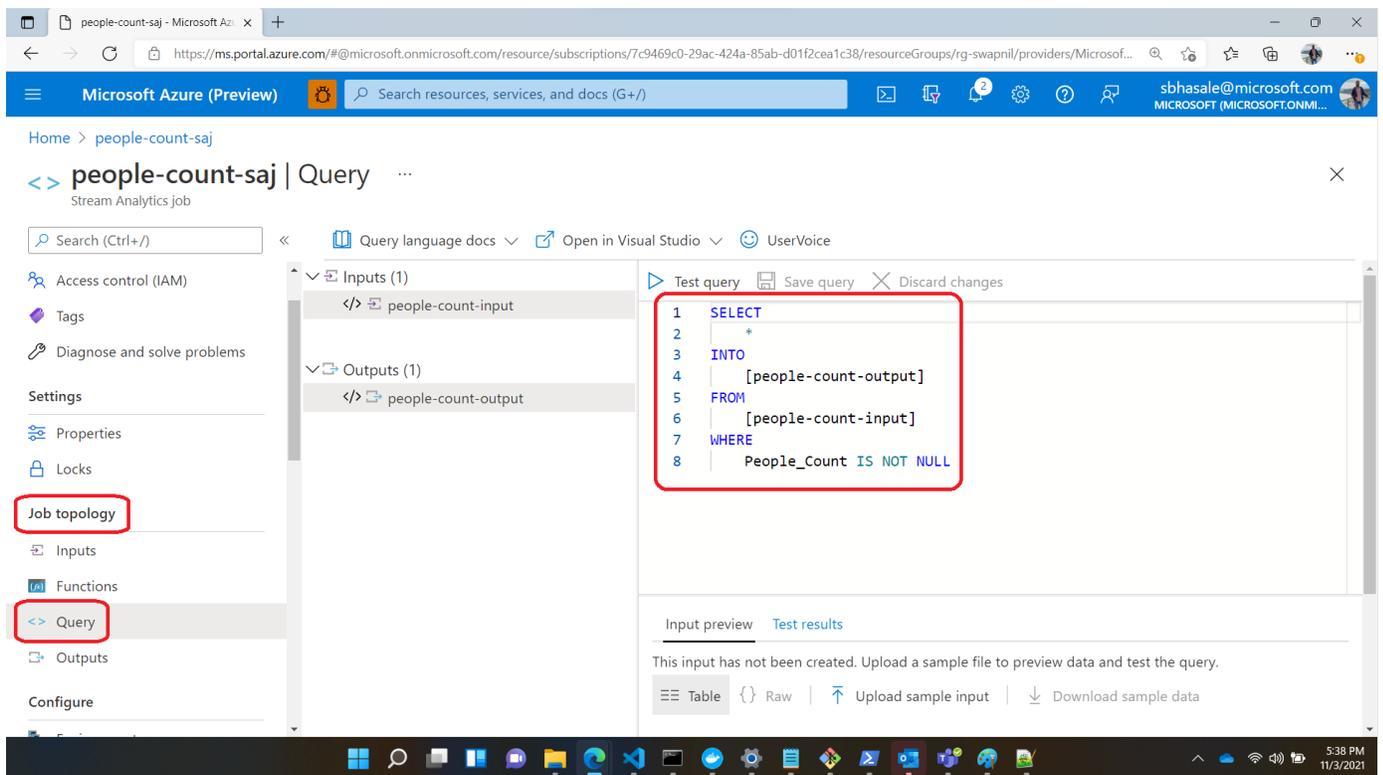
4. On the **Power BI - New output** pane, select **Authorize** and follow the prompts to sign into your Power BI account
5. Click **Save**

## Ex. 4 - Task 4 - Configure the query of the Stream Analytics job (5 min)

1. Under **Job topology**, select **Query**
2. Replace **[YourInputAlias]** with the input alias of the job
3. Replace **[YourOutputAlias]** with the output alias of the job
4. Add the following **WHERE** clause as the last line of the query. This line ensures that only messages with a **People\_Count** property will be forwarded to Power BI.

**WHERE People\_Count IS NOT NULL**

5. The query will look as follows -



The screenshot shows the Microsoft Azure portal interface for configuring a Stream Analytics job. The job name is 'people-count-saj'. The 'Query' tab is selected in the left-hand navigation pane, which is highlighted with a red box. The main area displays the query editor with the following SQL query:

```
1 SELECT
2 *
3 INTO
4 [people-count-output]
5 FROM
6 [people-count-input]
7 WHERE
8 People_Count IS NOT NULL
```

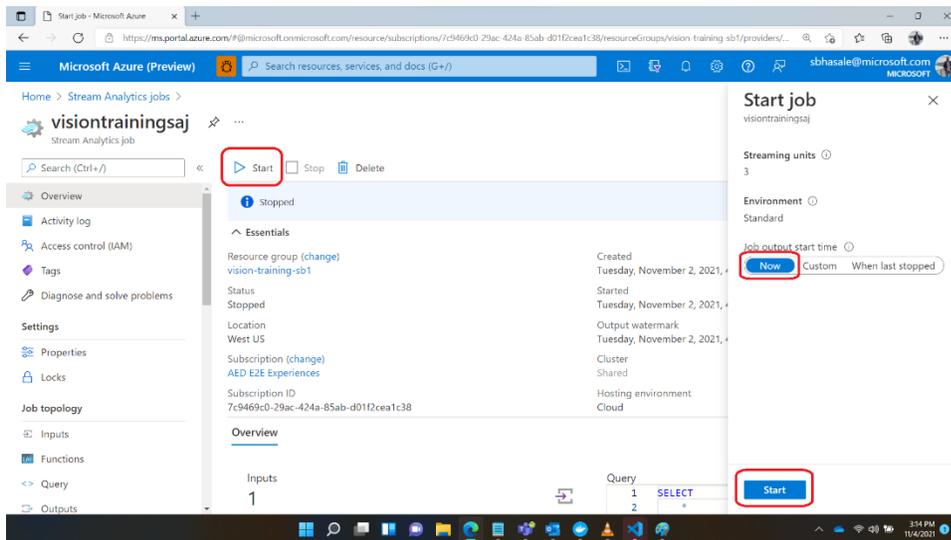
The query text is enclosed in a red box. Below the query editor, there are tabs for 'Input preview' and 'Test results'. A message states: 'This input has not been created. Upload a sample file to preview data and test the query.' The system tray at the bottom right shows the time as 5:38 PM on 11/3/2021.

6. Click **Save Query**

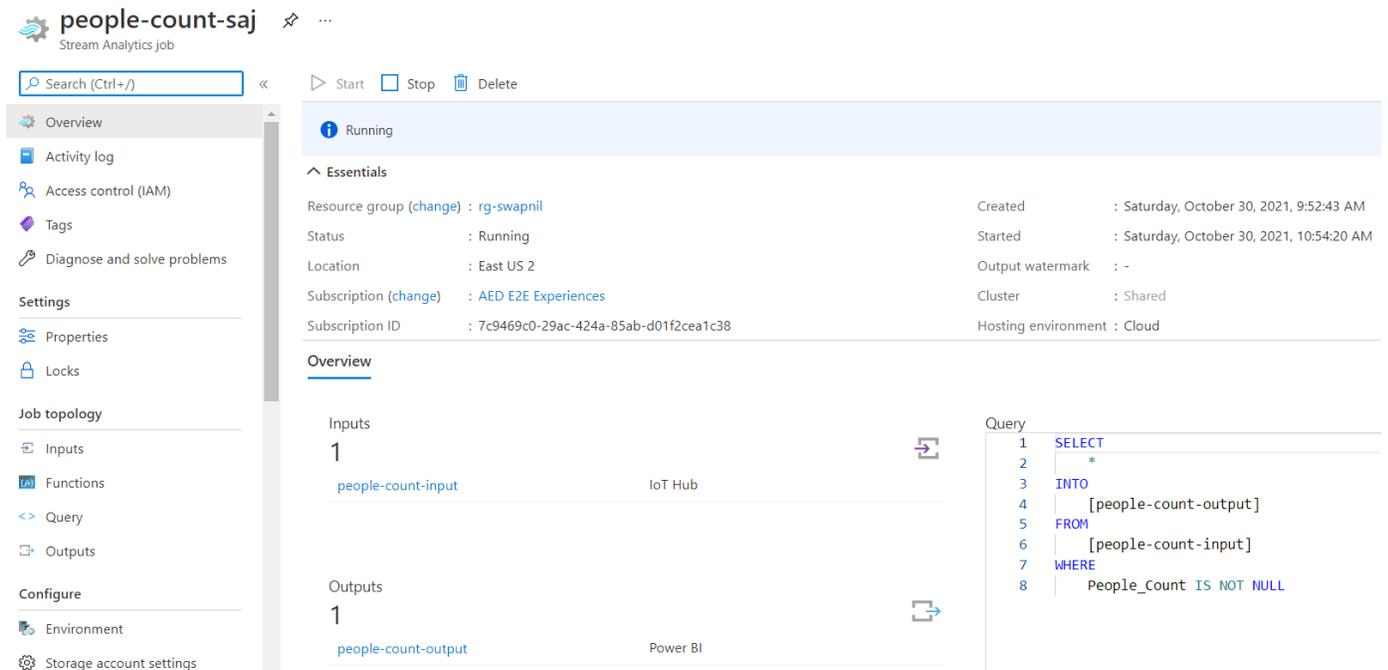
**Note-** The **People\_Count** property is sent from the countmodule to the IoT hub and is forwarded to the Stream Analytics job.

## Ex. 4 - Task 5 - Run the Stream Analytics job (2 min)

1. In the Stream Analytics job, select **Overview**, then select **Start > Now > Start**.

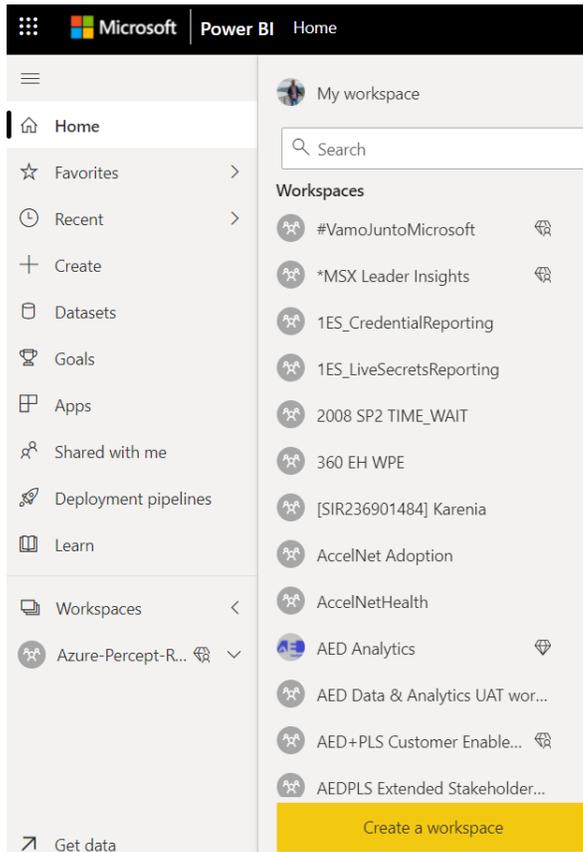


2. Once the job successfully starts, the job status changes from **Stopped** to **Running**

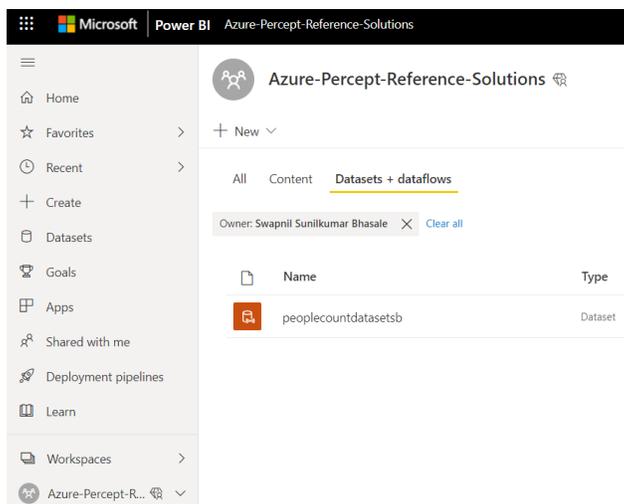


## Exercise 5 – Create and publish a PowerBI report to visualize data (5 min)

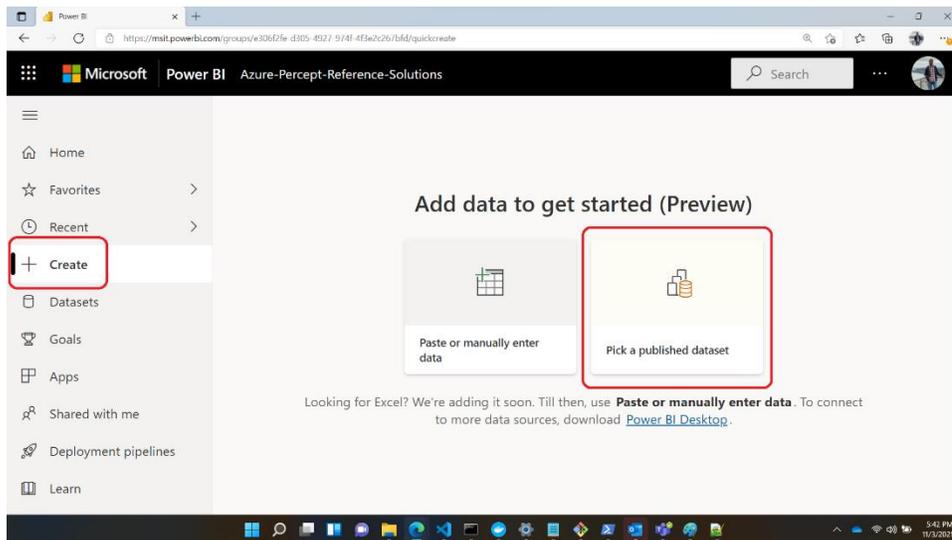
1. Log in to [Power BI](#) and select your Workspace (this is the same Group Workspace you used while creating the Stream Analytics job output)



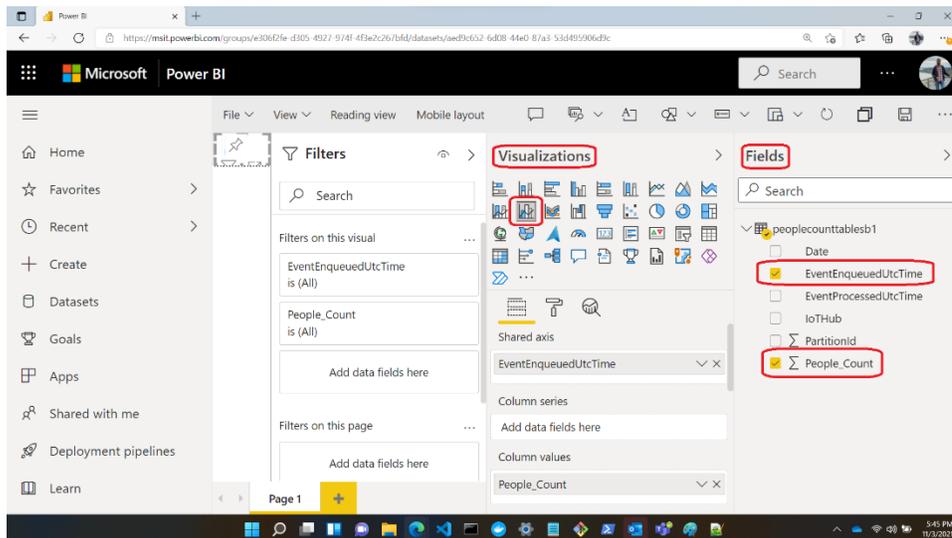
2. Verify that you see your dataset



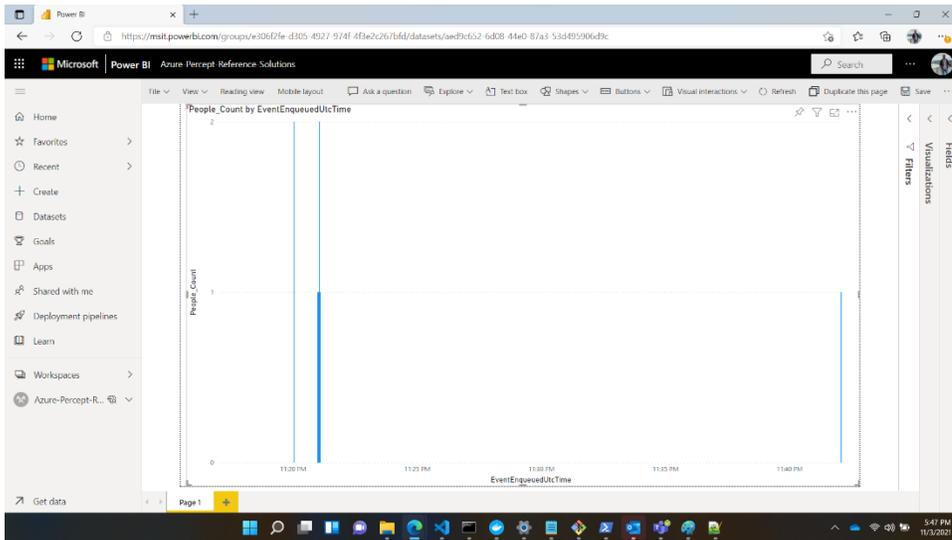
3. On the left scroll panel select **+ Create** and then click **Pick a published dataset**



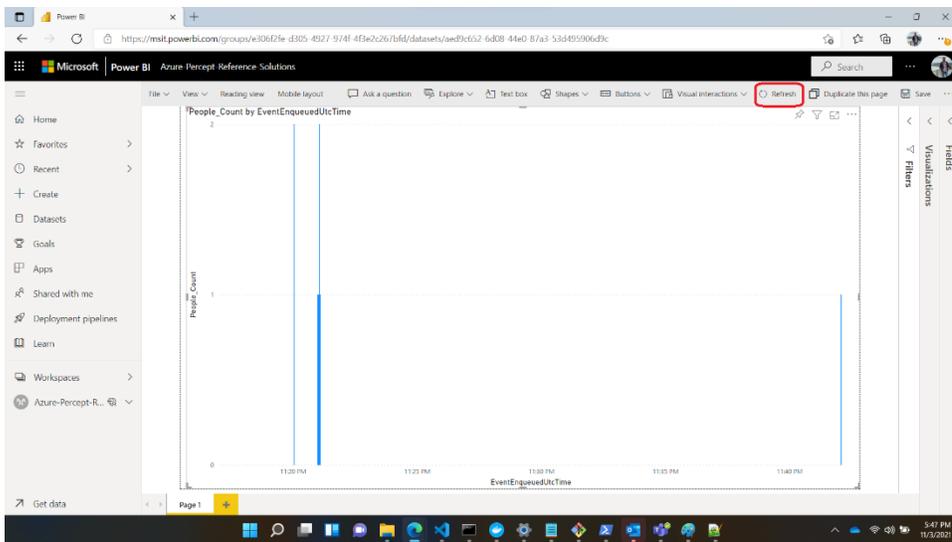
4. Select your dataset and click **Create**
5. On the right, expand the **Fields** dropdown and select **EventEnqueuedUtcTime** and **ΣPeople\_Count**
6. Under **Visualizations** select **Line and clustered column chart**



7. This will generate a graph as follows -



8. Click refresh periodically to update the graph



## Summary

In this lab you created a people detection and counting solution using the Azure Percept DK and the Azure Percept Vision view module. You used PowerBI dashboard to visualize the results.