

**Lab Manual- Detect, analyze, and identify faces (Lab using SDK)**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn918

**Contributor:**

Contents

[1. Objective 3](#_Toc167362836)

[2. Provision an Azure AI Services resource 3](#_Toc167362837)

[3. Copy Endpoint and Key 7](#_Toc167362838)

[4. Clone the repository in Visual Studio Code 8](#_Toc167362839)

[5. Install SDK Library 9](#_Toc167362840)

[6. Configure Endpoint and Key 10](#_Toc167362841)

[7. Use SDK 12](#_Toc167362842)

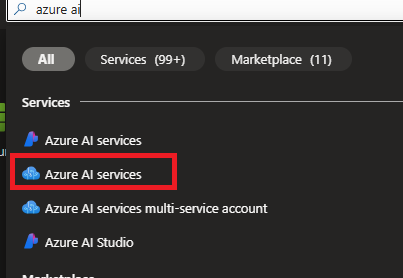
# Objective

The ability to detect and analyze human faces is a core AI capability. In this exercise, you'll explore two Azure AI Services that you can use to work with faces in images: the **Azure AI Vision** service, and the **Face** service and using it from a client application. The goal of the exercise is not to gain expertise in any particular service, but rather to become familiar with a general pattern for provisioning and working with Azure AI services as a developer.

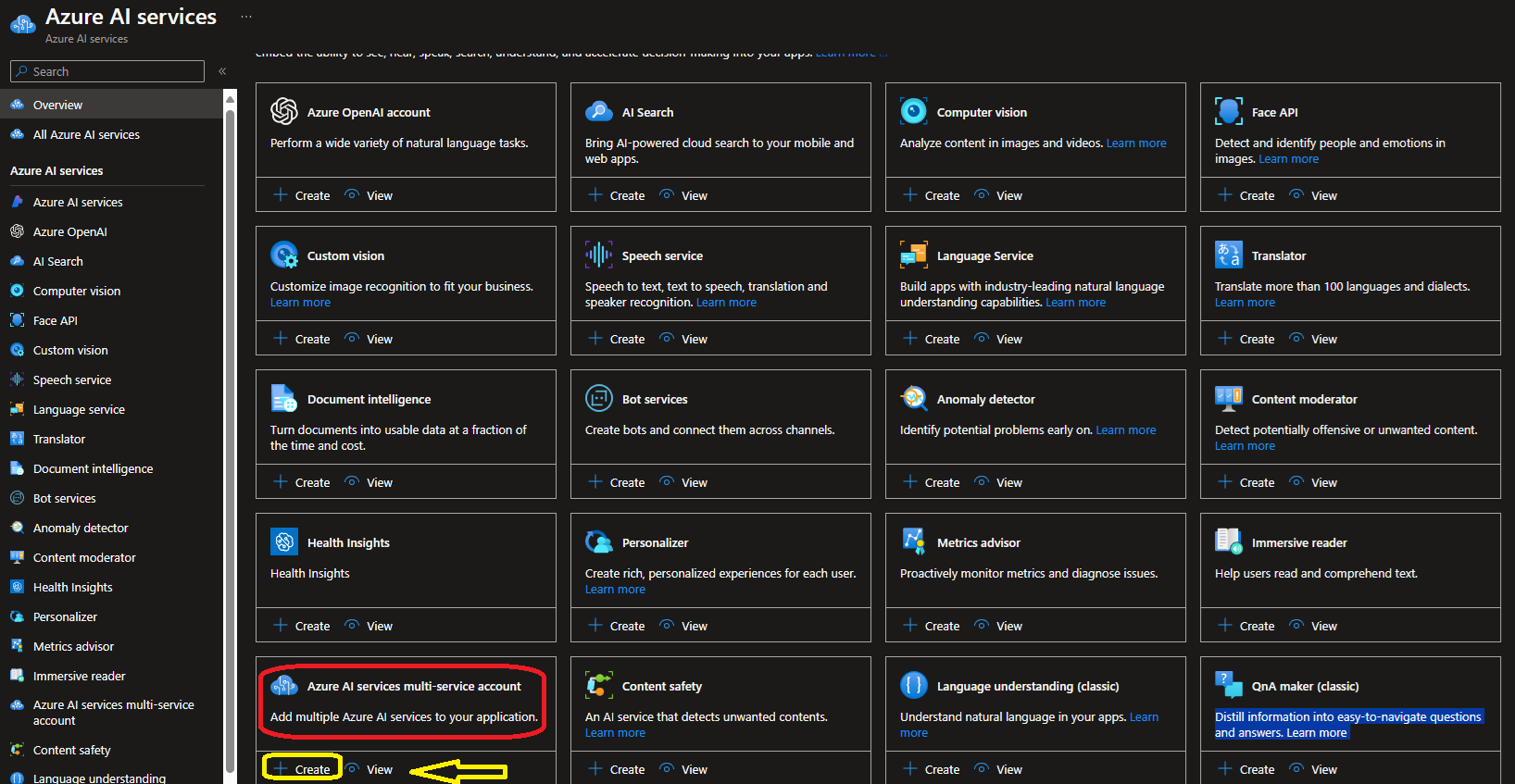
# Provision an Azure AI Services resource

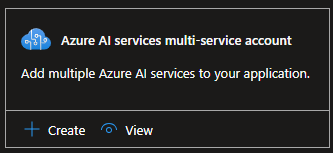
Azure AI Services are cloud-based services that encapsulate artificial intelligence capabilities you can incorporate into your applications. You can provision **individual Azure AI services** resources for **specific APIs** (for example, **Language** or **Vision**), or you can provision a single Azure AI Services resource that provides access to multiple Azure AI services APIs through a single **endpoint** and **key**. In this case, you'll use a single Azure AI Services resource

* Serach AI Service and select **Azure AI Services**

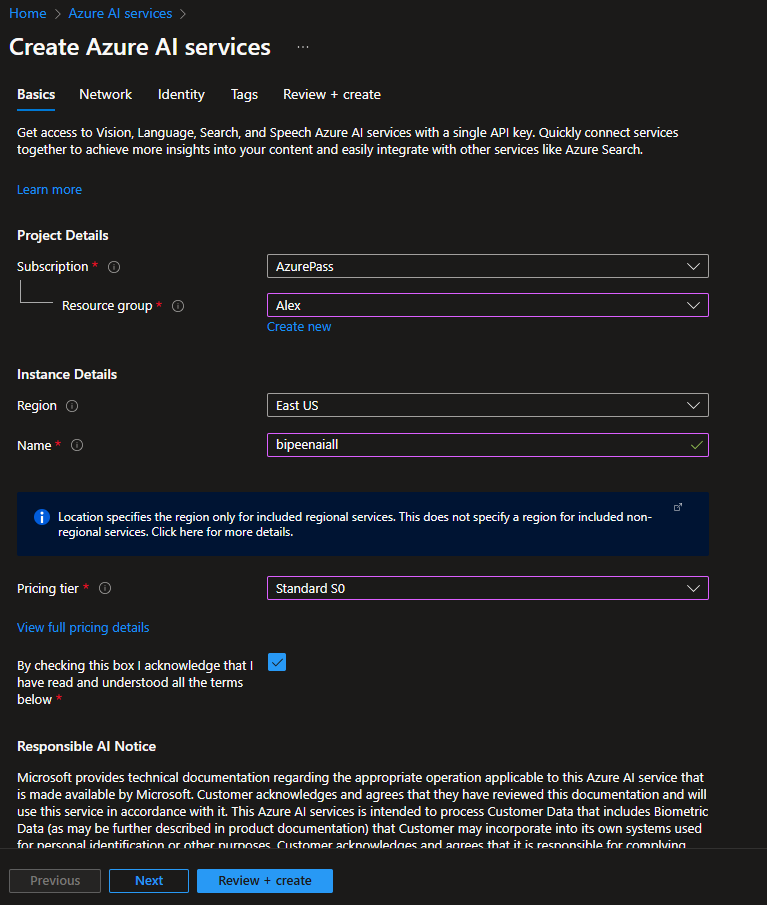


* Select **Azure Ai Service Muti-Service Account** and Click **create**

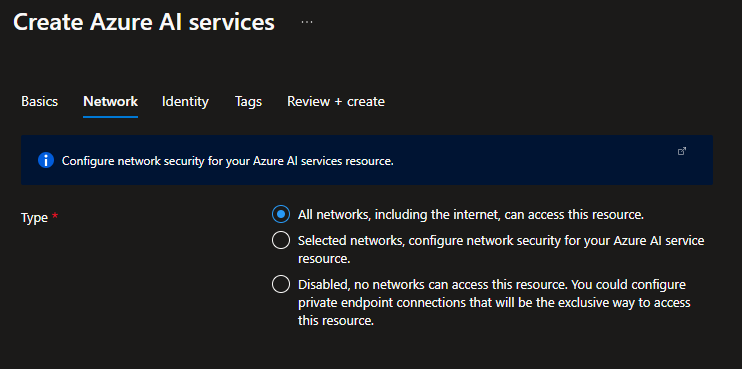




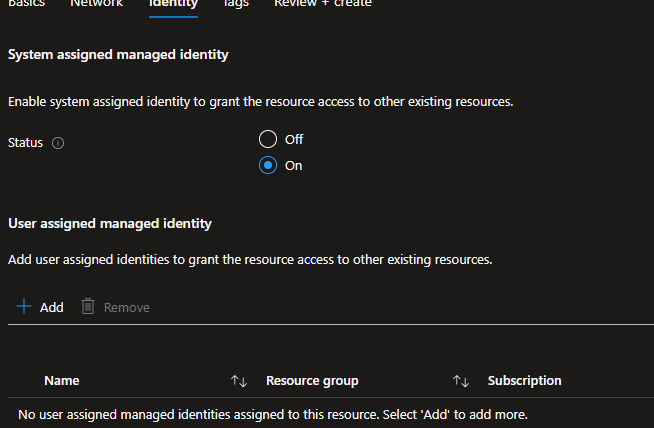
* Select your
  + Resource Group
  + Name of AI Instance
  + Pricing should be **Standard0**
  + **Responsible AI Terms- Checked**



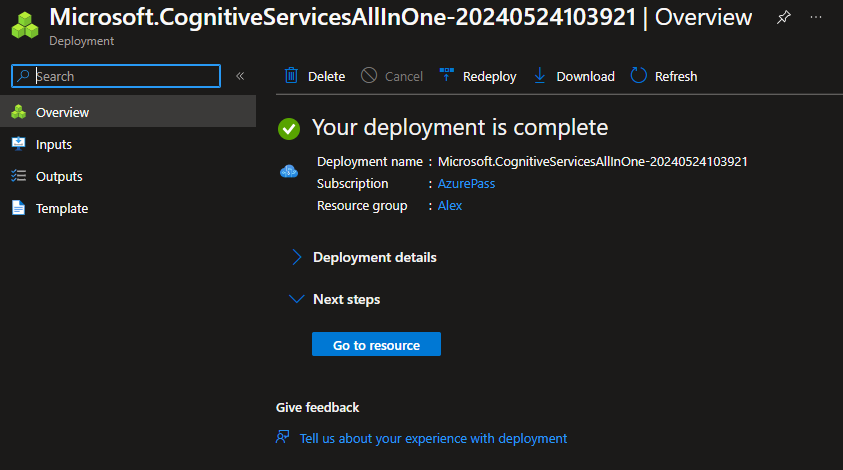
* Click **Next and Leave the default**



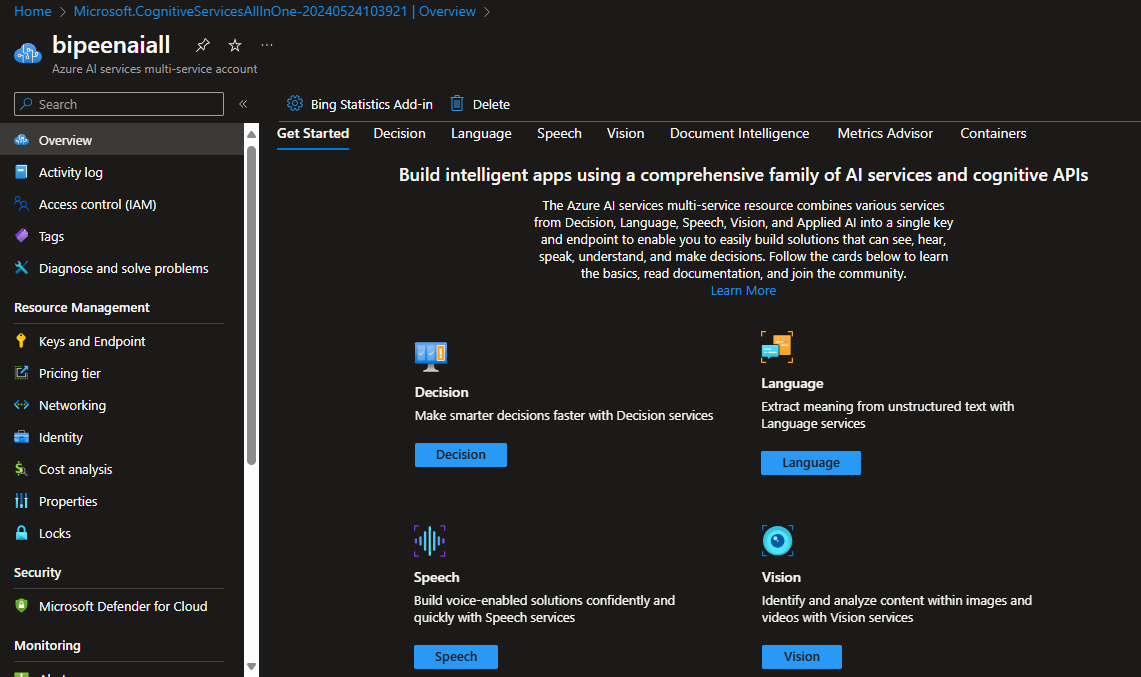
* Click **Next and on the System Managed identity**



* Click **Review and Create**

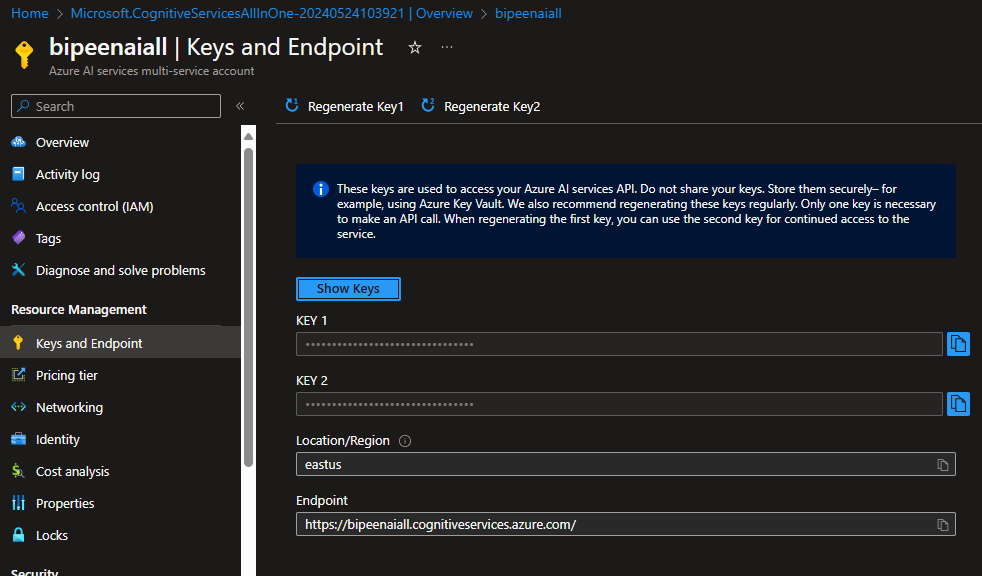


* Wait for deployment to complete, and then view the deployment details.



# Copy Endpoint and Key

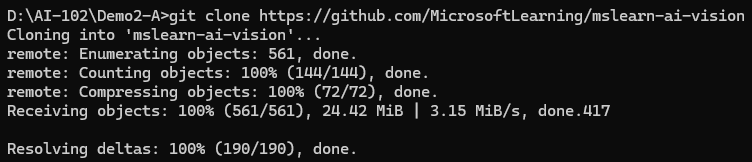
1. Wait for deployment to complete, and then view the deployment details.
2. Go to the resource and view its **Keys and Endpoint** page. This page contains the information that you will need to connect to your resource and use it from applications you develop. Specifically:
   * An HTTP *endpoint* to which client applications can send requests.
   * Two *keys* that can be used for authentication (client applications can use either key to authenticate).
   * The *location* where the resource is hosted. This is required for requests to some (but not all) APIs.



# Clone the repository in Visual Studio Code

1. Start Visual Studio Code / Command Prompt.
2. Run a **Git: Clone** command to clone the https://github.com/MicrosoftLearning/mslearn-ai-vision repository to a local folder (it doesn't matter which folder).

**git clone https://github.com/MicrosoftLearning/mslearn-ai-vision**

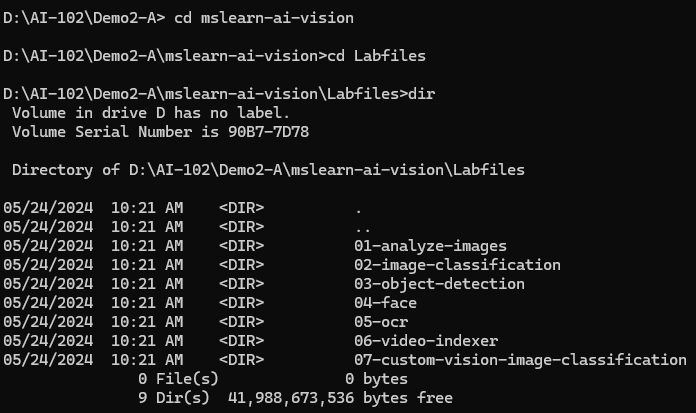


1. Go Inside **mslearn-ai-services/Labfiles** directory

**cd mslearn-ai-vision**

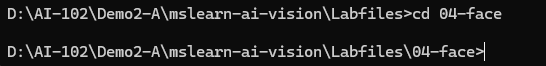
**Cd Labfiles**

**Dir**

****

1. Go Inside **04-face** directory under **Labfiles**

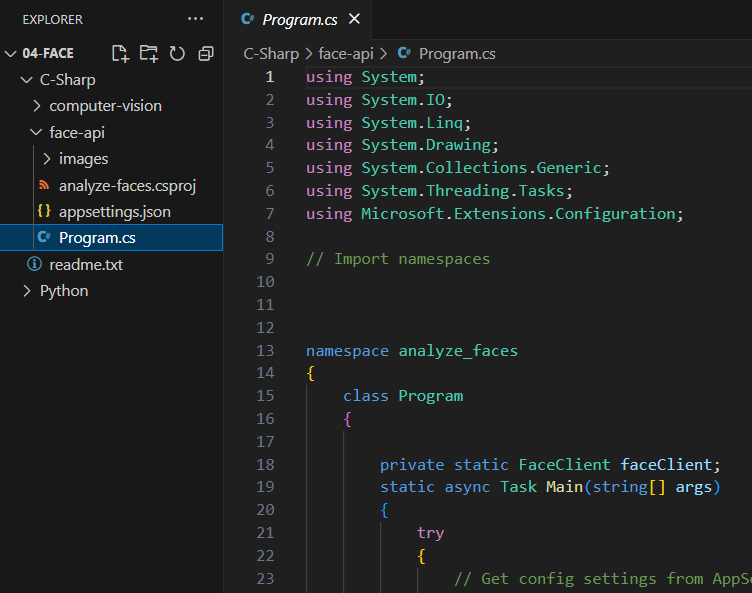
**cd 04-face**

****

1. Open the folder in Visual Studio Code.

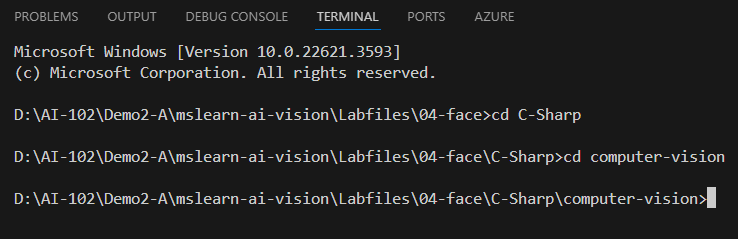
**Code .**

****

****

# Install SDK Library

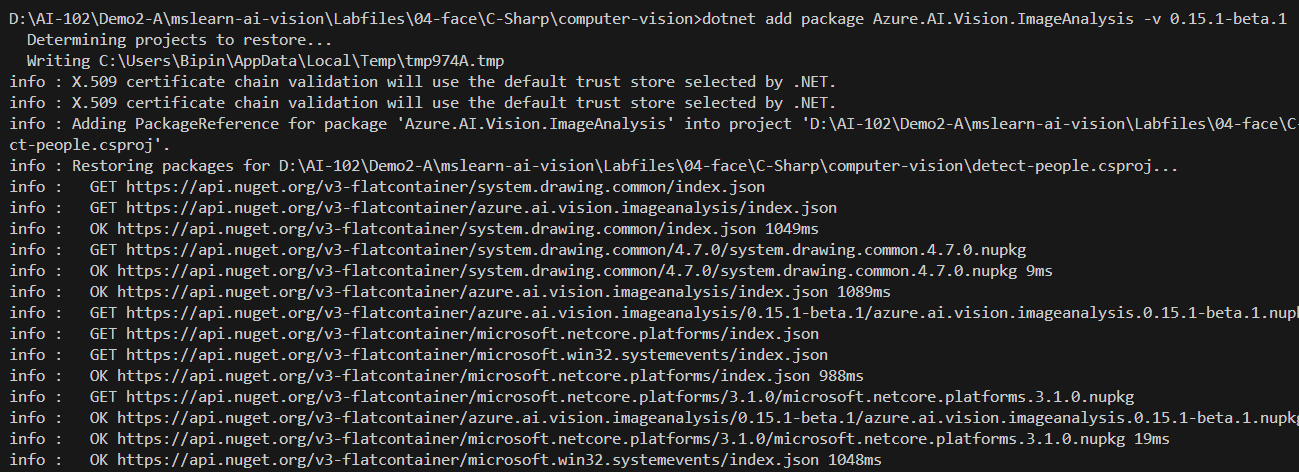
* In Visual Studio Code, Open Terminal
* Go Inside   **C-Sharp -> computer-vision**

****

* Install the Azure AI Vision SDK package.

**dotnet add package Azure.AI.Vision.ImageAnalysis -v 0.15.1-beta.1**

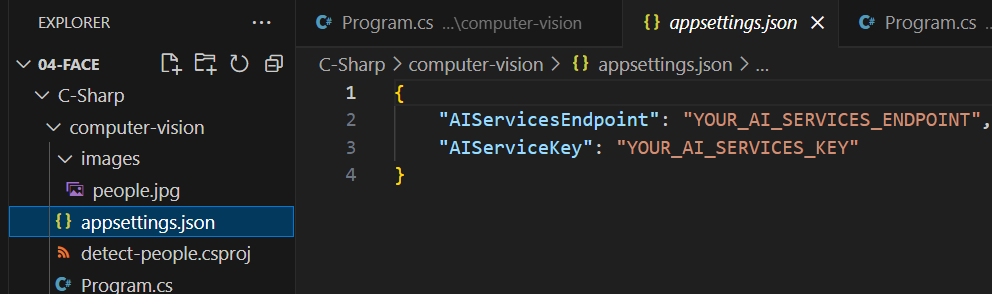




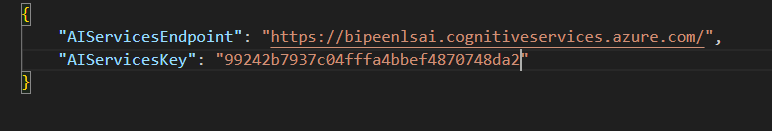
# Configure Endpoint and Key

* In Visual Studio Code, expand the **C-Sharp -> Computer-Vision**
* View the contents of the **Computer-Vision** folder, and note that it contains a file for configuration settings:

appsettings.json



* Open the configuration file and update the configuration values it contains to reflect the **endpoint** and an authentication **key** for your Azure AI services resource. **Save your** changes.



# Detect faces in an image – Update program.cs code

Open the code file **Program.cs** and review the code it contains, noting the following details:

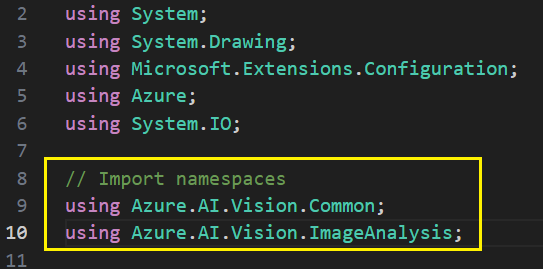
* Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Azure AI Vision SDK:

C#

// import namespaces

using Azure.AI.Vision.Common;

using Azure.AI.Vision.ImageAnalysis;



* Now you're ready to use the SDK to call the Vision service and detect faces in an image.

1. In the code file for your client application (**Program.cs** \_, in the **Main** function, note that the code to load the configuration settings has been provided. Then find the comment **Authenticate Azure AI Vision client**. Then, under this comment, add the following language-specific code to create and authenticate a Azure AI Vision client object:

**C#**

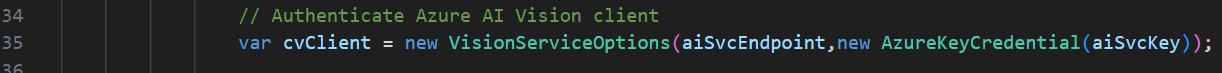
C#

// Authenticate Azure AI Vision client

var cvClient = new VisionServiceOptions(

aiSvcEndpoint,

new AzureKeyCredential(aiSvcKey));



1. In the **Main** function, under the code you just added, note that the code specifies the path to an image file and then passes the image path to a function named **AnalyzeImage**. This function is not yet fully implemented.
2. In the **AnalyzeImage** function, under the comment **Specify features to be retrieved (PEOPLE)**, add the following code:

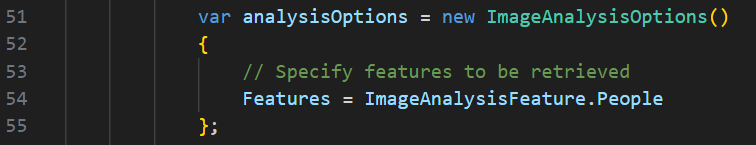
**C#**

C#

// Specify features to be retrieved (PEOPLE)

Features =

ImageAnalysisFeature.People



1. In the **AnalyzeImage** function, under the comment **Get image analysis**, add the following code:

**C#**

C

// Get image analysis

using var imageSource = VisionSource.FromFile(imageFile);

using var analyzer = new ImageAnalyzer(serviceOptions, imageSource, analysisOptions);

var result = analyzer.Analyze();

if (result.Reason == ImageAnalysisResultReason.Analyzed)

{

// Get people in the image

if (result.People != null)

{

Console.WriteLine($" People:");

// Prepare image for drawing

System.Drawing.Image image = System.Drawing.Image.FromFile(imageFile);

Graphics graphics = Graphics.FromImage(image);

Pen pen = new Pen(Color.Cyan, 3);

Font font = new Font("Arial", 16);

SolidBrush brush = new SolidBrush(Color.WhiteSmoke);

foreach (var person in result.People)

{

// Draw object bounding box if confidence &gt; 50%

if (person.Confidence &gt; 0.5)

{

// Draw object bounding box

var r = person.BoundingBox;

Rectangle rect = new Rectangle(r.X, r.Y, r.Width, r.Height);

graphics.DrawRectangle(pen, rect);

// Return the confidence of the person detected

Console.WriteLine($" Bounding box {person.BoundingBox}, Confidence {person.Confidence:0.0000}");

}

}

// Save annotated image

String output\_file = "detected\_people.jpg";

image.Save(output\_file);

Console.WriteLine(" Results saved in " + output\_file + "\n");

}

}

else

{

var errorDetails = ImageAnalysisErrorDetails.FromResult(result);

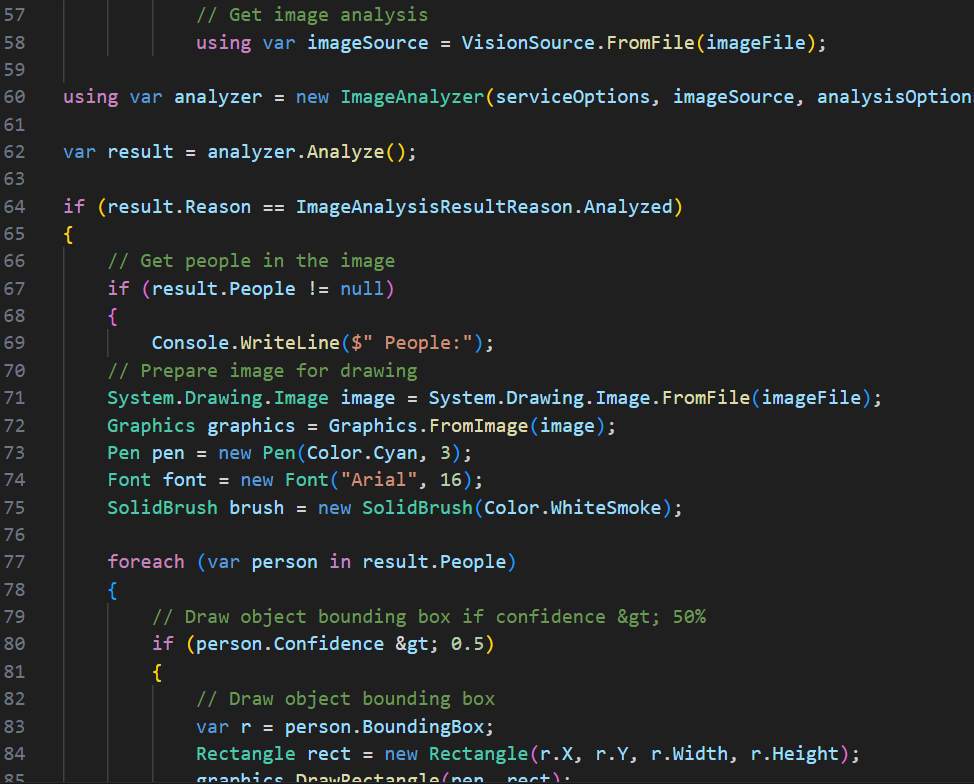
Console.WriteLine(" Analysis failed.");

Console.WriteLine($" Error reason : {errorDetails.Reason}");

Console.WriteLine($" Error code : {errorDetails.ErrorCode}");

Console.WriteLine($" Error message: {errorDetails.Message}\n");

}



**The Complete Code**:

// dotnet add package Azure.AI.Vision.ImageAnalysis -v 0.15.1-beta.1

using System;

using System.Drawing;

using Microsoft.Extensions.Configuration;

using Azure;

using System.IO;

// Import namespaces

using Azure.AI.Vision.Common;

using Azure.AI.Vision.ImageAnalysis;

namespace detect\_people

{

    class Program

    {

        static void Main(string[] args)

        {

            try

            {

                // Get config settings from AppSettings

                IConfigurationBuilder builder = new ConfigurationBuilder().AddJsonFile("appsettings.json");

                IConfigurationRoot configuration = builder.Build();

                string aiSvcEndpoint = configuration["AIServicesEndpoint"];

                string aiSvcKey = configuration["AIServiceKey"];

                // Get image

                string imageFile = "images/people.jpg";

                if (args.Length > 0)

                {

                    imageFile = args[0];

                }

                // Authenticate Azure AI Vision client

                var cvClient = new VisionServiceOptions(aiSvcEndpoint,new AzureKeyCredential(aiSvcKey));

                // Analyze image

                AnalyzeImage(imageFile, cvClient);

            }

            catch (Exception ex)

            {

                Console.WriteLine(ex.Message);

            }

        }

        static void AnalyzeImage(string imageFile, VisionServiceOptions serviceOptions)

        {

            Console.WriteLine($"\nAnalyzing {imageFile} \n");

            var analysisOptions = new ImageAnalysisOptions()

            {

                // Specify features to be retrieved

                Features = ImageAnalysisFeature.People

            };

            // Get image analysis

            using var imageSource = VisionSource.FromFile(imageFile);

using var analyzer = new ImageAnalyzer(serviceOptions, imageSource, analysisOptions);

var result = analyzer.Analyze();

if (result.Reason == ImageAnalysisResultReason.Analyzed)

{

    // Get people in the image

    if (result.People != null)

    {

        Console.WriteLine($" People:");

    // Prepare image for drawing

    System.Drawing.Image image = System.Drawing.Image.FromFile(imageFile);

    Graphics graphics = Graphics.FromImage(image);

    Pen pen = new Pen(Color.Cyan, 3);

    Font font = new Font("Arial", 16);

    SolidBrush brush = new SolidBrush(Color.WhiteSmoke);

    foreach (var person in result.People)

    {

        // Draw object bounding box if confidence &gt; 50%

        if (person.Confidence > 0.5)

        {

            // Draw object bounding box

            var r = person.BoundingBox;

            Rectangle rect = new Rectangle(r.X, r.Y, r.Width, r.Height);

            graphics.DrawRectangle(pen, rect);

            // Return the confidence of the person detected

            Console.WriteLine($"   Bounding box {person.BoundingBox}, Confidence {person.Confidence:0.0000}");

        }

    }

    // Save annotated image

    String output\_file = "detected\_people.jpg";

    image.Save(output\_file);

    Console.WriteLine("  Results saved in " + output\_file + "\n");

}

}

else

{

    var errorDetails = ImageAnalysisErrorDetails.FromResult(result);

    Console.WriteLine(" Analysis failed.");

    Console.WriteLine($"   Error reason : {errorDetails.Reason}");

    Console.WriteLine($"   Error code : {errorDetails.ErrorCode}");

    Console.WriteLine($"   Error message: {errorDetails.Message}\n");

}

        }

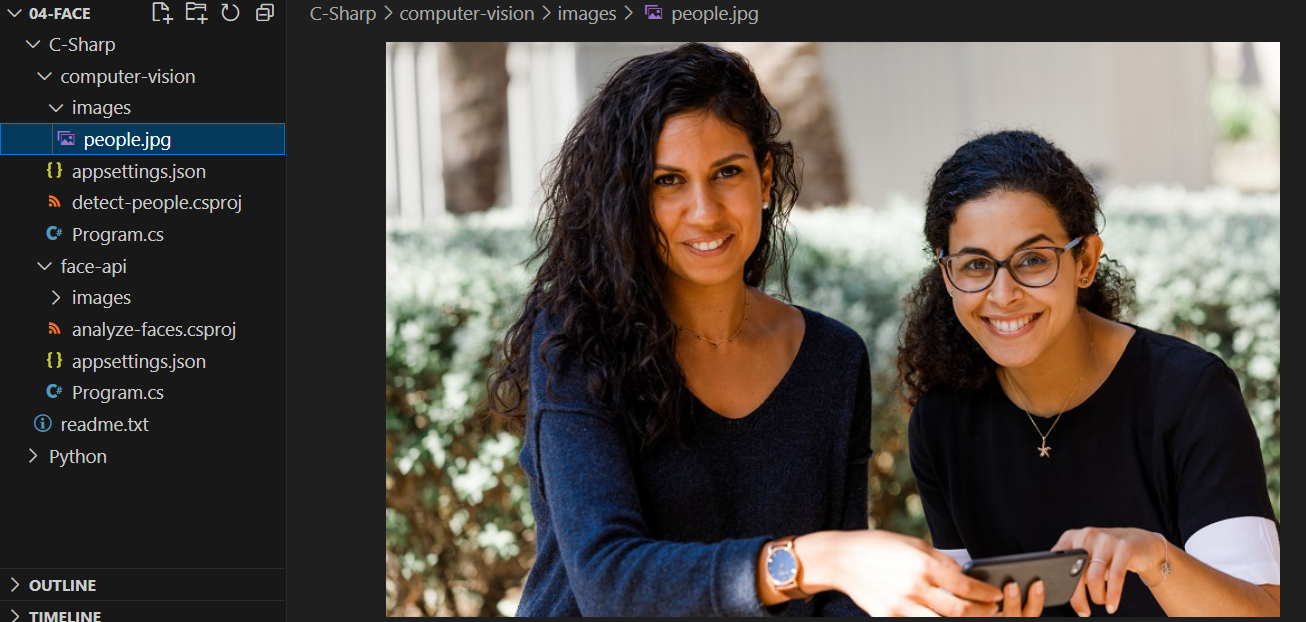
    }

}

# View the image you will analyze

In this exercise, you will use the Azure AI Vision service to analyze an image of people.

1. In Visual Studio Code, expand the **computer-vision** folder and the **images** folder it contains.
2. Select the **people.jpg** image to view it.

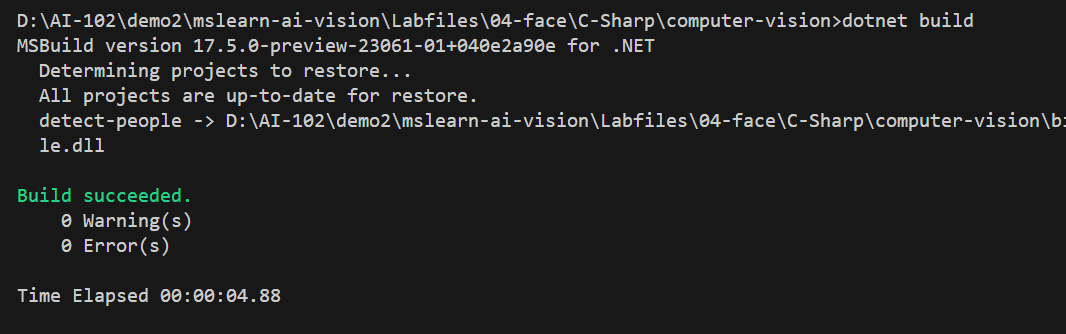


# Detect faces in an image using SDK

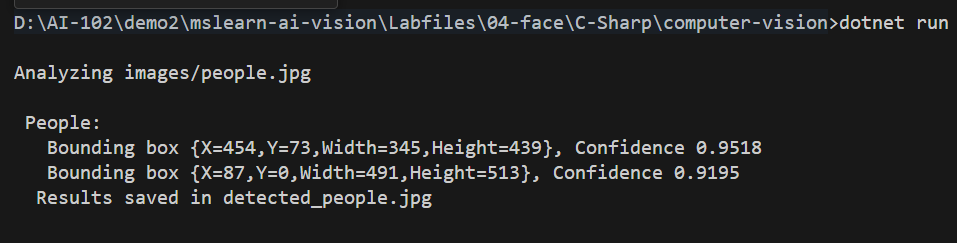
Now you're ready to use the SDK to call the Vision service and detect faces in an image.

* In VS Code terminal type

**Dotnet build**



**Dotnet Run**

****

Check Results saved in detected\_people.jpg

