100 XP

Exercise - Identify a queue

5 minutes

Sandbox activated! Time remaining: **1 hr 12 min**

You have used 1 of 10 sandboxes for today. More sandboxes will be available tomorrow.

Let's create a client application to work with a queue. Then we'll add our connection string to the code.

Note

You can create the client application on your local computer if you have .NET Core installed, or directly in the Cloud Shell environment.

Create the application

We'll create a .NET Core application that you can run on Linux, macOS, or Windows. Let's name it **QueueApp**. For simplicity, we'll use a single app that will both send and receive messages through our queue.

1. Use the `dotnet new` command to create a new console app with the name **QueueApp**. You can type commands into the Cloud Shell on the right, or if you are working locally, in a terminal/console window. This command creates a simple app with a single source file: `Program.cs`.

   Azure CLI — Copy

   ```
   dotnet new console -n QueueApp
   ```

2. Switch to the newly created `QueueApp` folder and build the app to verify that all is well.

   Azure CLI — Copy

   ```
   cd QueueApp
   ```

Azure CLI Copy

```
dotnet build
```

Get your connection string

Recall that your connection string is available in the Azure portal **Settings > Access keys** section of your storage account.

You can also retrieve it through the Azure CLI or PowerShell tools. Let's use the Azure CLI command. Remember to replace the `<name>` with the specific name of the storage account you created. You can use `az storage account list` if you need a reminder.

Azure CLI Copy

```
az storage account show-connection-string --name <name> --resource-group learn-
fbe5d791-54b8-403f-a0a5-26258f584781
```

This command will return a JSON block with your connection string. It will include the storage account name and the account key:

JSON Copy

```
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=
<name>;AccountKey=vyw6aKz2PtSAgQ4ljJQgJFgxbCETdXt39ZyYQ5fLqoBJj/gT+43TbrhoVco7Rqj/
AAJVlvFORRfnYqGHiX9QcQ=="
}
```

Add the connection string to the application

Finally, let's add the connection string into our app so it can access the storage account.

⚠ Warning

For simplicity, you will place the connection string in the **Program.cs** file. In a production application, you should store it in a secure location. For server side use, we recommend using Azure Key Vault.

1. Type `code .` in the terminal to open the online code editor. Alternatively, if you are working on your own you can use the IDE of your choice. We recommend Visual Studio

Code, which is an excellent cross-platform IDE.

2. Open the `Program.cs` source file in the project.

3. In the `Program` class, add a `const string` value to hold the connection string. You only need the value (it starts with the text **DefaultEndpointsProtocol**).

4. Save the file. You can click the ellipse "..." in the right corner of the cloud editor, or use the accelerator key (`Ctrl+S` on Windows and Linux, `Cmd+S` on macOS).

Your code should look something like this (the string value will be unique to your account).

C# Copy

```csharp
using System;

namespace QueueApp
{
    class Program
    {
        private const string ConnectionString = "DefaultEndpointsProtocol=https;
...";

        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Now that we have this starter project setup, let's look at how to work with a queue in code. It all starts with *messages*.

Next unit: Programmatically access a queue

Continue >

```
Azure Cloud Shell
Configuring Cloud Shell for sandbox
access...
```