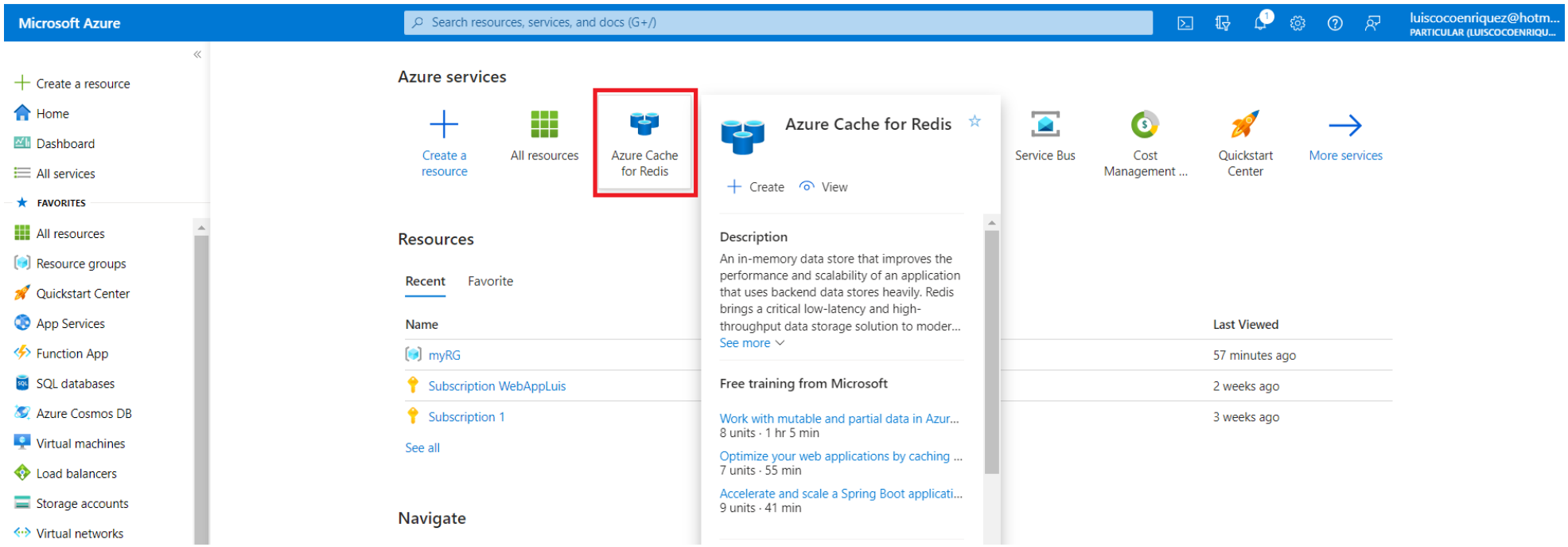
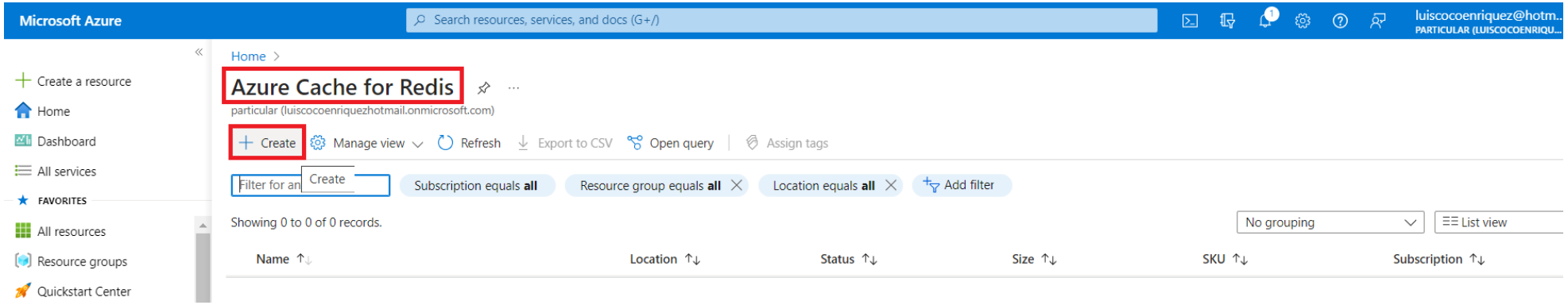
How to create .NET8 CRUD WebAPI Azure Cache for Redis Microservice

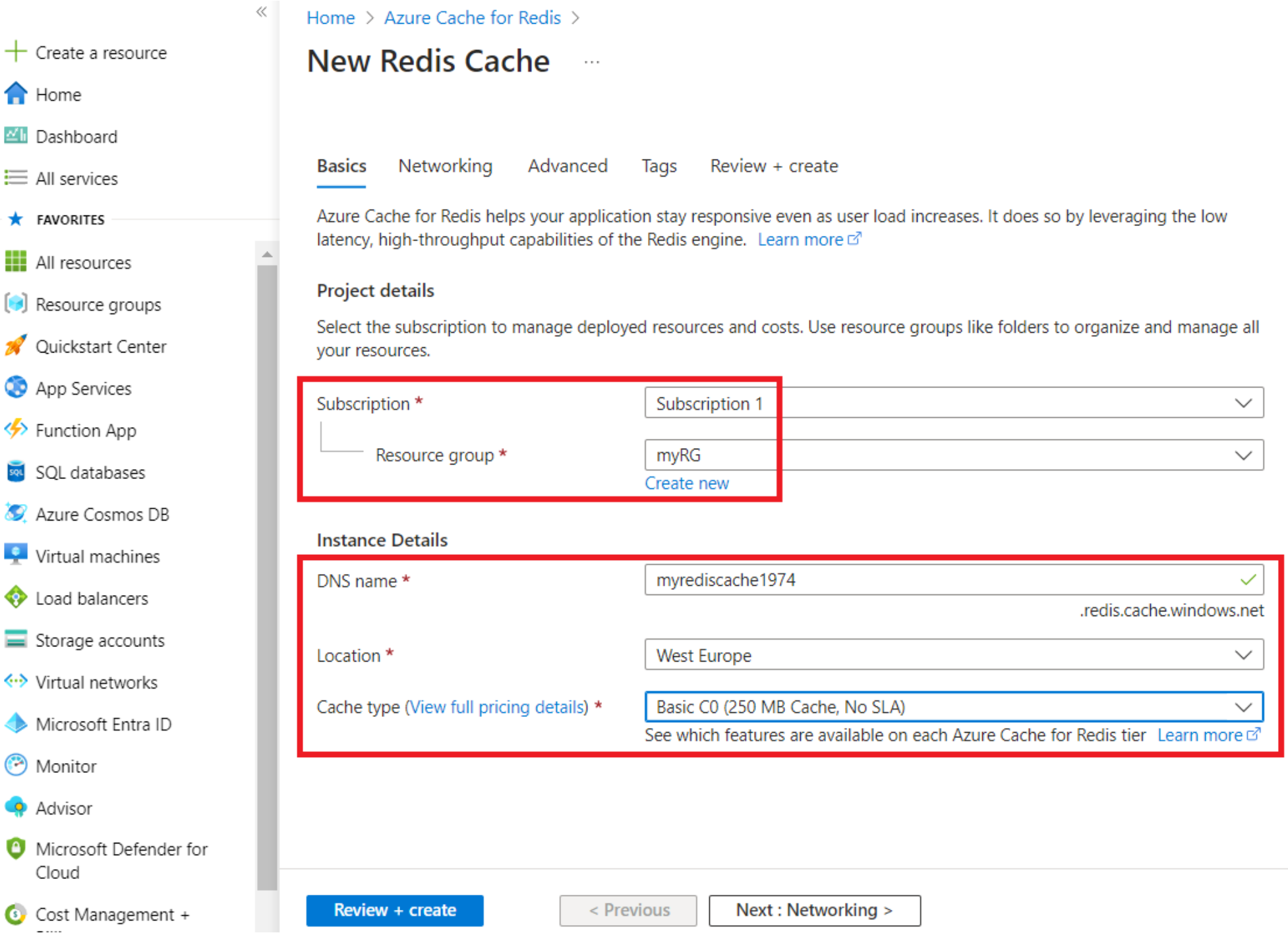
You can find the source code for this example in this github repo:

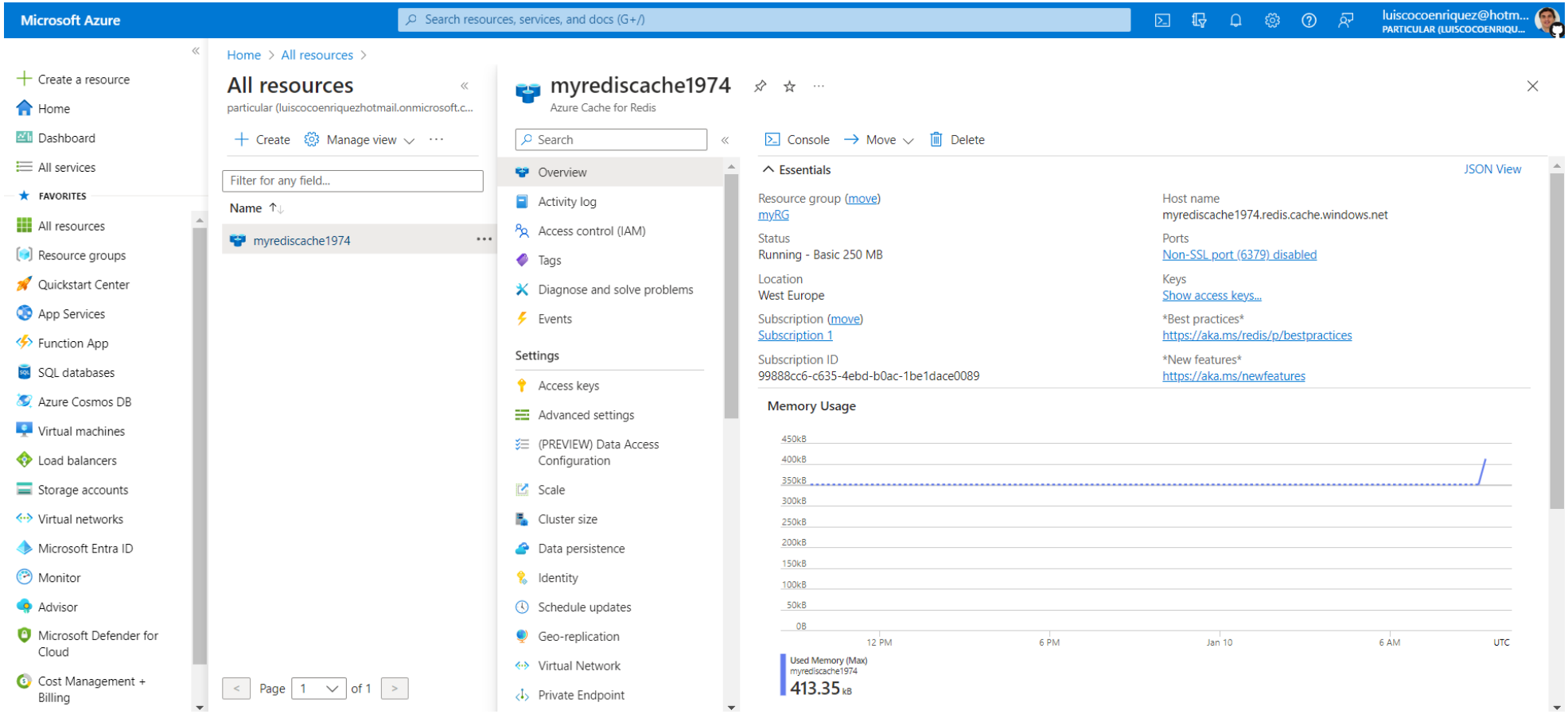
https://github.com/luiscoco/MicroServices_dotNET8_CRUD_WebAPI-Azure-Cache-for-Redis

1. Create Azure Cache for Redis in Azure Portal

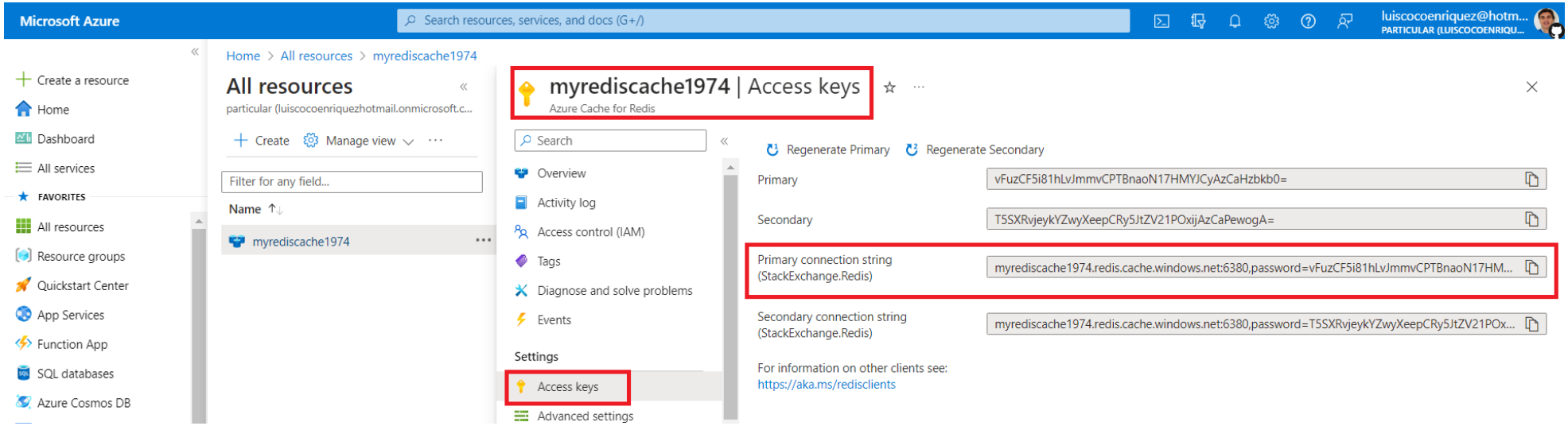



Microsoft Azure
Search resources, services, and docs (G+/)
luiscocoenriquez@hotm..
PARTICULAR (LUISCOCOENRIQU...
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
Home
Azure Cache for Redis
particular (luiscocoenriquezhotmail.onmicrosoft.com)
Create
Manage view
Refresh
Export to CSV
Open query
Assign tags
Filter for an
Create
Subscription equals all
Resource group equals all
Location equals all
Add filter
Showing 0 to 0 of 0 records.
No grouping
List view
Name
Location
Status
Size
SKU
Subscription


Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Microsoft Entra ID
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management +
Home > Azure Cache for Redis >
New Redis Cache
Basics
Networking
Advanced
Tags
Review + create
Azure Cache for Redis helps your application stay responsive even as user load increases. It does so by leveraging the low latency, high-throughput capabilities of the Redis engine. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Subscription 1
Resource group *
myRG
Create new
Instance Details
DNS name *
myrediscache1974
.redis.cache.windows.net
Location *
West Europe
Cache type (View full pricing details) *
Basic C0 (250 MB Cache, No SLA)
See which features are available on each Azure Cache for Redis tier Learn more
Review + create
< Previous
Next : Networking >



We copy the connection string in the appsettings.json file



2. appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "RedisCache": "myrediscache1974.redis.cache.windows.net:6380,password=vFuzCF5i81hLvJmmvCPTBnaoN17HMYJCyAzCaHzbkb0=,ssl=True,a
  }
}
```

3. We set the Middleware (Program.cs)

```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using StackExchange.Redis;
using AzureCacheforRedis.Services;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();

// Add Swagger service
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "Azure Cache for Redis API", Version = "v1" });
});

builder.Services.AddSingleton<IConnectionMultiplexer>(
    ConnectionMultiplexer.Connect(builder.Configuration.GetConnectionString("RedisCache")));
builder.Services.AddScoped<RedisCacheService>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();

    // Enable middleware to serve generated Swagger as a JSON endpoint
    app.UseSwagger();

    // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.)
    app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Azure Cache for Redis API V1"));
}
```

```
app.UseAuthorization();

app.MapControllers();

app.Run();
```

4. We create the Service (RedisCacheService.cs)

```
using StackExchange.Redis;
using System.Threading.Tasks;


namespace AzureCacheforRedis.Services
{
    public class RedisCacheService
    {
        private readonly IConnectionMultiplexer _connectionMultiplexer;

        public RedisCacheService(IConnectionMultiplexer connectionMultiplexer)
        {
            _connectionMultiplexer = connectionMultiplexer;
        }

        public async Task SetAsync(string key, string value)
        {
            var db = _connectionMultiplexer.GetDatabase();
            await db.StringSetAsync(key, value);
        }

        public async Task<string> GetAsync(string key)
        {
            var db = _connectionMultiplexer.GetDatabase();
            return await db.StringGetAsync(key);
        }
```

```
        public async Task DeleteAsync(string key)
        {
            var db = _connectionMultiplexer.GetDatabase();
            await db.KeyDeleteAsync(key);
        }

        // Other CRUD operations as needed

    }
}
```

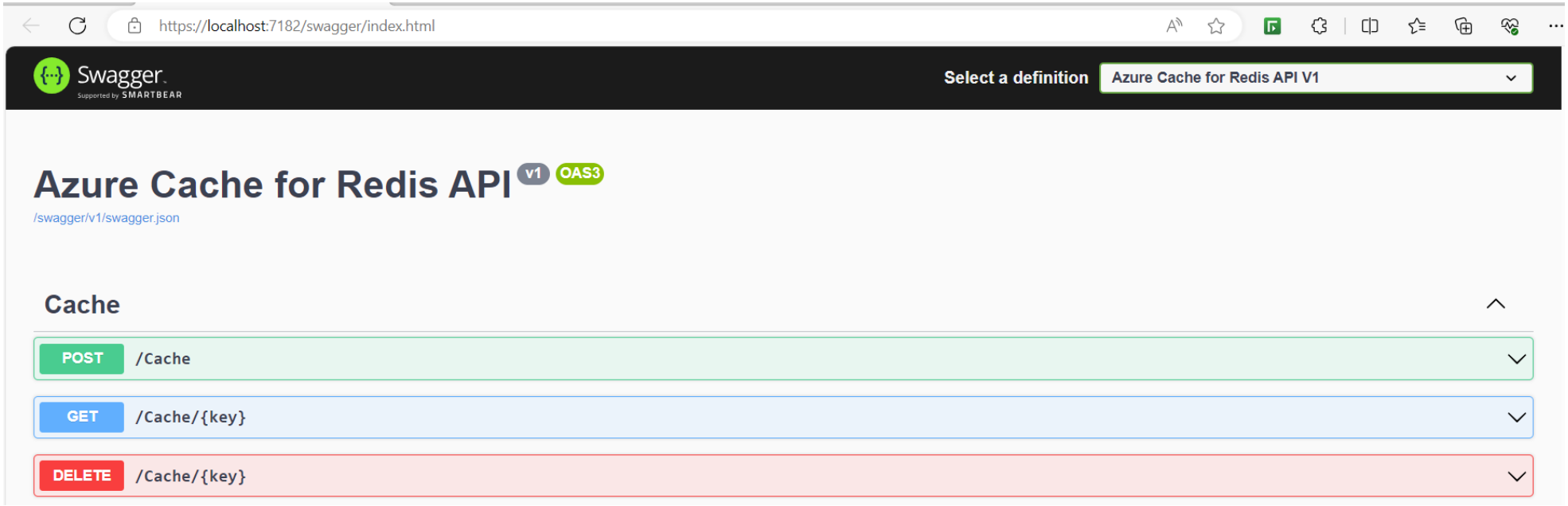
5. We create the Controller (CacheController.cs)

```
using AzureCacheforRedis.Services;
using Microsoft.AspNetCore.Mvc;
using System.Threading.Tasks;


namespace AzureCacheforRedis.Controllers
{
    [ApiController]
    [Route("[controller]")]

    public class CacheController : ControllerBase
    {
        private readonly RedisCacheService _cacheService;

        public CacheController(RedisCacheService cacheService)
        {
            _cacheService = cacheService;
        }

        [HttpPost]
        public async Task<IActionResult> Set(string key, string value)
        {
```

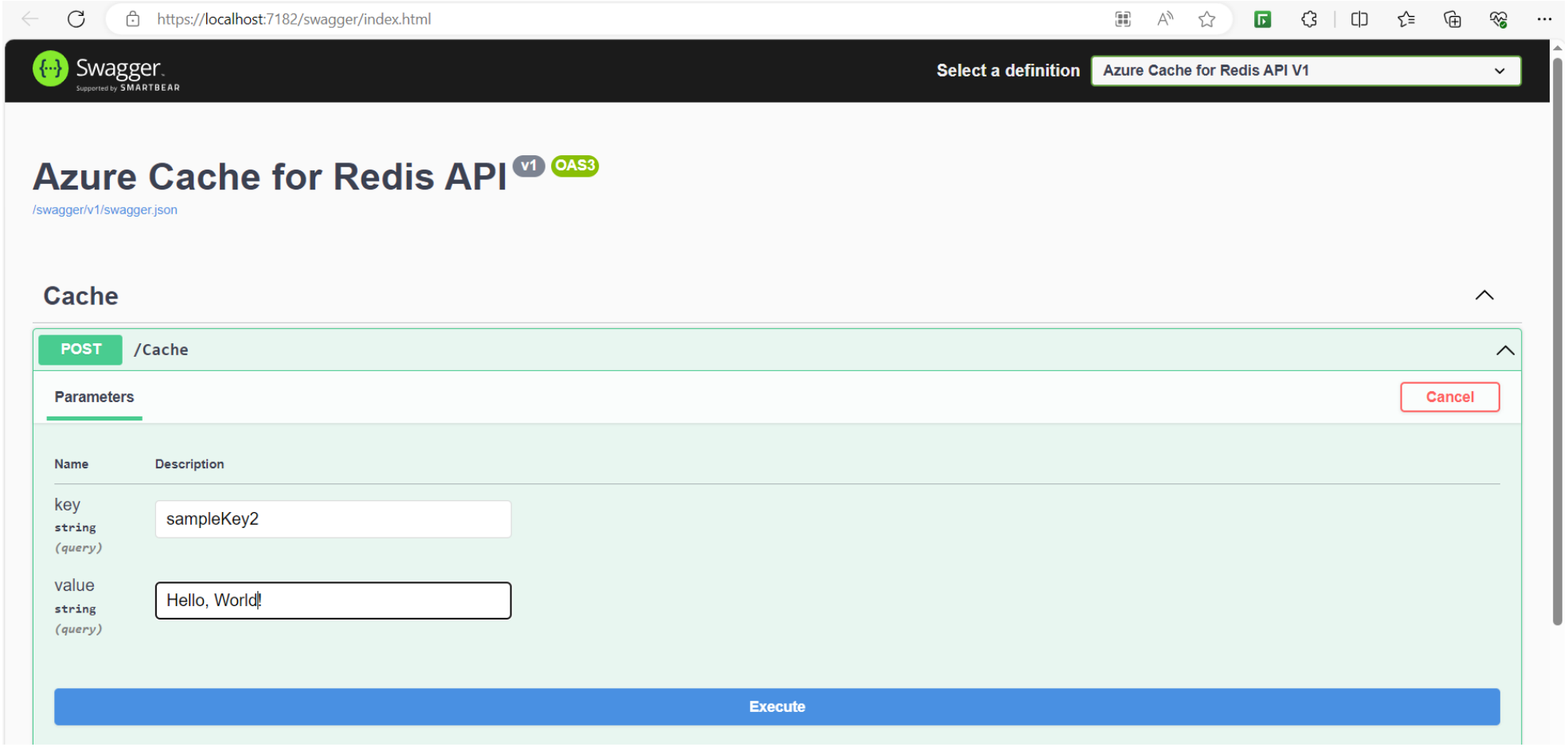
```
            await _cacheService.SetAsync(key, value);
            return Ok();
        }

        [HttpGet("{key}")]
        public async Task<IActionResult> Get(string key)
        {
            var value = await _cacheService.GetAsync(key);
            return Ok(value);
        }

        [HttpDelete("{key}")]
        public async Task<IActionResult> Delete(string key)
        {
            await _cacheService.DeleteAsync(key);
            return Ok();
        }

        // Add other endpoints for CRUD operations

    }
}
```

6. Verify the application

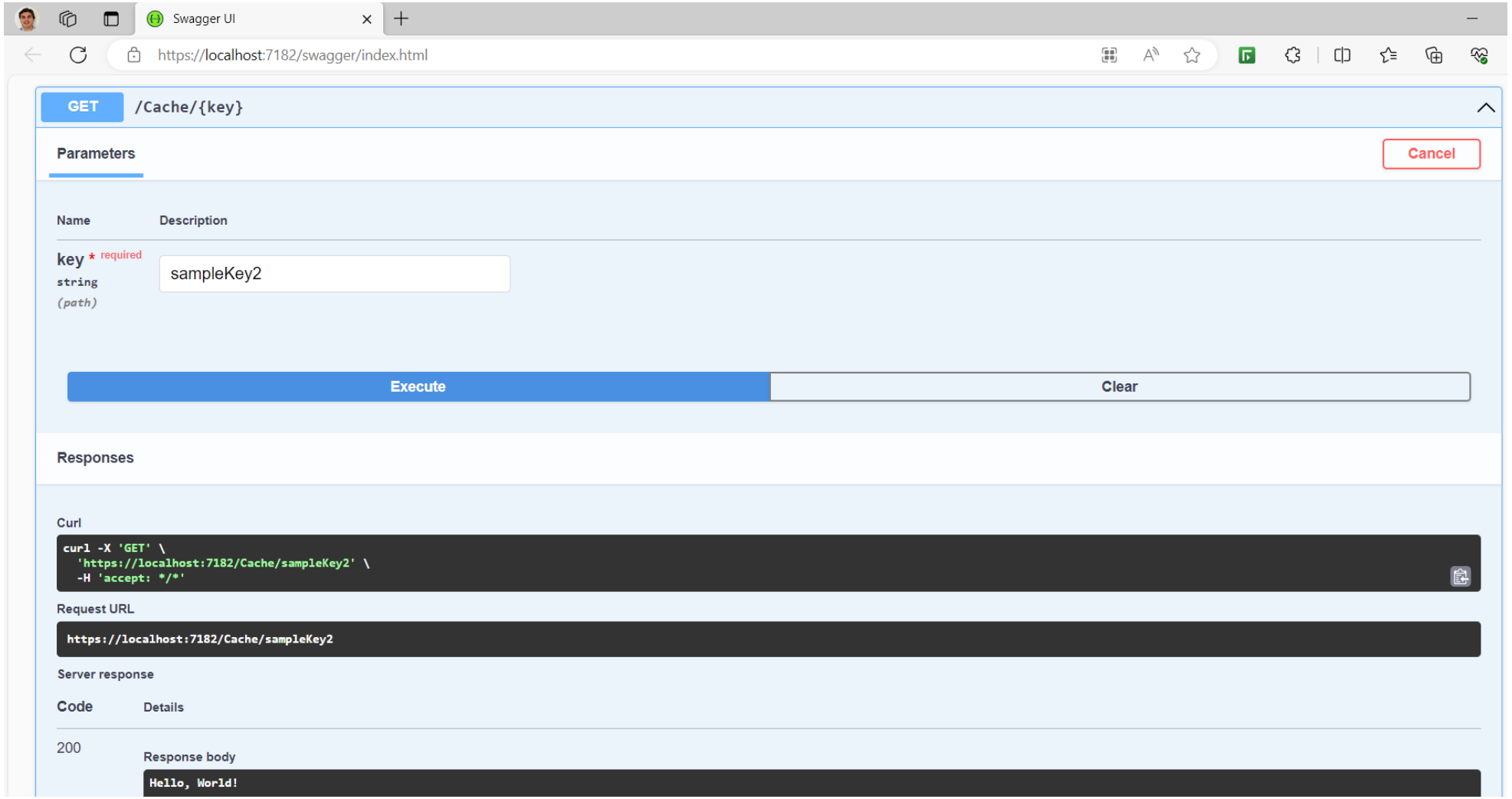
https://localhost:7182/swagger/index.html



We send a POST request



We send a GET request


Swagger UI
https://localhost:7182/swagger/index.html
GET
/Cache/{key}
Parameters
Cancel
Name
Description
key * required
string
(path)
sampleKey2
Execute
Clear
Responses
Curl
curl -X 'GET' \
'https://localhost:7182/Cache/sampleKey2' \
-H 'accept: */*'
Request URL
https://localhost:7182/Cache/sampleKey2
Server response
Code
Details
200
Response body
Hello, World!