**How to Use Azure Redis Cache**

[AZURE.SELECTOR]

* [.Net](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md)
* [Node.js](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-nodejs-get-started.md)
* [Java](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-java-get-started.md)
* [Python](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-python-get-started.md)

This guide shows you how to get started using **Azure Redis Cache**. Microsoft Azure Redis Cache is based on the popular open source Redis Cache. It gives you access to a secure, dedicated Redis cache, managed by Microsoft. A cache created using Azure Redis Cache is accessible from any application within Microsoft Azure.

Microsoft Azure Redis Cache is available in the following tiers:

* **Basic** – Single node. Multiple sizes up to 53 GB.
* **Standard** – Two-node Primary/Replica. Multiple sizes up to 53 GB. 99.9% SLA.
* **Premium** – Two-node Primary/Replica with up to 10 shards. Multiple sizes from 6 GB to 530 GB (contact us for more). All Standard tier features and more including support for [Redis cluster](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-clustering.md), [Redis persistence](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-persistence.md), and [Azure Virtual Network](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-vnet.md). 99.9% SLA.

Each tier differs in terms of features and pricing. For information on pricing, see [Cache Pricing Details](http://www.windowsazure.com/pricing/details/cache/).

This guide shows you how to use the [StackExchange.Redis](http://github.com/StackExchange/StackExchange.Redis) client using C# code. The scenarios covered include **creating and configuring a cache**, **configuring cache clients**, and **adding and removing objects from the cache**. For more information on using Azure Redis Cache, refer to the [Next Steps](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#next-steps) section. For a step-by-step tutorial of building an ASP.NET MVC web app with Redis Cache, see [How to create a Web App with Redis Cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-web-app-howto.md).

**Get Started with Azure Redis Cache**

Getting started with Azure Redis Cache is easy. To get started, you provision and configure a cache. Next, you configure the cache clients so they can access the cache. Once the cache clients are configured, you can begin working with them.

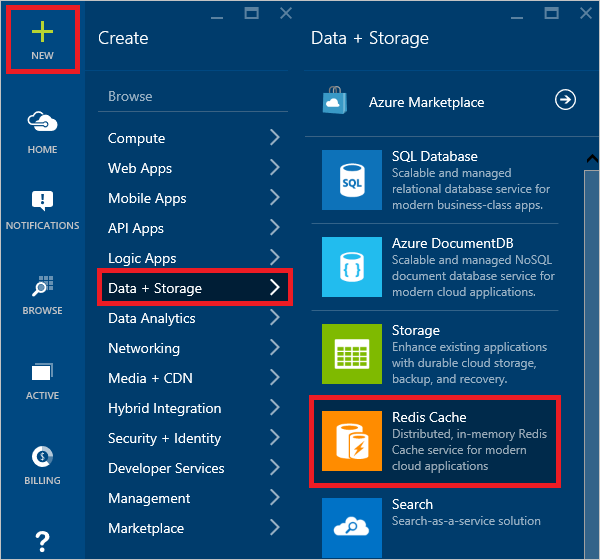
* [Create the cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#create-cache)
* [Configure the cache clients](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#NuGet)

**Create a cache**

To create a cache, first sign in to the [Azure Portal](https://portal.azure.com/), and click **New**, **Data + Storage**, **Redis Cache**.

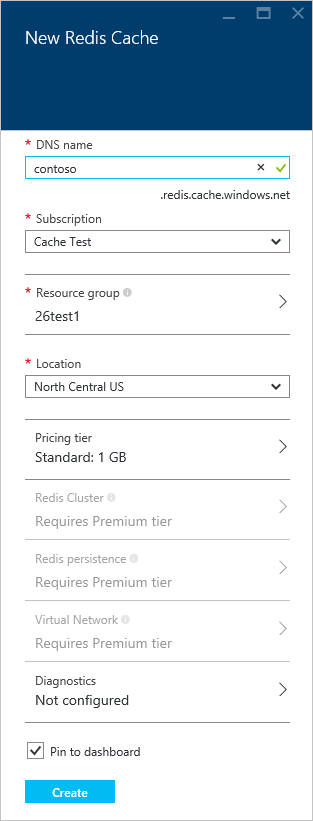
[AZURE.NOTE] In addition to creating caches in the Azure Portal, you can also create them using ARM templates, PowerShell, or Azure CLI.

* To create a cache using ARM templates, see [Create a Redis cache using a template](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-redis-cache-arm-provision.md).
* To create a cache using Azure PowerShell, see [Manage Azure Redis Cache with Azure PowerShell](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-howto-manage-redis-cache-powershell.md).
* To create a cache using Azure CLI, see [How to create and manage Azure Redis Cache using the Azure Command-Line Interface (Azure CLI)](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-manage-cli.md).

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-new-cache-menu.png)

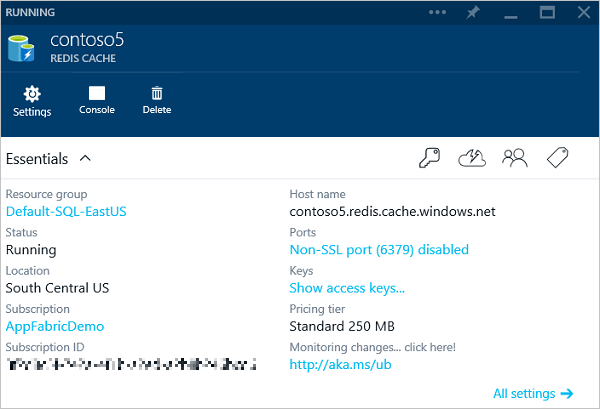
[AZURE.NOTE] If you don't have an Azure account, you can [Open an Azure account for free](https://azure.microsoft.com/pricing/free-trial/?WT.mc_id=redis_cache_hero) in just a couple of minutes.

In the **New Redis Cache** blade, specify the desired configuration for the cache.

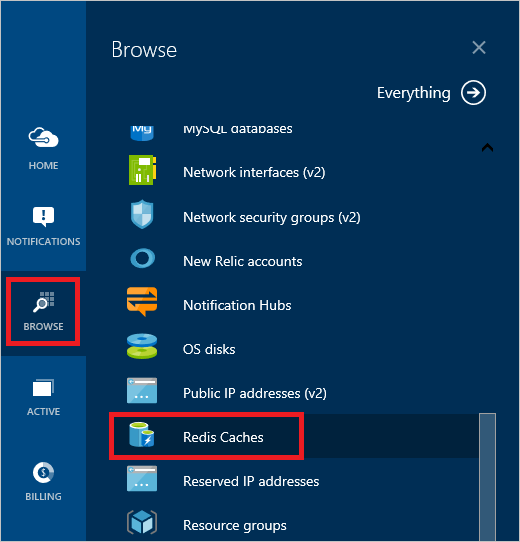
[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-cache-create.png)

* In **Dns name**, enter a cache name to use for the cache endpoint. The cache name must be a string between 1 and 63 characters and contain only numbers, letters, and the - character. The cache name cannot start or end with the -character, and consecutive - characters are not valid.
* For **Subscription**, select the Azure subscription that you want to use for the cache. If your account has only one subscription, it will be automatically selected and the **Subscription** drop-down will not be displayed.
* In **Resource group**, select or create a resource group for your cache. For more information, see [Using Resource groups to manage your Azure resources](http://azure.microsoft.com/documentation/articles/resource-group-overview/).
* Use **Location** to specify the geographic location in which your cache is hosted. For the best performance, Microsoft strongly recommends that you create the cache in the same region as the cache client application.
* Use **Pricing Tier** to select the desired cache size and features.
* **Redis cluster** allows you to create caches larger than 53 GB and to shard data across multiple Redis nodes. For more information, see [How to configure clustering for a Premium Azure Redis Cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-clustering.md).
* **Redis persistence** offers the ability to persist your cache to an Azure Storage account. For instructions on configuring persistence, see [How to configure persistence for a Premium Azure Redis Cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-persistence.md).
* **Virtual Network** provides enhanced security and isolation by restricting access to your cache to only those clients within the specified Azure Virtual Network. You can use all the features of VNet such as subnets, access control policies, and other features to further restrict access to Redis. For more information, see [How to configure Virtual Network support for a Premium Azure Redis Cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-premium-vnet.md).
* Use **Diagnostics** to specify a storage account for cache metrics. For more informations on configuring and viewing cache metrics, see [How to monitor Azure Redis Cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-monitor.md).

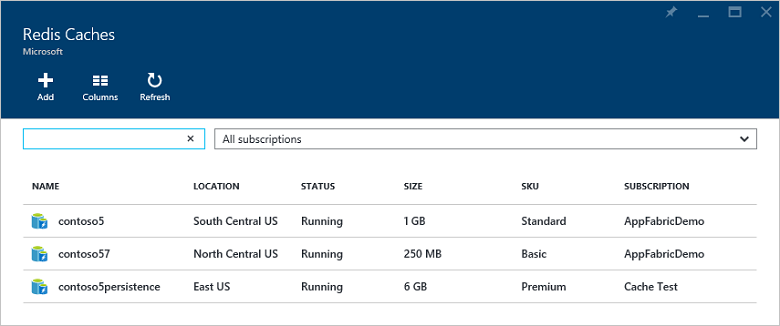
Once the new cache options are configured, click **Create**. It can take a few minutes for the cache to be created. To check the status, you can monitor the progress on the startboard. After the cache has been created, your new cache has a **Running**status and is ready for use with default settings.

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-cache-created.png)

Once your cache is created, you can access it from the **Browse** blade.

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-browse-caches.png)

Click **Redis Caches** to view your caches.

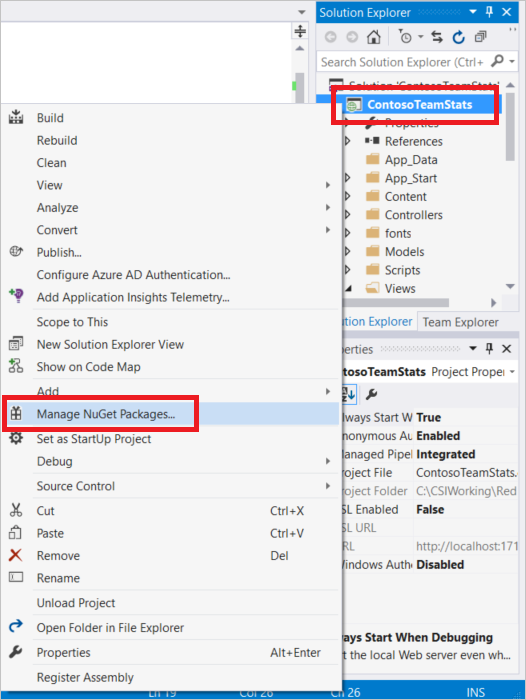
[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-caches.png)

**Configure the cache clients**

A cache created using Azure Redis Cache is accessible from any Azure application. .NET applications developed in Visual Studio can use the **StackExchange.Redis** cache client, which can be configured using a NuGet package that simplifies the configuration of cache client applications.

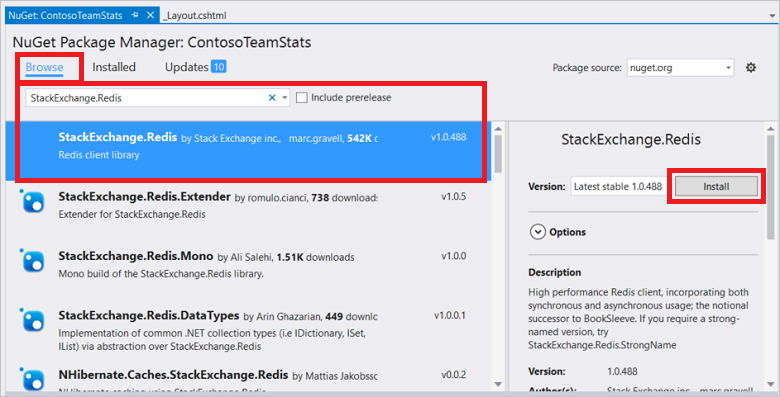
[AZURE.NOTE] For more information, see the [StackExchange.Redis](http://github.com/StackExchange/StackExchange.Redis) github page and the [StackExchange.Redis cache client documentation](http://github.com/StackExchange/StackExchange.Redis" \l "documentation).

To configure a client application in Visual Studio using the StackExchange.Redis NuGet package, right-click the project in **Solution Explorer** and choose **Manage NuGet Packages**.

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-manage-nuget-menu.png)

Type **StackExchange.Redis** or **StackExchange.Redis.StrongName** into the search text box, select the desired version from the results, and click **Install**.

[AZURE.NOTE] If you prefer to use a strong-named version of the **StackExchange.Redis** client library, choose **StackExchange.Redis.StrongName**; otherwise choose **StackExchange.Redis**.

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-stackexchange-redis.png)

The NuGet package downloads and adds the required assembly references for your client application to access Azure Redis Cache with the StackExchange.Redis cache client.

Once your client project is configured for caching, you can use the techniques described in the following sections for working with your cache.

**Working with Caches**

The steps in this section describe how to perform common tasks with Cache.

* [Connect to the cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#connect-to-cache)
* [Add and retrieve objects from the cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#add-object)
* [Work with .NET objects in the cache](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-dotnet-how-to-use-azure-redis-cache.md#work-with-net-objects-in-the-cache)

**Connect to the cache**

In order to programmatically work with a cache, you need a reference to the cache. Add the following to the top of any file from which you want to use the StackExchange.Redis client to access an Azure Redis Cache.

using StackExchange.Redis;

[AZURE.NOTE] The StackExchange.Redis client requires .NET Framework 4 or higher.

The connection to the Azure Redis Cache is managed by the ConnectionMultiplexer class. This class is designed to be shared and reused throughout your client application, and does not need to be created on a per operation basis.

To connect to an Azure Redis Cache and be returned an instance of a connected ConnectionMultiplexer, call the static Connect method and pass in the cache endpoint and key like the following example. Use the key generated from the Azure Portal as the password parameter.

ConnectionMultiplexer connection = ConnectionMultiplexer.Connect("contoso5.redis.cache.windows.net,abortConnect=false,ssl=true,password=...");

[AZURE.IMPORTANT] Warning: Never store credentials in source code. To keep this sample simple, I’m showing them in the source code. See [How Application Strings and Connection Strings Work](http://azure.microsoft.com/blog/2013/07/17/windows-azure-web-sites-how-application-strings-and-connection-strings-work/) for information on how to store credentials.

If you don't want to use SSL, either set ssl=false or omit the ssl parameter.

[AZURE.NOTE] The non-SSL port is disabled by default for new caches. For instructions on enabling the non-SSL port, see the [Access Ports](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-configure.md#access-ports)..

One approach to sharing a ConnectionMultiplexer instance in your application is to have a static property that returns a connected instance, similar to the following example. This provides a thread-safe way to initialize only a single connected ConnectionMultiplexer instance. In these examples abortConnect is set to false, which means that the call will succeed even if a connection to the Azure Redis Cache is not established. One key feature of ConnectionMultiplexer is that it will automatically restore connectivity to the cache once the network issue or other causes are resolved.

private static Lazy<ConnectionMultiplexer> lazyConnection = new Lazy<ConnectionMultiplexer>(() =>

{

return ConnectionMultiplexer.Connect("contoso5.redis.cache.windows.net,abortConnect=false,ssl=true,password=...");

});

public static ConnectionMultiplexer Connection

{

get

{

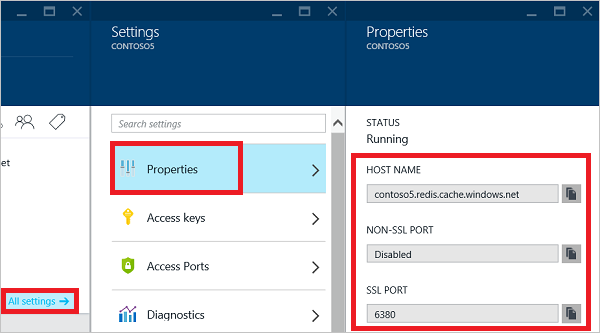
return lazyConnection.Value;

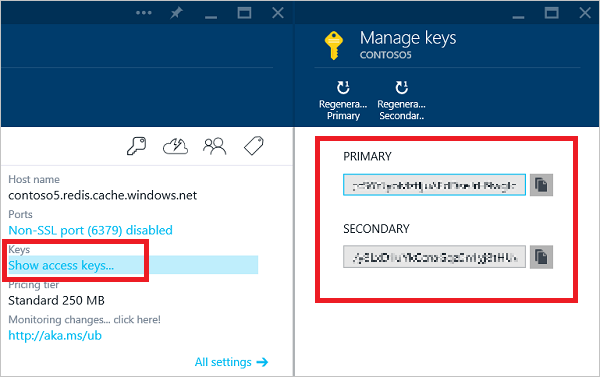
}

}

For more information on advanced connection configuration options, see [StackExchange.Redis configuration model](http://github.com/StackExchange/StackExchange.Redis/blob/master/Docs/Configuration.md).

The cache endpoint and keys can be obtained from the **Redis Cache** blade for your cache instance.

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-properties.png)

[](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/media/cache-dotnet-how-to-use-azure-redis-cache/redis-cache-manage-keys.png)

Once the connection is established, return a reference to the redis cache database by calling the ConnectionMultiplexer.GetDatabase method. The object returned from the GetDatabase method is a lightweight pass-through object and does not need to be stored.

// Connection refers to a property that returns a ConnectionMultiplexer

// as shown in the previous example.

IDatabase cache = Connection.GetDatabase();

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

// Simple put of integral data types into the cache

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

cache.StringSet("key2", 25);

// Simple get of data types from the cache

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

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

Now that you know how to connect to an Azure Redis Cache instance and return a reference to the cache database, let's take a look at working with the cache.

**Add and retrieve objects from the cache**

Items can be stored in and retrieved from a cache by using the StringSet and StringGet methods.

// If key1 exists, it is overwritten.

cache.StringSet("key1", "value1");

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

Redis stores most data as Redis strings, but these strings can contain many types of data, including serialized binary data, which can be used when storing .NET objects in the cache.

When calling StringGet, if the object exists, it is returned, and if it does not, null is returned. In this case you can retrieve the value from the desired data source and store it in the cache for subsequent use. This is known as the cache-aside pattern.

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

if (value == null)

{

// The item keyed by "key1" is not in the cache. Obtain

// it from the desired data source and add it to the cache.

value = GetValueFromDataSource();

cache.StringSet("key1", value);

}

To specify the expiration of an item in the cache, use the TimeSpan parameter of StringSet.

cache.StringSet("key1", "value1", TimeSpan.FromMinutes(90));

**Work with .NET objects in the cache**

Azure Redis Cache can cache .NET objects as well as primitive data types, but before a .NET object can be cached it must be serialized. This is the responsibility of the application developer, and gives the developer flexibility in the choice of the serializer.

One simple way to serialize objects is to use the JsonConvert serialization methods in [Newtonsoft.Json.NET](https://www.nuget.org/packages/Newtonsoft.Json/8.0.1-beta1) and serialize to and from JSON. The following example shows a get and set using an Employee object instance.

class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public Employee(int EmployeeId, string Name)

{

this.Id = EmployeeId;

this.Name = Name;

}

}

// Store to cache

cache.StringSet("e25", JsonConvert.SerializeObject(new Employee(25, "Clayton Gragg")));

// Retrieve from cache

Employee e25 = JsonConvert.DeserializeObject<Employee>(cache.StringGet("e25"));

**Next Steps**

Now that you've learned the basics, follow these links to learn more about Azure Redis Cache.

* Check out the ASP.NET providers for Azure Redis Cache.
  + [Azure Redis Session State Provider](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-aspnet-session-state-provider.md)
  + [Azure Redis Cache ASP.NET Output Cache Provider](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-aspnet-output-cache-provider.md)
* [Enable cache diagnostics](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-monitor.md#enable-cache-diagnostics) so you can [monitor](https://github.com/rgl/azure-content/blob/master/articles/redis-cache/cache-how-to-monitor.md) the health of your cache. You can view the metrics in the Azure Portal and you can also [download and review](https://github.com/rustd/RedisSamples/tree/master/CustomMonitoring) them using the tools of your choice.
* Check out the [StackExchange.Redis cache client documentation](http://github.com/StackExchange/StackExchange.Redis" \l "documentation).
  + Azure Redis Cache can be accessed from many Redis clients and development languages. For more information, see <http://redis.io/clients>.
  + Azure Redis Cache can also be used with services such as Redsmin. For more information, see [How to retrieve an Azure Redis connection string and use it with Redsmin](https://redsmin.uservoice.com/knowledgebase/articles/485711-how-to-connect-redsmin-to-azure-redis-cache).
* See the [redis](http://redis.io/documentation) documentation and read about [redis data types](http://redis.io/topics/data-types) and [a fifteen minute introduction to Redis data types](http://redis.io/topics/data-types-intro).