Explore face recognition

> ⓘ **Note** To complete this lab, you will need an [Azure subscription](#) in which you have administrative access.

Computer vision solutions often require an artificial intelligence (AI) solution to be able to detect human faces. For example, suppose the retail company Northwind Traders wants to locate where customers are standing in a store to best assist them. One way to accomplish this is to determine if there are any faces in the images, and if so, to identify the bounding box coordinates around the faces.

To test the capabilities of the Face service, we'll use a simple command-line application that runs in the Cloud Shell. The same principles and functionality apply in real-world solutions, such as web sites or phone apps.

Create a *Face API* resource

You can use the Face service by creating a **Face** resource. (Face API is no longer available in Cognitive Services)

If you haven't already done so, create a **Face API** resource in your Azure subscription.

1. In another browser tab, open the Azure portal at https://portal.azure.com, signing in with your Microsoft account.
2. Click the **+ Create a resource** button, search for *Face*, and create a **Face** resource with the following settings:

   - **Subscription**: *Your Azure subscription*.
   - **Resource group**: *Select or create a resource group with a unique name*.
   - **Region**: *Choose any available region*.
   - **Name**: *Enter a unique name*.
   - **Pricing tier**: Free F0

3. Review and create the resource, and wait for deployment to complete. Then go to the deployed resource.
4. View the **Keys and Endpoint** page for your Face resource. You will need the endpoint and keys to connect from client applications.
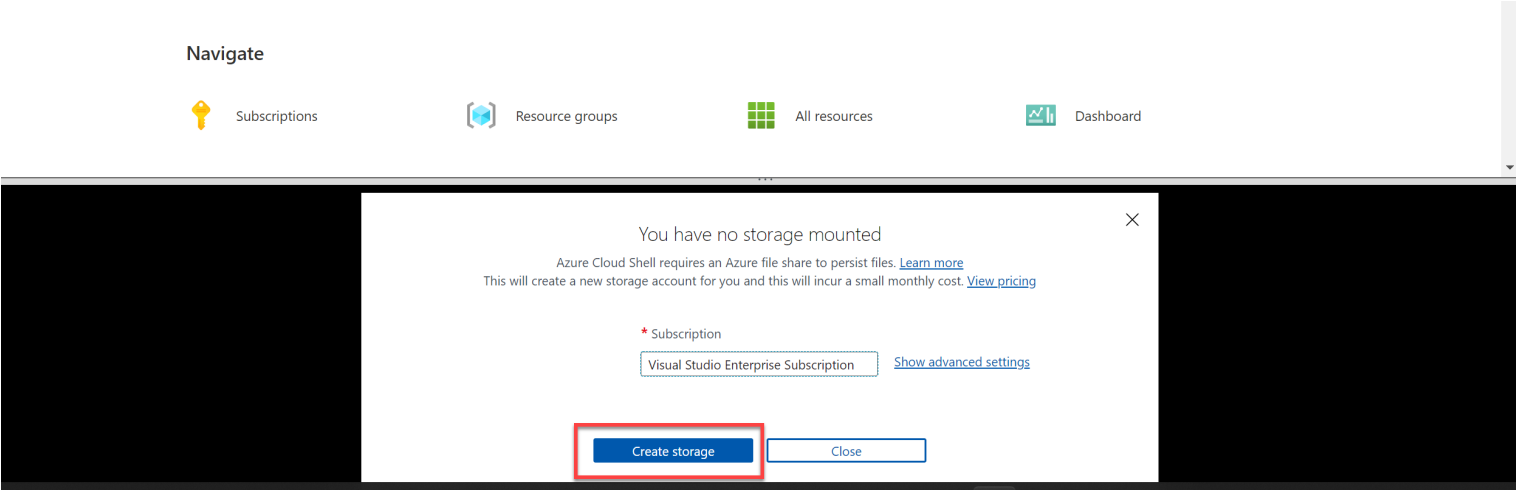
Run Cloud Shell

To test the capabilities of the Face service, we'll use a simple command-line application that runs in the Cloud Shell on Azure.

1. In the Azure portal, select the **[>_]** (*Cloud Shell*) button at the top of the page to the right of the search box. This opens a Cloud Shell pane at the bottom of the portal.
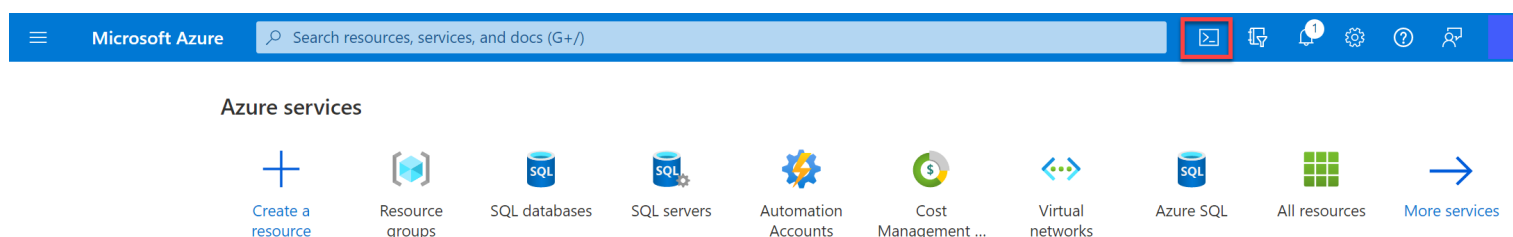
2. The first time you open the Cloud Shell, you may be prompted to choose the type of shell you want to use (*Bash* or *PowerShell*). Select **PowerShell**. If you do not see this option, skip the step.

3. If you are prompted to create storage for your Cloud Shell, ensure your subscription is specified and select **Create storage**. Then wait a minute or so for the storage to be created.

You have no storage mounted

×

Azure Cloud Shell requires an Azure file share to persist files. Learn more
This will create a new storage account for you and this will incur a small monthly cost. View pricing

* Subscription

Visual Studio Enterprise Subscription   Show advanced settings
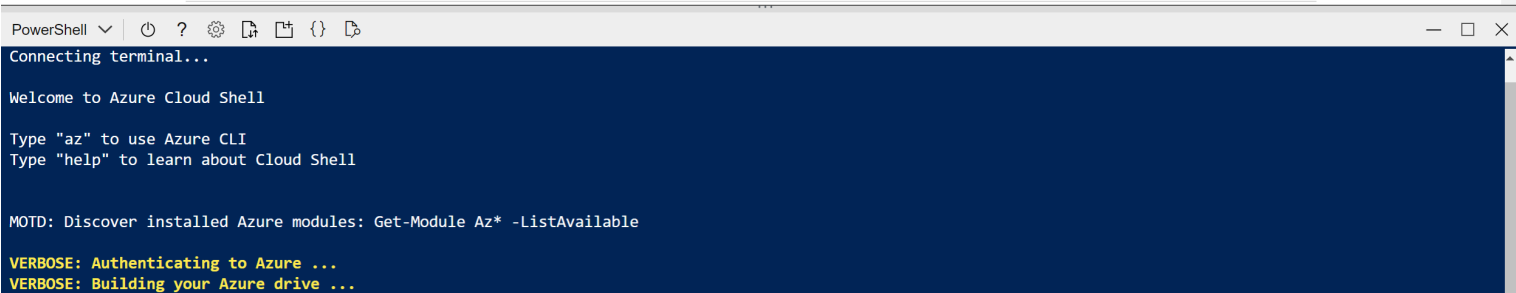
Create storage   Close

4. Make sure the type of shell indicated on the top left of the Cloud Shell pane is switched to *PowerShell*. If it is *Bash*, switch to *PowerShell* by using the drop-down menu.

```
Bash ⌄   ⏻  ?  ⚙  ⎘  ⎙  { }  ⎙
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell
```
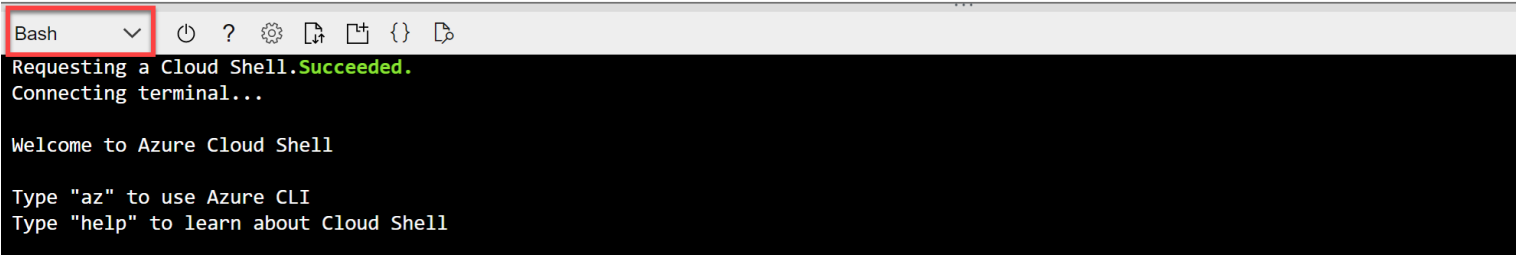
5. Wait for PowerShell to start. You should see the following screen in the Azure portal:

```
PowerShell ⌄   ⏻  ?  ⚙  ⎘  ⎙  { }  ⎙                                    —  □  ×
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

MOTD: Discover installed Azure modules: Get-Module Az* -ListAvailable

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
```

Configure and run a client application

Now that you have a custom model, you can run a simple client application that uses the Face service.

1. In the command shell, enter the following command to download the sample application and save it to a folder called ai-900.

Code	🗐 Copy

```
git clone https://github.com/MicrosoftLearning/AI-900-AIFundamentals ai-900
```

> **Tip** If you already used this command in another lab to clone the *ai-900* repository, you can skip this step.

2. The files are downloaded to a folder named **ai-900**. Now we want to see all of the files in your Cloud Shell storage and work with them. Type the following command into the shell:

Code	🗐 Copy

```
code .
```

Notice how this opens up an editor like the one in the image below:

PowerShell ∨ | ⏻ ? ⚙ ⤓ ⤒ {} 🗋

Untitled

FILES   ↻   1 |
▷ .azure
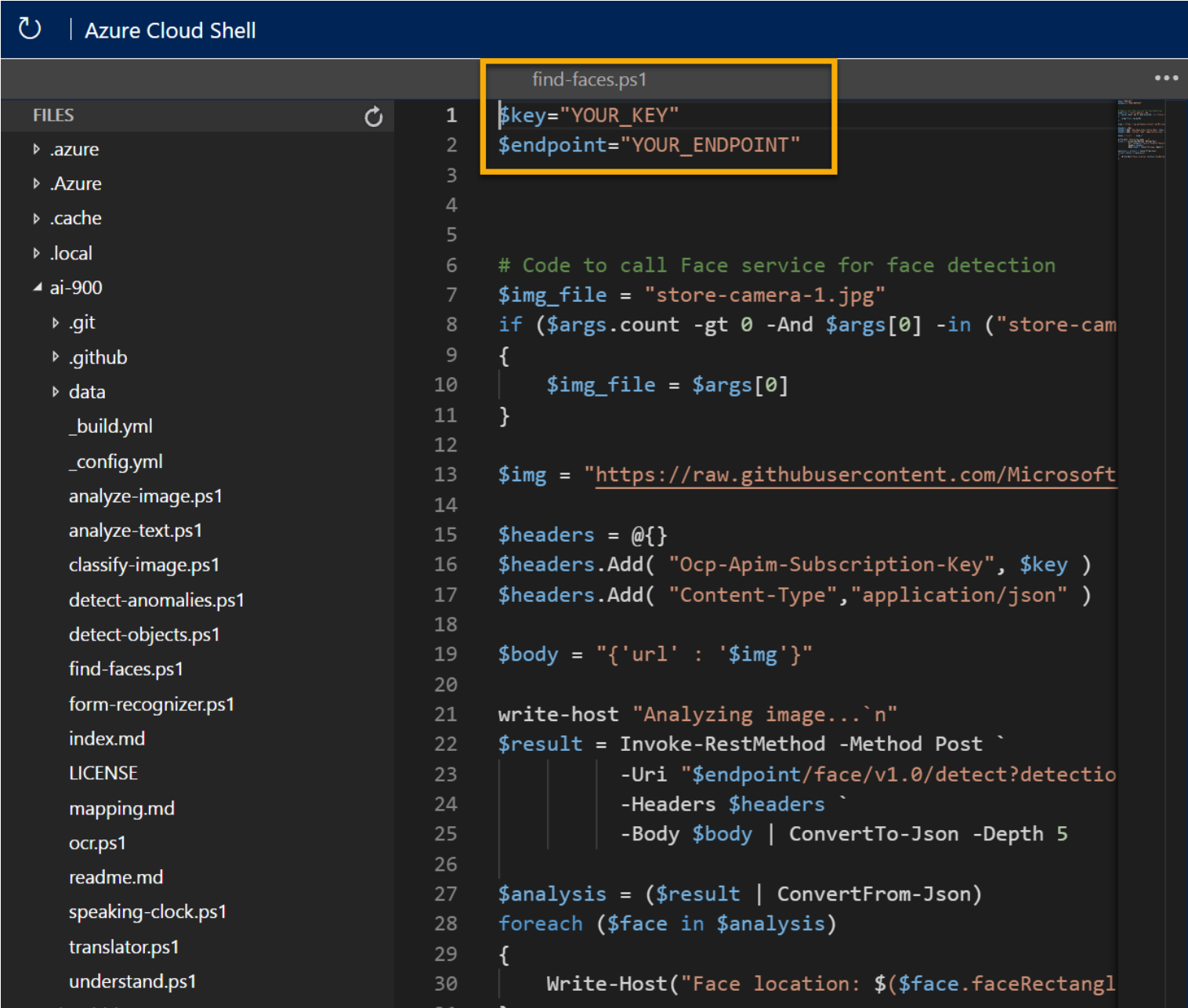▷ .Azure
▷ .cache
▷ .local
▷ clouddrive
  .bash_logout
  .bashrc

VERBOSE: Building your Azure drive ...
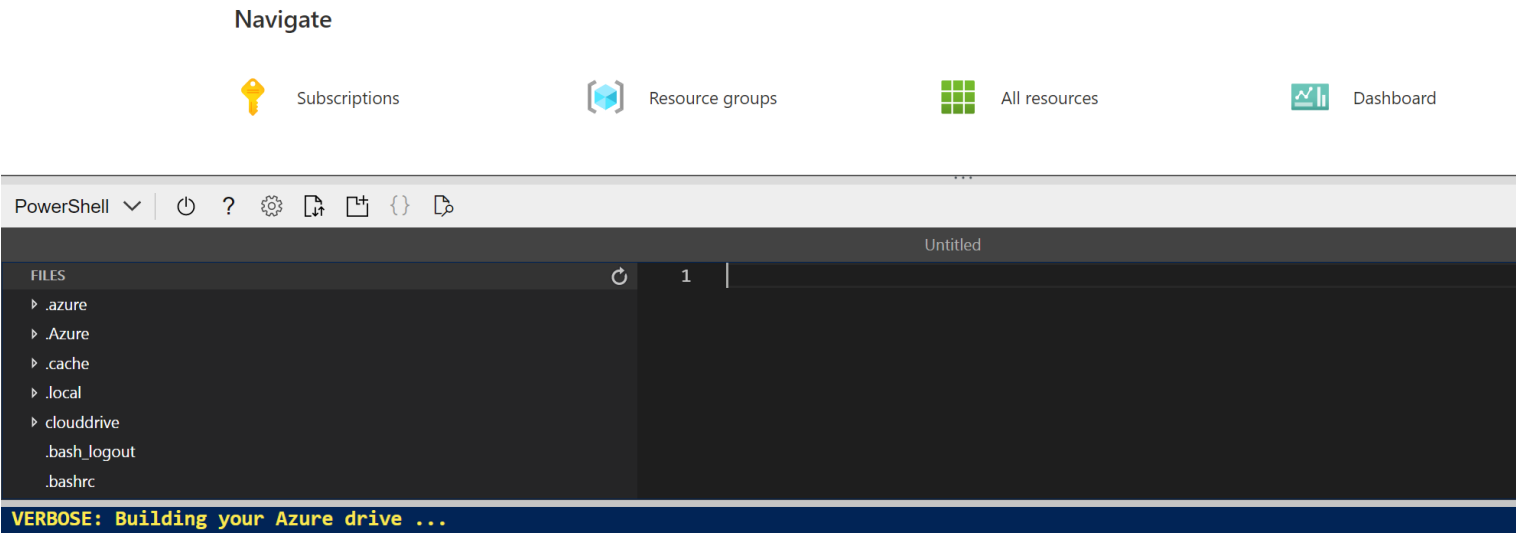
3. In the **Files** pane on the left, expand **ai-900** and select **find-faces.ps1**. This file contains some code that uses the Face service to detect and analyze faces in an image, as shown here:

↻ | Azure Cloud Shell

```
                            find-faces.ps1                              ...
FILES                 ↻   1  $key="YOUR_KEY"
▷ .azure                  2  $endpoint="YOUR_ENDPOINT"
▷ .Azure                  3
▷ .cache                  4
▷ .local                  5
▲ ai-900                  6  # Code to call Face service for face detection
  ▷ .git                  7  $img_file = "store-camera-1.jpg"
  ▷ .github               8  if ($args.count -gt 0 -And $args[0] -in ("store-cam
  ▷ data                  9  {
    _build.yml           10      $img_file = $args[0]
    _config.yml          11  }
    analyze-image.ps1    12
    analyze-text.ps1     13  $img = "https://raw.githubusercontent.com/Microsoft
    classify-image.ps1   14
    detect-anomalies.ps1 15  $headers = @{}
    detect-objects.ps1   16  $headers.Add( "Ocp-Apim-Subscription-Key", $key )
    find-faces.ps1       17  $headers.Add( "Content-Type","application/json" )
    form-recognizer.ps1  18
    index.md             19  $body = "{'url' : '$img'}"
    LICENSE              20
    mapping.md           21  write-host "Analyzing image...`n"
    ocr.ps1              22  $result = Invoke-RestMethod -Method Post `
    readme.md            23          -Uri "$endpoint/face/v1.0/detect?detectio
    speaking-clock.ps1   24          -Headers $headers `
    translator.ps1       25          -Body $body | ConvertTo-Json -Depth 5
    understand.ps1       26
                         27  $analysis = ($result | ConvertFrom-Json)
                         28  foreach ($face in $analysis)
                         29  {
                         30      Write-Host("Face location: $($face.faceRectangl
```

4. Don't worry too much about the details of the code, the important thing is that it needs the endpoint URL and either of the keys for your Face resource. Copy these from the **Keys and Endpoints** page for your resource from the Azure portal and paste them into the code editor, replacing the **YOUR_KEY** and **YOUR_ENDPOINT** placeholder values respectively.

> 🗲 **Tip** You may need to use the separator bar to adjust the screen area as you work with the **Keys and Endpoint** and **Editor** panes.

After pasting the key and endpoint values, the first two lines of code should look similar to this:

Code	🗗 Copy

```
$key="1a2b3c4d5e6f7g8h9i0j...."
$endpoint="https..."
```

5. At the top right of the editor pane, use the **...** button to open the menu and select **Save** to save your changes. Then open the menu again and select **Close Editor**.
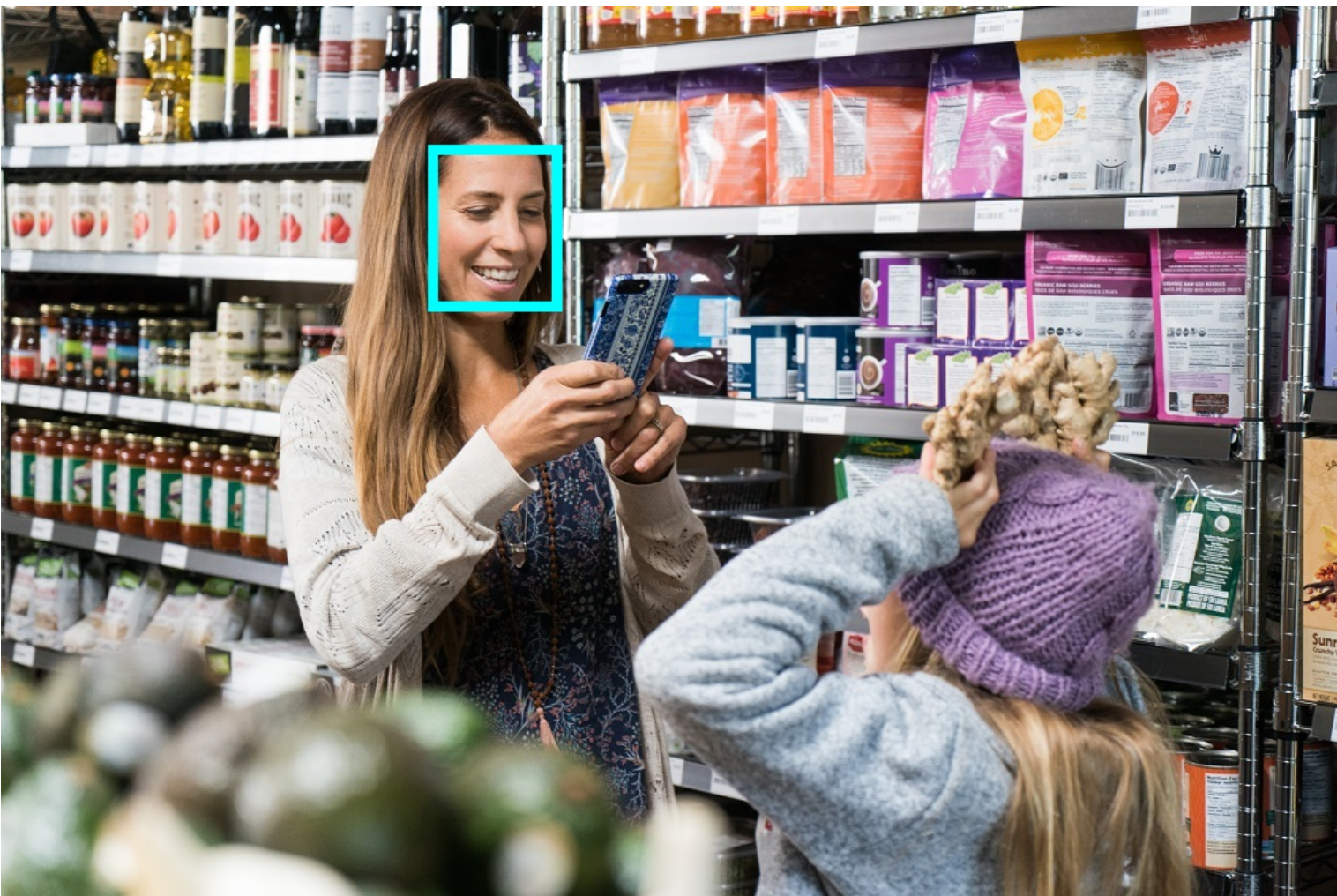
The sample client application will use your Face service to analyze the following image, taken by a camera in the Northwind Traders store:

6. In the PowerShell pane, enter the following commands to run the code:

```
Code                                                        Copy

  cd ai-900
  ./find-faces.ps1 store-camera-1.jpg
```

7. Review the returned information, which includes the location of the face in the image. The location of a face is indicated by the top-left coordinates, and the width and height of a *bounding box*, as shown here:

> ⓘ **Note** Face service capabilities that return personally identifiable features are restricted. See
> https://azure.microsoft.com/blog/responsible-ai-investments-and-safeguards-for-facial-recognition/ for details.

8. Now let's try another image:

To analyze the second image, enter the following command:

```
./find-faces.ps1 store-camera-2.jpg
```

9. Review the results of the face analysis for the second image.

10. Let's try one more:

To analyze the third image, enter the following command:

```
./find-faces.ps1 store-camera-3.jpg
```

11. Review the results of the face analysis for the third image.

Learn more

This simple app shows only some of the capabilities of the Face service. To learn more about what you can do with this service, see the Face API page.