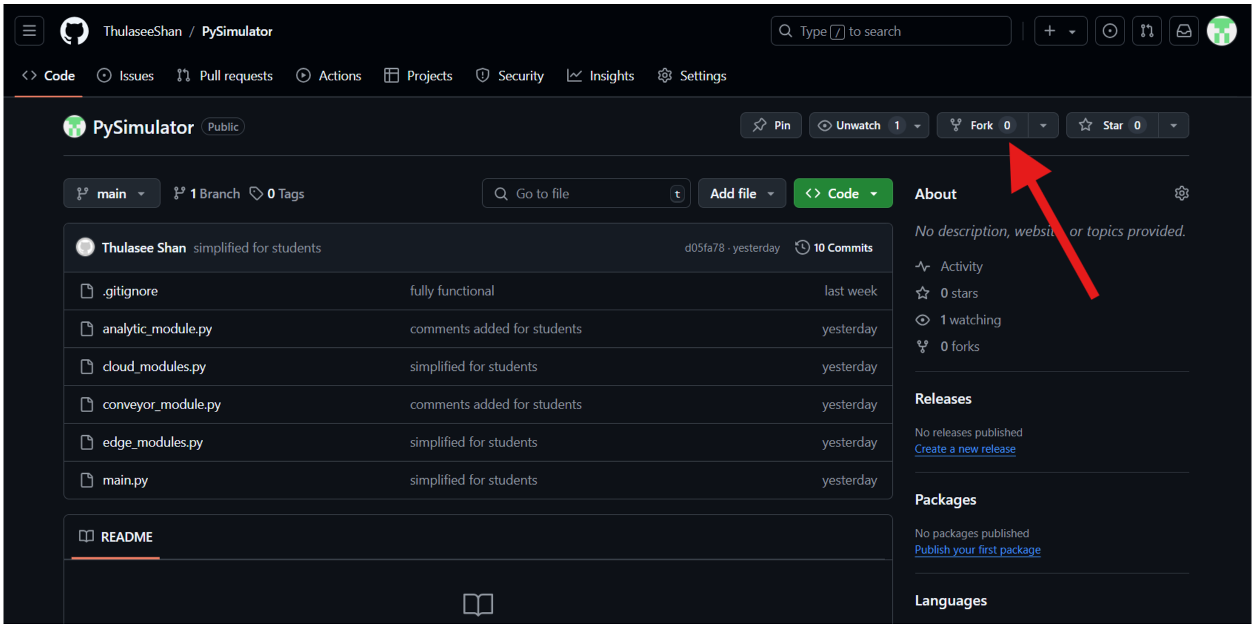
Edge Computing Hands-On Lab

Using the pre-built solution for cookie defect identification

Step 1: (Optional) Install Python and the Development tools

1. Install Python, VS Code and tools.
   Follow this tutorial: [Install and configure Visual Studio Code for Python development - Training | Microsoft Learn]
2. Ensure the following modules are installed.
   a. requests
   ```
   pip install requests
   ```
   b. pygame
   ```
   pip install pygame
   ```
3. Fork from this Git repository: https://github.com/ThulaseeShan/PySimulator to your own GitHub repository

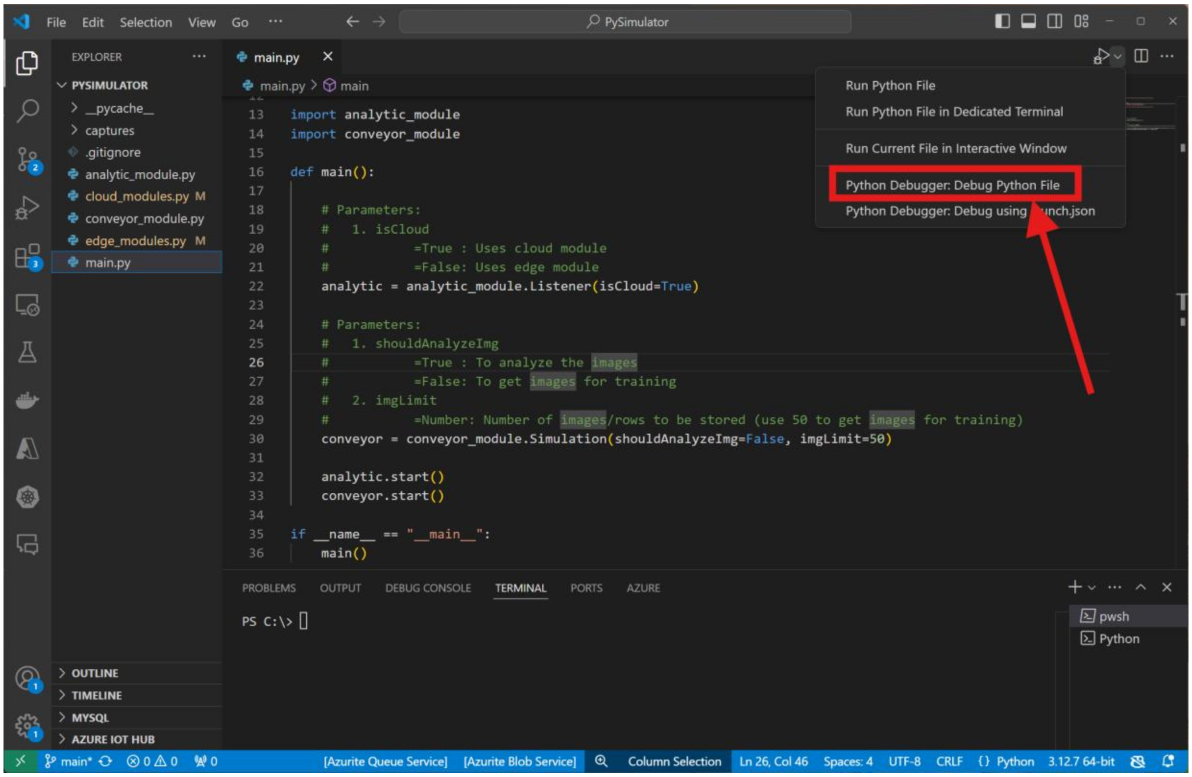


Thulasee Shan simplified for students		d05fa78 · yesterday 10 Commits
.gitignore	fully functional	last week
analytic_module.py	comments added for students	yesterday
cloud_modules.py	simplified for students	yesterday
conveyor_module.py	comments added for students	yesterday
edge_modules.py	simplified for students	yesterday
main.py	simplified for students	yesterday

4. Connect VS Code to your GitHub repository (we created this during the first lab session)
5. Sync up the code to your laptop / machine
   If you don't know how to work with Git repository using VS Code, refer this: [Collaborate on GitHub (visualstudio.com)]

Step 2: Run the Simulator for the first time.

1. Once the code is synced select the main.py and click run in Python Debugger (otherwise you might see some error messages). No code changes would be required for the 1st run.

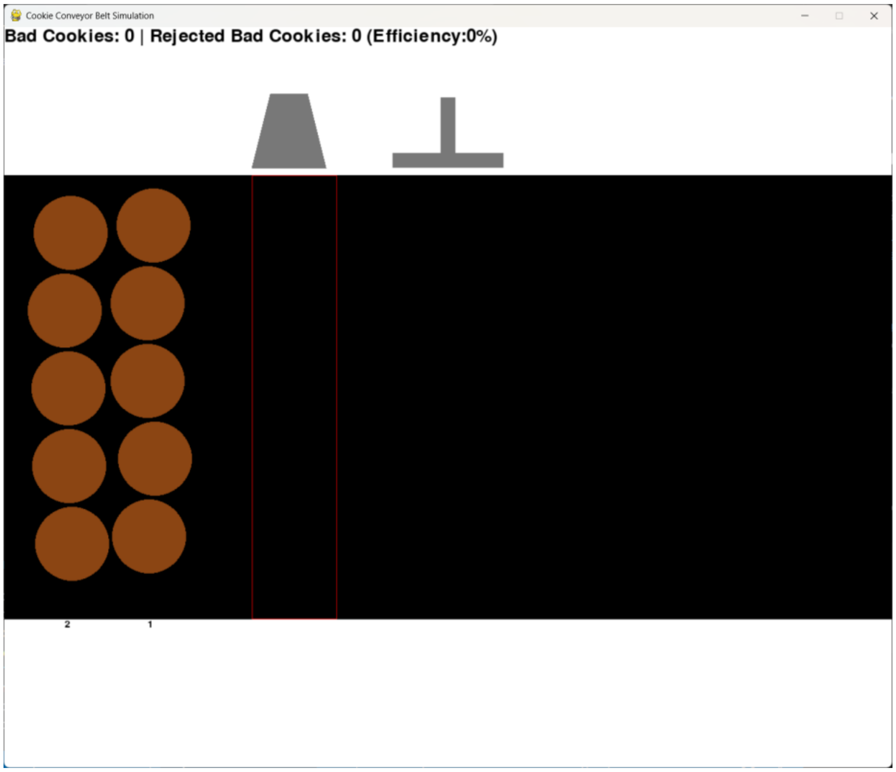


```
import analytic_module
import conveyor_module

def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=True)

    # Parameters:
    #   1. shouldAnalyzeImg
    #           =True : To analyze the images
    #           =False: To get images for training
    #   2. imgLimit
    #           =Number: Number of images/rows to be stored (use 50 to get images for training)
    conveyor = conveyor_module.Simulation(shouldAnalyzeImg=False, imgLimit=50)

    analytic.start()
    conveyor.start()

if __name__ == "__main__":
    main()
```

```
PS C:\>
```

File Edit Selection View Go ···
PySimulator
EXPLORER
PYSIMULATOR
__pycache__
captures
.gitignore
analytic_module.py
cloud_modules.py M
conveyor_module.py
edge_modules.py M
main.py
main.py > main
Run Python File
Run Python File in Dedicated Terminal
Run Current File in Interactive Window
Python Debugger: Debug Python File
Python Debugger: Debug using launch.json
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
pwsh
Python
OUTLINE
TIMELINE
MYSQL
AZURE IOT HUB
main* ⊗ 0 ⚠ 0 0
[Azurite Queue Service] [Azurite Blob Service]
Column Selection Ln 26, Col 46 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 64-bit

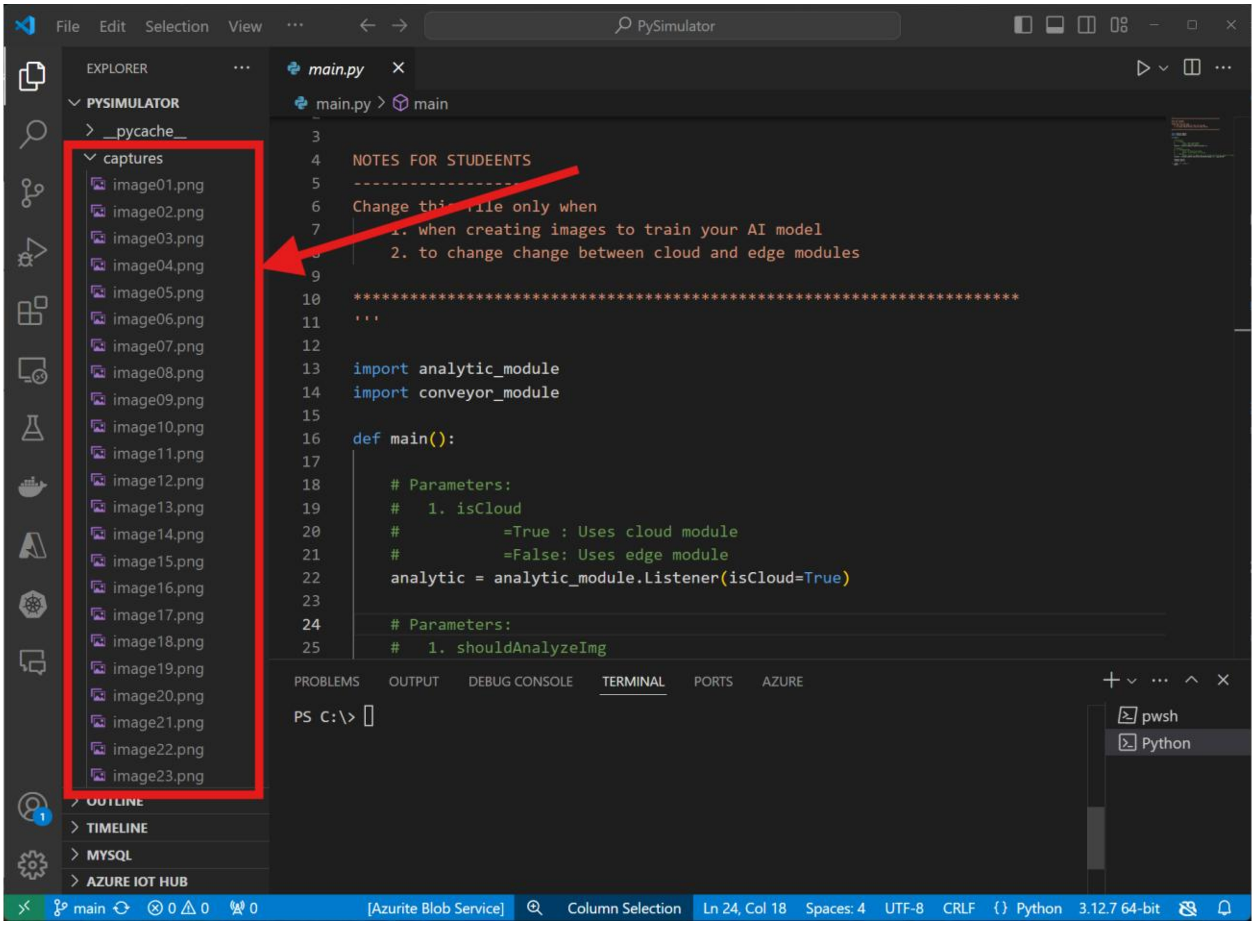
2. If everything is set up correctly, you should see the simulator running as follows



Cookie Conveyor Belt Simulation

Bad Cookies: 0 | Rejected Bad Cookies: 0 (Efficiency:0%)

3. Wait until you see 50 captured files created in the "captures" folder and then close the simulator. You may also watch the row numbers (on the bottom of the simulator screen) until they reach 55 - 60.



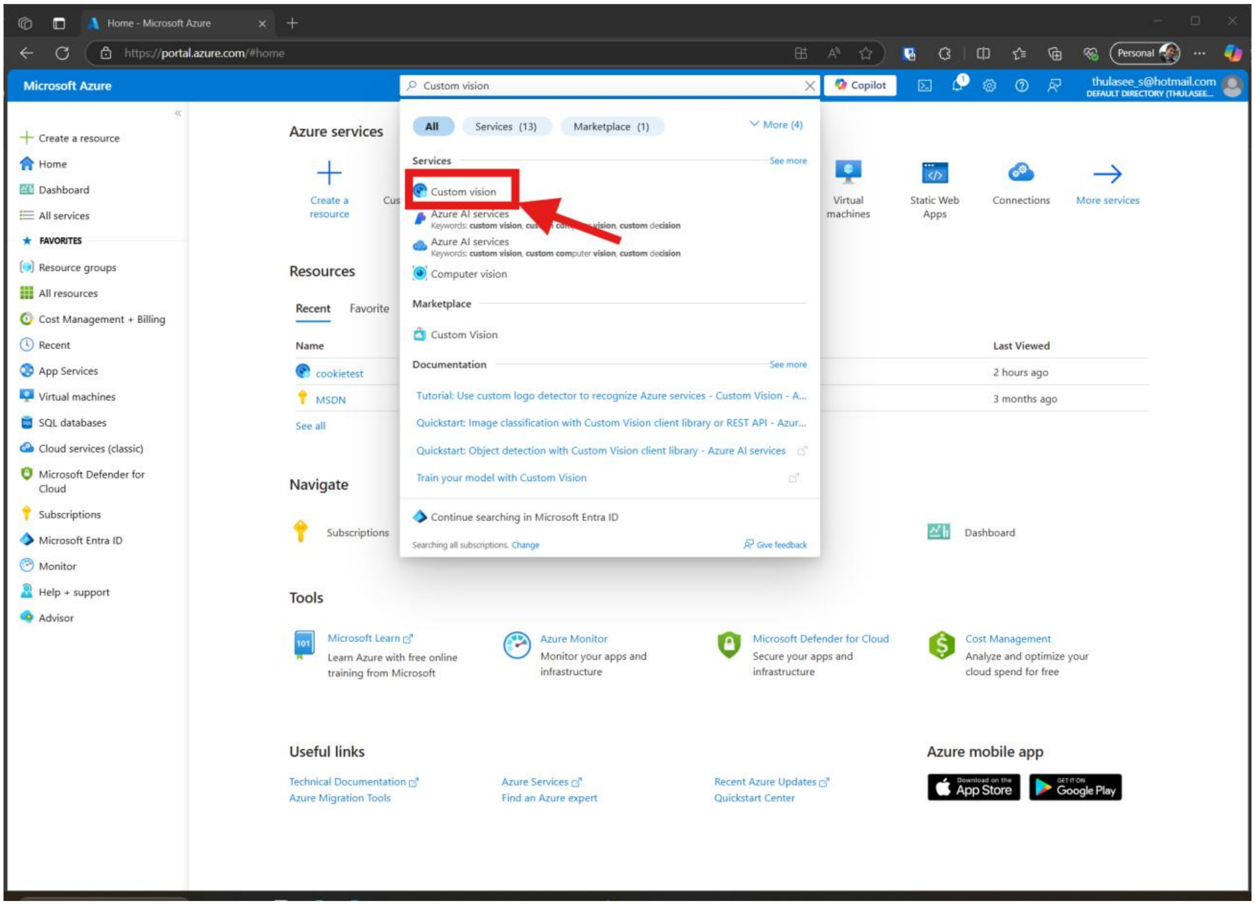
- __pycache__
- captures
  - image01.png
  - image02.png
  - image03.png
  - image04.png
  - image05.png
  - image06.png
  - image07.png
  - image08.png
  - image09.png
  - image10.png
  - image11.png
  - image12.png
  - image13.png
  - image14.png
  - image15.png
  - image16.png
  - image17.png
  - image18.png
  - image19.png
  - image20.png
  - image21.png
  - image22.png
  - image23.png

```

NOTES FOR STUDEENTS
-------------------
Change this file only when
    1. when creating images to train your AI model
    2. to change change between cloud and edge modules

**********************************************************************
'''


import analytic_module
import conveyor_module

def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=True)

    # Parameters:
    #   1. shouldAnalyzeImg
```

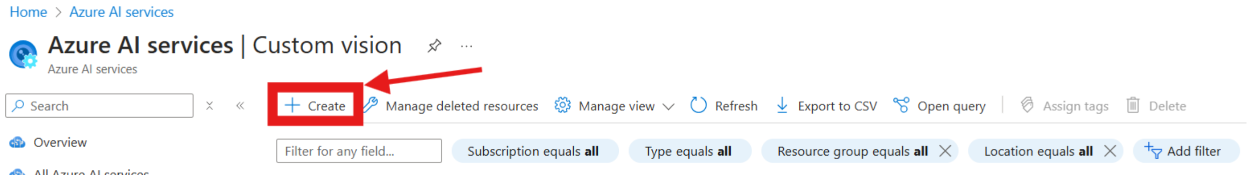
```
PS C:\>
```

Step 3: Setting up a Custom Vision AI Project in Azure Portal

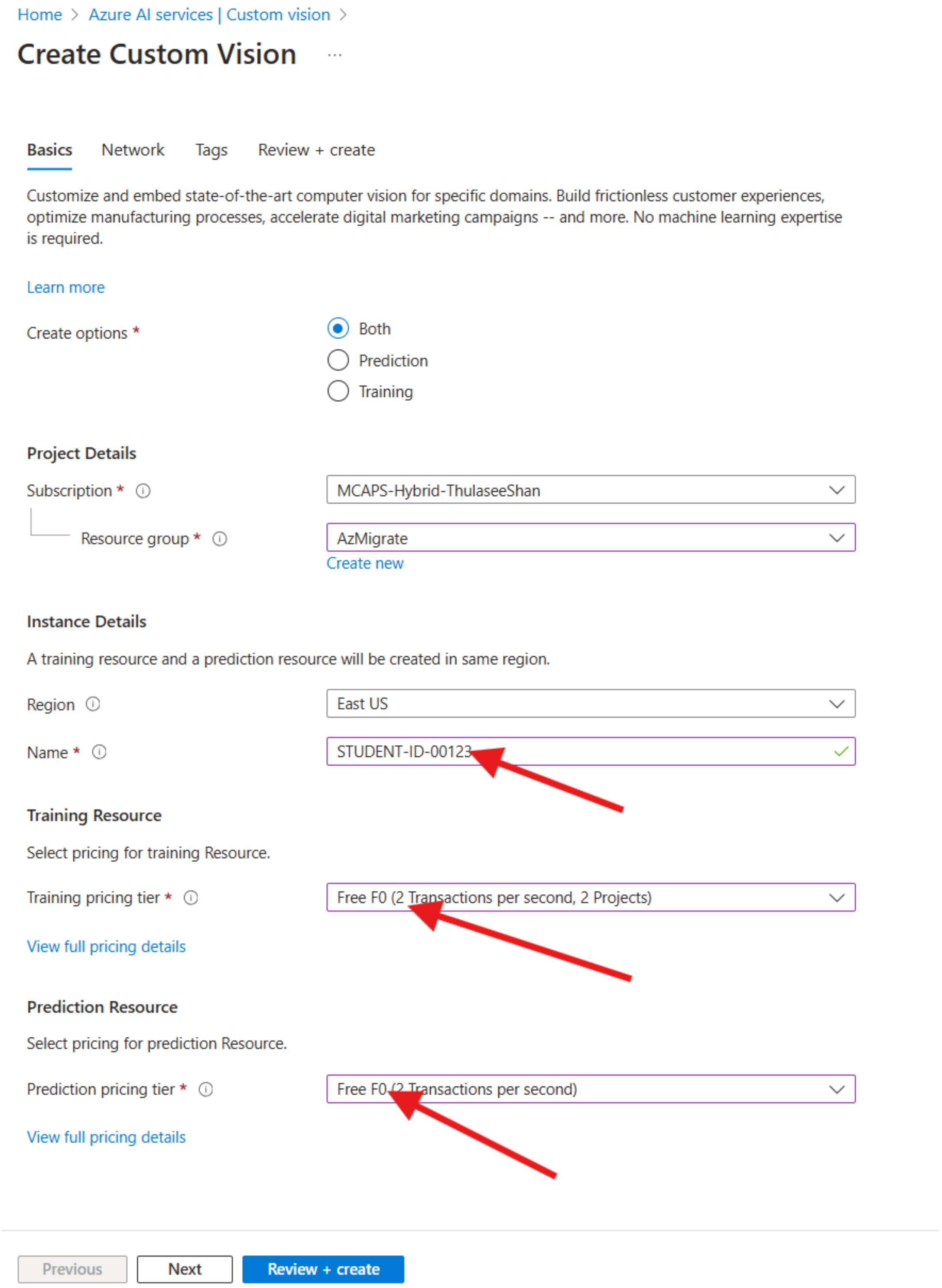
1. Login to the Azure portal in your browser and search for “custom vision” in the top search bar and select the option “Custom vision”



2. Click “Create” from the top tool bar



3. Fill out the information on the next screen. Ensure you have given your student id as the instance name and selected Free F0 as the pricing tier. You can have only 1 free resource. If you have already created it before, you will have to delete and recreate.



Home > Azure AI services | Custom vision >

Create Custom Vision ...

Basics   Network   Tags   Review + create

Customize and embed state-of-the-art computer vision for specific domains. Build frictionless customer experiences, optimize manufacturing processes, accelerate digital marketing campaigns -- and more. No machine learning expertise is required.

Learn more

Create options *

● Both
○ Prediction
○ Training

Project Details

Subscription * : MCAPS-Hybrid-ThulaseeShan

Resource group * : AzMigrate

Create new

Instance Details

A training resource and a prediction resource will be created in same region.

Region : East US

Name * : STUDENT-ID-00123 ✓

Training Resource

Select pricing for training Resource.

Training pricing tier * : Free F0 (2 Transactions per second, 2 Projects)

View full pricing details

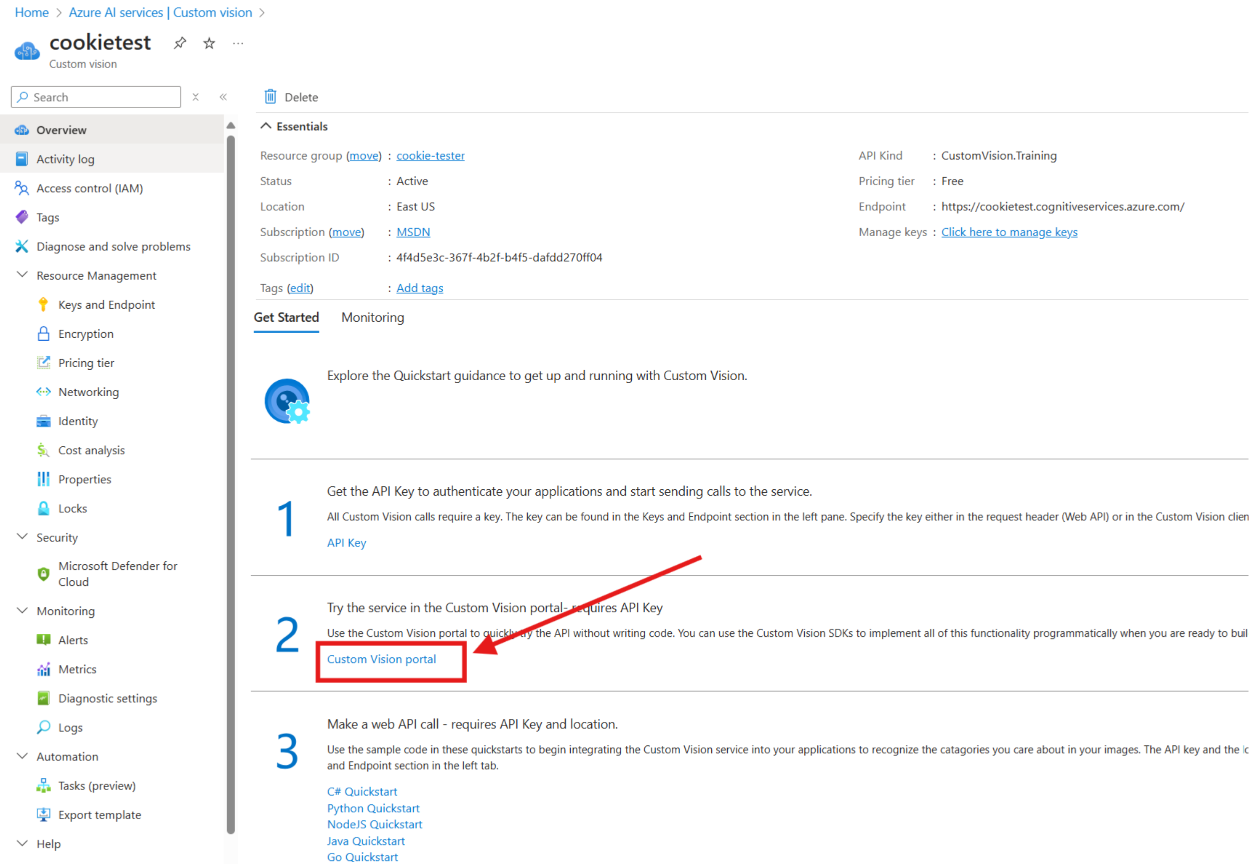
Prediction Resource

Select pricing for prediction Resource.

Prediction pricing tier * : Free F0 (2 Transactions per second)

View full pricing details

4. Once the instance is created click the 1st resource and then select “Custom Vision portal”. This will open in a new tab and then sign in to the portal (it should use the same login as Azure portal)



Resource group (move) : cookie-tester

Status : Active

Location : East US

Subscription (move) : MSDN

Subscription ID : 4f4d5e3c-367f-4b2f-b4f5-dafdd270ff04

Tags (edit) : Add tags

API Kind : CustomVision.Training

Pricing tier : Free

Endpoint : https://cookietest.cognitiveservices.azure.com/

Manage keys : Click here to manage keys

Explore the Quickstart guidance to get up and running with Custom Vision.

1 Get the API Key to authenticate your applications and start sending calls to the service.

All Custom Vision calls require a key. The key can be found in the Keys and Endpoint section in the left pane. Specify the key either in the request header (Web API) or in the Custom Vision clien

API Key

2 Try the service in the Custom Vision portal - requires API Key

Use the Custom Vision portal to quickly try the API without writing code. You can use the Custom Vision SDKs to implement all of this functionality programmatically when you are ready to buil

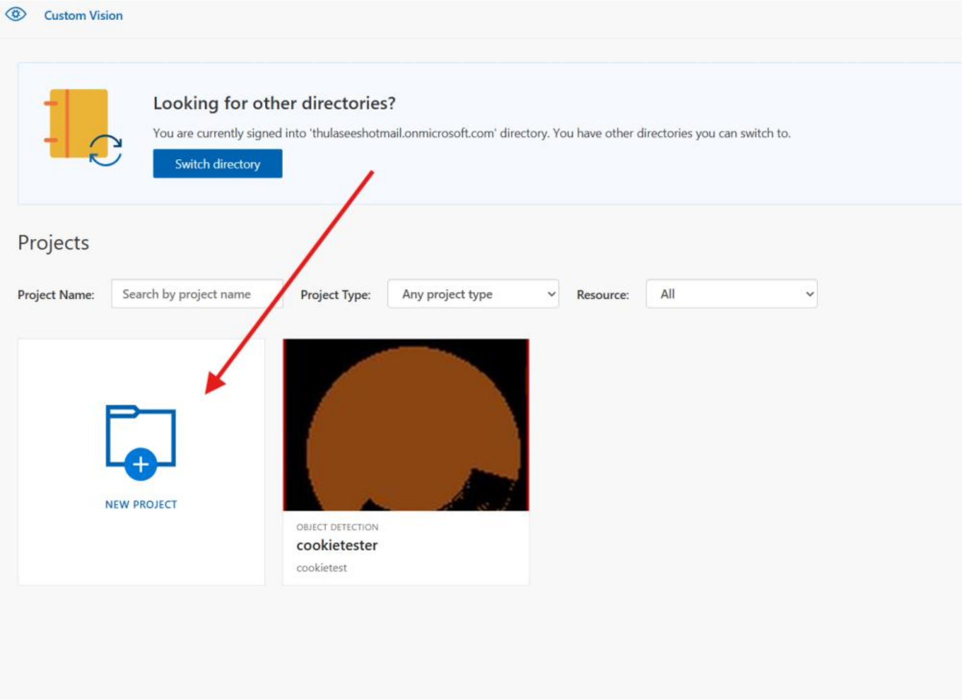
Custom Vision portal

3 Make a web API call - requires API Key and location.

Use the sample code in these quickstarts to begin integrating the Custom Vision service into your applications to recognize the catagories you care about in your images. The API key and the i and Endpoint section in the left tab.

C# Quickstart
Python Quickstart
NodeJS Quickstart
Java Quickstart
Go Quickstart

5. Once in the customvision.ai portal, click the new project link



Looking for other directories?

You are currently signed into 'thulaseeshotmail.onmicrosoft.com' directory. You have other directories you can switch to.

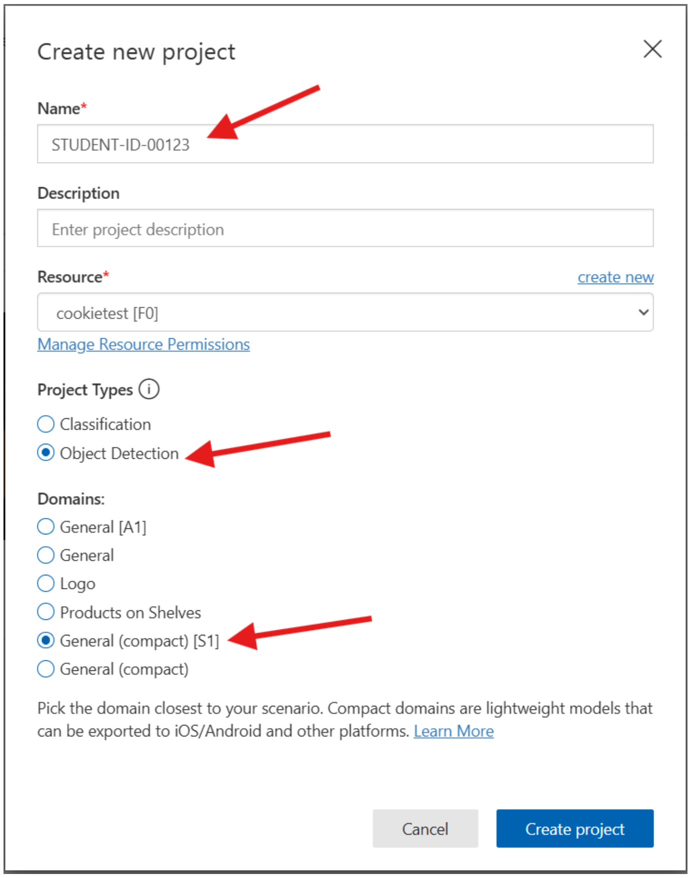
Projects

OBJECT DETECTION

cookietester

cookietest

6. Provide your student id again as the name of the project and select “Object Detection” as the project type and “General (compact) [S1]” as the Domain. If this is not selected, you won’t be able to build the Docker Container.



Create new project

Name*

STUDENT-ID-00123

Description

Enter project description

Resource*

cookietest [F0]

Project Types

○ Classification

● Object Detection

Domains:

○ General [A1]

○ General

○ Logo

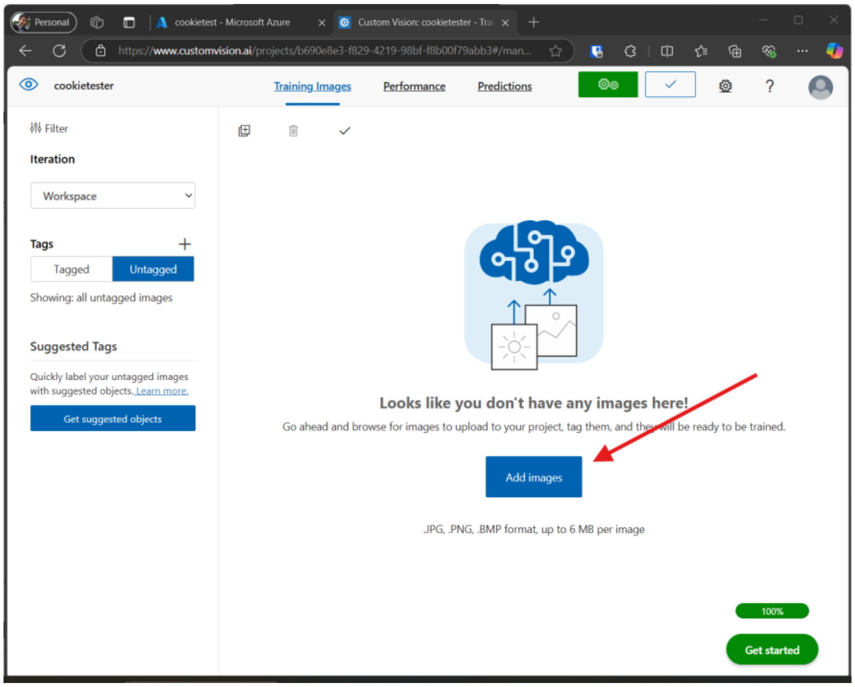
○ Products on Shelves

● General (compact) [S1]

○ General (compact)

Pick the domain closest to your scenario. Compact domains are lightweight models that can be exported to iOS/Android and other platforms. Learn More

7. Once the project is created, select the “Add images” button and upload all the 50 images generated by the simulator.



Showing: all untagged images

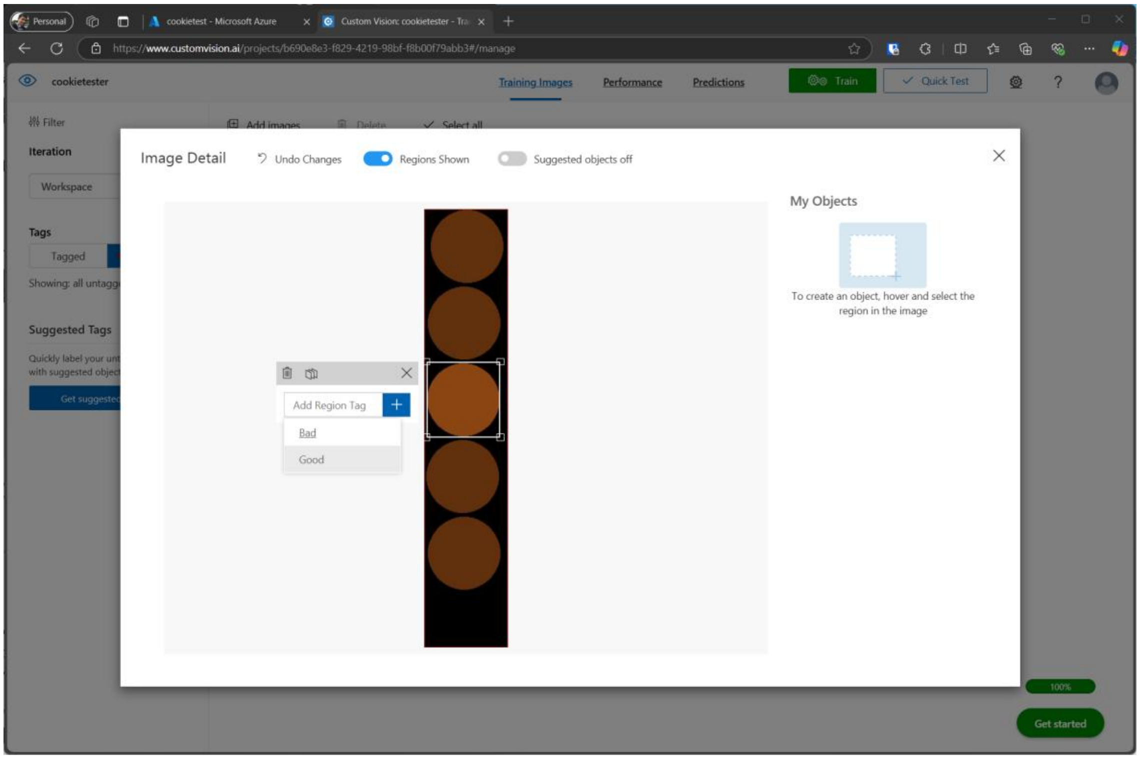
Quickly label your untagged images with suggested objects. Learn more.

Looks like you don't have any images here!

Go ahead and browse for images to upload to your project, tag them, and they will be ready to be trained.

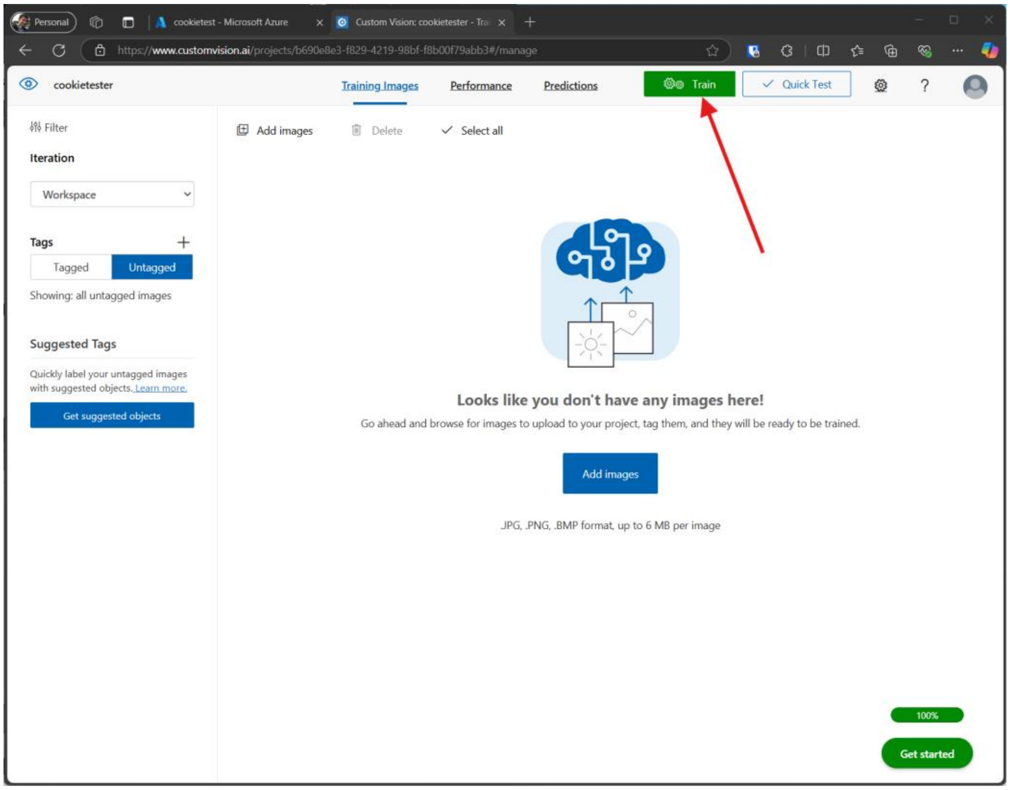
.JPG, .PNG, .BMP format, up to 6 MB per image

8. When the uploading is completed, select the image and tag each cookie appropriately. Select the good cookies and tag them as “Good” while the defect cookies as “Bad”. Pay attention to the letter-case, as python code may not work if you misspell them or change the letter-case



To create an object, hover and select the region in the image

9. After tagging all the 50 images, click on the "Train" from the top, to train the AI model



Showing: all untagged images

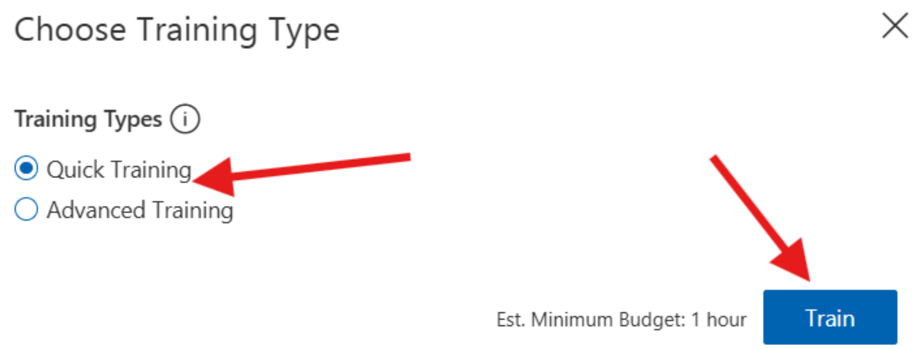
Quickly label your untagged images with suggested objects. Learn more.

Looks like you don't have any images here!

Go ahead and browse for images to upload to your project, tag them, and they will be ready to be trained.

.JPG, .PNG, .BMP format, up to 6 MB per image

10. Then select the "Quick Training", and then click "Train" button. This would take 30 to 60 minutes to complete.



Choose Training Type

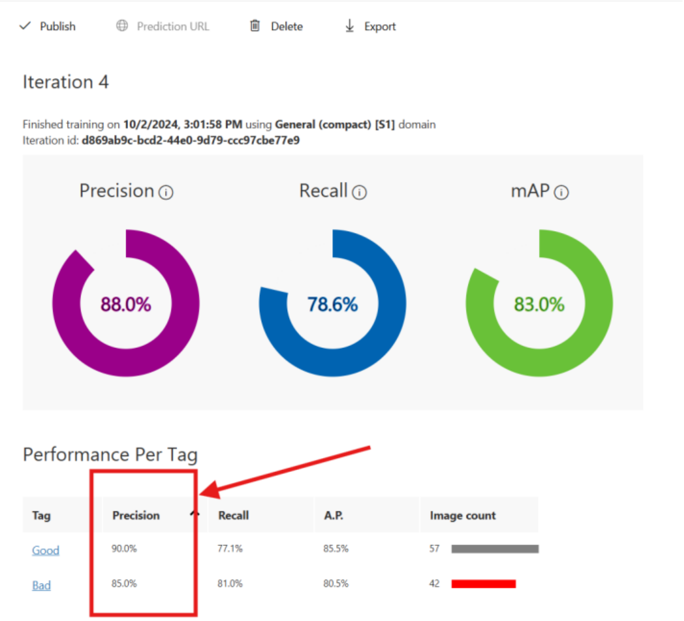
Training Types

● Quick Training

○ Advanced Training

Est. Minimum Budget: 1 hour

11. When the training is completed, you should see the precision above 85%. If not, the number of images you trained is not enough and you will have to bring more images and train the AI multiple times.



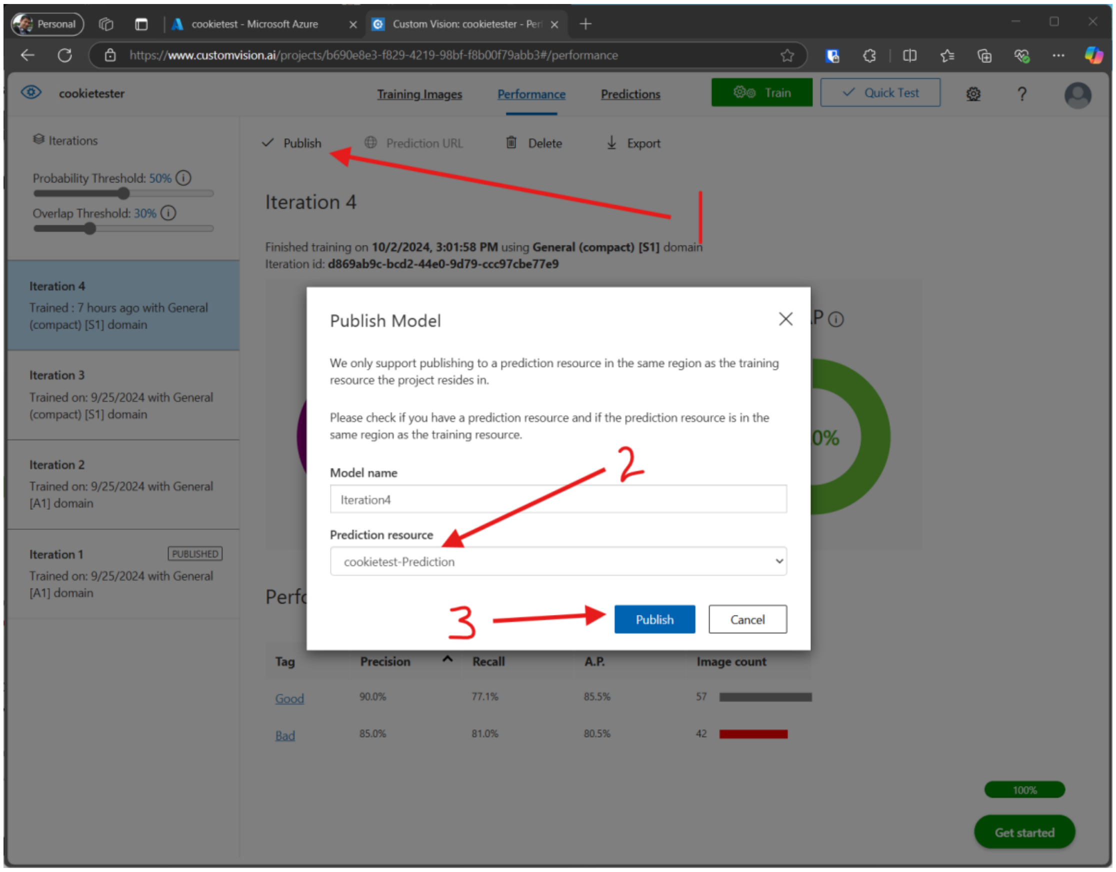
Iteration 4

Finished training on **10/2/2024, 3:01:58 PM** using **General (compact) [S1]** domain
Iteration id: **d869ab9c-bcd2-44e0-9d79-ccc97cbe77e9**

Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
Good	90.0%	77.1%	85.5%	57
Bad	85.0%	81.0%	80.5%	42

12. When you are satisfied with the results, click on “Publish” and select the Prediction resource you create earlier and publish


Personal
cookietest - Microsoft Azure
Custom Vision: cookietester - Per
https://www.customvision.ai/projects/b690e8e3-f829-4219-98bf-f8b00f79abb3#/performance
cookietester
Training Images
Performance
Predictions
Train
Quick Test
Iterations
Publish
Prediction URL
Delete
Export
2
3
100%
Get started

Probability Threshold: 50% ⓘ
Overlap Threshold: 30% ⓘ

Iteration 4
Trained : 7 hours ago with General (compact) [S1] domain

Iteration 3
Trained on: 9/25/2024 with General (compact) [S1] domain

Iteration 2
Trained on: 9/25/2024 with General [A1] domain

Iteration 1 PUBLISHED
Trained on: 9/25/2024 with General [A1] domain

Iteration 4

Finished training on **10/2/2024, 3:01:58 PM** using **General (compact) [S1]** domain
Iteration id: **d869ab9c-bcd2-44e0-9d79-ccc97cbe77e9**

Publish Model

We only support publishing to a prediction resource in the same region as the training resource the project resides in.

Please check if you have a prediction resource and if the prediction resource is in the same region as the training resource.

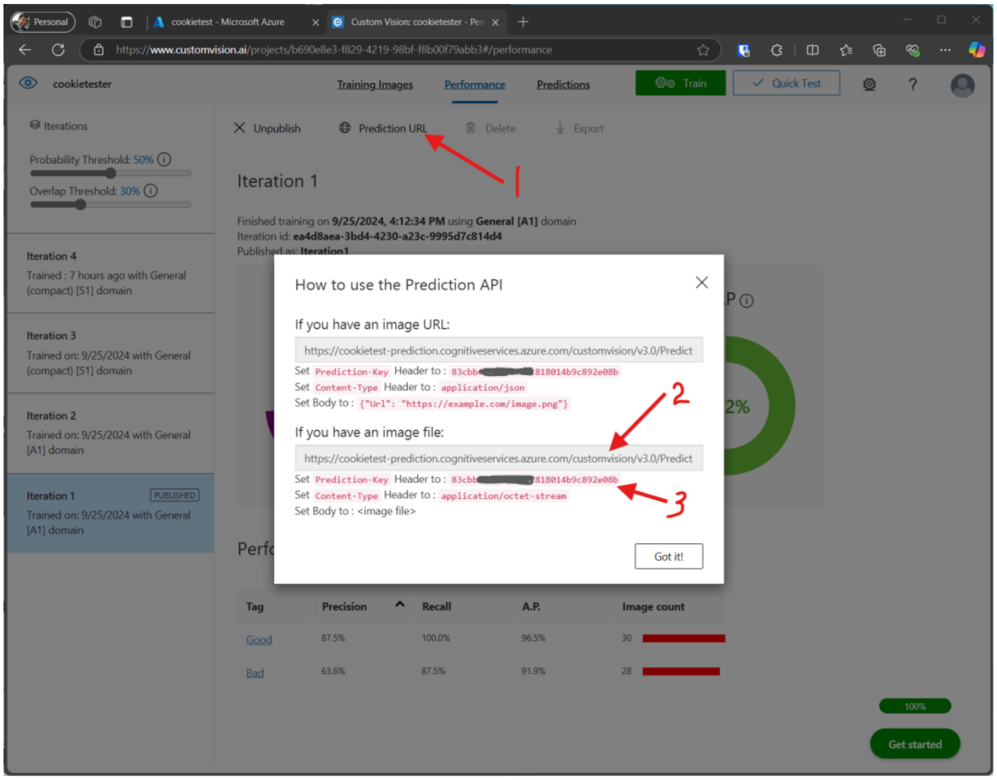
Model name
Iteration4

Prediction resource
cookietest-Prediction

Publish Cancel

Tag	Precision	Recall	A.P.	Image count
Good	90.0%	77.1%	85.5%	57
Bad	85.0%	81.0%	80.5%	42

13. After publishing, you should be able to get the URL and Prediction-Key (aka API Key) from the Prediction URL option from the top menu.



cookietester
Training Images
Performance
Predictions
Train
Quick Test
Iterations
Probability Threshold: 50%
Overlap Threshold: 30%
Iteration 4
Trained : 7 hours ago with General (compact) [S1] domain
Iteration 3
Trained on: 9/25/2024 with General (compact) [S1] domain
Iteration 2
Trained on: 9/25/2024 with General [A1] domain
Iteration 1
PUBLISHED
Trained on: 9/25/2024 with General [A1] domain
Unpublish
Prediction URL
Delete
Export
1
2
3
Got it!
100%
Get started

Iteration 1

Finished training on **9/25/2024, 4:12:34 PM** using **General [A1]** domain
Iteration id: **ea4d8aea-3bd4-4230-a23c-9995d7c814d4**
Published as: **Iteration1**

How to use the Prediction API

If you have an image URL:

https://cookietest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predict

Set `Prediction-Key` Header to : `83cbb[redacted]818014b9c892e08b`
Set `Content-Type` Header to : `application/json`
Set Body to : `{"Url": "https://example.com/image.png"}`

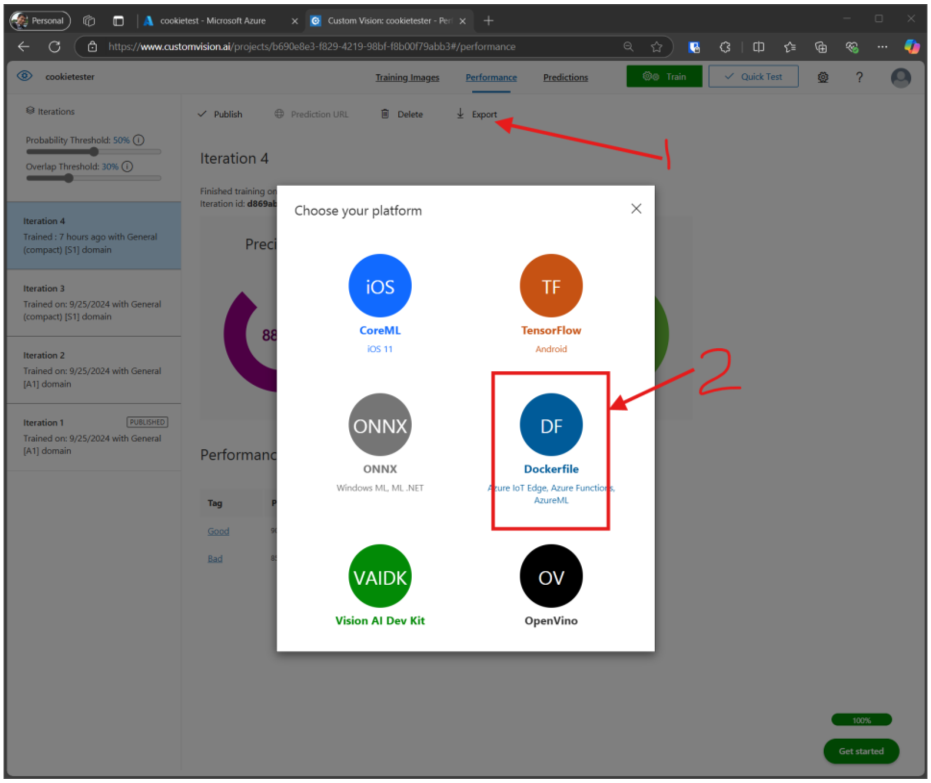
If you have an image file:

https://cookietest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predict

Set `Prediction-Key` Header to : `83cbb[redacted]818014b9c892e08b`
Set `Content-Type` Header to : `application/octet-stream`
Set Body to : <image file>

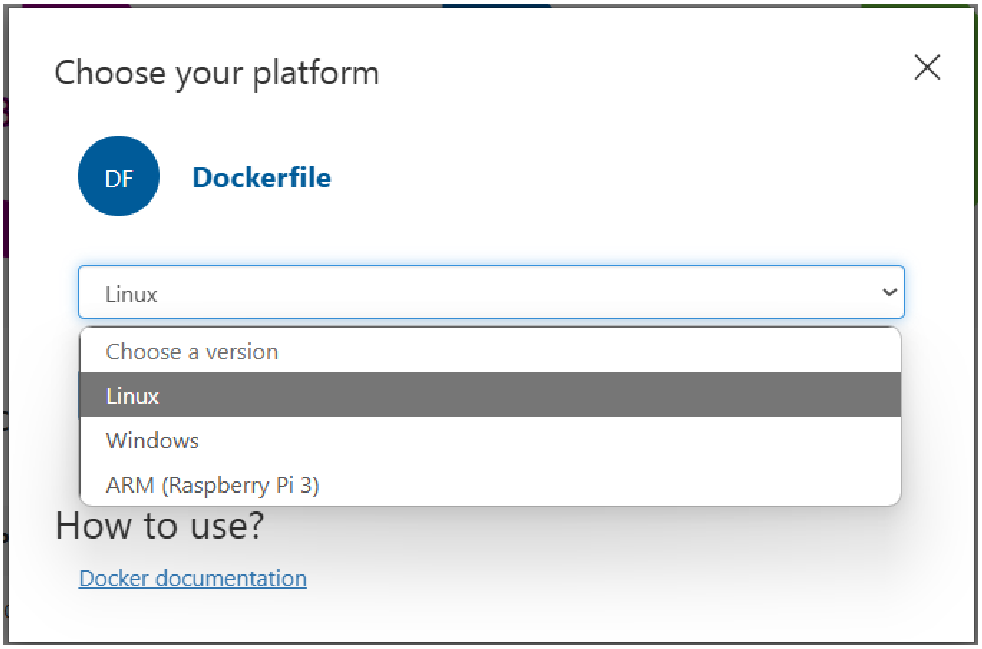
Tag	Precision	Recall	A.P.	Image count
Good	87.5%	100.0%	96.5%	30
Bad	63.6%	87.5%	91.9%	28

14. Export the Docker Container to build your Edge node



Choose your platform

15. Select appropriate OS, where you intend to run the docker and download the zip file



Choose your platform

DF Dockerfile

Linux

- Choose a version
- Linux
- Windows
- ARM (Raspberry Pi 3)

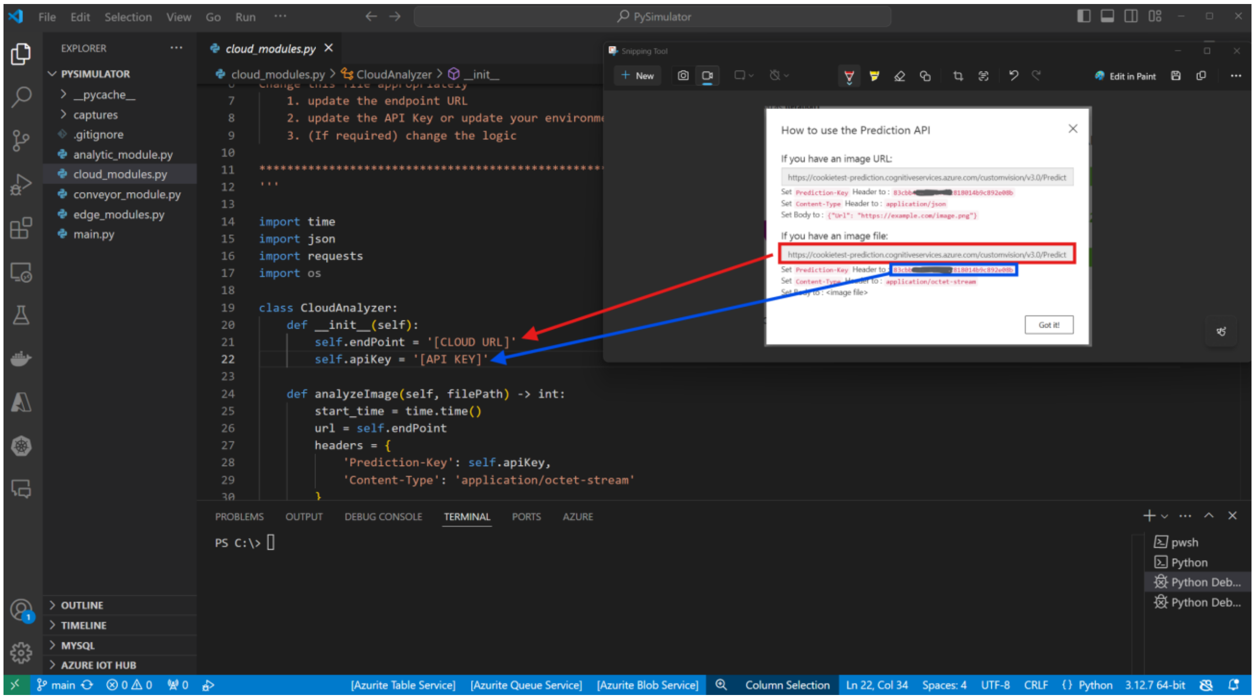
How to use?

Docker documentation

16. Unzip this to a folder and ensure you see the "Dockerfile" this the important file to build the docker container later.

Step 4: Testing the Cloud Analyzer with the Simulator

1. Go back to the python code and open the "cloud_module.py" file and replace the "[CLOUD URL]" and "[API KEY]" with the values from the publish settings



File Edit Selection View Go Run ...
PySimulator
EXPLORER
PYSIMULATOR
pycache
captures
.gitignore
analytic_module.py
cloud_modules.py
conveyor_module.py
edge_modules.py
main.py
OUTLINE
TIMELINE
MYSQL
AZURE IOT HUB
cloud_modules.py
cloud_modules.py > CloudAnalyzer > _init_
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
pwsh
Python
Python Deb...
Python Deb...
Snipping Tool
New
Edit in Paint
main 0 0 0
[Azurite Table Service] [Azurite Queue Service] [Azurite Blob Service]
Column Selection Ln 22, Col 34 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 64-bit

```
    Change this file appropriately
        1. update the endpoint URL
        2. update the API Key or update your environm
        3. (If required) change the logic

*************************************************
'''

import time
import json
import requests
import os

class CloudAnalyzer:
    def __init__(self):
        self.endPoint = '[CLOUD URL]'
        self.apiKey = '[API KEY]'

    def analyzeImage(self, filePath) -> int:
        start_time = time.time()
        url = self.endPoint
        headers = {
            'Prediction-Key': self.apiKey,
            'Content-Type': 'application/octet-stream'
        }
```

```
PS C:\>
```

How to use the Prediction API

If you have an image URL:

https://cookietest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predict

Set `Prediction-Key` Header to : `83cbb[redacted]818014b9c892e08b`

Set `Content-Type` Header to : `application/json`

Set Body to : `{"Url": "https://example.com/image.png"}`

If you have an image file:

https://cookietest-prediction.cognitiveservices.azure.com/customvision/v3.0/Predict

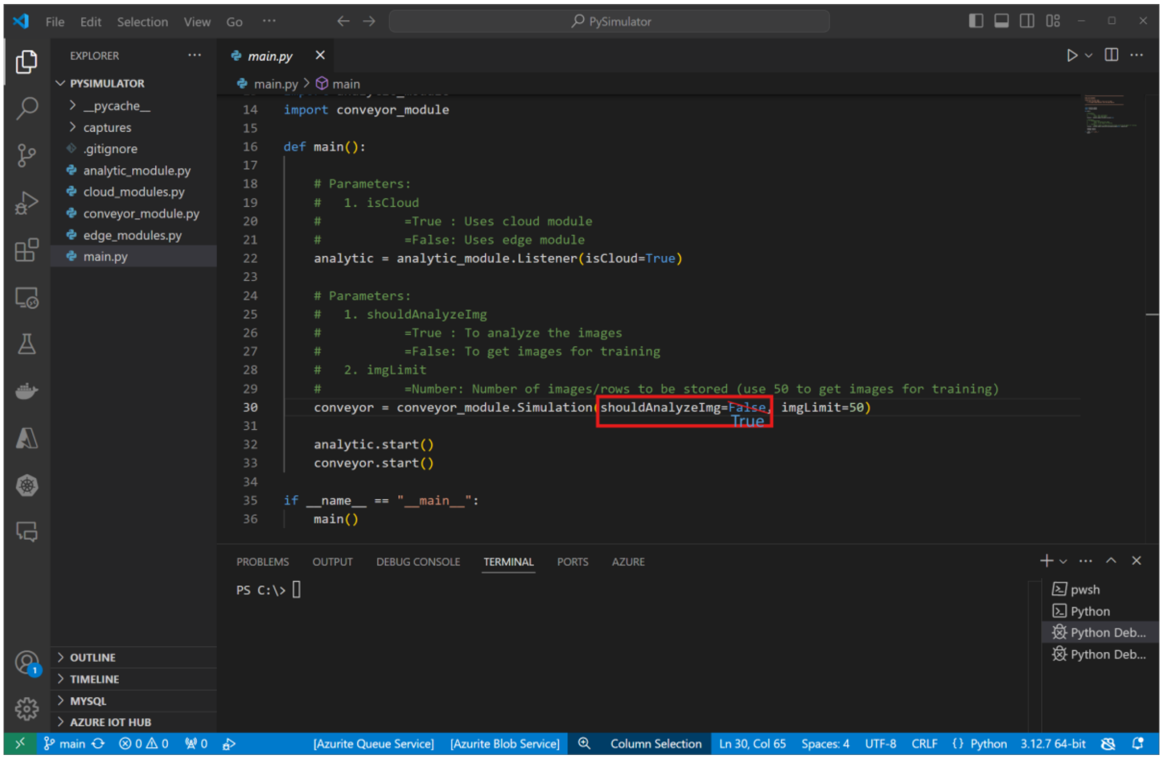
Set `Prediction-Key` Header to : `83cbb[redacted]818014b9c892e08b`

Set `Content-Type` Header to : `application/octet-stream`

Set Body to : <image file>

Got it!

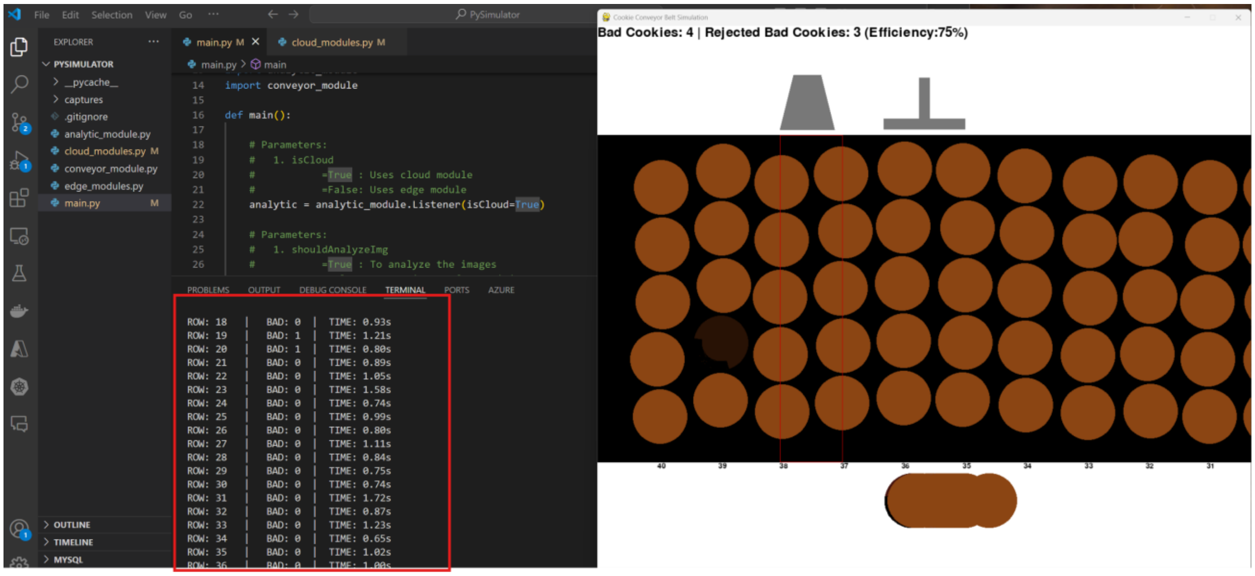
2. Save this file and open the main.py. Then set the shouldAnalyzeImg to True



```
import conveyor_module

def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=True)

    # Parameters:
    #   1. shouldAnalyzeImg
    #           =True : To analyze the images
    #           =False: To get images for training
    #   2. imgLimit
    #           =Number: Number of images/rows to be stored (use 50 to get images for training)
    conveyor = conveyor_module.Simulation(shouldAnalyzeImg=False, imgLimit=50)

    analytic.start()
    conveyor.start()

if __name__ == "__main__":
    main()
```

```
PS C:\>
```

3. Run the application in debug mode (refer to the point Step-2.1 above). Observe the RTT (round trip time) to get the results from the cloud.



Bad Cookies: 4 | Rejected Bad Cookies: 3 (Efficiency:75%)

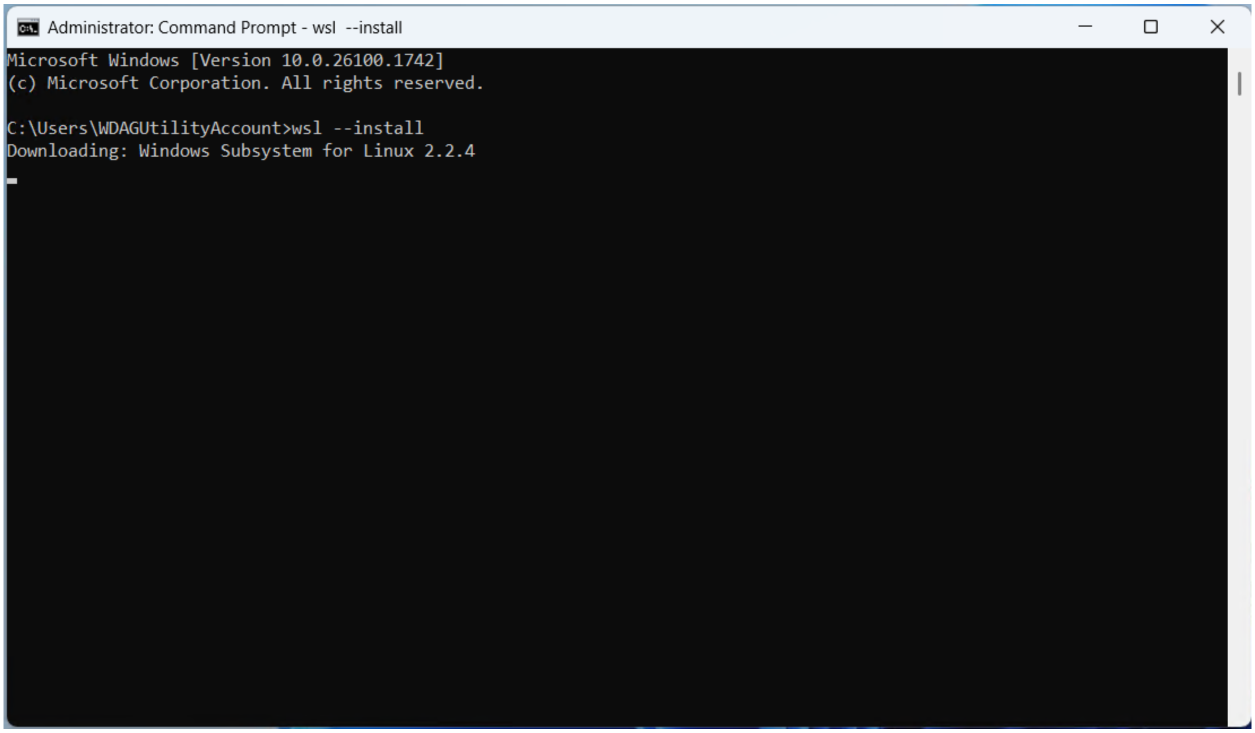
```
import conveyor_module

def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=True)

    # Parameters:
    #   1. shouldAnalyzeImg
    #           =True : To analyze the images
```

```
ROW: 18  |  BAD: 0  |  TIME: 0.93s
ROW: 19  |  BAD: 1  |  TIME: 1.21s
ROW: 20  |  BAD: 1  |  TIME: 0.80s
ROW: 21  |  BAD: 0  |  TIME: 0.89s
ROW: 22  |  BAD: 0  |  TIME: 1.05s
ROW: 23  |  BAD: 0  |  TIME: 1.58s
ROW: 24  |  BAD: 0  |  TIME: 0.74s
ROW: 25  |  BAD: 0  |  TIME: 0.99s
ROW: 26  |  BAD: 0  |  TIME: 0.80s
ROW: 27  |  BAD: 0  |  TIME: 1.11s
ROW: 28  |  BAD: 0  |  TIME: 0.84s
ROW: 29  |  BAD: 0  |  TIME: 0.75s
ROW: 30  |  BAD: 0  |  TIME: 0.74s
ROW: 31  |  BAD: 0  |  TIME: 1.72s
ROW: 32  |  BAD: 0  |  TIME: 0.87s
ROW: 33  |  BAD: 0  |  TIME: 1.23s
ROW: 34  |  BAD: 0  |  TIME: 0.65s
ROW: 35  |  BAD: 0  |  TIME: 1.02s
ROW: 36  |  BAD: 0  |  TIME: 1.00s
```

Step 5: (Optional) Setup Docker in Linux environment

Do this, only if you don't have Docker installed already.

1. Install WSL and Ubuntu. This will install Windows Subsystem for Linux, which is easier to work on Linux without leaving Windows. Run the following command on your command window "wsl –install"

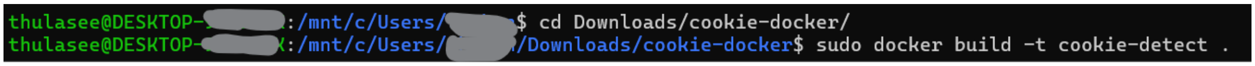


```
Microsoft Windows [Version 10.0.26100.1742]
(c) Microsoft Corporation. All rights reserved.

C:\Users\WDAGUtilityAccount>wsl --install
Downloading: Windows Subsystem for Linux 2.2.4
```

2. Install docker on Linux. Follow the instructions here: [How To Install and Use Docker on Ubuntu 20.04 | DigitalOcean]()

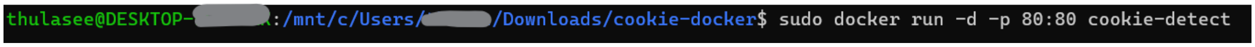
Step 6: Running and Testing Edge Module with Simulator

1. Extract the zip file you downloaded and then navigate to the folder where you find the “Dockerfile” on Linux. Run the docker build command as follows



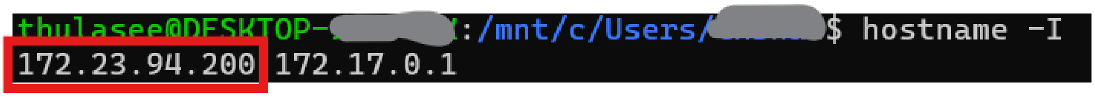
```
thulasee@DESKTOP-[redacted]:/mnt/c/Users/[redacted]$ cd Downloads/cookie-docker/
thulasee@DESKTOP-[redacted]:/mnt/c/Users/[redacted]/Downloads/cookie-docker$ sudo docker build -t cookie-detect .
```

2. Once the docker image is built, run the image using the following command



```
thulasee@DESKTOP-[redacted]:/mnt/c/Users/[redacted]/Downloads/cookie-docker$ sudo docker run -d -p 80:80 cookie-detect
```

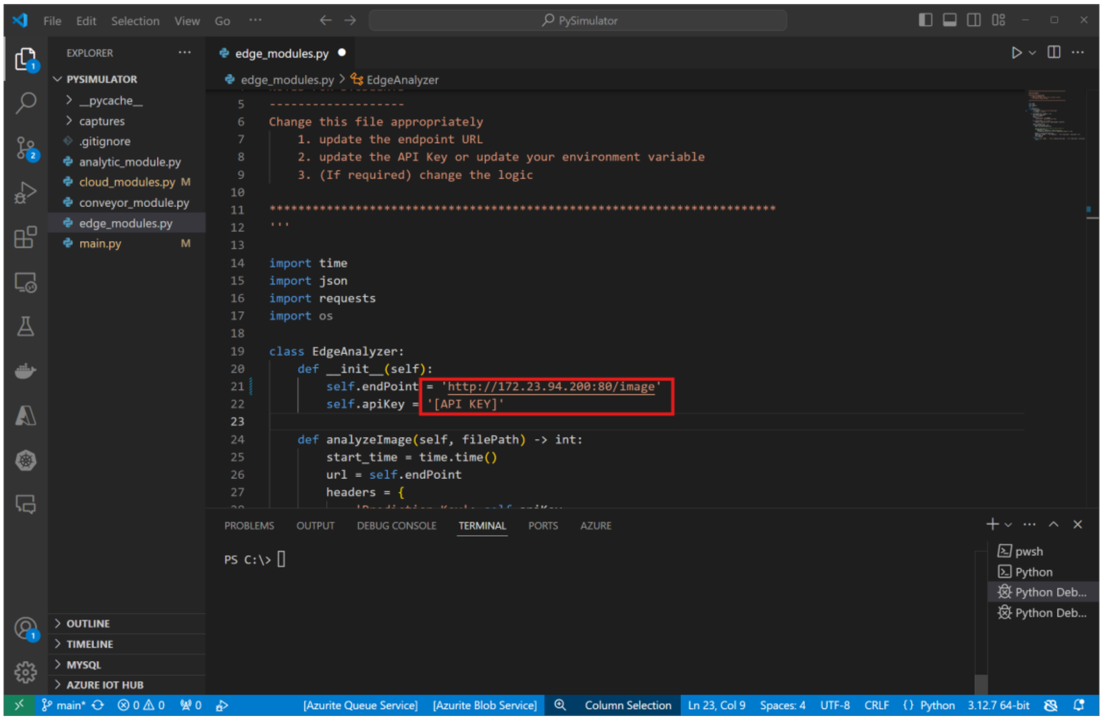
3. If it is running successfully, you should see a log code without any error messages.
4. Now get the IP address of your WSL / Linux



```
thulasee@DESKTOP-[redacted]:/mnt/c/Users/[redacted]$ hostname -I
172.23.94.200 172.17.0.1
```

5. This is our edge IP address that we will be using with our Edge module

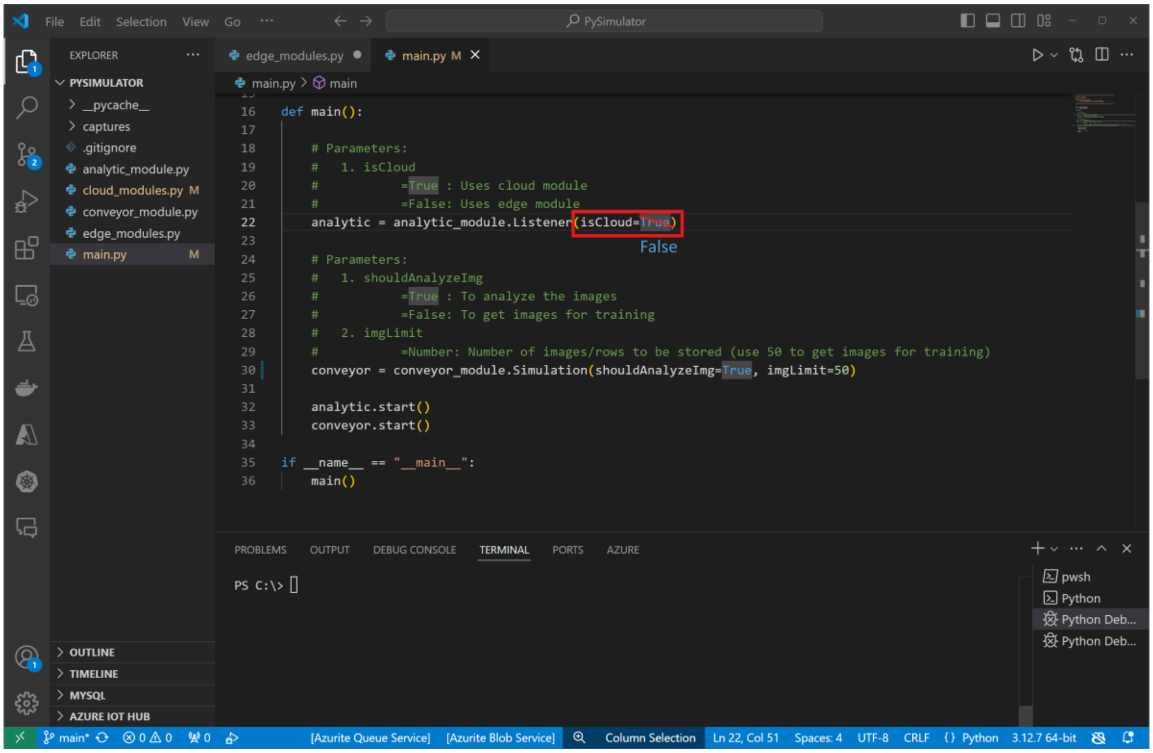
6. Go to the python code and open the "edge_module.py" and update "[EDGE URL]" with your IP address. E.g. if the IP address shows as 172.23.94.200, then the URL will be "http://172.23.94.200:80/image". Update the same API key used in the cloud module above (Step-4.1)



```
------------------
Change this file appropriately
    1. update the endpoint URL
    2. update the API Key or update your environment variable
    3. (If required) change the logic

************************************************************************
'''


import time
import json
import requests
import os

class EdgeAnalyzer:
    def __init__(self):
        self.endPoint = 'http://172.23.94.200:80/image'
        self.apiKey = '[API KEY]'

    def analyzeImage(self, filePath) -> int:
        start_time = time.time()
        url = self.endPoint
        headers = {
```

```
PS C:\>
```

7. Now set the isCLoud=True to False, in the "main.py" to run the edge module instead of the cloud module.

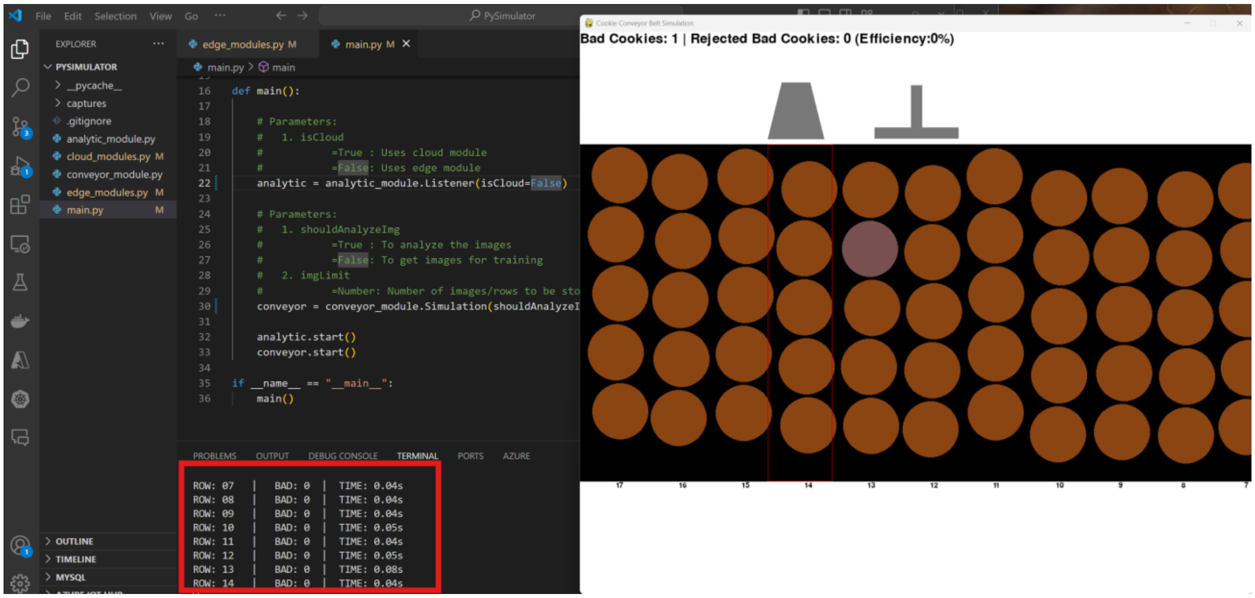


```
def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=True)

    # Parameters:
    #   1. shouldAnalyzeImg
    #           =True : To analyze the images
    #           =False: To get images for training
    #   2. imgLimit
    #           =Number: Number of images/rows to be stored (use 50 to get images for training)
    conveyor = conveyor_module.Simulation(shouldAnalyzeImg=True, imgLimit=50)

    analytic.start()
    conveyor.start()

if __name__ == "__main__":
    main()
```

False

```
PS C:\>
```

8. Run the application in debug mode (refer to the point Step-2.1 above). Observe the RTT (round trip time) to get the results from the edge.



```
def main():

    # Parameters:
    #   1. isCloud
    #           =True : Uses cloud module
    #           =False: Uses edge module
    analytic = analytic_module.Listener(isCloud=False)

    # Parameters:
    #   1. shouldAnalyzeImg
    #           =True : To analyze the images
    #           =False: To get images for training
    #   2. imgLimit
    #           =Number: Number of images/rows to be sto
    conveyor = conveyor_module.Simulation(shouldAnalyzeI

    analytic.start()
    conveyor.start()

if __name__ == "__main__":
    main()
```

```
ROW: 07  |   BAD: 0 |  TIME: 0.04s
ROW: 08  |   BAD: 0 |  TIME: 0.04s
ROW: 09  |   BAD: 0 |  TIME: 0.04s
ROW: 10  |   BAD: 0 |  TIME: 0.05s
ROW: 11  |   BAD: 0 |  TIME: 0.04s
ROW: 12  |   BAD: 0 |  TIME: 0.05s
ROW: 13  |   BAD: 0 |  TIME: 0.08s
ROW: 14  |   BAD: 0 |  TIME: 0.04s
```

Bad Cookies: 1 | Rejected Bad Cookies: 0 (Efficiency:0%)

Step 7: Assignment

1. Create a PPT file in your OneDrive (like assignment 1) and capture the following 7 screen into 7 different slides and share it with me. The title slide should have your name and student Id. All the text should be clear. If you miss anything, you will get 0 marks.
    i. Customvision.ai portal screen showing the iterations and the precision

    ii. The tagged images screen in the customvision.ai portal

    iii. Output of the published prediction URL

    iv. Output of the docker container in running state

    v. cloud_module.py code update screen

    vi. edge_module.py code update screen

    vii. Screen shot of edge_module output in the terminal window along with the simulator running side-by-side