Azure: Redis Cache



[**Sacha Barber**](https://www.codeproject.com/script/Membership/View.aspx?mid=569009), 25 Nov 2014 [CPOL](http://www.codeproject.com/info/cpol10.aspx)

|  |  |  |
| --- | --- | --- |
| https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-fill-md.png  https://codeproject.global.ssl.fastly.net/script/Ratings/Images/stars-empty-md.png | | 4.38 (6 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate this: | [vote 1vote 2vote 3vote 4vote 5](https://www.codeproject.com/Articles/846564/Azure-Redis-Cache) |  | | | |

Azure: Redis cache

This is a new post in a series of beginners' articles on how to do things in Azure. This series will be for absolute beginners, and if you are not one of those, this will not be for you.

You can find the complete set of posts that make us this series here:

* <https://sachabarbs.wordpress.com/azure/>

This time, we will look at how to use the preview of the Azure Redis Cache.

For this one, you will need access to the new Azure portal:

* <https://portal.azure.com/>

Why Redis Cache

I have used Redis in the past when I was looking into different NoSQL options, which you can read about here:

* <https://sachabarbs.wordpress.com/2012/05/21/document-dbs-a-quick-look-at-some-of-them/>

Redis was very easy to use and I found it very easy to store complex JSON object in it, at the time I was using the ServiceStack.Redis client libs.

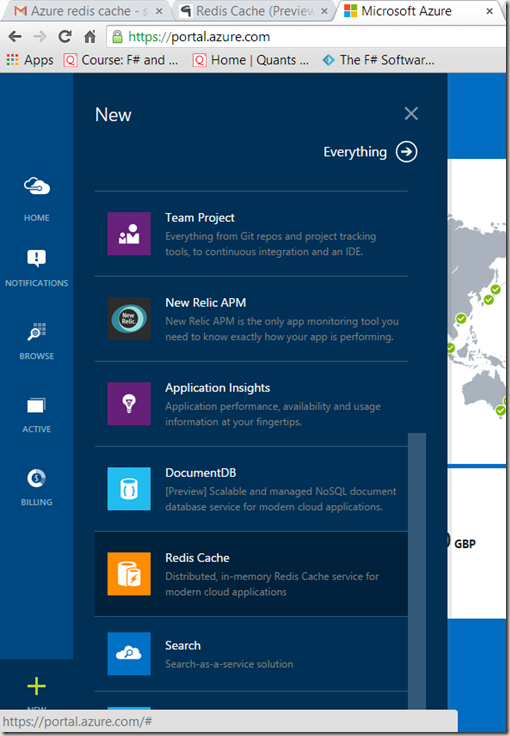
Here is what Scott Guthrie's blog has to say about the preview Redis Cache for Azure

*Unlike traditional caches which deal only with key-value pairs, Redis is popular for its support of high performance data types, on which you can perform atomic operations such as appending to a string, incrementing the value in a hash, pushing to a list, computing set intersection, union and difference, or getting the member with highest ranking in a sorted set.  Other features include support for transactions, pub/sub, Lua scripting, keys with a limited time-to-live, and configuration settings to make Redis behave more like a traditional cache.*

*Finally, Redis has a healthy, vibrant open source ecosystem built around it. This is reflected in the diverse set of Redis clients available across multiple languages. This allows it to be used by nearly any application, running on either Windows or Linux, that you host inside of Azure.*

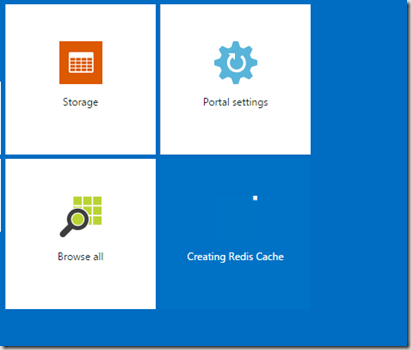
<http://weblogs.asp.net/scottgu/azure-redis-cache-disaster-recovery-to-azure-tagging-support-elastic-scale-for-sqldb-docdb>

Creating the Cache

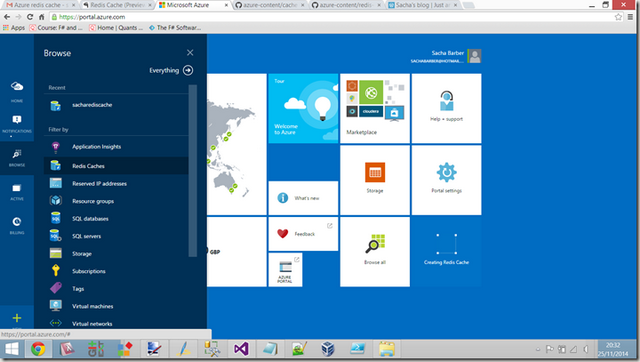


This will launch a wizard where you can pick the name/pricing plan, etc.

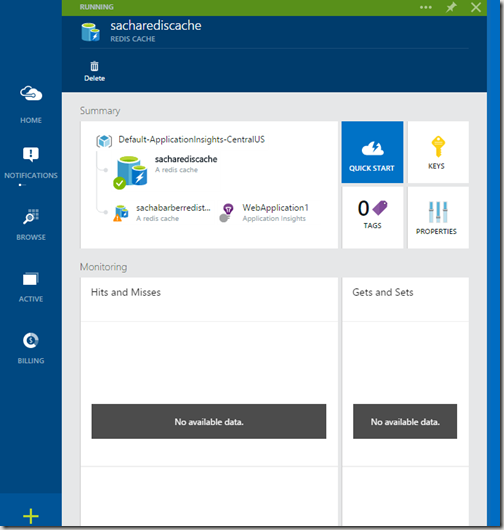
It may take a few minutes to configure the cache for the first time, you can see this in the portal, as shown below:



You may also check on the status of all your configured Azure items in the new Portal, using the BROWSE button, as shown below:



Once you have successfully created a Cache, you will see something like this:



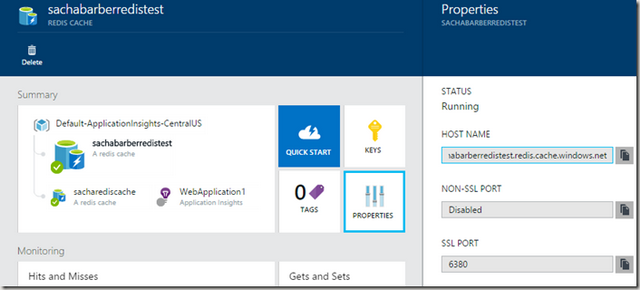
From here, you can grab the keys for your account, which we will use in the next bit of this post

Using A Created Cache

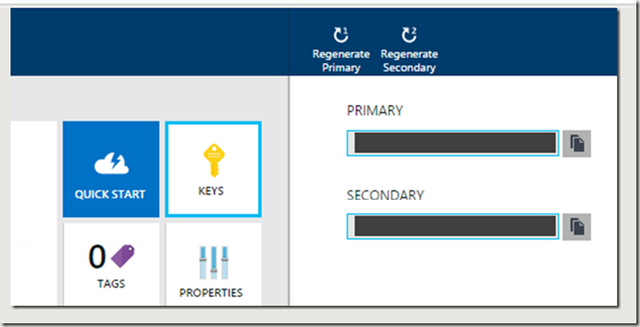
So once you have created a Redis Cache using the preview portal, you will likely want to connect and use it. So let's have a look at that, shall we.

We start by getting the connection details, this requires 2 steps:

1. Get the url, which is easy to get using the **properties** window, as shown below:



1. Once you have copied the host name url, we need to copy the password, which you can grab from the **keys** area:



Once you have these 2 bits of information, we are able to start using the Redis Azure cache.

Code Code Code

You will need to add the following NuGet package: **StackExchange.Redis**

NuGet command line > Install-Package StackExchange.Redis

Once you have that, it's finally time to start coding, so what do you need to use the cache.

Not much as it turns out. The following is a fully working program, which stores and retrieves some basic stringvalues, and also a fully serialized object.

**NOTE**: To see what types you can use with the Redis Cache, read this link: <http://redis.io/topics/data-types>

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using StackExchange.Redis;

namespace AzureRedisCacheDemo

{

class Program

{

static void Main(string[] args)

{

ConnectionMultiplexer connection = ConnectionMultiplexer.Connect(

"sachabarberredistest.redis.cache.windows.net,

ssl=true,password=\*\*\* YOUR ACCOUNT KEY HERE \*\*\*");

IDatabase cache = connection.GetDatabase();

*// Perform cache operations using the cache object...*

*// Simple put of integral data types into the cache*

cache.StringSet("key1", "dsyavdadsahda");

cache.StringSet("key2", 25);

Foo foo1 = new Foo() {Age = 1, Name = "Foo1"};

var serializedFoo = JsonConvert.SerializeObject(foo1);

cache.StringSet("serializedFoo", serializedFoo);

Foo foo3 = new Foo() { Age = 1, Name = "Foo3" };

var serializedFoo3 = JsonConvert.SerializeObject(foo1);

cache.StringSet("serializedFoo3", serializedFoo3);

*// Simple get of data types from the cache*

string key1 = cache.StringGet("key1");

int key2 = (int)cache.StringGet("key2");

var foo2 = JsonConvert.DeserializeObject<Foo>(cache.StringGet("serializedFoo"));

bool areEqual = foo1 == foo2;

var foo4 = JsonConvert.DeserializeObject<Foo>(cache.StringGet("serializedFoo3"));

bool areEqual2 = foo3 == foo4;

Console.ReadLine();

}

}

public class Foo : IEquatable<Foo>

{

public string Name { get; set; }

public int Age { get; set; }

public bool Equals(Foo other)

{

if (ReferenceEquals(null, other)) return false;

if (ReferenceEquals(this, other)) return true;

return string.Equals(Name, other.Name) && Age == other.Age;

}

public override bool Equals(object obj)

{

if (ReferenceEquals(null, obj)) return false;

if (ReferenceEquals(this, obj)) return true;

if (obj.GetType() != this.GetType()) return false;

return Equals((Foo) obj);

}

public override int GetHashCode()

{

unchecked

{

return ((Name != null ? Name.GetHashCode() : 0)\*397) ^ Age;

}

}

public static bool operator ==(Foo left, Foo right)

{

return Equals(left, right);

}

public static bool operator !=(Foo left, Foo right)

{

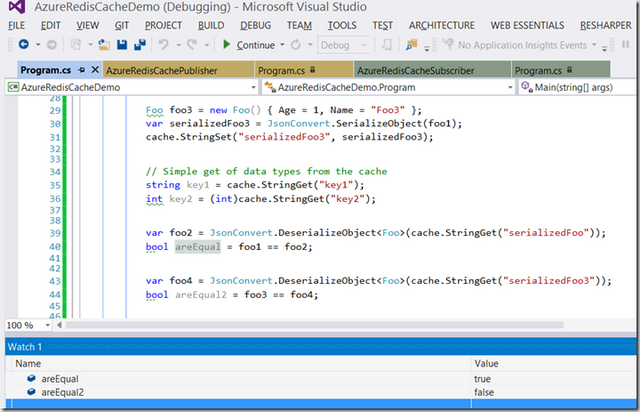
return !Equals(left, right);

}

}

}

When you run this code, you should see something like this:



It is a VERY simple demo, but I feel it demonstrates the cache nicely.

In this example, I am using JSON.Net to serialize objects to strings, which are then stored in the Redis Cache, you may have some other serializer you prefer, but this does illustrate the point of a working Redis Cache ok, I feel.

StackExchange.Redis Pub.Sub

Redis may also be used as a pub/sub framework, which you can use as follows:

Here is a basic publisher:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using StackExchange.Redis;

namespace AzureRedisCachePublisher

{

class Program

{

static void Main(string[] args)

{

ConnectionMultiplexer connection = ConnectionMultiplexer.Connect(

"sachabarberredistest.redis.cache.windows.net,ssl=true,

password=\*\*\* YOUR ACCOUNT KEY HERE \*\*\*");

ISubscriber sub = connection.GetSubscriber();

Console.WriteLine("Press a key to pubish");

Console.ReadLine();

sub.Publish("messages", "This is from the publisher");

Console.ReadLine();

}

}

}

And here is a basic Subscriber:

Which when run will give you something like this:

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using StackExchange.Redis;

namespace AzureRedisCacheSubscriber

{

class Program

{

static void Main(string[] args)

{

ConnectionMultiplexer connection = ConnectionMultiplexer.Connect(

"sachabarberredistest.redis.cache.windows.net,ssl=true,

password=\*\*\* YOUR ACCOUNT KEY HERE \*\*\*");

ISubscriber sub = connection.GetSubscriber();

sub.Subscribe("messages", (channel, message) =>

{

Console.WriteLine((string)message);

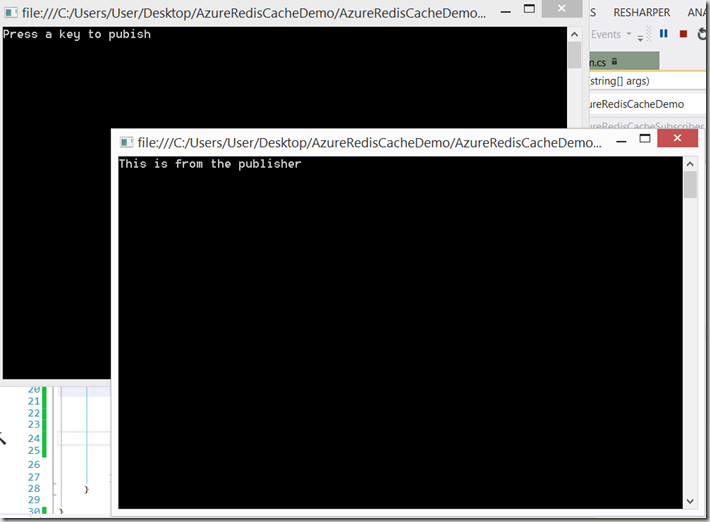
});

Console.ReadLine();

}

}

}

[](https://sachabarbs.files.wordpress.com/2014/11/image53.png)

Anyway, that is all for now, enjoy until next time.

This has barely scratched the surface of working with StackExhange.Redis, if you want to know more, read the documentation here: <https://github.com/StackExchange/StackExchange.Redis>.