# Titan - PoC

## Memcached as Distributed Cache in .Net Core
## Connecting a .Net Application to Oracle NoSQL Database Cloud Service

*Arunkumar Moola Balaji*
*SE Hub - OCI*

# Contents

# 1 Memcached as Distributed Cache in .Net Core

## 1.1 Create Windows Server

- Create a Windows Server compute instance on OCI
- Note down the IP address
- Login to the server as **OPC** user
- Download & Install Visual Studio 2019 - https://visualstudio.microsoft.com/downloads/

## 1.2 Create Memcached Server

- Create an OEL compute instance on OCI
- Note down the Private IP address
- Login to the server as **OPC** user to allow ingress connection through instance Firewall and install Memcached server

```
sudo apt-get install firewalld -y

sudo firewall-cmd --permanent --add-port=11211/tcp

sudo firewall-cmd --permanent --add-port=11211/udp

sudo firewall-cmd --reload
```

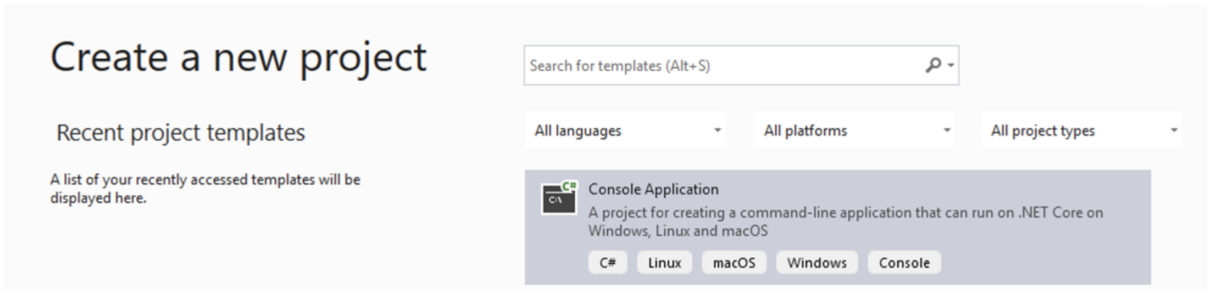
Install the Memcached server and the service starts automatically after installation and by default starts listening on port 11211. You can also launch multiple threads of Memcached by specifying the "-t" parameter while starting Memcached.

```
sudo apt-get -y install Memcached
```

- From OCI console, add an ingress rule to allow traffic from Windows Server through port **11211**

## 1.3 Creating Console Application

- Open Visual Studio 2019, create a new Console Application .NET Core



Create a new project

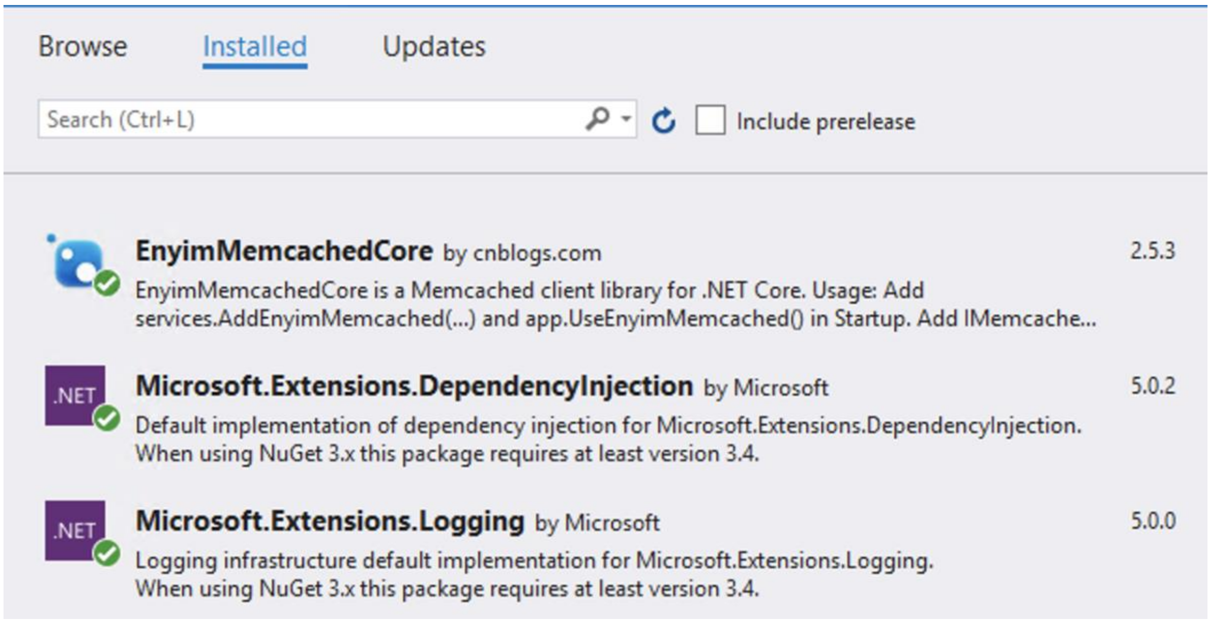
Recent project templates

A list of your recently accessed templates will be displayed here.

Console Application

A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS

- Add the following NuGet packages –
  - ***Microsoft.Extenstions.DependencyInjection***
  - ***EnyimMemcachedCore***
  - ***Microsoft.Extensions.Logging***



**EnyimMemcachedCore** by cnblogs.com 2.5.3

EnyimMemcachedCore is a Memcached client library for .NET Core. Usage: Add services.AddEnyimMemcached(...) and app.UseEnyimMemcached() in Startup. Add IMemcache...

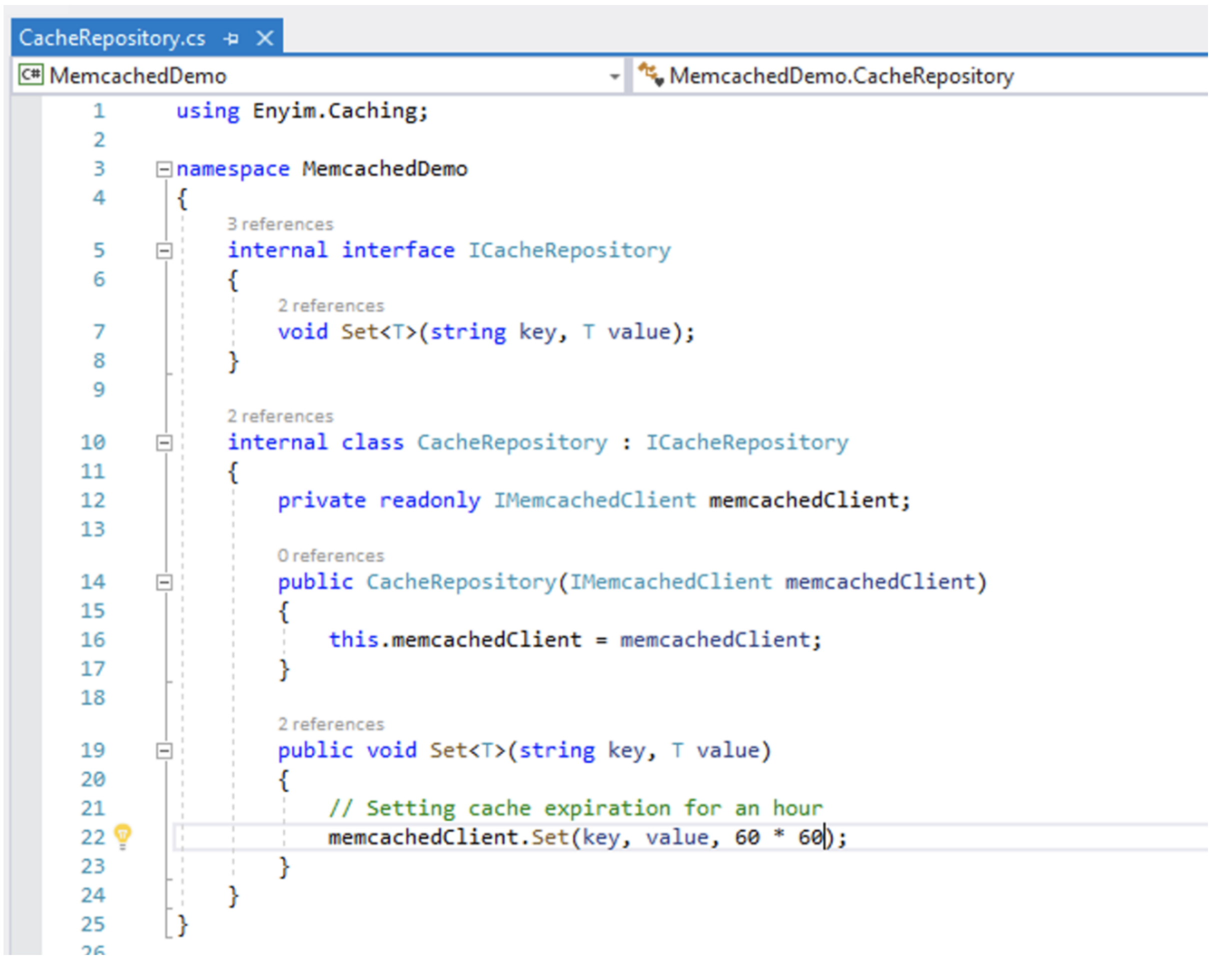
**Microsoft.Extensions.DependencyInjection** by Microsoft 5.0.2

Default implementation of dependency injection for Microsoft.Extensions.DependencyInjection. When using NuGet 3.x this package requires at least version 3.4.

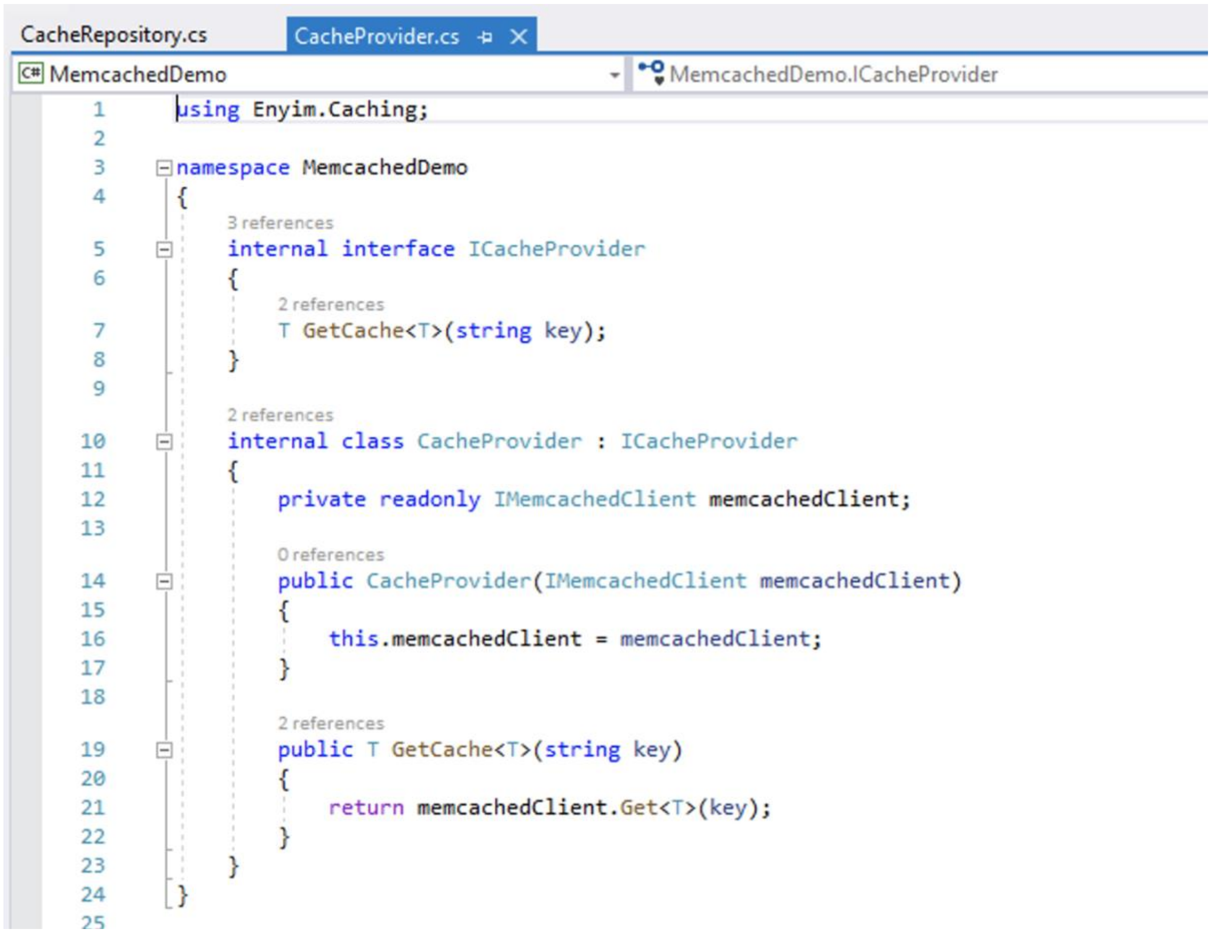
**Microsoft.Extensions.Logging** by Microsoft 5.0.0

Logging infrastructure default implementation for Microsoft.Extensions.Logging. When using NuGet 3.x this package requires at least version 3.4.

- Create a ***CacheRepository*** and ***CacheProvider*** classes. These classes will encapsulate the functionality of accessing Cache. This is important if we want to abstract the Cache provider from the rest of the application

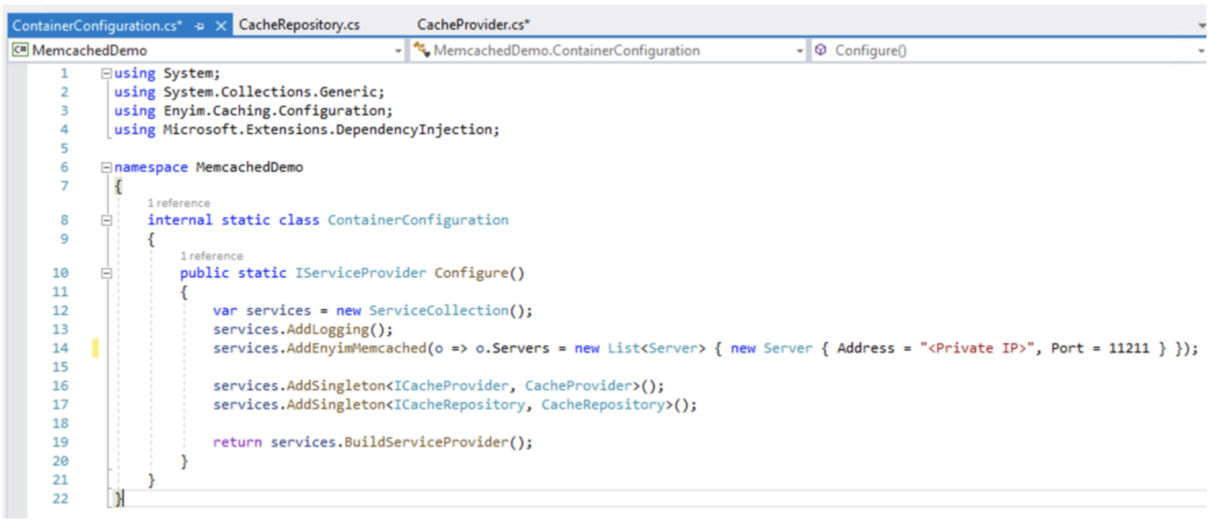


```
using Enyim.Caching;

namespace MemcachedDemo
{
    3 references
    internal interface ICacheRepository
    {
        2 references
        void Set<T>(string key, T value);
    }

    2 references
    internal class CacheRepository : ICacheRepository
    {
        private readonly IMemcachedClient memcachedClient;

        0 references
        public CacheRepository(IMemcachedClient memcachedClient)
        {
            this.memcachedClient = memcachedClient;
        }

        2 references
        public void Set<T>(string key, T value)
        {
            // Setting cache expiration for an hour
            memcachedClient.Set(key, value, 60 * 60);
        }
    }
}
```



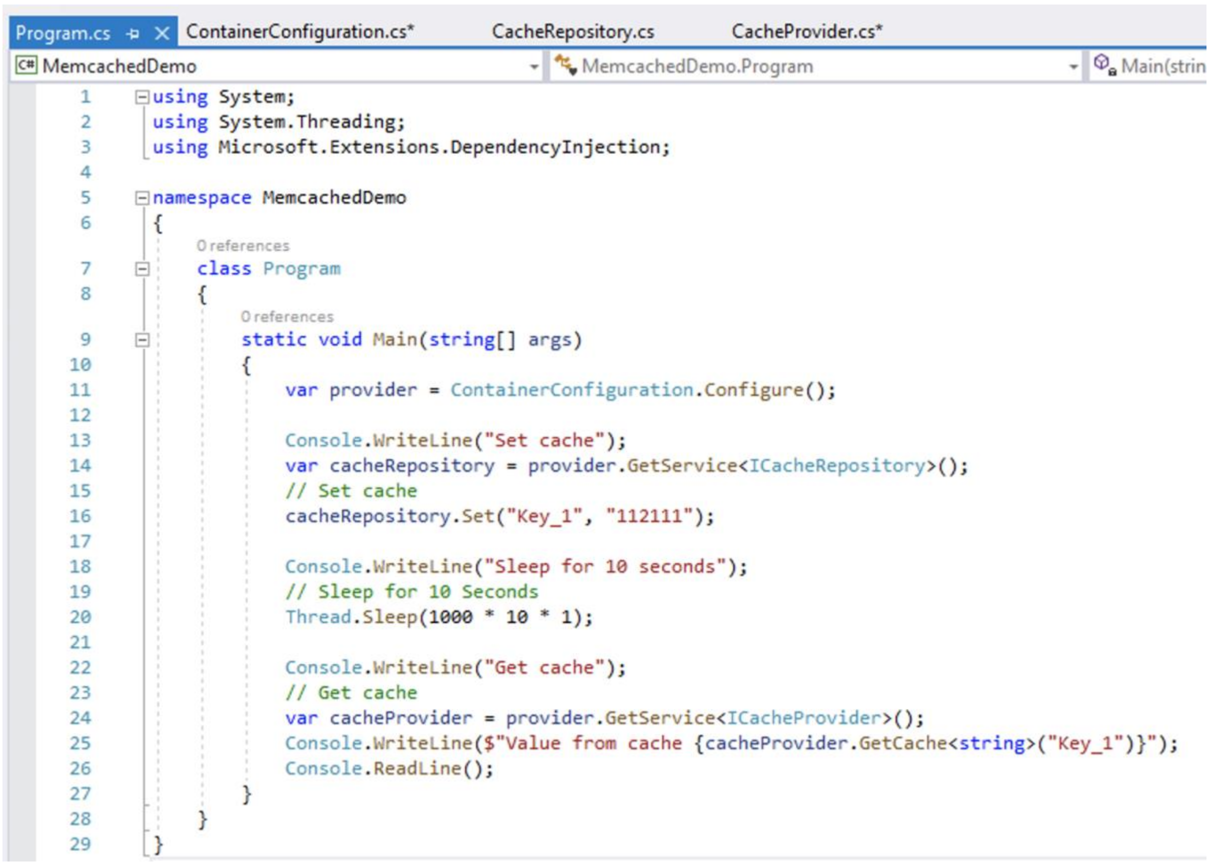
```
using Enyim.Caching;

namespace MemcachedDemo
{
    3 references
    internal interface ICacheProvider
    {
        2 references
        T GetCache<T>(string key);
    }

    2 references
    internal class CacheProvider : ICacheProvider
    {
        private readonly IMemcachedClient memcachedClient;

        0 references
        public CacheProvider(IMemcachedClient memcachedClient)
        {
            this.memcachedClient = memcachedClient;
        }

        2 references
        public T GetCache<T>(string key)
        {
            return memcachedClient.Get<T>(key);
        }
    }
}
```

- Create ***ContainerConfiguration*** class for register configuration for the application. Input the **Private IP** of the Memcached server



```
using System;
using System.Collections.Generic;
using Enyim.Caching.Configuration;
using Microsoft.Extensions.DependencyInjection;

namespace MemcachedDemo
{
    internal static class ContainerConfiguration
    {
        public static IServiceProvider Configure()
        {
            var services = new ServiceCollection();
            services.AddLogging();
            services.AddEnyimMemcached(o => o.Servers = new List<Server> { new Server { Address = "<Private IP>", Port = 11211 } });

            services.AddSingleton<ICacheProvider, CacheProvider>();
            services.AddSingleton<ICacheRepository, CacheRepository>();

            return services.BuildServiceProvider();
        }
    }
}
```

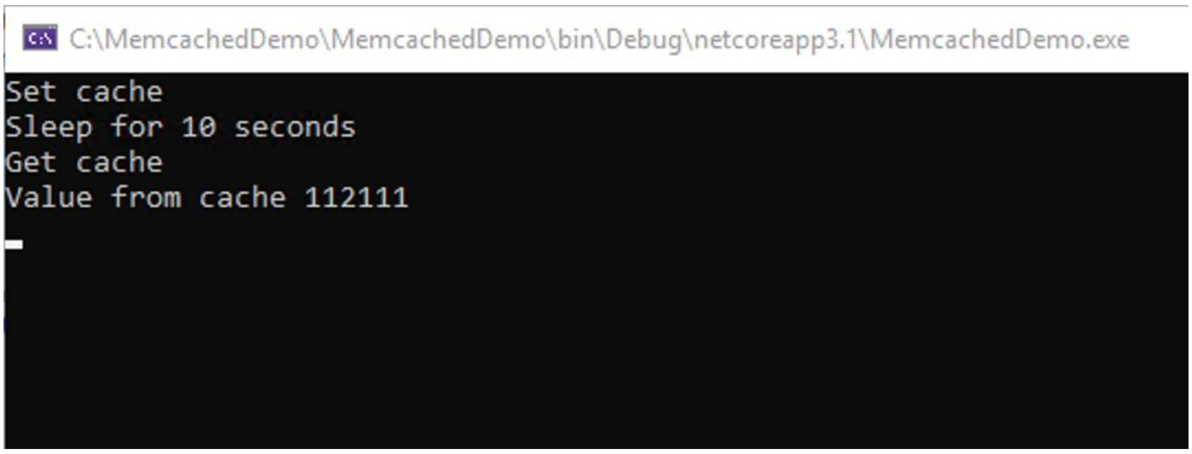
- Finally, update the ***Program*** class to set and get cache.



```
using System;
using System.Threading;
using Microsoft.Extensions.DependencyInjection;

namespace MemcachedDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            var provider = ContainerConfiguration.Configure();

            Console.WriteLine("Set cache");
            var cacheRepository = provider.GetService<ICacheRepository>();
            // Set cache
            cacheRepository.Set("Key_1", "112111");

            Console.WriteLine("Sleep for 10 seconds");
            // Sleep for 10 Seconds
            Thread.Sleep(1000 * 10 * 1);

            Console.WriteLine("Get cache");
            // Get cache
            var cacheProvider = provider.GetService<ICacheProvider>();
            Console.WriteLine($"Value from cache {cacheProvider.GetCache<string>("Key_1")}");
            Console.ReadLine();
        }
    }
}
```

- Run the project

There will be only 1 copy of key present in memcache instances , please refer https://github.com/memcached/memcached/wiki/TutorialCachingStory


C:\MemcachedDemo\MemcachedDemo\bin\Debug\netcoreapp3.1\MemcachedDemo.exe

```
Set cache
Sleep for 10 seconds
Get cache
Value from cache 112111
```

- To ensure that the .net application has set the cache in Memcached server, login to the server and run the following command

```
[opc@tit-memcache ~]$ memcached-tool localhost:11211 dump
Dumping memcache contents
  Number of buckets: 1
  Number of items  : 1
Dumping bucket 1 - 1 total items
add Key_1 274 1635838750 6
112111
```

For more information please refer - https://docs.oracle.com/en-us/iaas/Content/Resources/Assets/whitepapers/deploying-memcached-and-redis-on-oci.pdf

# 2 Connecting a .Net Application to Oracle NoSQL Database Cloud Service

## 2.1 Credentials

- Acquire credentials. See Acquire Credentials. You will need these:
  - Tenancy ID
  - User ID
  - API signing key (private key file in PEM format
  - Fingerprint for the public key uploaded to the user's account
  - Private key pass phrase, needed only if the private key is encrypted
  - Region

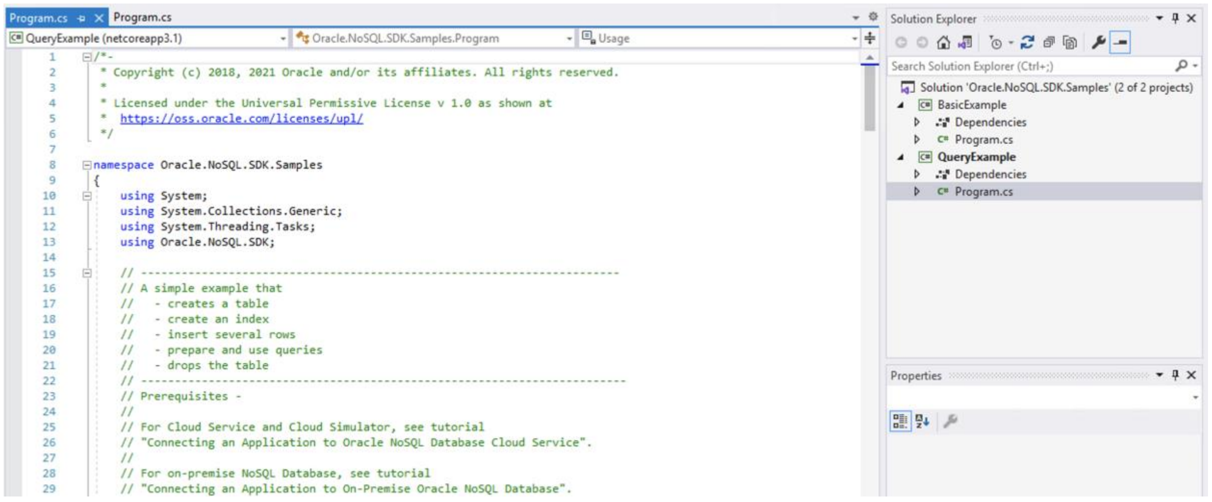
## 2.2 Config

- Put the information in a configuration file, **~/.oci/config**, where ~ is a value of *USERPROFILE* environment variable.

```
[DEFAULT]

tenancy=<your-tenancy-ocid>

user=<your-user-ocid>

fingerprint=<fingerprint-of-your-public-key>

key_file=<path-to-your-private-key-file>

pass_phrase=<your-private-key-passphrase>

region=<your-region-identifier>
```

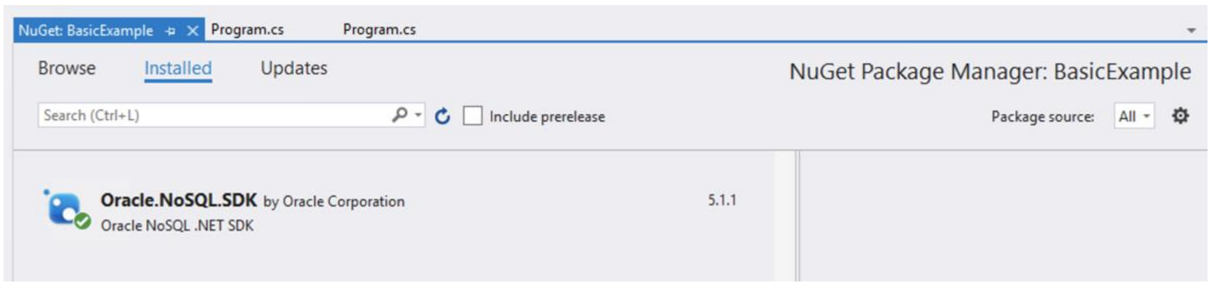
## 2.3 Sample Application

- Download/Clone the Oracle NoSQL Database .NET SDK from https://github.com/oracle/nosql-dotnet-sdk
- Open Visual Studio 2019, open the Samples solution located at Oracle.NoSQL.SDK.Samples/Oracle.NoSQL.SDK.Samples.sln


Program.cs
Program.cs
QueryExample (netcoreapp3.1)
Oracle.NoSQL.SDK.Samples.Program
Usage
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Oracle.NoSQL.SDK.Samples' (2 of 2 projects)
BasicExample
Dependencies
Program.cs
QueryExample
Dependencies
Program.cs
Properties

```
/*-
 * Copyright (c) 2018, 2021 Oracle and/or its affiliates. All rights reserved.
 *
 * Licensed under the Universal Permissive License v 1.0 as shown at
 *  https://oss.oracle.com/licenses/upl/
 */

namespace Oracle.NoSQL.SDK.Samples
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Oracle.NoSQL.SDK;

    // -----------------------------------------------------------------------
    // A simple example that
    //   - creates a table
    //   - create an index
    //   - insert several rows
    //   - prepare and use queries
    //   - drops the table
    // ------------------------------------------------------------------------
    // Prerequisites -
    //
    // For Cloud Service and Cloud Simulator, see tutorial
    // "Connecting an Application to Oracle NoSQL Database Cloud Service".
    //
    // For on-premise NoSQL Database, see tutorial
    // "Connecting an Application to On-Premise Oracle NoSQL Database".
```

- Add the following NuGet package –
    - ***Oracle.NoSQL.SDK***



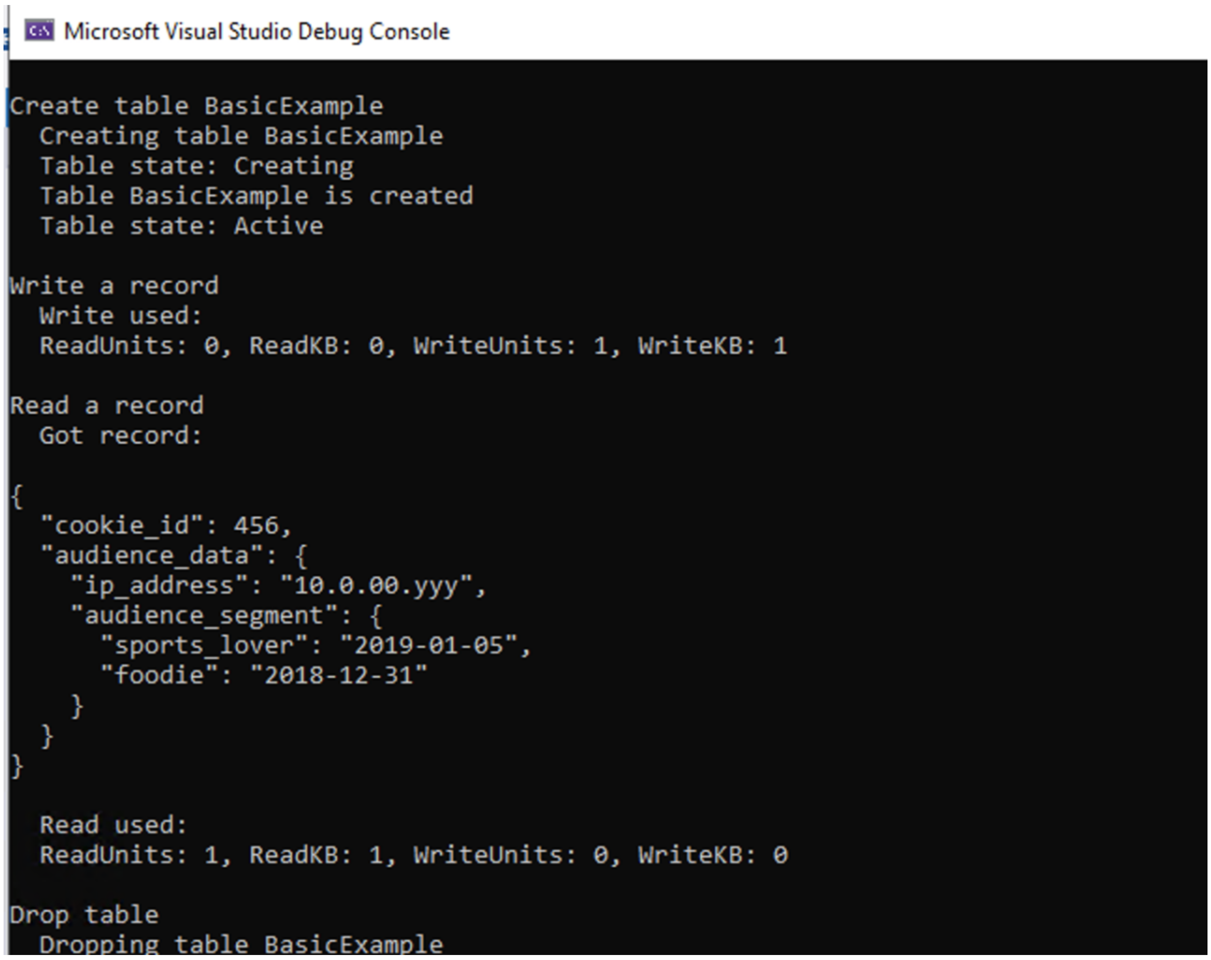
NuGet: BasicExample   Program.cs   Program.cs

Browse   Installed   Updates   NuGet Package Manager: BasicExample

Search (Ctrl+L)   Include prerelease   Package source: All

Oracle.NoSQL.SDK by Oracle Corporation   5.1.1
Oracle NoSQL .NET SDK

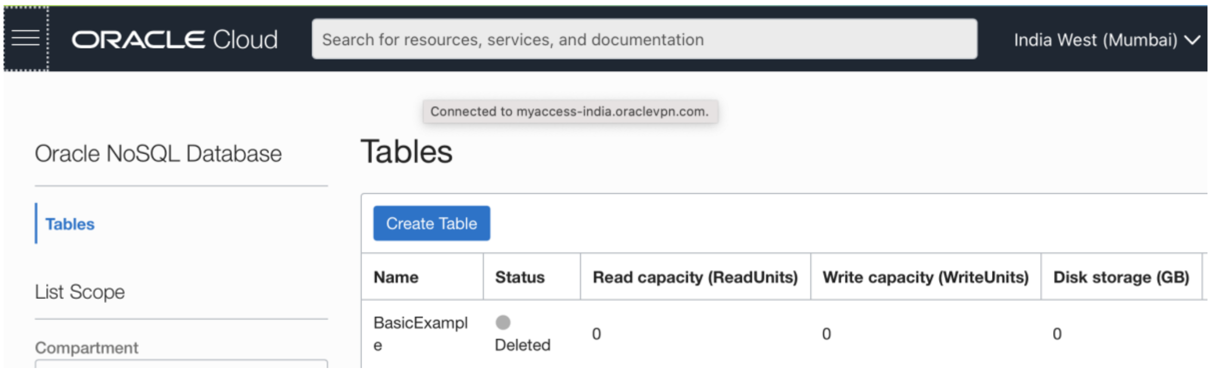
- Run any of the project – in this case, we run BasicExample



Microsoft Visual Studio Debug Console

```
Create table BasicExample
  Creating table BasicExample
  Table state: Creating
  Table BasicExample is created
  Table state: Active

Write a record
  Write used:
  ReadUnits: 0, ReadKB: 0, WriteUnits: 1, WriteKB: 1

Read a record
  Got record:

{
  "cookie_id": 456,
  "audience_data": {
    "ip_address": "10.0.00.yyy",
    "audience_segment": {
      "sports_lover": "2019-01-05",
      "foodie": "2018-12-31"
    }
  }
}

  Read used:
  ReadUnits: 1, ReadKB: 1, WriteUnits: 0, WriteKB: 0

Drop table
  Dropping table BasicExample
```

- Login to the OCI console, under Oracle NoSQL Database, you can see the name of the table that was created and deleted



ORACLE Cloud   Search for resources, services, and documentation   India West (Mumbai)

Connected to myaccess-india.oraclevpn.com.

Oracle NoSQL Database   Tables

Tables   List Scope   Compartment

Create Table

| Name | Status | Read capacity (ReadUnits) | Write capacity (WriteUnits) | Disk storage (GB) |
|---|---|---|---|---|
| BasicExample | ● Deleted | 0 | 0 | 0 |

For more Information please refer - https://docs.oracle.com/en/cloud/paas/nosql-cloud/csnsd/connecting-using-c.html

Try out the below code to verify if ONLY 1 copy of key is being placed in memcache instances

```
using System;
using System.Threading;
using Microsoft.Extensions.DependencyInjection;

namespace MemcachedDemo
{
 class Program
 {
  static void Main(string[] args)
  {
   var provider = ContainerConfiguration.Configure();
   var cacheRepository = provider.GetService<ICacheRepository>();
   var cacheProvider = provider.GetService<ICacheProvider>();
   for (int i = 1001; i < 2000; i++)
   {
    Console.WriteLine("Set cache for cache Key_"+i);

    // Set cache
    cacheRepository.Set("Key_"+i, i);

    // Console.WriteLine("Sleep for 10 seconds");
     //Sleep for 10 Seconds
    //Thread.Sleep(1000 * 10 * 1);

    Console.WriteLine("Get cache");
    // Get cache

    Console.WriteLine($"Value from cache {cacheProvider.GetCache<string>("Key_"+i)}");
   }

   Console.ReadLine();
  }
 }
```