Lesson 3- Securing Your Azure OpenAI Credentials with API Keys, Key Vault, and .NET Secrets

Microsoft Agent Framework

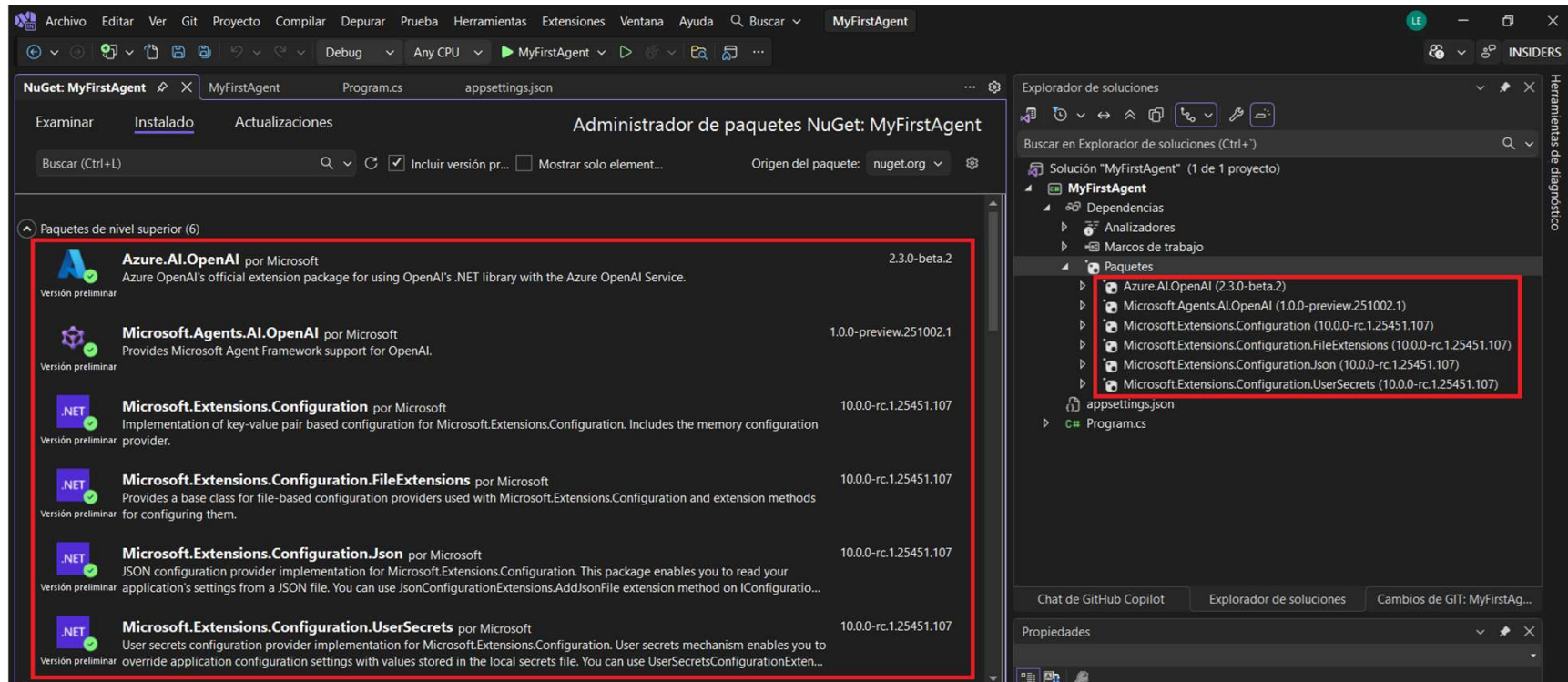https://github.com/microsoft/agent-framework/tree/main/dotnet/samples

Create a C# Console Application

Load Nuget Packages

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with.NET Secrets

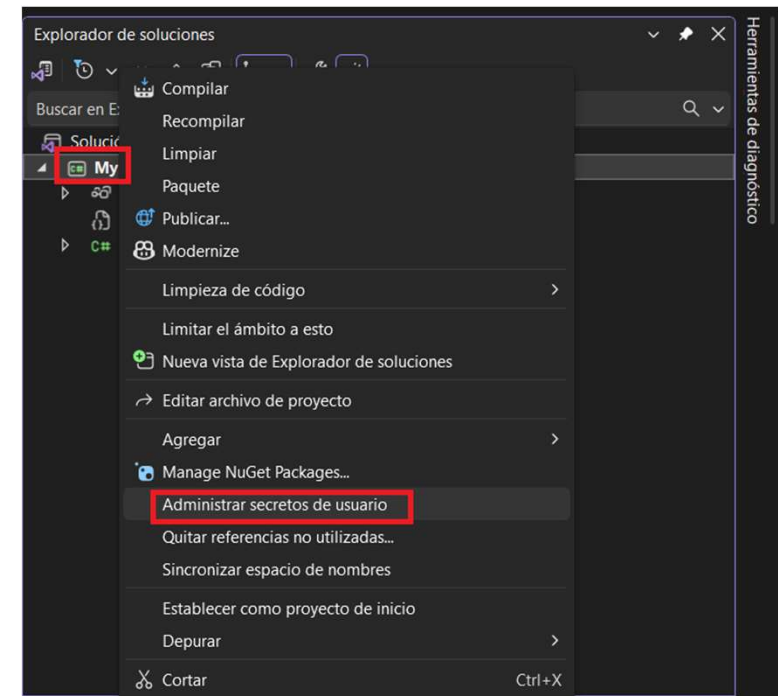
secrets.json

Securing with.NET Secrets

appsettings.json

```json
{
    "AzureOpenAI": {
        "Endpoint": "https://myaiserviceluiscoco.openai.azure.com/",
        "DeploymentName": "gpt-4o-mini"
    }
}
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with.NET Secrets

program.cs

```csharp
using Azure;                    // AzureKeyCredential
using Azure.AI.OpenAI;
using Microsoft.Agents.AI;
using Microsoft.Extensions.Configuration;
using OpenAI;

var configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true) // Endpoint & DeploymentName
    .AddUserSecrets<Program>(optional: false)                              // ApiKey from user secrets
    .Build();

// Read non-secrets from appsettings.json
var endpoint = configuration["AzureOpenAI:Endpoint"]
    ?? throw new InvalidOperationException("Missing AzureOpenAI:Endpoint in appsettings.json.");

var deploymentName = configuration["AzureOpenAI:DeploymentName"] ?? "gpt-4o-mini";

// Read secret from .NET User Secrets
var apiKey = configuration["AzureOpenAI:ApiKey"]
    ?? throw new InvalidOperationException("Missing AzureOpenAI:ApiKey in user secrets.");
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with.NET Secrets

program.cs

```csharp
// Create agent using API key auth (not AAD)
const string JokerName = "Joker";
const string JokerInstructions = "You are good at telling jokes.";

AIAgent agent = new AzureOpenAIClient(
        new Uri(endpoint),
        new AzureKeyCredential(apiKey))
    .GetChatClient(deploymentName)
    .CreateAIAgent(JokerInstructions, JokerName);

// Use the agent
Console.WriteLine(await agent.RunAsync("Tell me a joke about a pirate."));

await foreach (var chunk in agent.RunStreamingAsync("Another pirate joke, please."))
{
    Console.WriteLine(chunk);
}
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Microsoft
Agent Framework

Securing with.NET Secrets

MyFirstAgent.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk">

    <PropertyGroup>
        <OutputType>Exe</OutputType>
        <TargetFramework>net10.0</TargetFramework>
        <ImplicitUsings>enable</ImplicitUsings>
        <Nullable>enable</Nullable>
        <UserSecretsId>dbc6a46c-8812-48bc-8b2f-4d7f471c8552</UserSecretsId>
    </PropertyGroup>

    <!-- Ensure appsettings.json is copied to the build output -->
    <ItemGroup>
        <None Update="appsettings.json">
            <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
        </None>
    </ItemGroup>

    <ItemGroup>
        <PackageReference Include="Azure.AI.OpenAI" Version="2.3.0-beta.2" />
        <PackageReference Include="Microsoft.Agents.AI.OpenAI" Version="1.0.0-preview.251002.1" />
        <PackageReference Include="Microsoft.Extensions.Configuration" Version="10.0.0-rc.1.25451.107" />
        <PackageReference Include="Microsoft.Extensions.Configuration.FileExtensions" Version="10.0.0-rc.1.25451.107" />
        <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="10.0.0-rc.1.25451.107" />
        <PackageReference Include="Microsoft.Extensions.Configuration.UserSecrets" Version="10.0.0-rc.1.25451.107" />
    </ItemGroup>

</Project>
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents
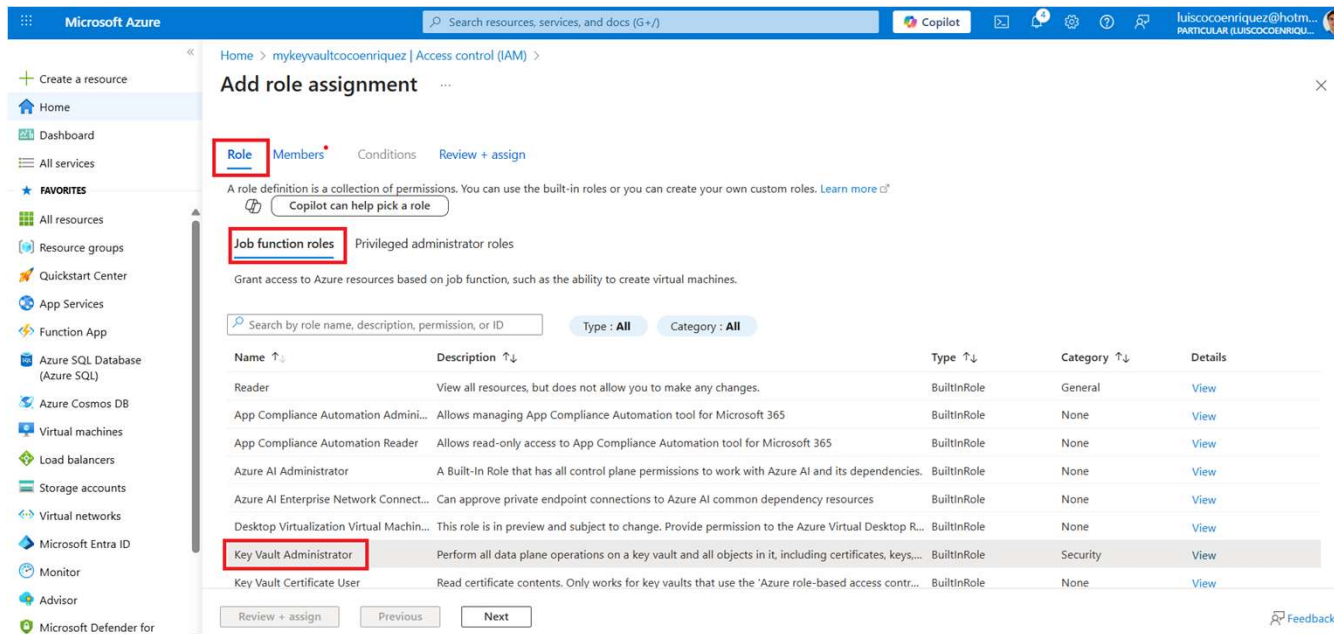
Securing with Azure KeyVault service

We search for the Azure KeyVault Service

Securing with Azure KeyVault service

Microsoft Agent Framework
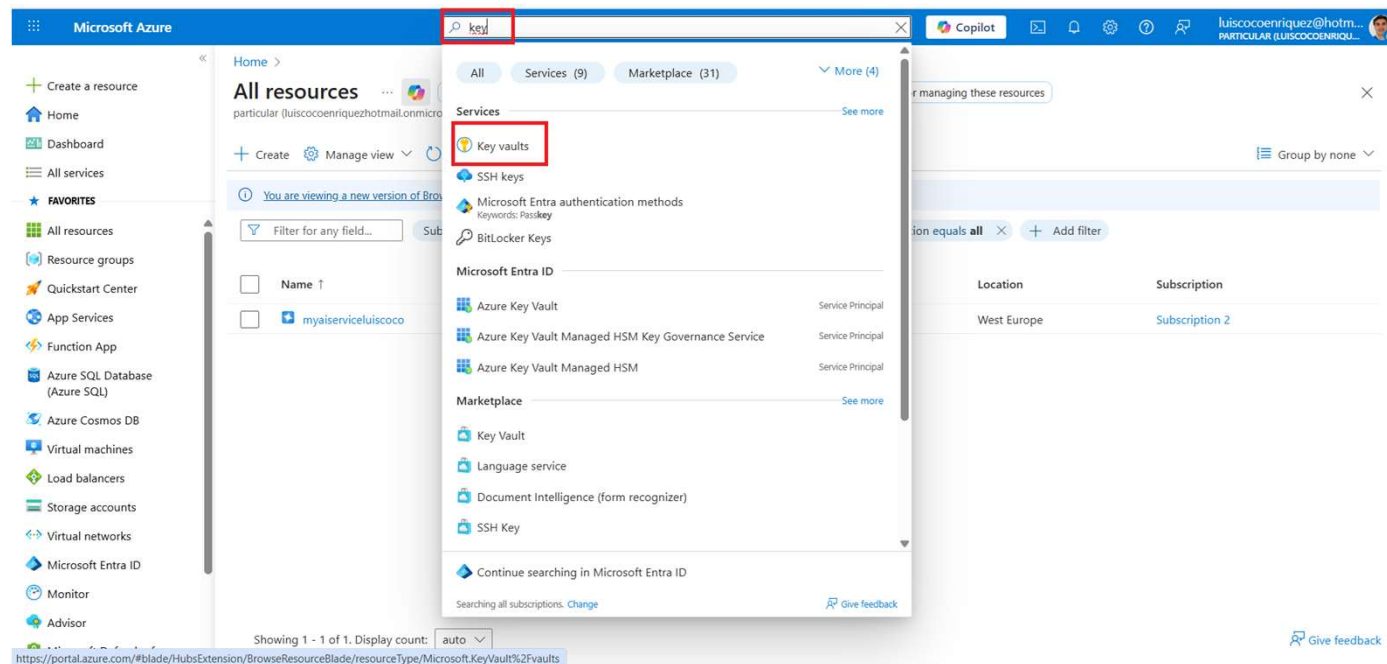
We **create** a new **Azure KeyVault** service

Securing with Azure KeyVault service

We input the KeyVault parameters

Securing with Azure KeyVault service

We copy the **Azure KeyVault URI** to input in the **appsettings.json** file in your C# console application

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

We grant permission to the KeyVault for creating a new Secret

Securing with Azure KeyVault service

Microsoft Agent Framework

We select the KeyVault Administrator

Securing with Azure KeyVault service

We select the KeyVault Administrator

Securing with Azure KeyVault service

We review and assign the role to the Azure KeyVault service

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

We now can create the **Secret** to store the Azure OpenAI API Key inside the Azure KeyVault

Securing with Azure KeyVault service
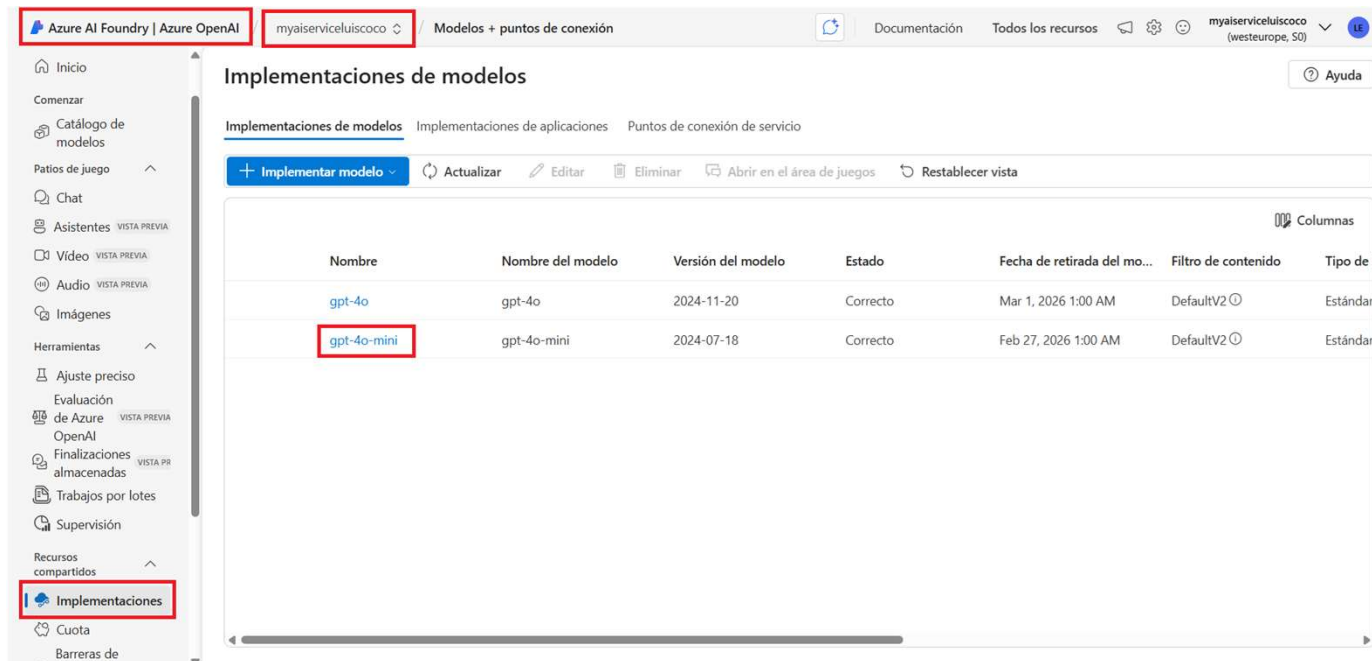
We search for the **Azure OpenAI APIKey** in **Azure Foundry**

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

We copy the Azure OpenAI **APIKey** Value

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service
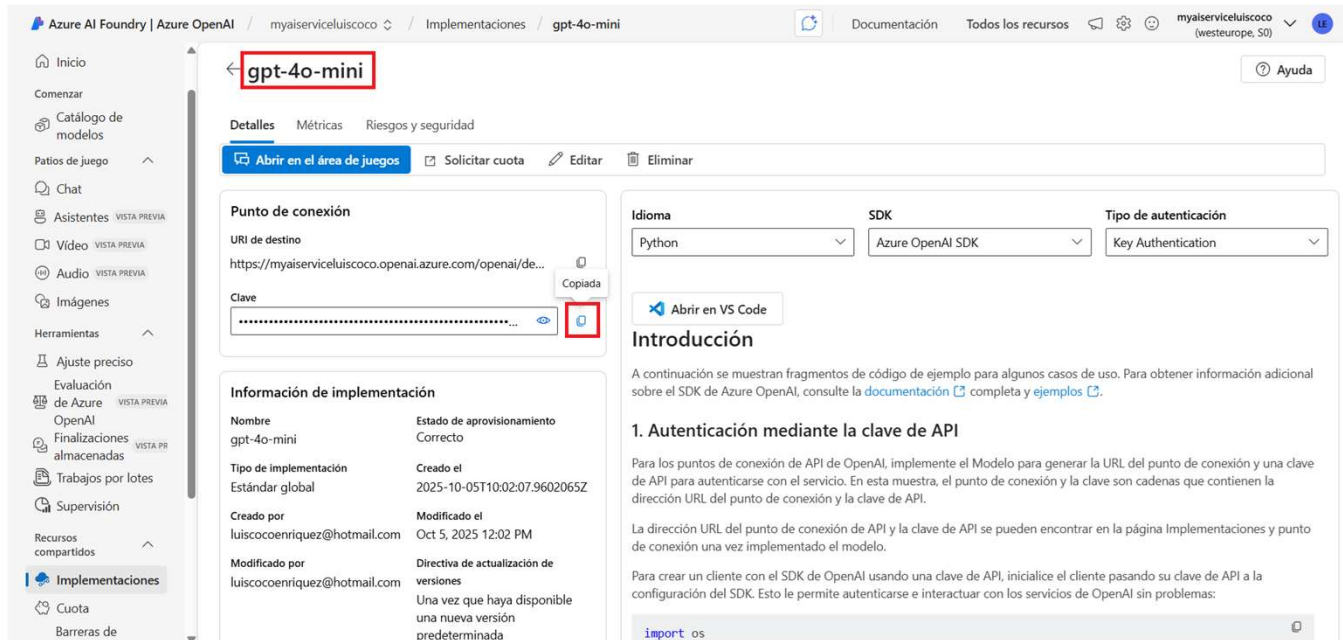
We paste the **Azure OpenAI APIKey** value into the **Secret value**

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

Microsoft Agent Framework

We **Register** a new **Application** in **Microsoft Entra ID**
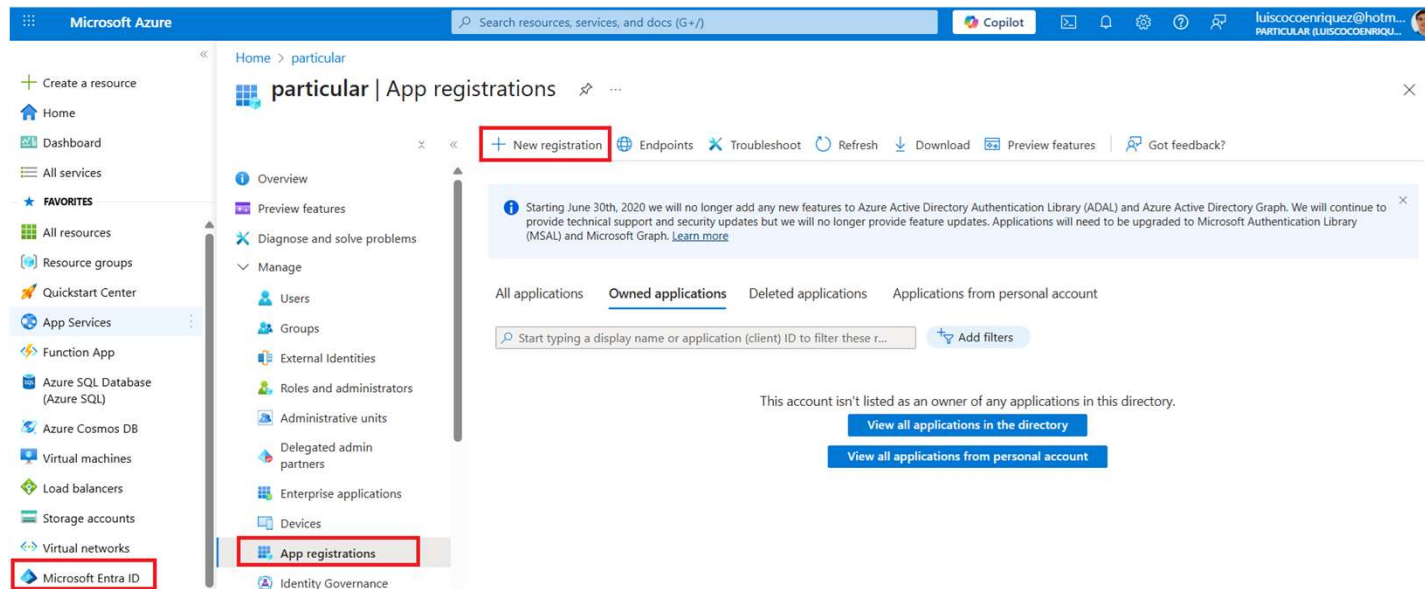
Securing with Azure KeyVault service

We input the new application information

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

We copy the ClientID and the Tenant values to create environmental variables

Securing with Azure KeyVault service

Microsoft Agent Framework

We create a new **Secret** and we also store in an environmental variable

Securing with Azure KeyVault service

We create a new **Secret** and we also store in an environmental variable
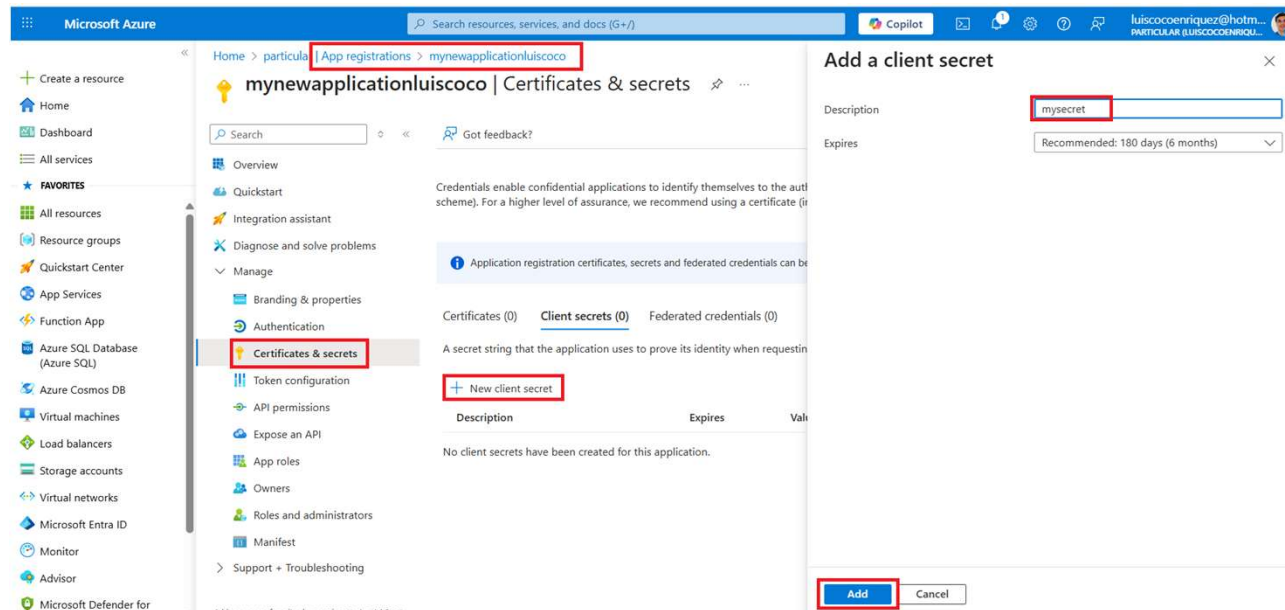
Securing with Azure KeyVault service

We copy the **Secret Value** and we store it in an environmental variable

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

Microsoft Agent Framework

We store the **Secret** in an environmental variable

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents
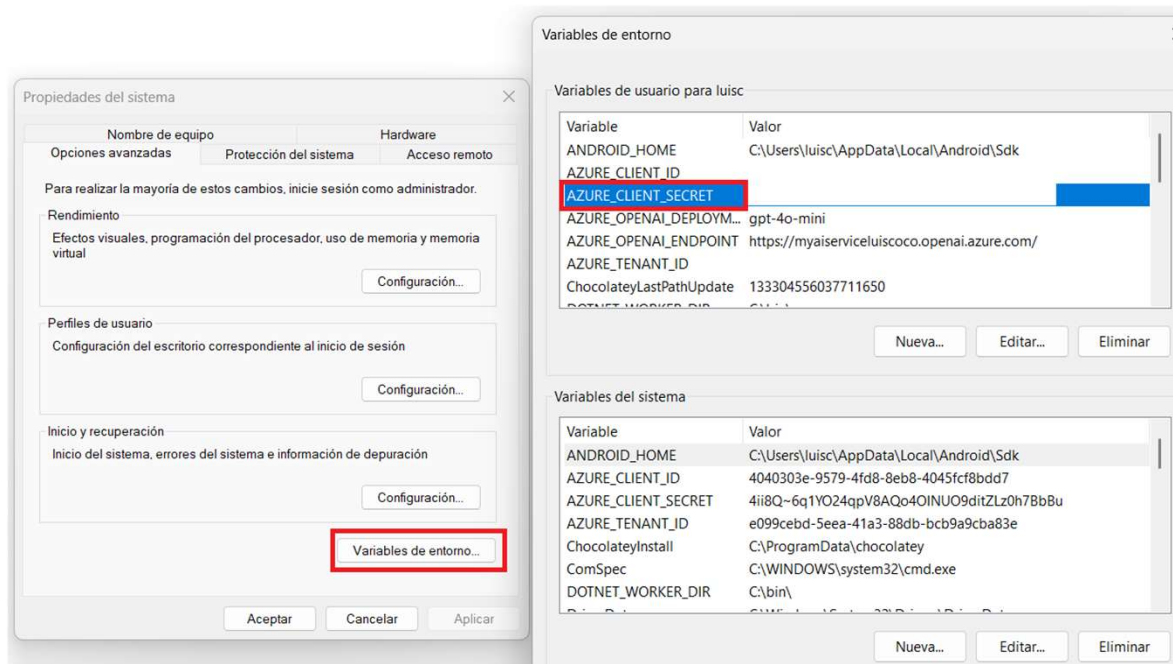
Securing with Azure KeyVault service

In the Azure KeyVault, we assign a new role "Key Vault Secrets User" to the Registered application

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

Assign access to: **User, group, or service principal**.
Select members: search your app by **Application Name**

Securing with Azure KeyVault service

We input the Azure **OpenAI URI** and the **Deployment name** in the **appsettings.json**

Securing with Azure KeyVault service

We input the Azure **KeyVault URI** in the **appsettings.json** file

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

We input the Azure **KeyVault Secret Name** in the **appsettings.json** file

```
"AzureOpenAI": {
    "Endpoint": "https://myaiserviceluiscoco.openai.azure.com/",
    "DeploymentName": "gpt-4o-mini",
    "KeyVault": {
        "VaultUri": "https://mykeyvaultcocoenriquez.vault.azure.net/",
        "SecretName": "my-Azure-OpenAI-API-Key"
    }
}
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

Microsoft Agent Framework

program.cs

```
using Azure;
using Azure.AI.OpenAI;
using Azure.Core;
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;
using Microsoft.Agents.AI;
using Microsoft.Extensions.Configuration;
using OpenAI;
using System;
using System.IO;

1 referencia
static TokenCredential CreateCredential(out string used)
{
    var tenantId = Environment.GetEnvironmentVariable("AZURE_TENANT_ID");
    var clientId = Environment.GetEnvironmentVariable("AZURE_CLIENT_ID");
    var clientSecret = Environment.GetEnvironmentVariable("AZURE_CLIENT_SECRET");

    if (!string.IsNullOrWhiteSpace(tenantId) &&
        !string.IsNullOrWhiteSpace(clientId) &&
        !string.IsNullOrWhiteSpace(clientSecret))
    {
        used = "ClientSecretCredential (env)";
        return new ClientSecretCredential(tenantId, clientId, clientSecret);
    }

    used = "DefaultAzureCredential";
    return new DefaultAzureCredential();
}
```
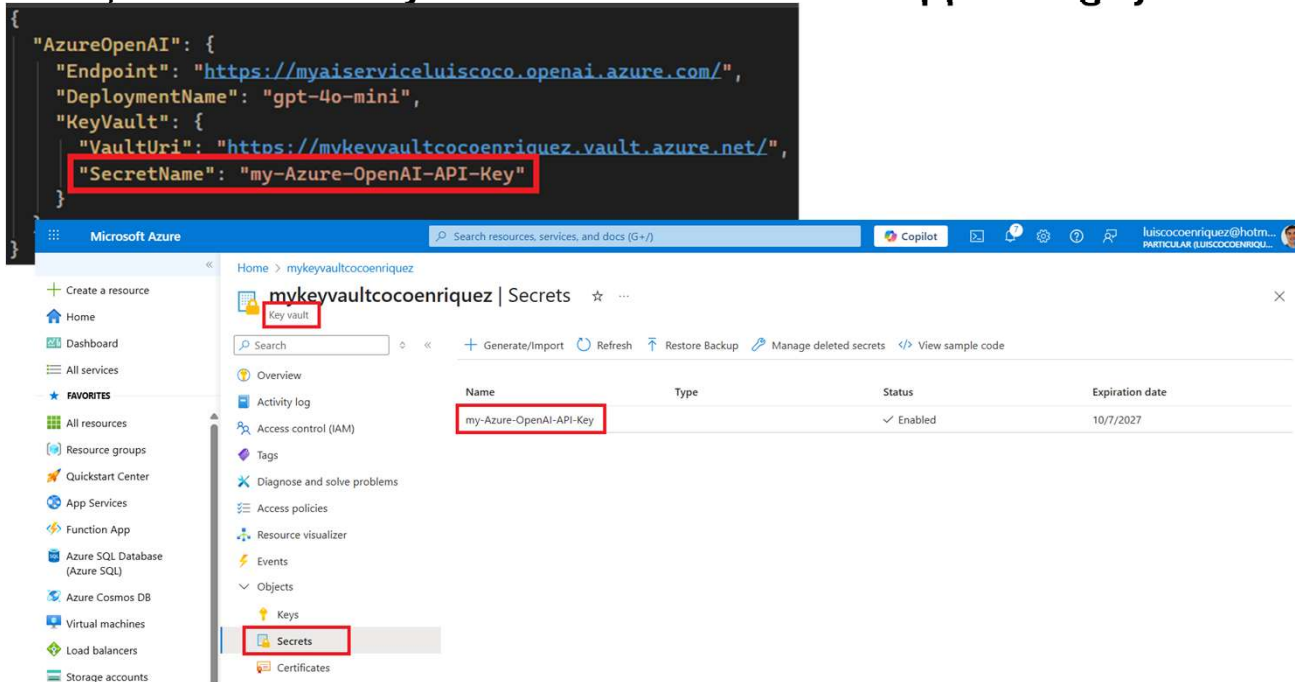
```
try
{
    // 1) Load non-secret config
    var cfg = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
        .Build();

    var endpoint = cfg["AzureOpenAI:Endpoint"]
        ?? throw new InvalidOperationException("Missing AzureOpenAI:Endpoint.");
    var deploymentName = cfg["AzureOpenAI:DeploymentName"] ?? "gpt-4o-mini";

    var vaultUri = cfg["AzureOpenAI:KeyVault:VaultUri"]
        ?? throw new InvalidOperationException("Missing AzureOpenAI:KeyVault:VaultUri.");
    var secretName = cfg["AzureOpenAI:KeyVault:SecretName"] ?? "AzureOpenAI--ApiKey";

    // 2) Credential
    var credential = CreateCredential(out var usedCred);
    Console.WriteLine($"[Auth] Using: {usedCred}");

    // 3) Fetch the secret directly (no listing, no readMetadata)
    var secretClient = new SecretClient(new Uri(vaultUri), credential);
    KeyVaultSecret secret = await secretClient.GetSecretAsync(secretName);
    var apiKey = secret.Value
        ?? throw new InvalidOperationException($"Secret '{secretName}' has no value.");
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents

Securing with Azure KeyVault service

program.cs

```csharp
try
{
    // 4) Build the agent
    const string JokerName = "Joker";
    const string JokerInstructions = "You are good at telling jokes.";

    AIAgent agent = new AzureOpenAIClient(new Uri(endpoint), new AzureKeyCredential(apiKey))
        .GetChatClient(deploymentName)
        .CreateAIAgent(JokerInstructions, JokerName);

    Console.WriteLine(await agent.RunAsync("Tell me a joke about a pirate."));
    await foreach (var chunk in agent.RunStreamingAsync("Another pirate joke, please."))
    {
        Console.WriteLine(chunk);
    }
}
catch (AuthenticationFailedException ex)
{
    Console.Error.WriteLine("Authentication failed: " + ex.Message);
    Console.Error.WriteLine("If using ClientSecretCredential, ensure the client secret is valid and not expired.");
    throw;
}
catch (RequestFailedException ex)
{
    Console.Error.WriteLine($"Key Vault error ({ex.Status}): {ex.Message}");
    Console.Error.WriteLine("Ensure this identity has at least Secret GET permission (RBAC role or access policy).");
    throw;
}
```

https://github.com/microsoft/agent-framework/tree/main/dotnet/samples/GettingStarted/Agents