# Azure IoT 学院专题二: 动手实验 2.2

## 动手实验 2.2: 视频

这个动手实验室建立在之前的实验室的基础上，利用以下内容来获得类似的结果：

- Azure 数字孪生 SDK
- 视觉工作室代码

使用 Azure 数字孪生的开发人员通常会编写客户端应用程序，以便与他们的 Azure 数字孪生服务实例进行交互。 这个以开发人员为中心的动手实验介绍了使用 .NET (C#) 的 Azure 数字孪生 SDK 与 Azure 数字孪生进行交互的编程。 我们将引导您从头开始逐步编写 C# 控制台应用程序。

在本实验室中，您将...

- 设置本地 Azure 凭据。
- 创建一个 C# 项目。
- 将代码添加到您的 C# 项目中：
    - 创建模型
    - 创建数字孪生
    - 建立关系
    - 列出关系
    - 查询数字孪生

**Course Content**



## 1.1. 课程准备

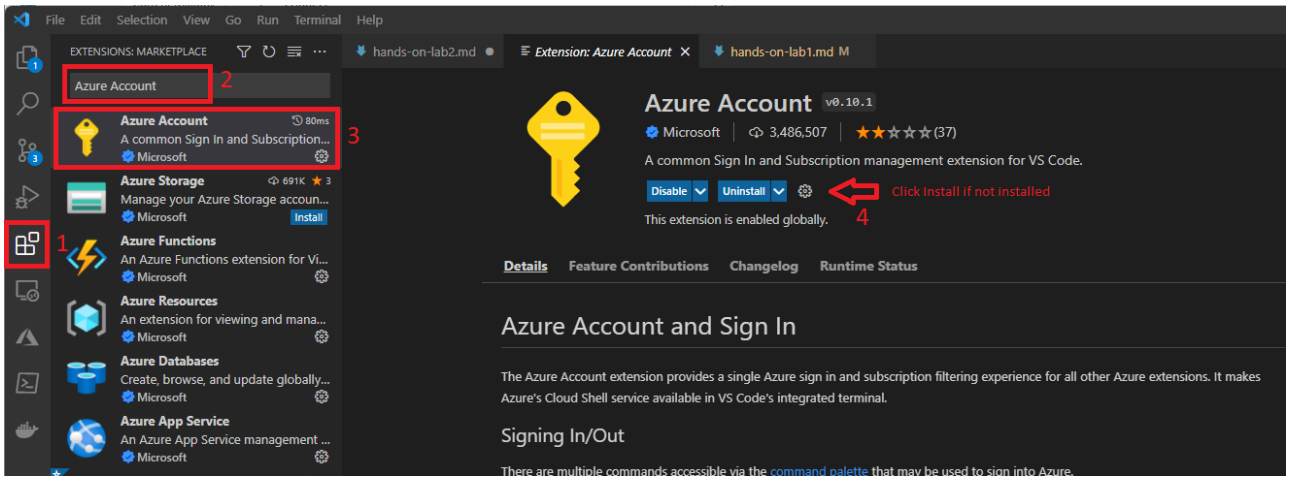
此 Azure 数字孪生实验室使用命令行进行设置和项目工作。 因此，您可以使用任何代码编辑器来完成练习。

你需要准备：

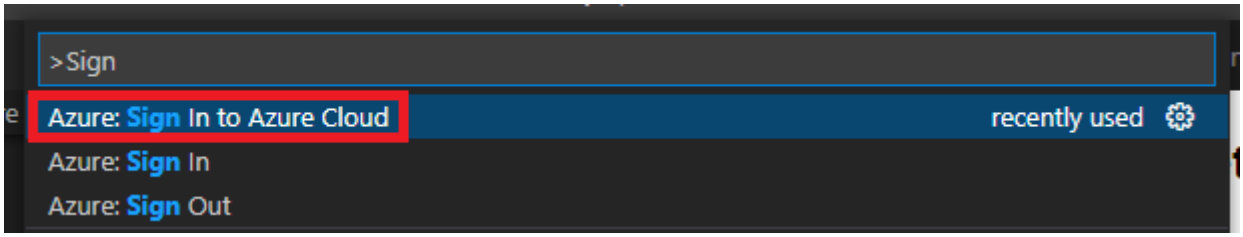
- 任何代码编辑器（我们将使用 Visual Studio Code 进行演示）
- .NET Core 3.1 在您的开发机器上。 您可以从 下载 .NET Core 3.1 下载适用于多个平台的 .NET Core SDK.

### 1.1.1. 设置本地 Azure 凭据

1. 打开 Visual Studio Code
2. Azure Account 插件




3. 点击 "View"->"Command Palette...", 选择 "Sign In" 找到正确的命令. 选择 "Azure: Sign In to Azure Cloud", 之后输入账号登入.

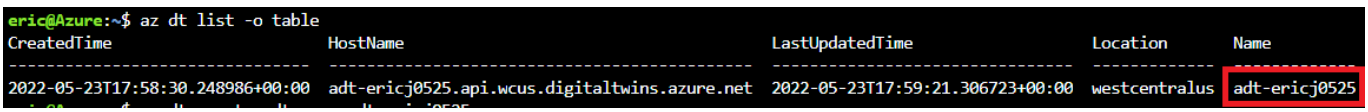



### 1.1.2. 创建或重置 Azure 数字孪生资源

有关如何创建 Azure 数字孪生资源的说明，请参阅 动手实验 2.2的第 1.2.3 节。

如果资源已经存在，打开 cloud shell ，然后找到资源的名字

```
az dt list -o table
```




使用资源的名称，运行以下命令，将名称替换为上表中的名称

```
az dt reset --dt-name [name] --yes
```

```
eric@Azure:~$ az dt reset --dt-name adt-ericj0525 --yes
This command is in preview and under development. Reference and support levels: https://aka.ms/CLI_refstatus
Found 4 twin(s).
Found 1 relationship(s) associated with twin Room1.
Found 1 relationship(s) associated with twin Floor0.
Found 0 relationship(s) associated with twin Floor1.
Found 0 relationship(s) associated with twin Room0.
eric@Azure:~$
```

## 1.2. 项目代码

### 1.2.1. 创建项目并添加依赖

在 VS Code 中打开一个新终端，并创建一个空的项目目录，以便在此实验期间存储您的工作。 将目录命名为您想要的任何名称（例如，DigitalTwinsCodeLab）。

导航到新目录。

进入项目目录后，创建一个空的 .NET 控制台应用程序项目。 在命令窗口中，您可以运行以下命令来为控制台创建一个 C# 项目：

```
dotnet new console
```

此命令将在您的目录中创建多个文件，包括一个名为 ***Program.cs*** 的文件，您将在其中编写大部分代码。

接下来，将两个依赖项添加到使用 Azure 数字孪生所需的项目中。 第一个是适用于 .NET 的 Azure 数字孪生 SDK 包，第二个提供了帮助针对 Azure 进行身份验证的工具。

```
dotnet add package Azure.DigitalTwins.Core
dotnet add package Azure.Identity
```

### 1.2.2. 开始使用项目代码

在本部分中，你将开始为新应用项目编写代码以使用 Azure 数字孪生。 涵盖的行动包括：

- 针对服务进行身份验证
- 上传模型
- 捕捉错误
- 创建数字孪生
- 建立关系
- 查询数字孪生

实验结束时还有一个部分显示了完整的代码。 您可以使用本节作为参考，随时检查您的程序。

首先，在任何代码编辑器中打开文件 ***Program.cs***。 您将看到一个看起来像这样的最小代码模板：

```
1   // See https://aka.ms/new-console-template for more information
2   Console.WriteLine("Hello, World!");
```

首先，将现有代码替换为以下代码以引入必要的依赖项

```
using Azure.DigitalTwins.Core;
using Azure.Identity;
```

接下来，将代码添加到此文件以启用某些功能。

### 1.2.3. 针对服务进行身份验证

您的应用需要做的第一件事是针对 Azure 数字孪生服务进行身份验证。 然后，您可以创建一个服务客户端类来访问 SDK 功能。

若要进行身份验证，你需要 Azure 数字孪生实例的主机名。

在 ***Program.cs*** 中，将以下代码粘贴到“Hello, World!”下方 Main 方法中的打印输出行。 将 adtInstanceUrl 的值设置为你的 Azure 数字孪生实例主机名。

```
string adtInstanceUrl = "https://<your-Azure-Digital-Twins-host-name>";

var credential = new DefaultAzureCredential();
var client = new DigitalTwinsClient(new Uri(adtInstanceUrl), credential);
Console.WriteLine($"Service client created – ready to go");
```

保存文件

In your command window, run the code with this command:

```
dotnet run
```

该命令将在第一次运行时恢复依赖关系，然后执行程序。

- 如果没有发生错误，程序将打印：“Service client created - ready to go”。
- 由于这个项目还没有任何错误处理，如果有任何问题，你会看到代码抛出的异常。 请验证它是否已正确复制，并且您的 Azure 数字孪生主机名已复制到代码中的正确位置。

## 1.3. 创建模型

Azure 数字孪生没有内在的领域词汇。 你可以在 Azure 数字孪生中表示的环境中的元素类型由你使用模型定义。 模型类似于面向对象编程语言中的类； 他们为数字孪生提供用户定义的模板，以供以后遵循和实例化。 它们是用一种称为数字孪生定义语言 (DTDL) 的类 JSON 语言编写的。

创建 Azure 数字孪生解决方案的第一步是在 DTDL 文件中定义至少一个模型。

在您创建项目的目录中，创建一个名为 ***SampleModel.json*** 的新 .json 文件。 粘贴到以下文件正文中：

```
{
  "@id": "dtmi:example:SampleModel;1",
  "@type": "Interface",
  "displayName": "SampleModel",
  "contents": [
    {
      "@type": "Relationship",
      "name": "contains"
    },
    {
      "@type": "Property",
      "name": "data",
      "schema": "string"
    }
  ],
  "@context": "dtmi:dtdl:context;2"
}
```

### 1.3.1. 上传模型

接下来，向 ***Program.cs*** 添加更多代码，以将您创建的模型上传到您的 Azure 数字孪生实例中。

首先，在文件顶部添加一些 using 语句：

```
using System.IO;
using System.Collections.Generic;
using Azure;
```

接下来是与 Azure 数字孪生服务交互的第一段代码。 此代码加载您从磁盘创建的 DTDL 文件，然后将其上传到您的 Azure 数字孪生服务实例。

将以下代码粘贴到您之前添加的授权代码下。

```
Console.WriteLine();
Console.WriteLine($"Upload a model");
string dtdl = File.ReadAllText("SampleModel.json");
var models = new List<string> { dtdl };
// Upload the model to the service
client.CreateModels(models);
```

在您的命令窗口中，使用以下命令运行程序：

```
dotnet run
```

输出中会打印“上传模型 - Upload a model”，表示已到达此代码，但尚无输出表明上传是否成功。

要添加显示已成功上传到实例的所有模型的打印语句，请在上一节之后添加以下代码：

```
foreach (DigitalTwinsModelData md in client.GetModels())
{
    Console.WriteLine($"Model: {md.Id}");
}
```

在再次运行程序以测试此新代码之前，请回想一下上次运行程序时，您已经上传了模型。 Azure 数字孪生不会让您两次上传相同的模型，因此如果您尝试再次上传相同的模型，程序应该会引发异常。

记住这些信息，在命令窗口中使用以下命令再次运行程序：

```
dotnet run
```

程序应该抛出异常。 当您尝试上传已上传的模型时，服务会通过 REST API 返回“错误请求”错误。 因此，Azure 数字孪生客户端 SDK 将依次为每个服务返回代码而不是成功引发异常。

下一节将讨论此类异常以及如何在代码中处理它们。code.

### 1.3.2. 错误捕获

为了防止程序崩溃，您可以在模型上传代码周围添加异常代码。 在 try/catch 处理程序中包装现有的客户端调用“client.CreateModels(typeList)”，如下所示：

```
try
{
    client.CreateModels(models);
    Console.WriteLine("Models uploaded to the instance:");
}
catch (RequestFailedException e)
{
    Console.WriteLine($"Upload model error: {e.Status}: {e.Message}");
}
```

在命令窗口中使用 dotnet run 再次运行该程序。 您将看到返回有关模型上传问题的更多详细信息，包括说明 ModelIdAlreadyExists 的错误代码。

从现在开始，实验室将对服务方法的所有调用包装在 try/catch 处理程序中。

## 1.4. 创建数字孪生

现在你已将模型上传到 Azure 数字孪生，你可以使用此模型定义来创建数字孪生。 数字孪生是模型的实例，代表您的业务环境中的实体——例如农场中的传感器、建筑物中的房间或汽车中的灯。 本节根据您之前上传的模型创建一些数字孪生。

将以下代码添加到 Main 方法的末尾，以创建并初始化三个基于此模型的数字孪生。

```
var twinData = new BasicDigitalTwin();
twinData.Metadata.ModelId = "dtmi:example:SampleModel;1";
twinData.Contents.Add("data", $"Hello World!");

string prefix = "sampleTwin-";
for (int i = 0; i < 3; i++)
{
    try
    {
        twinData.Id = $"{prefix}{i}";
        client.CreateOrReplaceDigitalTwin<BasicDigitalTwin>(twinData.Id,
twinData);
        Console.WriteLine($"Created twin: {twinData.Id}");
    }
    catch(RequestFailedException e)
    {
        Console.WriteLine($"Create twin error: {e.Status}: {e.Message}");
    }
}
```

## 1.5. 建立关系

接下来，您可以在已创建的孪生之间创建关系，将它们连接成孪生图。 双图用于表示您的整个环境。

在 Main 方法下面的 Program 类中添加一个新的静态方法（代码现在有两个方法）：

```
void CreateRelationship(DigitalTwinsClient client, string srcId, string
targetId)
{
    var relationship = new BasicRelationship
    {
        TargetId = targetId,
        Name = "contains"
    };

    try
    {
        string relId = $"{srcId}-contains->{targetId}";
        client.CreateOrReplaceRelationship(srcId, relId, relationship);
        Console.WriteLine("Created relationship successfully");
    }
    catch (RequestFailedException e)
    {
        Console.WriteLine($"Create relationship error: {e.Status}:
{e.Message}");
    }
}
```

接下来，将以下代码添加到 Main 方法的末尾，以调用 CreateRelationship 方法并使用您刚刚编写的代码：

```
// Connect the twins with relationships
CreateRelationship(client, "sampleTwin-0", "sampleTwin-1");
CreateRelationship(client, "sampleTwin-0", "sampleTwin-2");
```

在命令窗口中，使用 `dotnet run` 运行程序。 在输出中，查找表明两个关系已成功创建的打印语句。

如果已存在具有相同 ID 的另一个关系，Azure 数字孪生将不允许您创建关系，因此如果您多次运行该程序，您将看到创建关系时出现异常。 此代码捕获异常并忽略它们。

### 1.5.1. 列出关系

您将添加的下一个代码允许您查看已创建的关系列表。

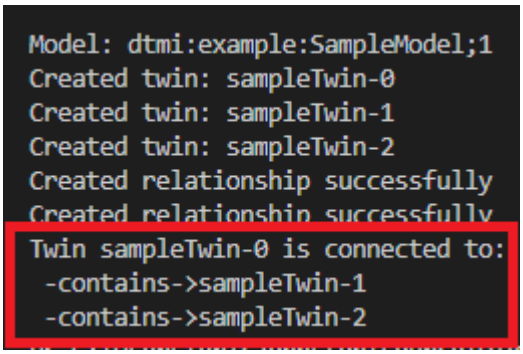
将以下新方法添加到 Program 类：

```
void ListRelationships(DigitalTwinsClient client, string srcId)
{
    try
    {
        Pageable<BasicRelationship> results =
client.GetRelationships<BasicRelationship>(srcId);
        Console.WriteLine($"Twin {srcId} is connected to:");
        foreach (BasicRelationship rel in results)
        {
            Console.WriteLine($" -{rel.Name}->{rel.TargetId}");
        }
    }
    catch (RequestFailedException e)
    {
        Console.WriteLine($"Relationship retrieval error: {e.Status}:
{e.Message}");
    }
}
```

然后，在 Main 方法的末尾添加以下代码来调用 ListRelationships 代码：

```
//List the relationships
ListRelationships(client, "sampleTwin-0");
```

在命令窗口中，使用 `dotnet run` 运行程序。 您应该会在如下所示的输出语句中看到您创建的所有关系的列表：




## 1.6. 查询数字孪生

Azure 数字孪生的一个主要功能是能够 查询

本教程中要添加的最后一段代码针对 Azure 数字孪生实例运行查询。 此示例中使用的查询返回实例中的所有数字孪生。

添加此 `using` 语句以启用 `JsonSerializer` 类来帮助呈现数字孪生信息：

```
using System.Text.Json;
```

Then, add the following code to the end of the `Main` method:

```
// Run a query for all twins
string query = "SELECT * FROM digitaltwins";
Pageable<BasicDigitalTwin> queryResult = client.Query<BasicDigitalTwin>
(query);

foreach (BasicDigitalTwin twin in queryResult)
{
    Console.WriteLine(JsonSerializer.Serialize(twin));
    Console.WriteLine("---------------");
}
```

在命令窗口中，使用 `dotnet run` 运行程序。 您应该在输出中看到此实例中的所有数字孪生。

## 1.7. Recap

此时在实验室中，你拥有一个完整的客户端应用程序，可以对 Azure 数字孪生执行基本操作。 作为参考，***Program.cs*** 中程序的完整代码应类似于以下代码：

```
using System;
using System.IO;
using System.Collections.Generic;
using Azure;
using Azure.DigitalTwins.Core;
using Azure.Identity;
using System.Text.Json;
```

```
string adtInstanceUrl = "https://<your-Azure-Digital-Twins-host-name>";

var credential = new DefaultAzureCredential();
var client = new DigitalTwinsClient(new Uri(adtInstanceUrl), credential);
Console.WriteLine($"Service client created – ready to go");
Console.WriteLine();
Console.WriteLine($"Upload a model");
string dtdl = File.ReadAllText("SampleModel.json");
var models = new List<string> { dtdl };
// Upload the model to the service
try
{
    client.CreateModels(models);
    Console.WriteLine("Models uploaded to the instance:");
}
catch (RequestFailedException e)
{
    Console.WriteLine($"Upload model error: {e.Status}: {e.Message}");
}

foreach (DigitalTwinsModelData md in client.GetModels())
{
    Console.WriteLine($"Model: {md.Id}");
}

var twinData = new BasicDigitalTwin();
twinData.Metadata.ModelId = "dtmi:example:SampleModel;1";
twinData.Contents.Add("data", $"Hello World!");

string prefix = "sampleTwin-";
for (int i = 0; i < 3; i++)
{
    try
    {
        twinData.Id = $"{prefix}{i}";
        client.CreateOrReplaceDigitalTwin<BasicDigitalTwin>(twinData.Id,
twinData);
        Console.WriteLine($"Created twin: {twinData.Id}");
    }
    catch(RequestFailedException e)
    {
        Console.WriteLine($"Create twin error: {e.Status}: {e.Message}");
    }
}

// Connect the twins with relationships
CreateRelationship(client, "sampleTwin-0", "sampleTwin-1");
CreateRelationship(client, "sampleTwin-0", "sampleTwin-2");

//List the relationships
ListRelationships(client, "sampleTwin-0");

// Run a query for all twins
```

```
string query = "SELECT * FROM digitaltwins";
Pageable<BasicDigitalTwin> queryResult = client.Query<BasicDigitalTwin>
(query);

foreach (BasicDigitalTwin twin in queryResult)
{
    Console.WriteLine(JsonSerializer.Serialize(twin));
    Console.WriteLine("---------------");
}

void ListRelationships(DigitalTwinsClient client, string srcId)
{
    try
    {
        Pageable<BasicRelationship> results =
client.GetRelationships<BasicRelationship>(srcId);
        Console.WriteLine($"Twin {srcId} is connected to:");
        foreach (BasicRelationship rel in results)
        {
            Console.WriteLine($" -{rel.Name}->{rel.TargetId}");
        }
    }
    catch (RequestFailedException e)
    {
        Console.WriteLine($"Relationship retrieval error: {e.Status}:
{e.Message}");
    }
}

void CreateRelationship(DigitalTwinsClient client, string srcId, string
targetId)
{
    var relationship = new BasicRelationship
    {
        TargetId = targetId,
        Name = "contains"
    };

    try
    {
        string relId = $"{srcId}-contains->{targetId}";
        client.CreateOrReplaceRelationship(srcId, relId, relationship);
        Console.WriteLine("Created relationship successfully");
    }
    catch (RequestFailedException e)
    {
        Console.WriteLine($"Create relationship error: {e.Status}:
{e.Message}");
    }
}
```

## 1.8. 清空资源

转到下一个动手实验中 动手实验 2.2.