

**Lab Manual- face detection (Lab using SDK)**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

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

**Contributor:**

Contents

[1. Objective 3](#_Toc167444188)

[2. Provision an Azure AI Services resource 3](#_Toc167444189)

[3. Copy Endpoint and Key 7](#_Toc167444190)

[4. Clone the repository in Visual Studio Code 8](#_Toc167444191)

[5. Install SDK Library 10](#_Toc167444192)

[6. Configure Endpoint and Key 11](#_Toc167444193)

[7. Detect faces in an image ( Prepare to use the Face SDK) 12](#_Toc167444194)

[8. View the image you will analyze 17](#_Toc167444195)

[9. Detect faces in an image using SDK 17](#_Toc167444196)

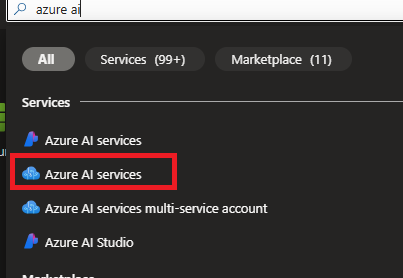
# Objective

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.

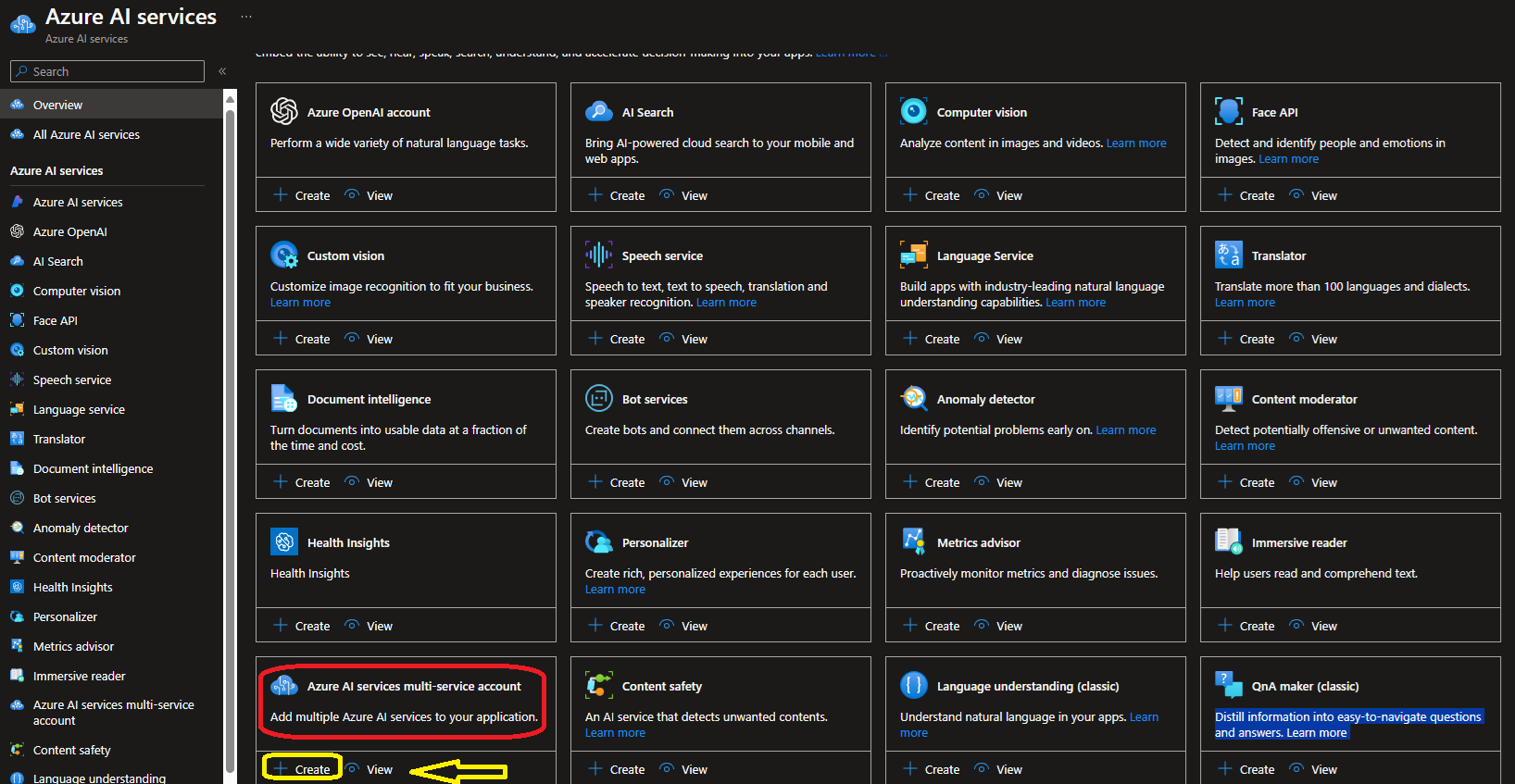
# Provision an Azure AI Services resource

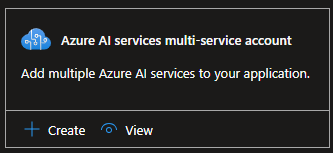
Azure AI Services are cloud-based services that encapsulate artificial intelligence capabilities you can incorporate into your applications. You can provision **individual Azure AI services** resources for **specific APIs** (for example, **Language** or **Vision**), or you can provision a single Azure 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**

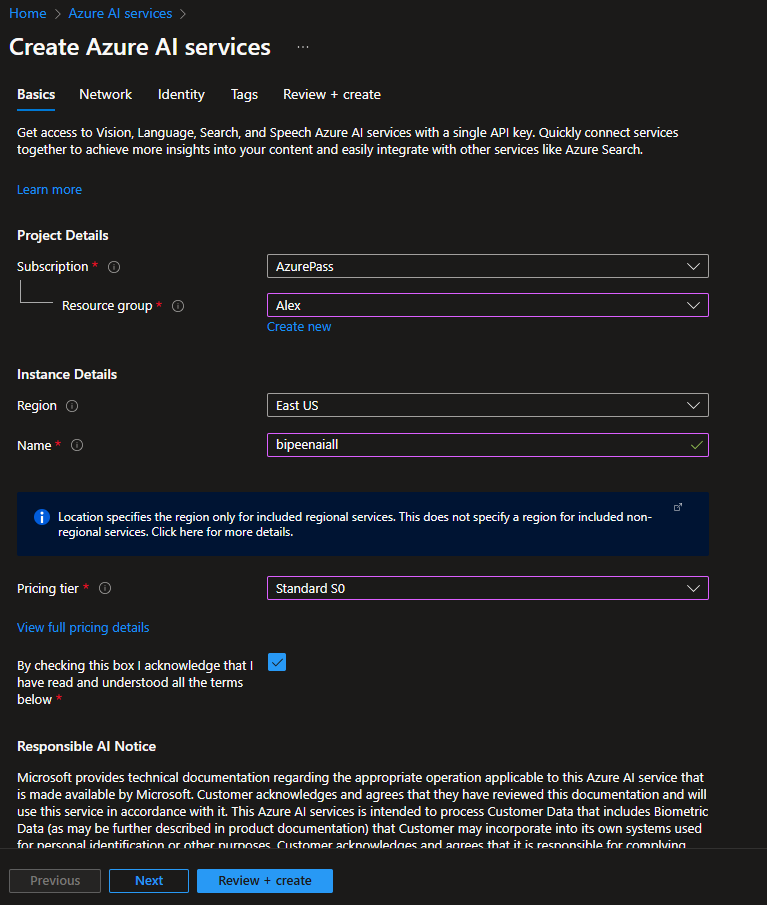


* Select **Azure Ai Service Muti-Service Account** and Click **create**

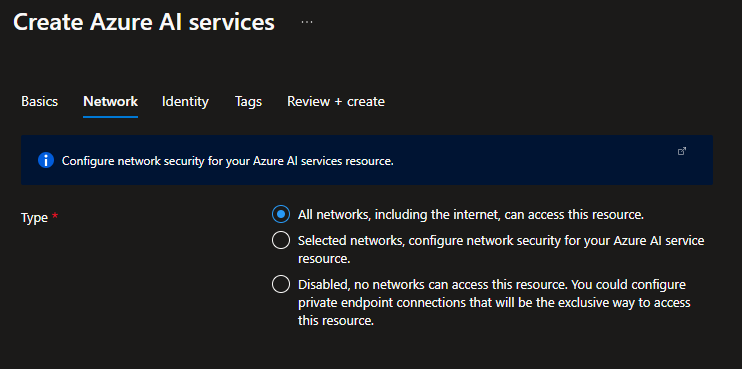




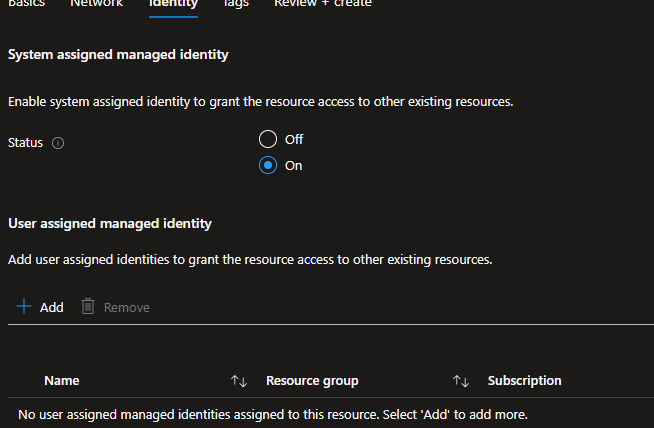
* Select your
  + Resource Group
  + Name of AI Instance
  + Pricing should be **Standard0**
  + **Responsible AI Terms- Checked**



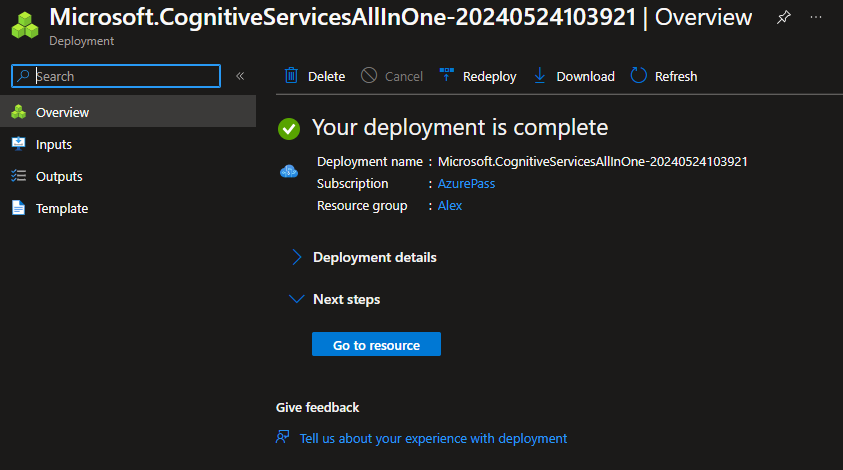
* Click **Next and Leave the default**



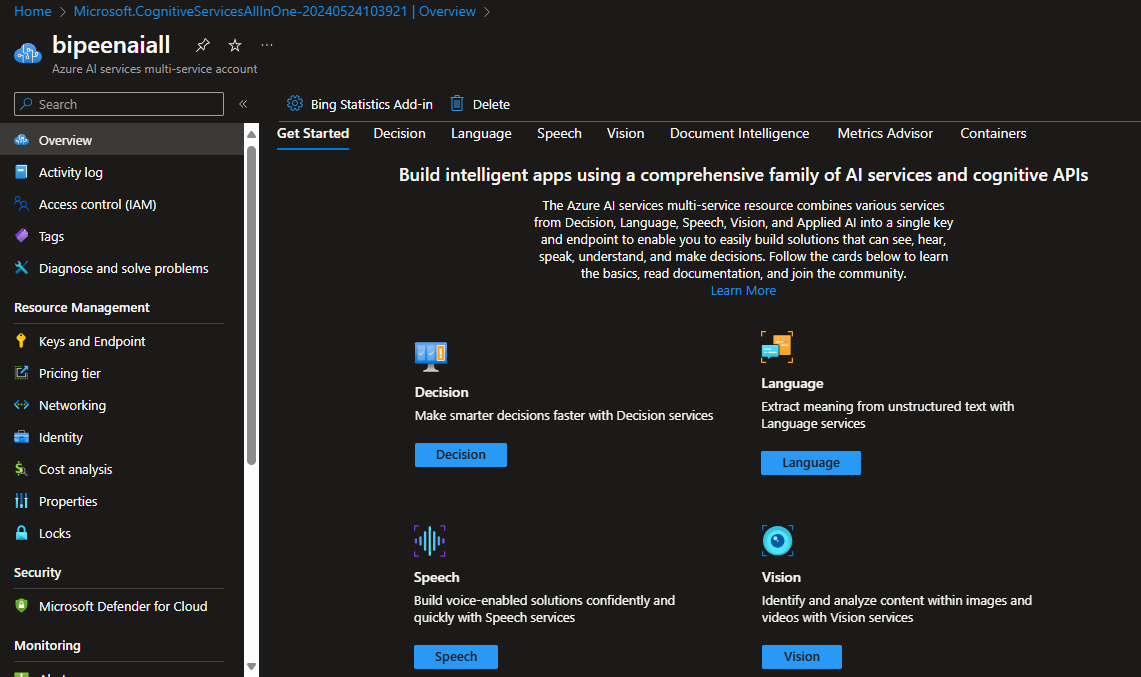
* Click **Next and on the System Managed identity**



* Click **Review and Create**

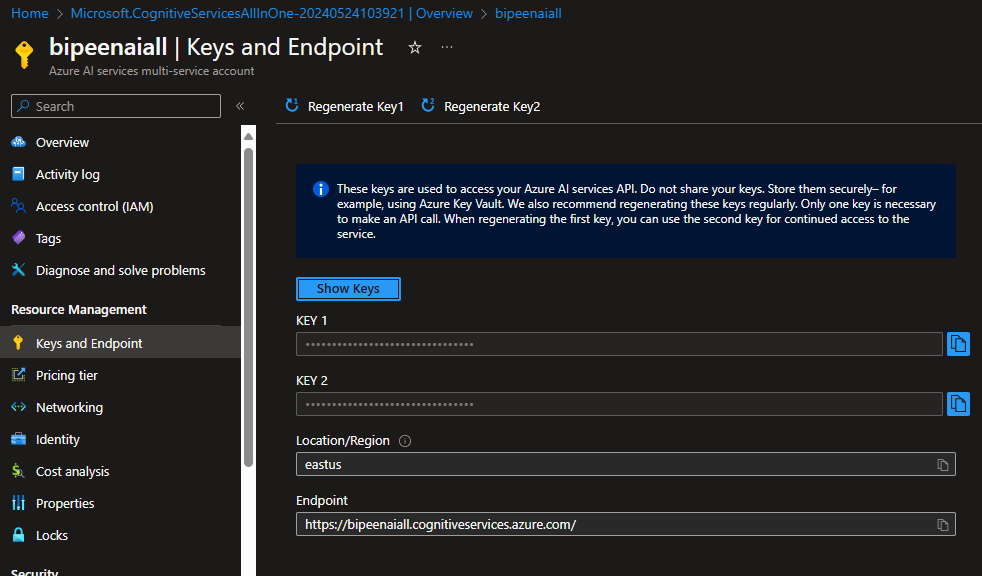


* Wait for deployment to complete, and then view the deployment details.



# Copy Endpoint and Key

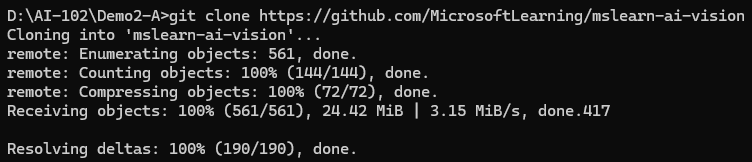
1. Wait for deployment to complete, and then view the deployment details.
2. Go to the resource and view its **Keys and Endpoint** page. This page contains the information that you will need to connect to your resource and use it from applications you develop. Specifically:
   * An HTTP *endpoint* to 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**

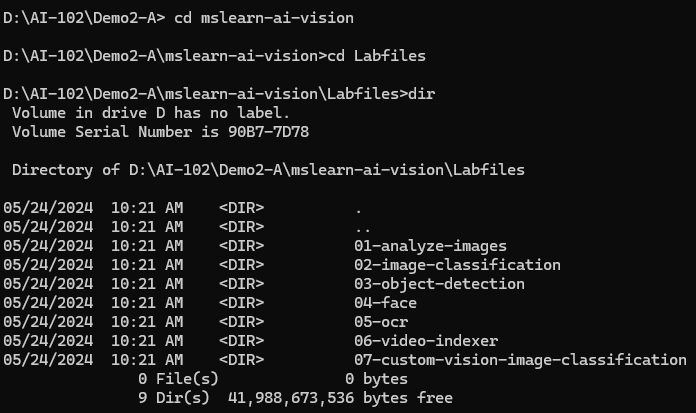


1. Go Inside **mslearn-ai-services/Labfiles** directory

**cd mslearn-ai-vision**

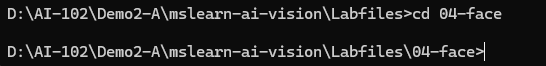
**Cd Labfiles**

**Dir**

****

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

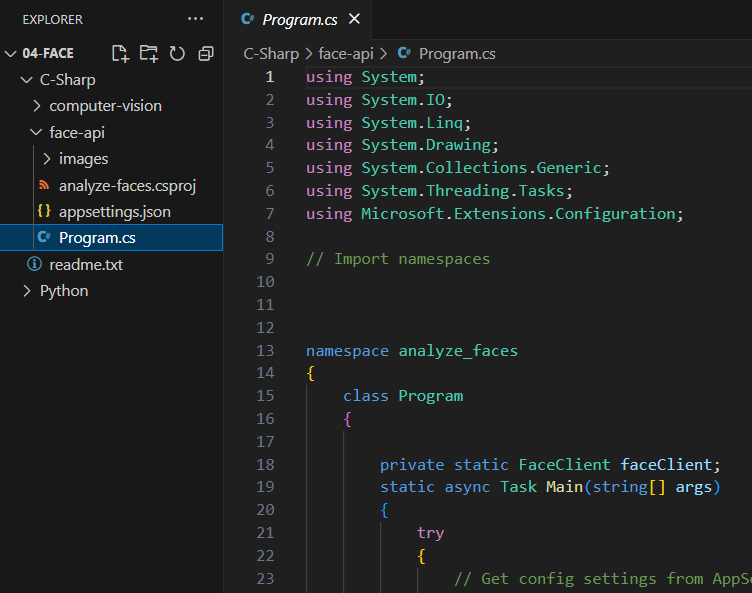
**cd 04-face**

****

1. Open the folder in Visual Studio Code.

**Code .**

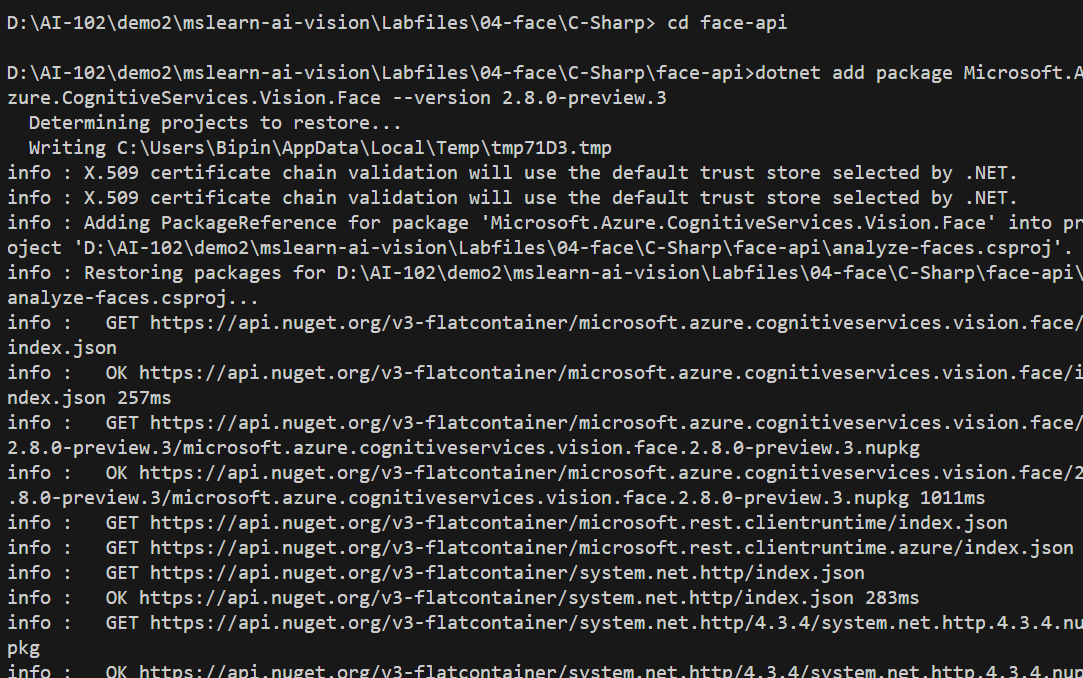
****

****

# Install SDK Library

* In Visual Studio Code, Open Terminal
* Go Inside   **C-Sharp -> Face-api**
* Install the Azure AI Vision SDK package.

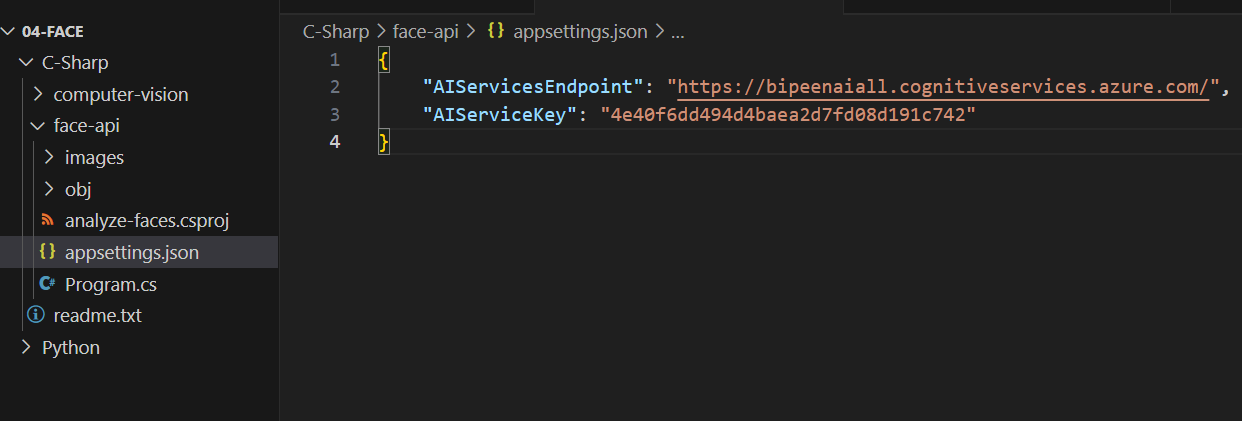
**dotnet add package Microsoft.Azure.CognitiveServices.Vision.Face --version 2.8.0-preview.3**



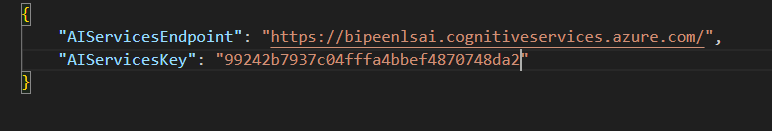
# Configure Endpoint and Key

* In Visual Studio Code, expand the **C-Sharp -> face-api**
* View the contents of the **face-api** 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 ( Prepare to use the Face SDK)

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

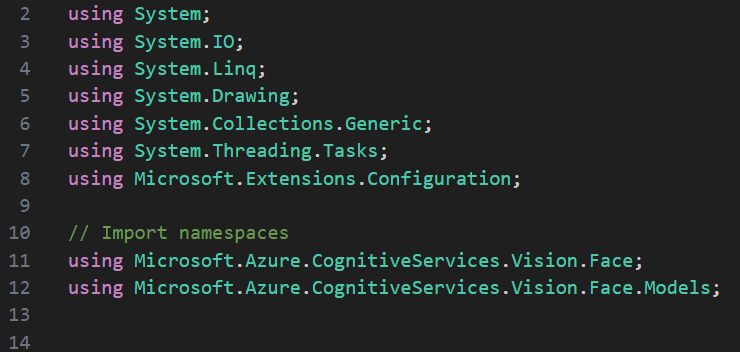
* Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Azure AI Vision SDK:

C#

// Import namespaces

using Microsoft.Azure.CognitiveServices.Vision.Face;

using Microsoft.Azure.CognitiveServices.Vision.Face.Models;



* 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#**

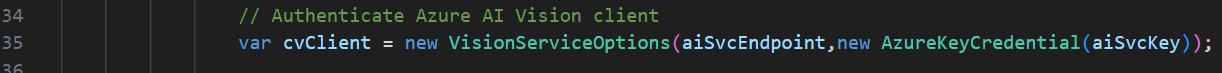
C#

// 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:

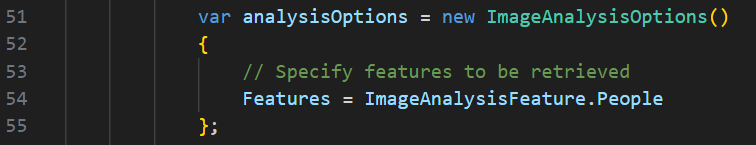
**C#**

C#

// Specify features to be retrieved (PEOPLE)

Features =

ImageAnalysisFeature.People



1. In the **DetectFaces** function, under the code you just added, find the comment **Get faces** and add the following code:

**C#**

C

// Get faces

using (var imageData = File.OpenRead(imageFile))

{

var detected\_faces = await faceClient.Face.DetectWithStreamAsync(imageData, returnFaceAttributes: features, returnFaceId: false);

if (detected\_faces.Count() > 0)

{

Console.WriteLine($"{detected\_faces.Count()} faces detected.");

// Prepare image for drawing

Image image = Image.FromFile(imageFile);

Graphics graphics = Graphics.FromImage(image);

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

Font font = new Font("Arial", 4);

SolidBrush brush = new SolidBrush(Color.White);

int faceCount=0;

// Draw and annotate each face

foreach (var face in detected\_faces)

{

faceCount++;

Console.WriteLine($"\nFace number {faceCount}");

// Get face properties

Console.WriteLine($" - Mouth Occluded: {face.FaceAttributes.Occlusion.MouthOccluded}");

Console.WriteLine($" - Eye Occluded: {face.FaceAttributes.Occlusion.EyeOccluded}");

Console.WriteLine($" - Blur: {face.FaceAttributes.Blur.BlurLevel}");

Console.WriteLine($" - Glasses: {face.FaceAttributes.Glasses}");

// Draw and annotate face

var r = face.FaceRectangle;

Rectangle rect = new Rectangle(r.Left, r.Top, r.Width, r.Height);

graphics.DrawRectangle(pen, rect);

string annotation = $"Face number {faceCount}";

graphics.DrawString(annotation,font,brush,r.Left, r.Top);

}

// Save annotated image

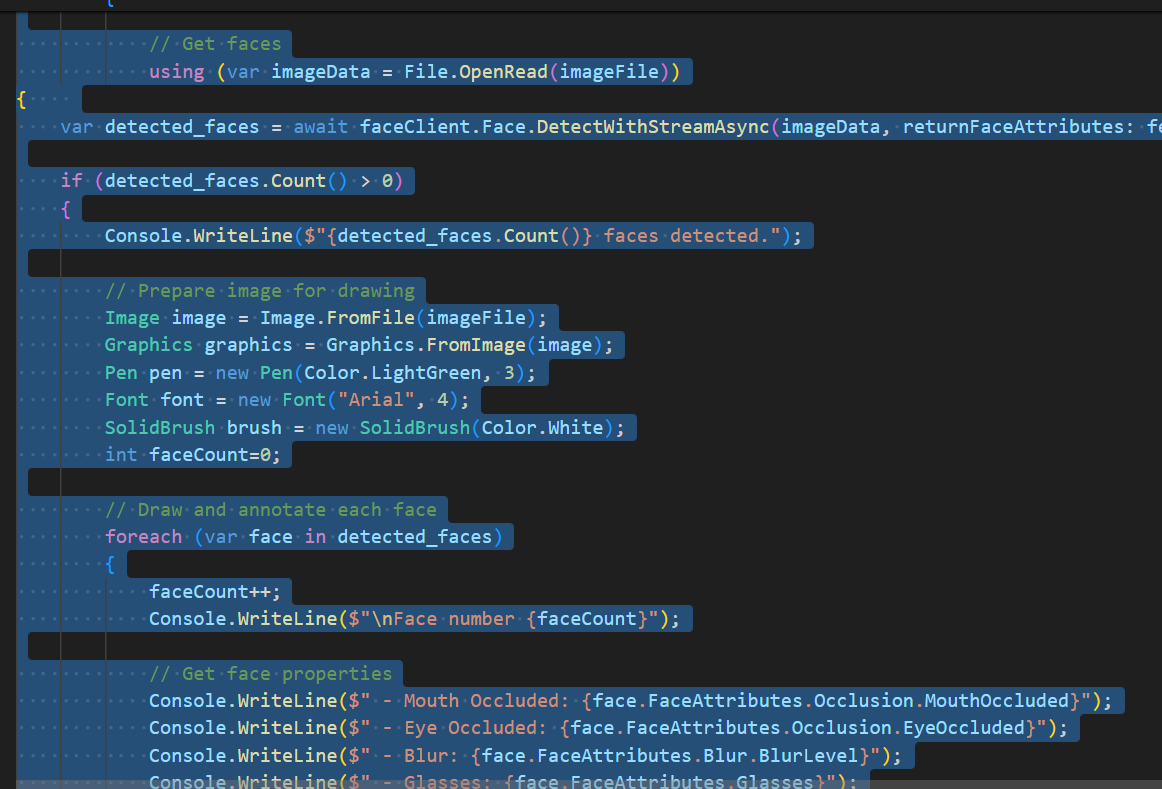
String output\_file = "detected\_faces.jpg";

image.Save(output\_file);

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

}

}



**The Complete Code**:

// dotnet add package Microsoft.Azure.CognitiveServices.Vision.Face --version 2.8.0-preview.3

using System;

using System.IO;

using System.Linq;

using System.Drawing;

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.Extensions.Configuration;

// Import namespaces

using Microsoft.Azure.CognitiveServices.Vision.Face;

using Microsoft.Azure.CognitiveServices.Vision.Face.Models;

namespace analyze\_faces

{

    class Program

    {

        private static FaceClient faceClient;

        static async Task Main(string[] args)

        {

            try

            {

                // Get config settings from AppSettings

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

                IConfigurationRoot configuration = builder.Build();

                string cogSvcEndpoint = configuration["AIServicesEndpoint"];

                string cogSvcKey = configuration["AIServiceKey"];

                // Authenticate Face client

                ApiKeyServiceClientCredentials credentials = new ApiKeyServiceClientCredentials(cogSvcKey);

                faceClient = new FaceClient(credentials)

                {

                    Endpoint = cogSvcEndpoint

                };

                // Menu for face functions

                Console.WriteLine("1: Detect faces\nAny other key to quit");

                Console.WriteLine("Enter a number:");

                string command = Console.ReadLine();

                switch (command)

                {

                    case "1":

                        await DetectFaces("images/people.jpg");

                        break;

                    default:

                        break;

                }

            }

            catch (Exception ex)

            {

                Console.WriteLine(ex.Message);

            }

        }

        static async Task DetectFaces(string imageFile)

        {

            Console.WriteLine($"Detecting faces in {imageFile}");

            // Specify facial features to be retrieved

            IList<FaceAttributeType> features = new FaceAttributeType[]

            {

                FaceAttributeType.Occlusion,

                FaceAttributeType.Blur,

                FaceAttributeType.Glasses

            };

            // Get faces

            using (var imageData = File.OpenRead(imageFile))

{

    var detected\_faces = await faceClient.Face.DetectWithStreamAsync(imageData, returnFaceAttributes: features, returnFaceId: false);

    if (detected\_faces.Count() > 0)

    {

        Console.WriteLine($"{detected\_faces.Count()} faces detected.");

        // Prepare image for drawing

        Image image = Image.FromFile(imageFile);

        Graphics graphics = Graphics.FromImage(image);

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

        Font font = new Font("Arial", 4);

        SolidBrush brush = new SolidBrush(Color.White);

        int faceCount=0;

        // Draw and annotate each face

        foreach (var face in detected\_faces)

        {

            faceCount++;

            Console.WriteLine($"\nFace number {faceCount}");

            // Get face properties

            Console.WriteLine($" - Mouth Occluded: {face.FaceAttributes.Occlusion.MouthOccluded}");

            Console.WriteLine($" - Eye Occluded: {face.FaceAttributes.Occlusion.EyeOccluded}");

            Console.WriteLine($" - Blur: {face.FaceAttributes.Blur.BlurLevel}");

            Console.WriteLine($" - Glasses: {face.FaceAttributes.Glasses}");

            // Draw and annotate face

            var r = face.FaceRectangle;

            Rectangle rect = new Rectangle(r.Left, r.Top, r.Width, r.Height);

            graphics.DrawRectangle(pen, rect);

            string annotation = $"Face number {faceCount}";

            graphics.DrawString(annotation,font,brush,r.Left, r.Top);

        }

        // Save annotated image

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

        image.Save(output\_file);

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

    }

}

        }

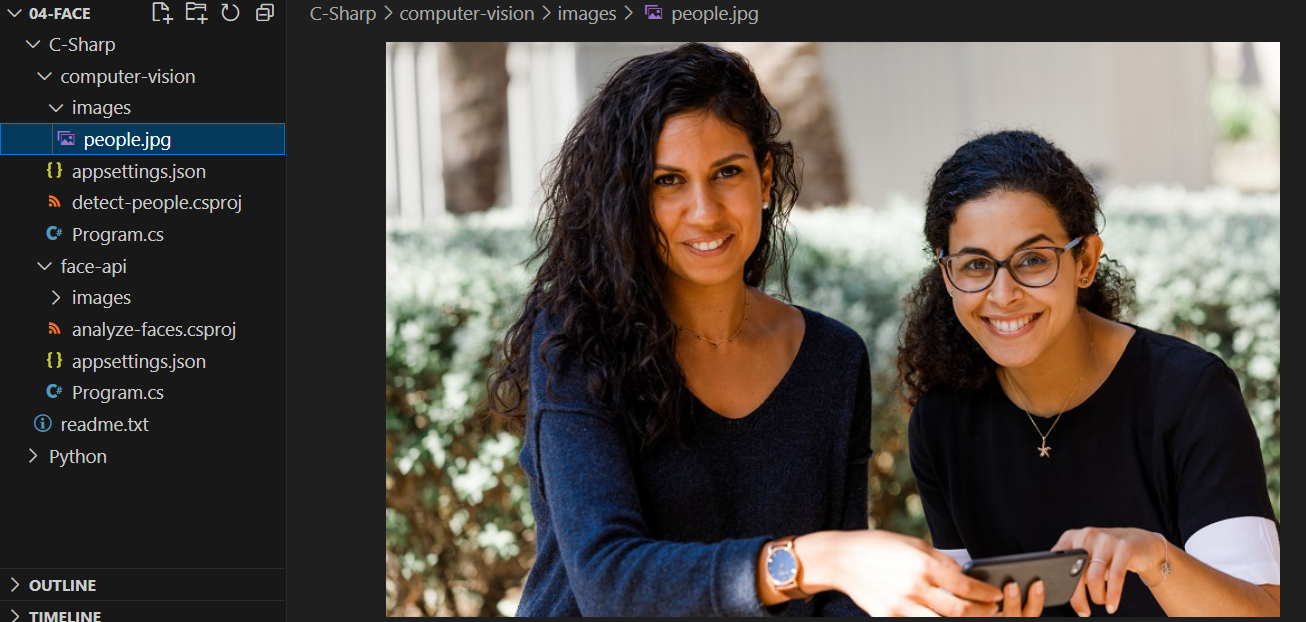
    }

}

# 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.

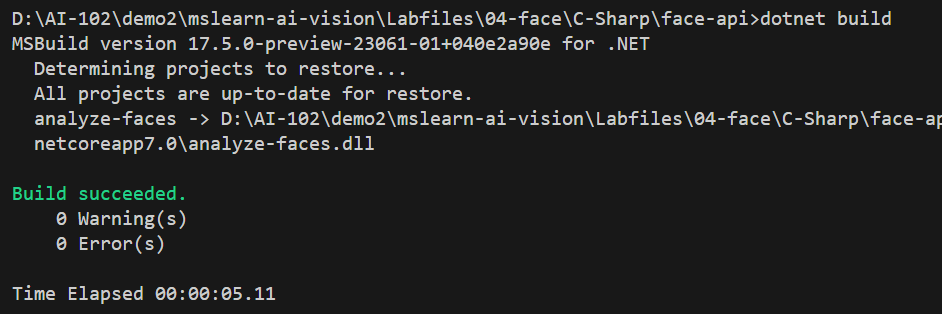


# Detect faces in an image using SDK

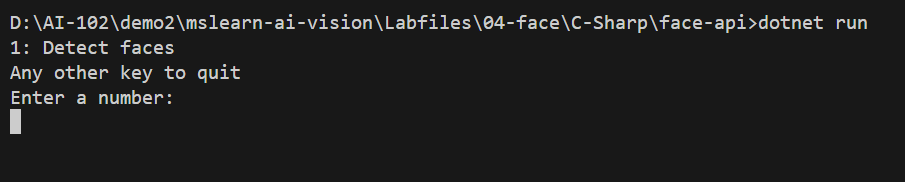
Now you're ready to use the SDK to call the Vision service and detect faces in an image.

* In VS Code terminal type

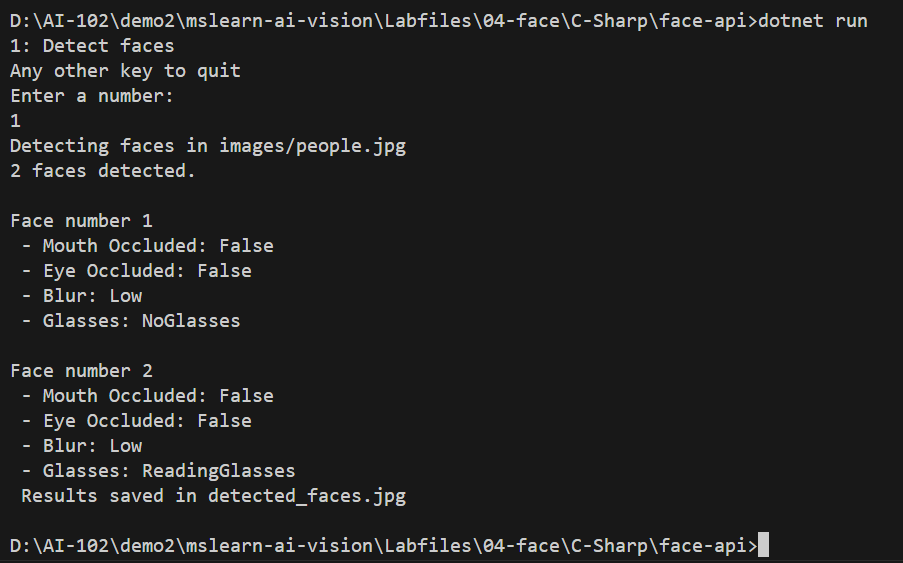
**Dotnet build**



**Dotnet Run**

****

1. When prompted, enter **1** and observe the output, which should include the ID and attributes of each face detected.
2. View the **detected\_faces.jpg** file that is generated in the same folder as your code file to see the annotated faces.

****