**Exercise: Connect an app to Azure Cache for Redis by using .NET Core**

In this exercise you will learn how to:

* Create a new Redis Cache instance by using Azure CLI commands.
* Create a .NET Core console app to add and retrieve values from the cache by using the **StackExchange.Redis** package.

**Prerequisites**

* An **Azure account** with an active subscription. If you don't already have one, you can sign up for a free trial at <https://azure.com/free>.
* [Visual Studio Code](https://code.visualstudio.com/) on one of the [supported platforms](https://code.visualstudio.com/docs/supporting/requirements#_platforms).
* The [C# extension](https://marketplace.visualstudio.com/items?itemName=ms-dotnettools.csharp) for Visual Studio Code.
* [.NET Core 3.1](https://dotnet.microsoft.com/download/dotnet/3.1) is the target framework for the steps below.

**Create Azure resources**

1. Launch VS Code and open a terminal by selecting **Terminal > New Terminal** from the menu
2. Login to Azure using command in the terminal

**az login**

1. Create myLocation variable

$myLocation="southeastasia"

$myGroup="az204-redis-rg"

1. Create a resource group for Azure resources. Replace <myLocation> with a region near you.

**az group create** --name $myGroup --location $myLocation

1. Create an Azure Cache for Redis instance by using the az redis create command. The instance name needs to be unique and the script below will attempt to generate one for you, replace <myLocation> with the region you used in the previous step. This command will take a few minutes to complete.

$redisName=az204redis$RANDOM

**az redis create** --location $myLocation --resource-group $myGroup --name $redisName --sku Basic --vm-size c0

1. In the Azure portal navigate to the new Redis Cache you created.
2. Select **Access keys** in the **Settings** section of the Navigation Pane and leave the portal open. We'll copy the **Primary connection string (StackExchange.Redis)** value to use in the app later.

**Create the console application**

1. Create a console app by running the command below in the Visual Studio Code terminal.

dotnet new console -o Rediscache –f netcoreapp3.1

1. Open the app in Visual Studio Code by selecting **File > Open Folder** and choosing the folder for the app.
2. Add the StackExchange.Redis package to the project.

dotnet add package StackExchange.Redis

1. In the Program.cs file add the using statement below at the top.

using StackExchange.Redis;

using System.Threading.Tasks;

1. Add the following variable to the Program class, replace <REDIS\_CONNECTION\_STRING> with the **Primary connection string (StackExchange.Redis)** from the portal.
2. // connection string to your Redis Cache

       static string connectionString = "<REDIS\_CONNECTION\_STRING>"

1. Replace the Main method with the following code.

static async Task Main(string[] args) {

  //The connection to the Azure Cache for Redis is managed by the

// ConnectionMultiplexer class.

  using (var cache = ConnectionMultiplexer.Connect(connectionString)) {

    IDatabase db = cache.GetDatabase();

    // Snippet below executes a PING to test the server connection

    var result = await db.ExecuteAsync("ping");

    Console.WriteLine($"PING = {result.Type} : {result}");

// Call StringSetAsync on the IDatabase object to set the key "test:key" to

// the value "100"

    bool setValue = await db.StringSetAsync("test:key", "100");

    Console.WriteLine($"SET: {setValue}");

    // StringGetAsync takes the key to retrieve and return the value

    string getValue = await db.StringGetAsync("test:key");

    Console.WriteLine($"GET: {getValue}");

  }

}

1. In the Visual Studio Code terminal run the commands below to build the app to check for errors, and then run the app using the commands below

dotnet build

dotnet run

The output should be similar to the following:

PING = SimpleString : PONG

SET: True

GET: 100

1. Return to the portal and select **Activity log** in the **Azure Cache for Redis** blade. You can view the operations in the log.

**Clean up resources**

When the resources are no longer needed, you can use the az group delete command to remove the resource group.

az group delete -n $myGroup --no-wait