Detect objects in images

The **Azure AI Custom Vision** service enables you to create computer vision models that are trained on your own images. You can use it to train *image classification* and *object detection* models; which you can then publish and consume from applications.

In this exercise, you will use the Custom Vision service to train an *object detection* model that can detect and locate three classes of fruit (apple, banana, and orange) in an image.

While this exercise is based on the Azure Custom Vision Python SDK, you can develop vision applications using multiple language-specific SDKs; including:

- Azure Custom Vision for JavaScript (training)
- Azure Custom Vision for JavaScript (prediction)
- Azure Custom Vision for Microsoft .NET (training)
- Azure Custom Vision for Microsoft .NET (prediction)
- Azure Custom Vision for Java (training)
- Azure Custom Vision for Java (prediction)

This exercise takes approximately **45** minutes.

Create Custom Vision resources

Before you can train a model, you will need Azure resources for *training* and *prediction*. You can create **Custom Vision** resources for each of these tasks, or you can create a single resource and use it for both. In this exercise, you'll create **Custom Vision** resources for training and prediction.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using your Azure credentials. Close any welcome messages or tips that are displayed.
2. Select **Create a resource**.
3. In the search bar, search for `Custom Vision`, select **Custom Vision**, and create the resource with the following settings:
   - **Create options**: Both
   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *Create or select a resource group*
   - **Region**: *Choose any available region*
   - **Name**: *A valid name for your Custom Vision resource*
   - **Training pricing tier**: F0
   - **Prediction pricing tier**: F0
4. Create the resource and wait for deployment to complete, and then view the deployment details. Note that two Custom Vision resources are provisioned; one for training, and another for prediction.

   Note: Each resource has its own *endpoint* and *keys*, which are used to manage access from your code. To train an image classification model, your code must use the *training* resource (with its endpoint and key); and to use the trained model to predict image classes, your code must use the *prediction* resource (with its endpoint and key).

5. When the resources have been deployed, go to the resource group to view them. You should see two custom vision resources, one with the suffix ***-Prediction***.

Create a Custom Vision project in the Custom Vision portal

To train an object detection model, you need to create a Custom Vision project based on your training resource. To do this, you'll use the Custom Vision portal.

1. Open a new browser tab (keeping the Azure portal tab open - you'll return to it later).

2. In the new browser tab, open the [Custom Vision portal](https://customvision.ai) at `https://customvision.ai`. If prompted, sign in using your Azure credentials and agree to the terms of service.
3. Create a new project with the following settings:
   - **Name**: `Detect Fruit`
   - **Description**: `Object detection for fruit.`
   - **Resource**: *Your Custom Vision resource*
   - **Project Types**: Object Detection
   - **Domains**: General
4. Wait for the project to be created and opened in the browser.

Upload and tag images

Now that you have an object detection project, you can upload and tag images to train a model.

Upload and tag images in the Custom Vision portal

The Custom Vision portal includes visual tools that you can use to upload images and tag regions within them that contain multiple types of object.

1. In a new browser tab, download the [training images](https://github.com/MicrosoftLearning/mslearn-ai-vision/raw/main/Labfiles/object-detection/training-images.zip) from

```
https://github.com/MicrosoftLearning/mslearn-ai-vision/raw/main/Labfiles/object-
detection/training-images.zip
```

   and extract the zip folder to view its contents. This folder contains images of fruit.
2. In the Custom Vision portal, in your object detection project, select **Add images** and upload all of the images in the extracted folder.
3. After the images have been uploaded, select the first one to open it.
4. Hold the mouse over any object in the image until an automatically detected region is displayed like the image below. Then select the object, and if necessary resize the region to surround it.

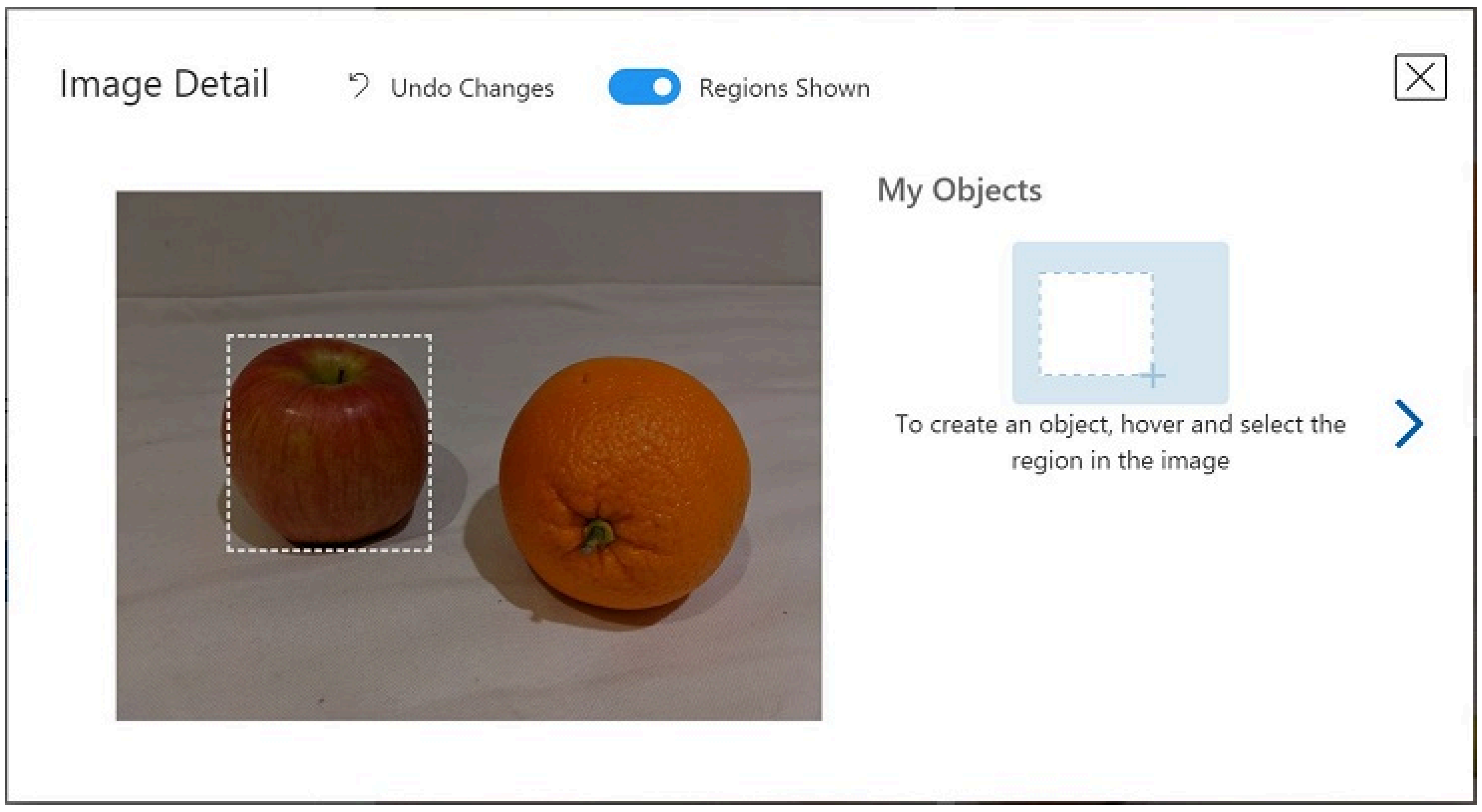


Image Detail
Undo Changes
Regions Shown
My Objects
To create an object, hover and select the region in the image

   Alternatively, you can simply drag around the object to create a region.

5. When the region surrounds the object, add a new tag with the appropriate object type (*apple*, *banana*, or *orange*) as shown here:

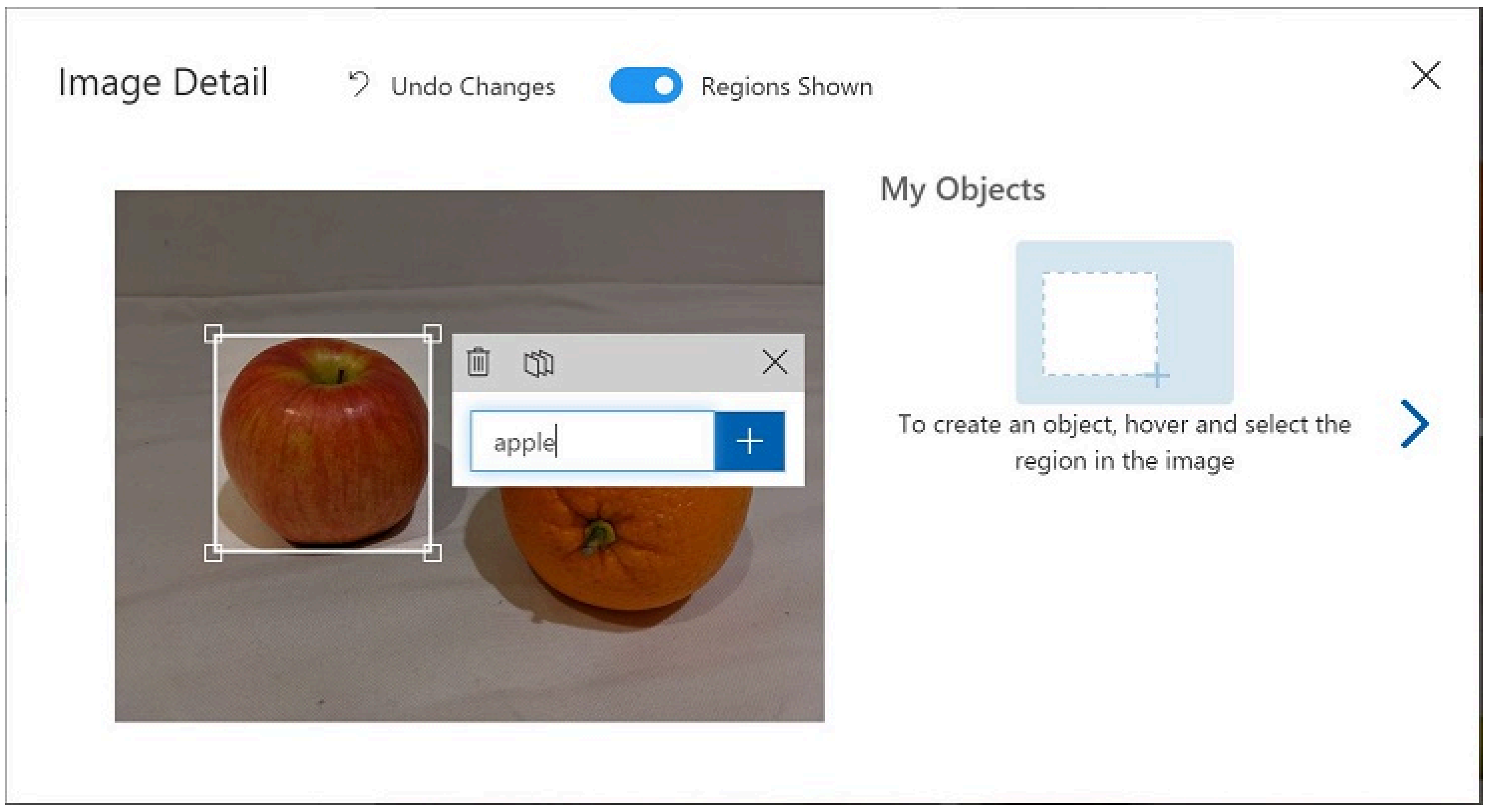
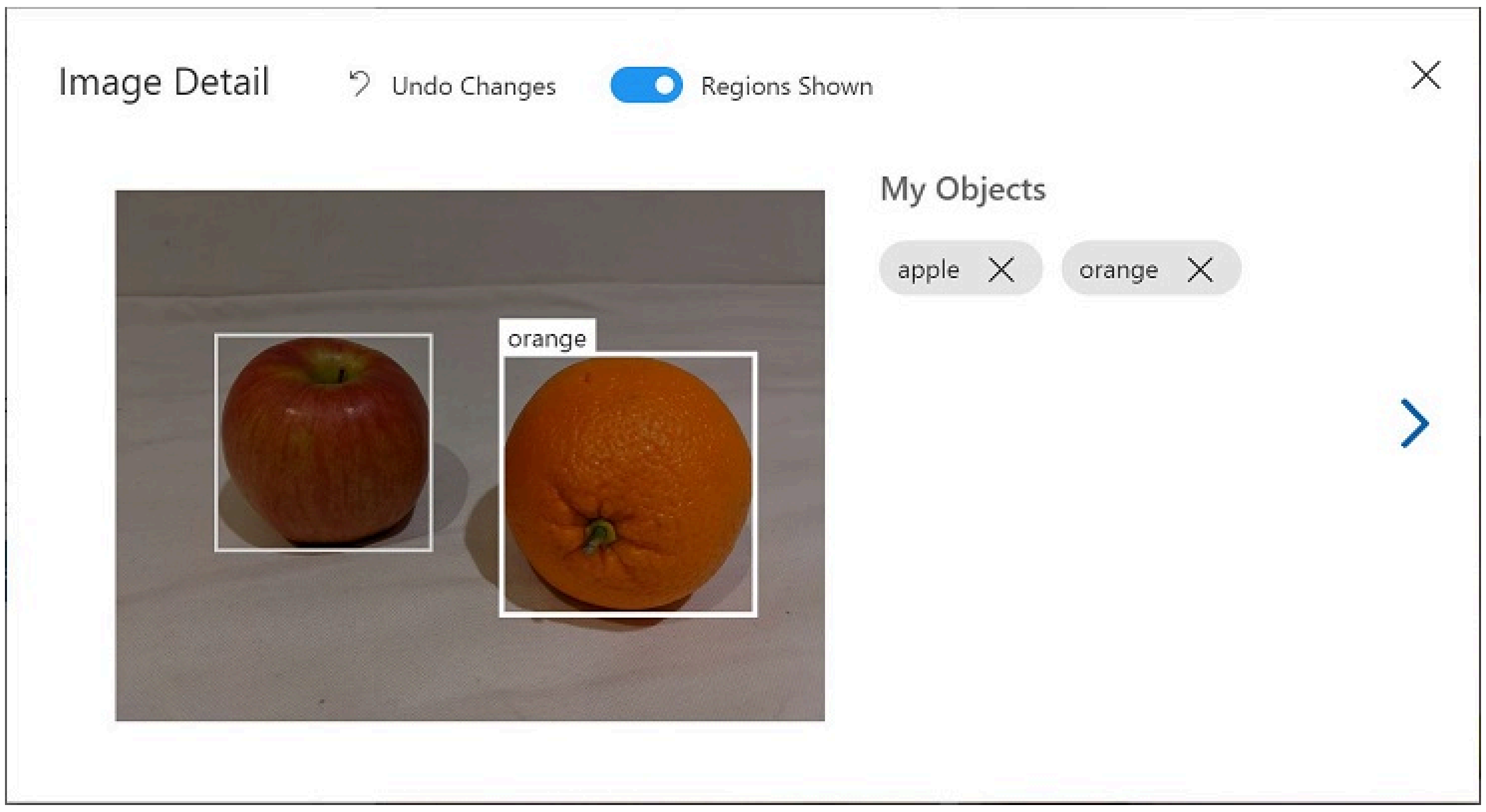
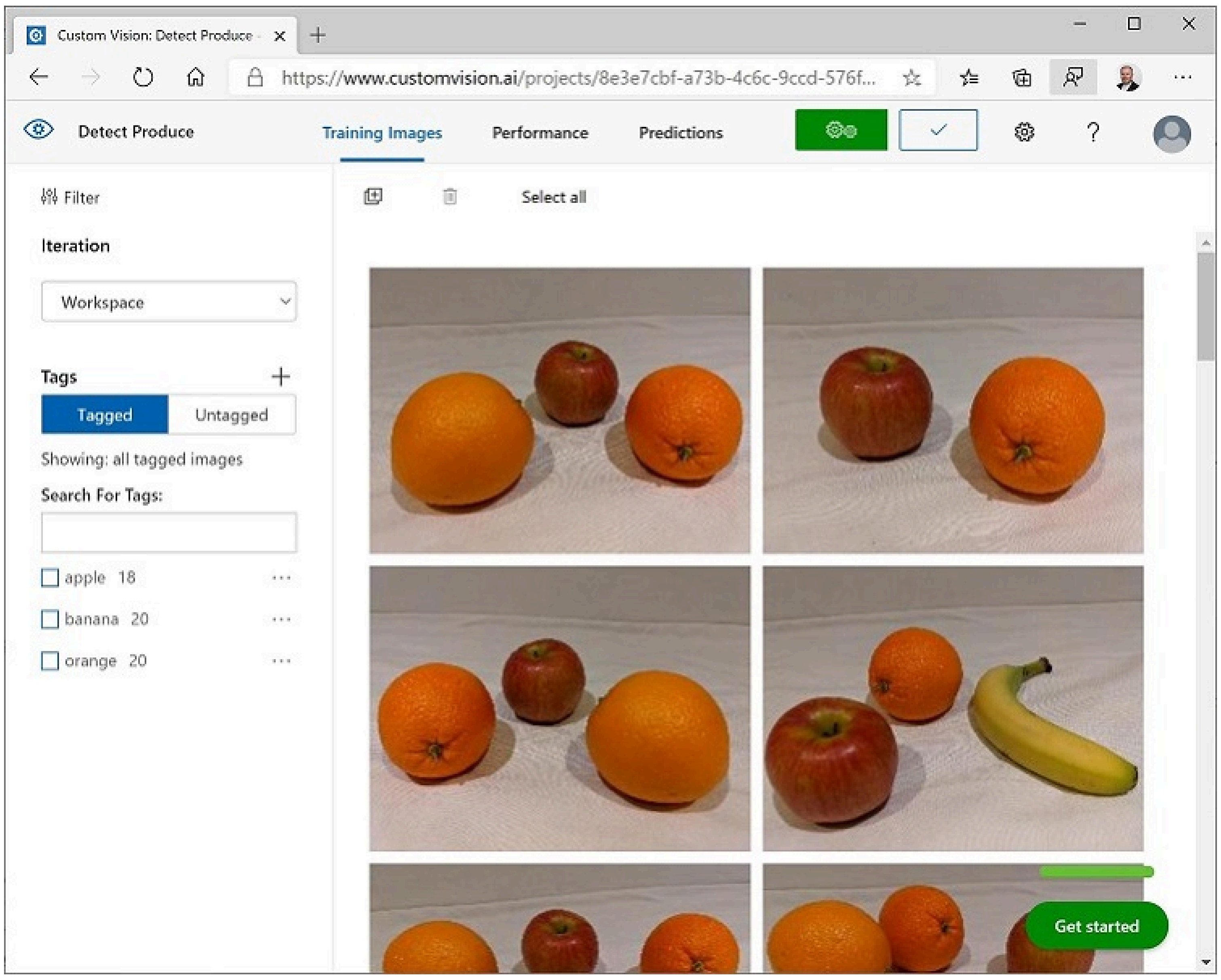


Image Detail
Undo Changes
Regions Shown
apple
My Objects
To create an object, hover and select the region in the image

6. Select and tag each other object in the image, resizing the regions and adding new tags as required.



7. Use the **>** link on the right to go to the next image, and tag its objects. Then just keep working through the entire image collection, tagging each apple, banana, and orange.
8. When you have finished tagging the last image, close the **Image Detail** editor. On the **Training Images** page, under **Tags**, select **Tagged** to see all of your tagged images:



Use the Custom Vision SDK to upload images

You can use the UI in the Custom Vision portal to tag your images, but many AI development teams use other tools that generate files containing information about tags and object regions in images. In scenarios like this, you can use the Custom Vision training API to upload tagged images to the project.

1. Click the *settings* (⚙) icon at the top right of the **Training Images** page in the Custom Vision portal to view the project settings.
2. Under **General** (on the left), note the **Project Id** that uniquely identifies this project.
3. On the right, under **Resources** note that the **Key** and **Endpoint** are shown. These are the details for the *training* resource (you can also obtain this information by viewing the resource in the Azure portal).
4. Return to the browser tab containing the Azure portal (keeping the Custom Vision portal tab open - you'll return to it later).
5. In the Azure portal, use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment with no storage in your subscription.

   The cloud shell provides a command-line interface in a pane at the bottom of the Azure portal.

   > **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

6. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

7. Resize the cloud shell pane so you can see more of it.

   > **Tip**" You can resize the pane by dragging the top border. You can also use the minimize and maximize buttons to switch between the cloud shell and the main portal interface.

8. In the cloud shell pane, enter the following commands to clone the GitHub repo containing the code files for this exercise (type the command, or copy it to the clipboard and then right-click in the command line and paste as plain text):

Code Copy

```
rm -r mslearn-ai-vision -f
git clone https://github.com/MicrosoftLearning/mslearn-ai-vision
```

Tip: As you paste commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

9. After the repo has been cloned, use the following command to navigate to the application code files:

   Code Copy

   ```
   cd mslearn-ai-vision/Labfiles/object-detection/python/train-detector
   ls -a -l
   ```

   The folder contains application configuration and code files for your app. It also contains a **tagged-images.json** file which contains bounding box coordinates for objects in multiple images, and an **/images** subfolder, which contains the images.

10. Install the Azure AI Custom Vision SDK package for training and any other required packages by running the following commands:

    Code Copy

    ```
    python -m venv labenv
    ./labenv/bin/Activate.ps1
    pip install -r requirements.txt azure-cognitiveservices-vision-customvision
    ```

11. Enter the following command to edit the configuration file for your app:

    Code Copy

    ```
    code .env
    ```

    The file is opened in a code editor.
12. In the code file, update the configuration values it contains to reflect the **Endpoint** and an authentication **Key** for your Custom Vision *training* resource, and the **Project ID** for the custom vision project you created previously.
13. After you've replaced the placeholders, within the code editor, use the **CTRL+S** command to save your changes and then use the **CTRL+Q** command to close the code editor while keeping the cloud shell command line open.

14. In the cloud shell command line, enter the following command to open the **tagged-images.json** file to see the tagging information for the image files in the **/images** subfolder:

    Code Copy

    ```
    code tagged-images.json
    ```

    JSON defines a list of images, each containing one or more tagged regions. Each tagged region includes a tag name, and the top and left coordinates and width and height dimensions of the bounding box containing the tagged object.

> **Note**: The coordinates and dimensions in this file indicate relative points on the image. For example, a *height* value of 0.7 indicates a box that is 70% of the height of the image. Some tagging tools generate other formats of file in which the coordinate and dimension values represent pixels, inches, or other units of measurements.

15. Close the JSON file without saving any changes (*CTRL_Q*).
16. In the cloud shell command line, enter the following command to open the code file for the client application:

Code Copy

```
code add-tagged-images.py
```

17. Note the following details in the code file:
    - The namespaces for the Azure AI Custom Vision SDK are imported.
    - The **Main** function retrieves the configuration settings, and uses the key and endpoint to create an authenticated **CustomVisionTrainingClient**, which is then used with the project ID to create a **Project** reference to your project.
    - The **Upload_Images** function extracts the tagged region information from the JSON file and uses it to create a batch of images with regions, which it then uploads to the project.
18. Close the code editor (*CTRL+Q*) and enter the following command to run the program:

Code Copy

```
python add-tagged-images.py
```

19. Wait for the program to end.
20. Switch back to the browser tab containing the Custom Vision portal (keeping the Azure portal cloud shell tab open), and view the **Training Images** page for your project (refreshing the browser if necessary).
21. Verify that some new tagged images have been added to the project.

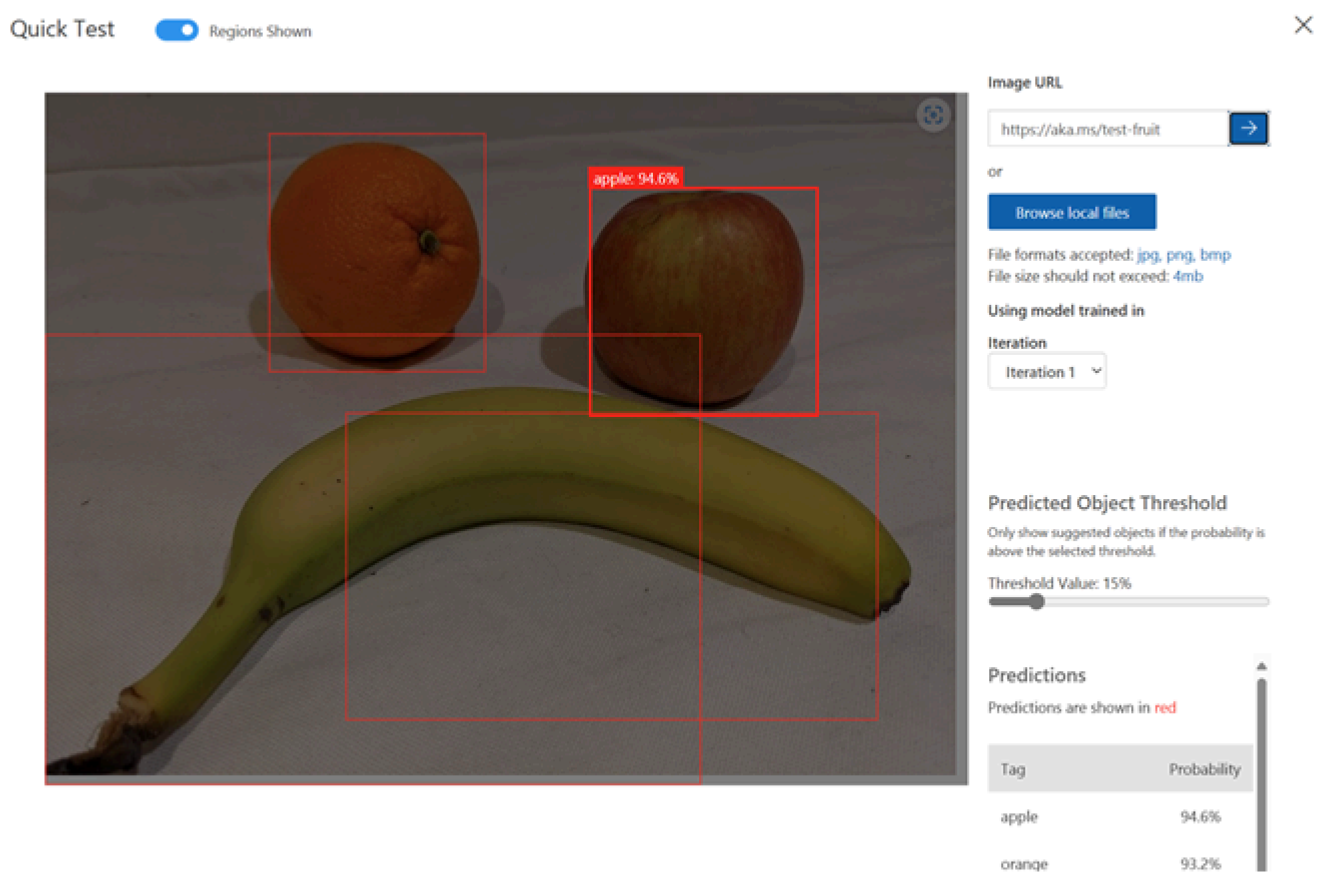
Train and test a model

Now that you've tagged the images in your project, you're ready to train a model.

1. In the Custom Vision project, click **Train** (⚙⚙) to train an object detection model using the tagged images. Select the **Quick Training** option.
2. Wait for training to complete (it might take ten minutes or so).

> **Tip**: The Azure cloud shell has a 20-minute inactivity timeout, after which the session is abandoned. While you wait for training to finish, occassionally return to the cloud shell and enter a colland like `ls` to keep the session active.

3. In the Custom Vision portal, when training has finished, review the *Precision*, *Recall*, and *mAP* performance metrics - these measure the prediction accuracy of the object detection model, and should all be high.
4. At the top right of the page, click **Quick Test**, and then in the **Image URL** box, type `https://aka.ms/test-fruit` and click the *quick test image* (➔) button.
5. View the prediction that is generated.


Quick Test
Regions Shown
apple: 94.6%
Image URL
https://aka.ms/test-fruit
or
Browse local files
File formats accepted: jpg, png, bmp
File size should not exceed: 4mb
Using model trained in
Iteration
Iteration 1
Predicted Object Threshold
Only show suggested objects if the probability is above the selected threshold.
Threshold Value: 15%
Predictions
Predictions are shown in red

Tag	Probability
apple	94.6%
orange	93.2%

6. Close the **Quick Test** window.

Use the object detector in a client application

Now you’re ready to publish your trained model and use it in a client application.

Publish the object detection model

1. In the Custom Vision portal, on the **Performance** page, click ✓ **Publish** to publish the trained model with the following settings:
   - **Model name**: `fruit-detector`
   - **Prediction Resource**: *The **prediction** resource you created previously which ends with “-Prediction” (<u>not</u> the training resource).*
2. At the top left of the **Project Settings** page, click the *Projects Gallery* (👁) icon to return to the Custom Vision portal home page, where your project is now listed.
3. On the Custom Vision portal home page, at the top right, click the *settings* (⚙) icon to view the settings for your Custom Vision service. Then, under **Resources**, find your *prediction* resource which ends with “-Prediction” (<u>not</u> the training resource) to determine its **Key** and **Endpoint** values (you can also obtain this information by viewing the resource in the Azure portal).

Use the image classifier from a client application

Now that you’ve published the image classification model, you can use it from a client application. Once again, you can choose to use **C#** or **Python**.

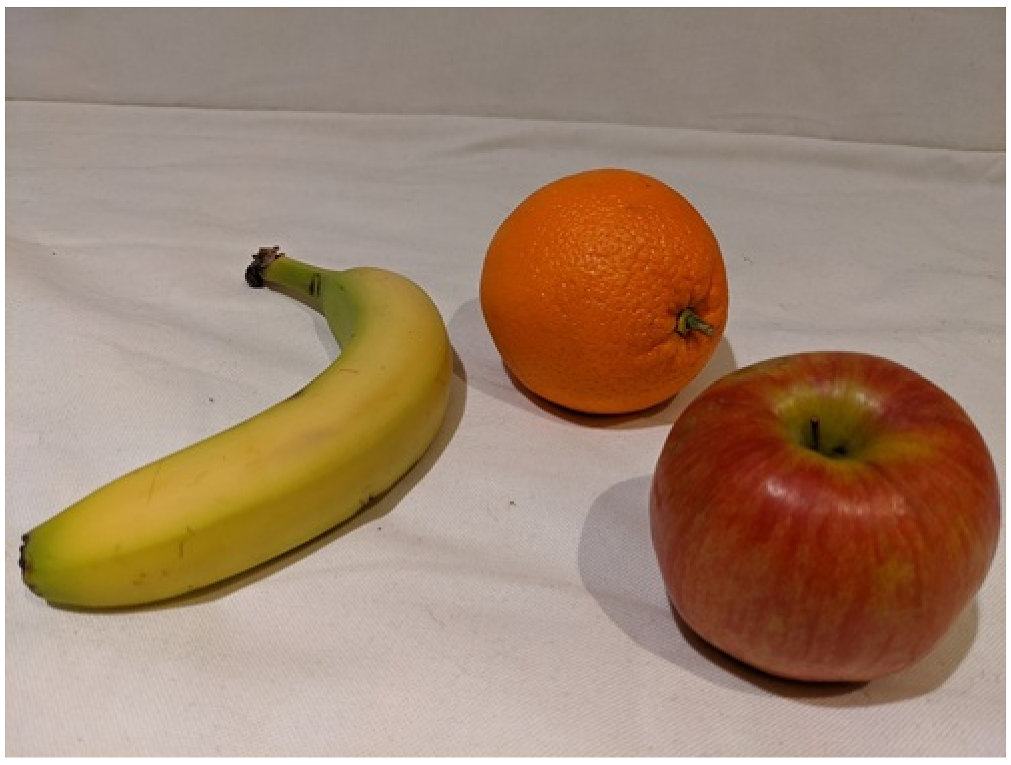
1. Return to the browser tab containing the Azure portal and the cloud shell pane.

2. In cloud shell, run the following commands to switch to the folder for you client application and view the files it contains:

   Code

   ```
   cd ../test-detector
   ls -a -l
   ```

   The folder contains application configuration and code files for your app. It also contains the following **produce.jpg** image file, which you’ll use to test your model.



3. Install the Azure AI Custom Vision SDK package for prediction and any other required packages by running the following commands:

Code Copy

```
python -m venv labenv
./labenv/bin/Activate.ps1
pip install -r requirements.txt azure-cognitiveservices-vision-customvision
```

4. Enter the following command to edit the configuration file for your app:

Code Copy

```
code .env
```

The file is opened in a code editor.

5. Update the configuration values to reflect the **Endpoint** and **Key** for your Custom Vision *prediction* resource, the **Project ID** for the object detection project, and the name of your published model (which should be *fruit-detector*). Save your changes (*CTRL+S*) and close the code editor (*CTRL+Q*).

6. In the cloud shell command line, enter the following command to open the code file for the client application:

Code Copy

```
code test-detector.py
```

7. Review the code, noting the following details:
   - The namespaces for the Azure AI Custom Vision SDK are imported.
   - The **Main** function retrieves the configuration settings, and uses the key and endpoint to create an authenticated **CustomVisionPredictionClient**.
   - The prediction client object is used to get object detection predictions for the **produce.jpg** image, specifying the project ID and model name in the request. The predicted tagged regions are then drawn on the image, and the result is saved as **output.jpg**.

8. Close the code editor and enter the following command to run the program:

Code Copy

```
python test-detector.py
```

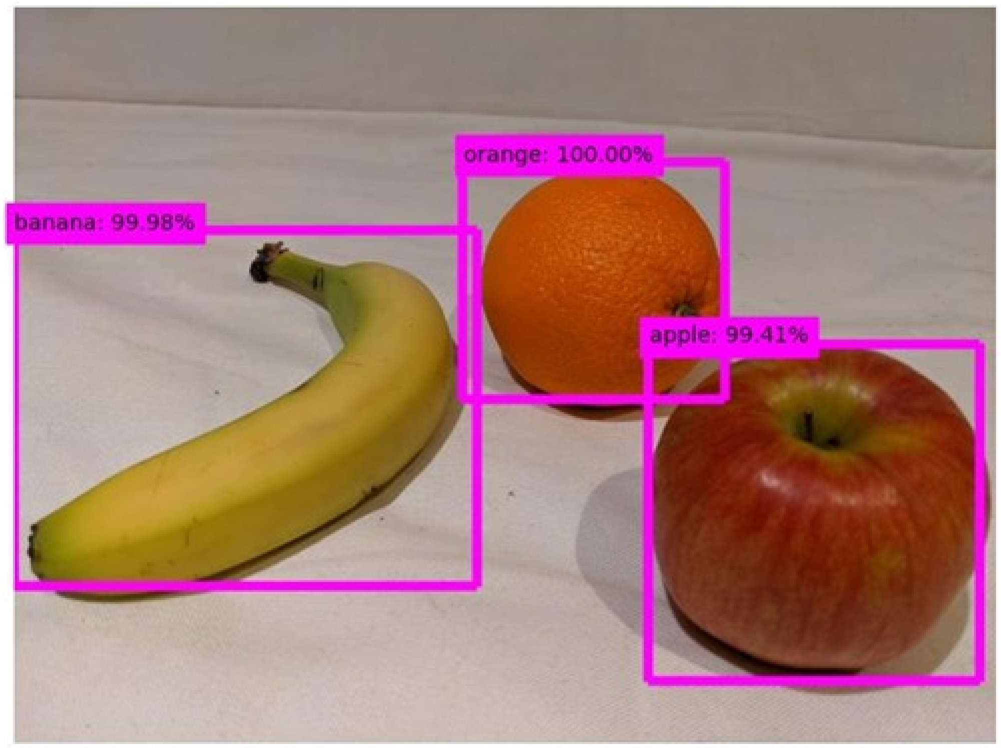
9. Review the program output, which lists each object detected in the image.

10. Note that an image file named **output.jpg** is generated. Use the (Azure cloud shell-specific) **download** command to download it:

Code

```
download output.jpg
```

The download command creates a popup link at the bottom right of your browser, which you can select to download and open the file. The image should look simlar to this:



Clean up resources

If you’re not using the Azure resources created in this lab for other training modules, you can delete them to avoid incurring further charges.

1. Open the Azure portal at `https://portal.azure.com`, and in the top search bar, search for the resources you created in this lab.

2. On the resource page, select **Delete** and follow the instructions to delete the resource. Alternatively, you can delete the entire resource group to clean up all resources at the same time.

More information

For more information about object detection with the Custom Vision service, see the Custom Vision documentation.