

Microsoft Azure Cloud Services



Authorized & published by Summitworks Technologies Inc

Agenda

- **Azure services**
 - Data services
 - Azure Storage
 - Table
 - Queue

Azure Table Storage

Azure Table storage

Azure Table storage is a service that stores structured NoSQL data in the cloud, providing a key/attribute store with a schemaless design. Because Table storage is schemaless, it's easy to adapt your data as the needs of your application evolve. Access to Table storage data is fast and cost-effective for many types of applications, and is typically lower in cost than traditional SQL for similar volumes of data.

You can use Table storage to store flexible datasets like user data for web applications, address books, device information, or other types of metadata your service requires. You can store any number of entities in a table, and a storage account may contain any number of tables, up to the capacity limit of the storage account.

Azure Table storage

What is Table storage

Azure Table storage stores large amounts of structured data. The service is a NoSQL data store which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of Table storage include:

- Storing TBs of structured data capable of serving web scale applications
- Storing datasets that don't require complex joins, foreign keys, or stored procedures and can be denormalized for fast access
- Quickly querying data using a clustered index
- Accessing data using the OData protocol and LINQ queries with WCF Data Service .NET Libraries

You can use Table storage to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

Azure Table storage

Table storage concepts

Table storage contains the following components:

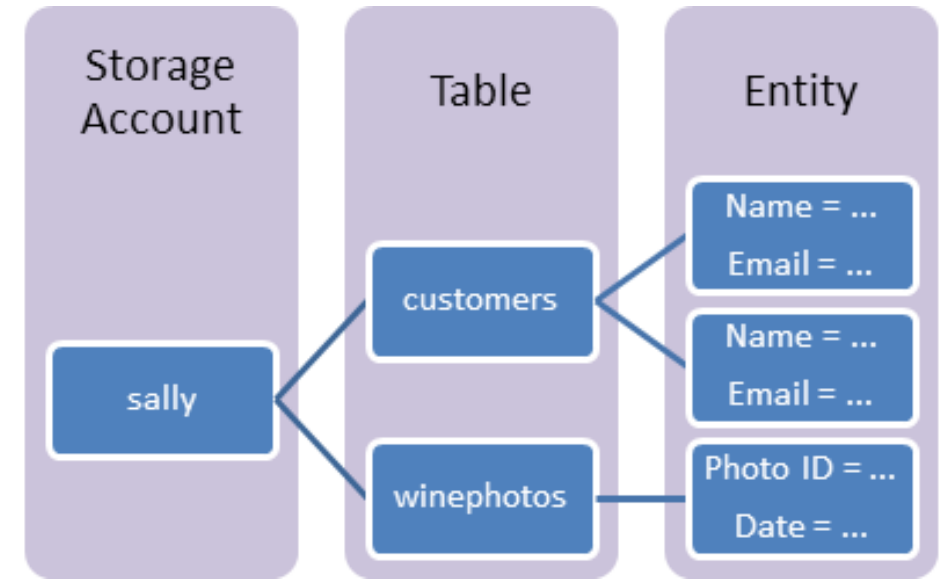
- URL format

Azure Table Storage accounts use this format:

<http://<storage account>.table.core.windows.net/<table>>

Azure Cosmos DB Table API accounts use this format:

<http://<storage account>.table.cosmosdb.azure.com/<table>>



Azure Table storage

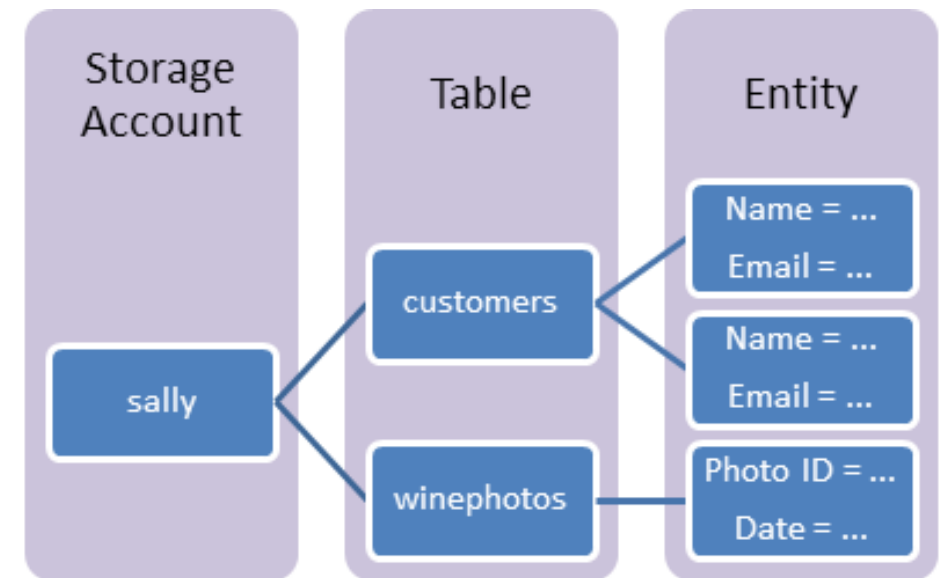
- **Accounts**

All access to Azure Storage is done through a storage account. See [Azure Storage Scalability and Performance Targets](#) for details about storage account capacity.

All access to Azure Cosmos DB is done through a Table API account. See [Create a Table API account](#) for details creating a Table API account.

- **Table**

A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties.



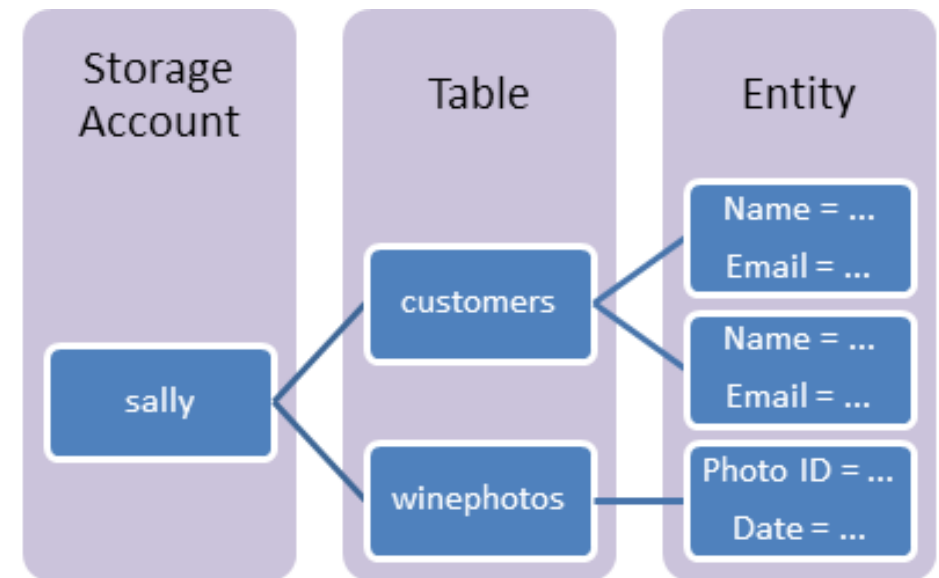
Azure Table storage

- **Entity**

An entity is a set of properties, similar to a database row. An entity in Azure Storage can be up to 1MB in size. An entity in Azure Cosmos DB can be up to 2MB in size.

- **Properties:**

A property is a name-value pair. Each entity can include up to 252 properties to store data. Each entity also has three system properties that specify a partition key, a row key, and a timestamp. Entities with the same partition key can be queried more quickly, and inserted/updated in atomic operations. An entity's row key is its unique identifier within a partition.

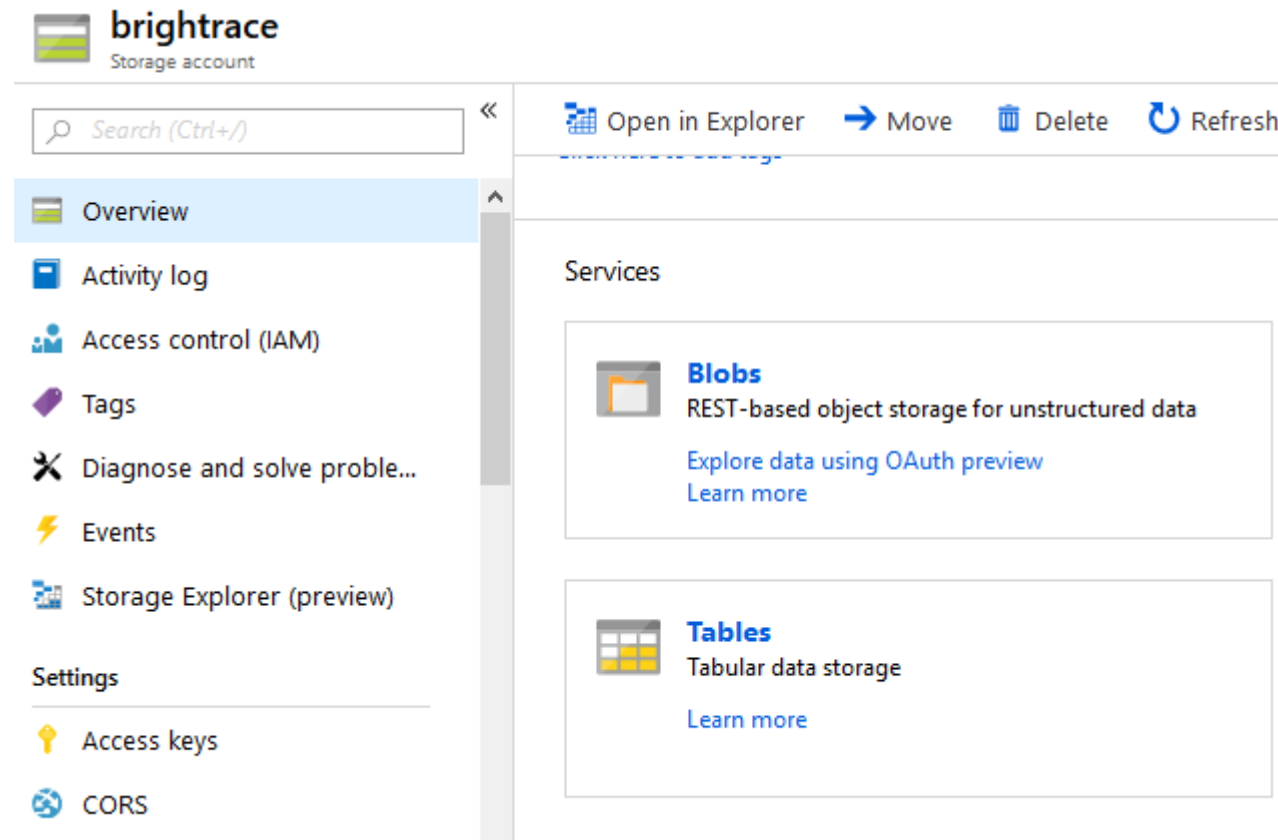


Create an Azure Storage table (Portal)

Add a table

You can now use Table service in the Azure portal to create a table.

1. Click Overview > Tables.



Create an Azure Storage table (Portal)

2. Click **+ Table**.
3. Type a name for your table in the **Table name** box, then click **OK**.

+ Table Refresh Delete tables

Add table

* Table name

myTable ✓

OK Cancel

+ Table Refresh Delete tables

i Check out premium Table experience with Azure Cosmos DB →

Storage account: [brightrace](#)

Search tables by prefix

TABLE	URL
myTable	https://brightrace.table.core.windows.net/myTable

Azure Table Storage in .NET

- Download the project from:

<https://github.com/Azure-Samples/storage-table-dotnet-getting-started/archive/master.zip>

To run the sample using the Storage Service:

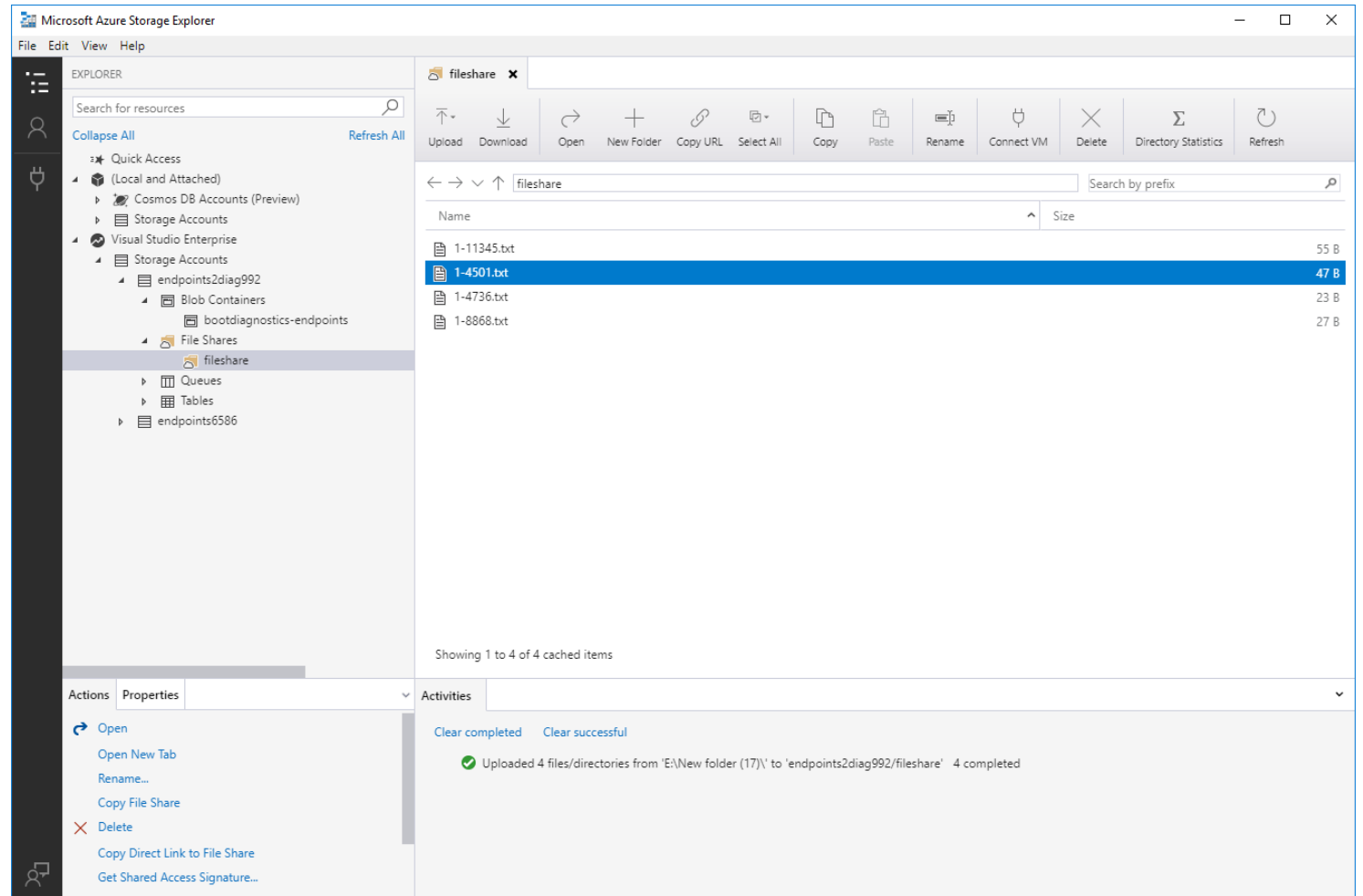
- Go to your Azure Storage account in the Azure Portal and under "SETTINGS" click on "Access keys". Copy either key1 or key2's "CONNECTION STRING".
- Open the app.config file and comment out the connection string for the emulator (UseDevelopmentStorage=True) and uncomment the connection string for the storage service (AccountName=[]...)
- Load the project in Visual Studio
- Run

Get started with Storage Explorer

Azure Storage Explorer is a standalone app that enables you to easily work with Azure Storage data on Windows, macOS, and Linux. In this article, you learn several ways of connecting to and managing your Azure storage accounts.

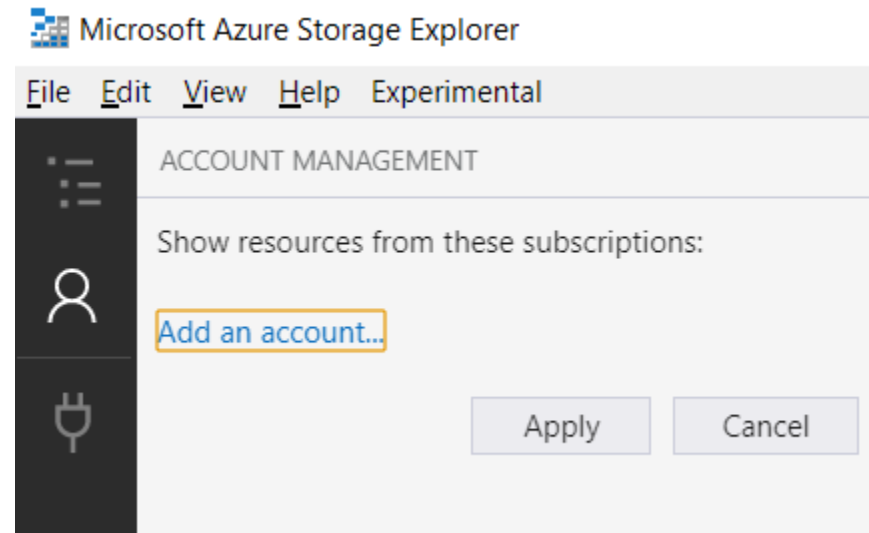
Get Storage Explorer from:

<https://azure.microsoft.com/en-us/features/storage-explorer/>



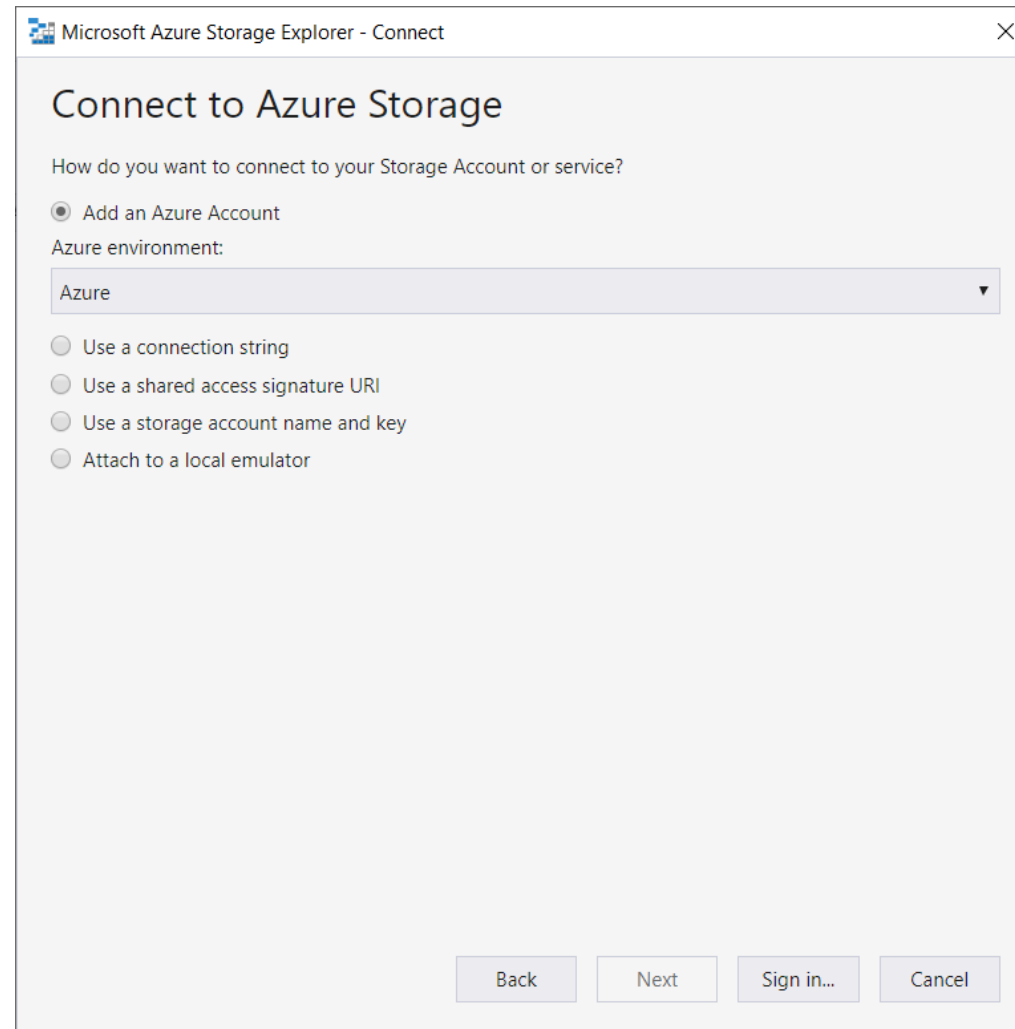
Connect to a storage account or service

1. In Storage Explorer, select **Manage Accounts** to go to the **Account Management Panel**.
2. The left pane now displays all the Azure accounts you've signed in to. To connect to another account, select **Add an account**



Connect to a storage account or service

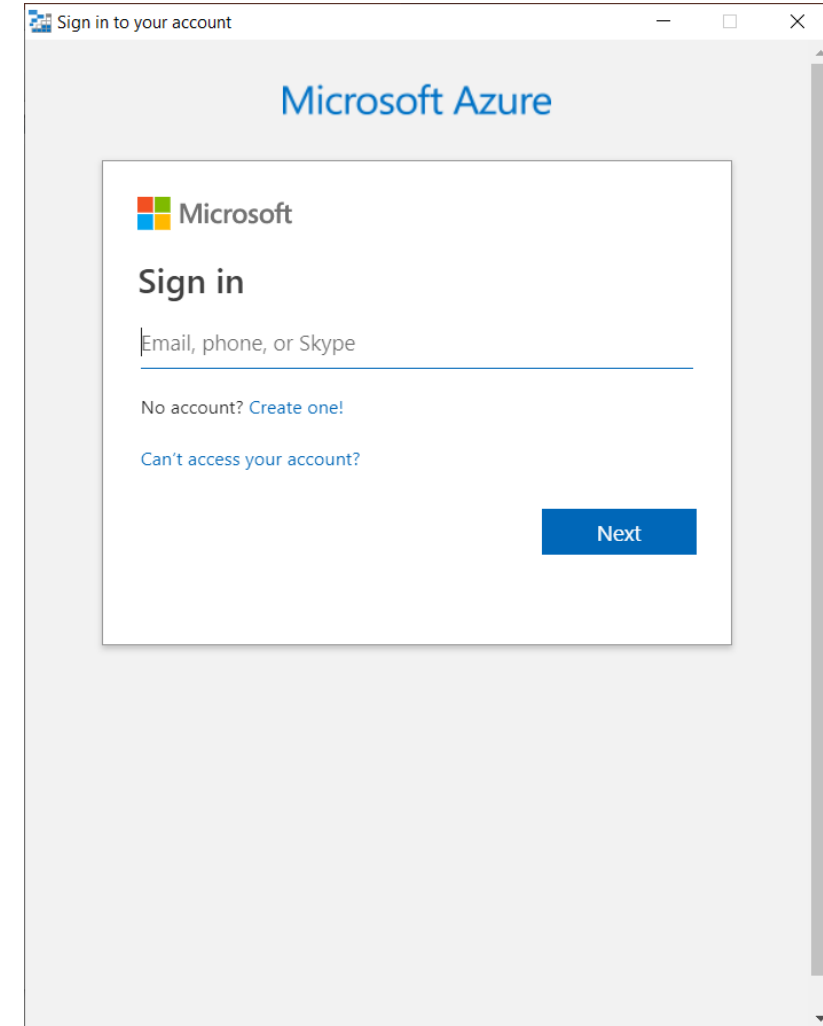
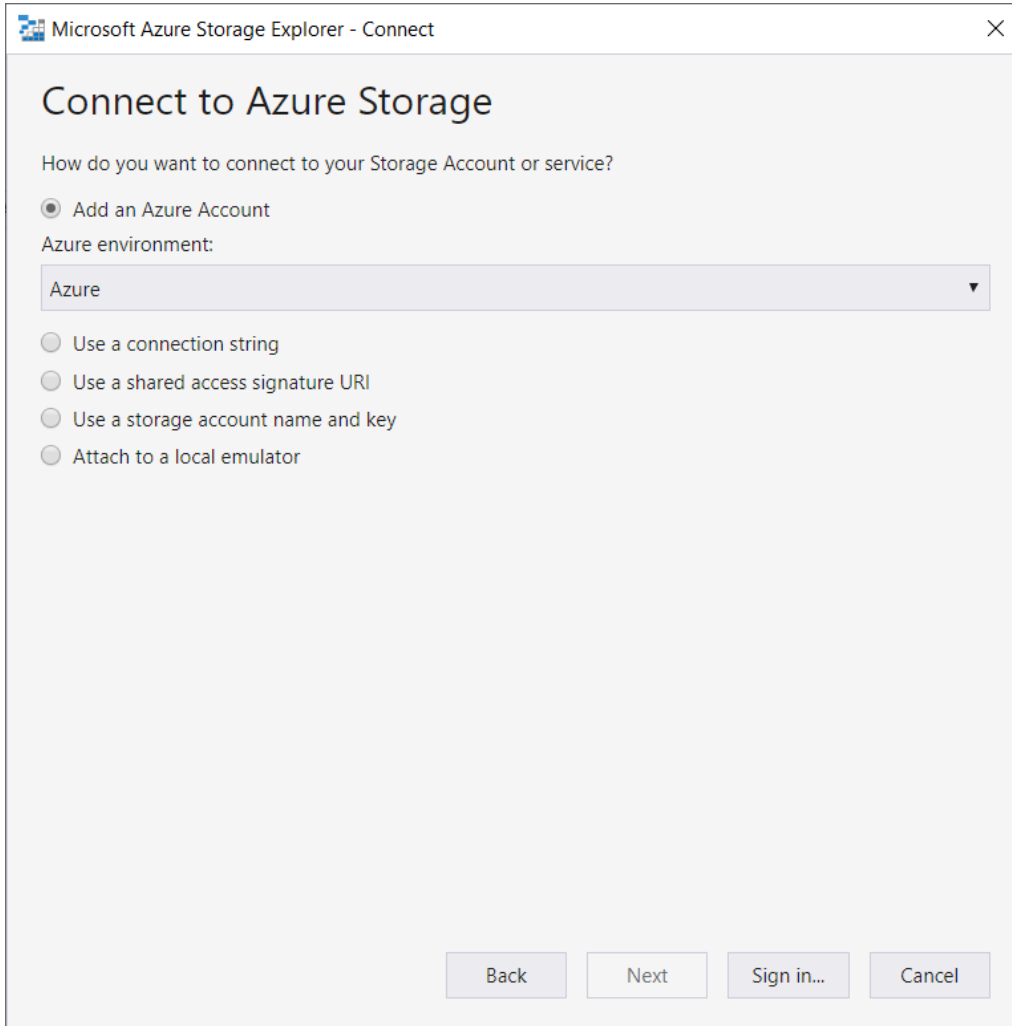
Click “Add an Azure Account”, then click next.



The screenshot shows a dialog box titled "Microsoft Azure Storage Explorer - Connect". The main heading is "Connect to Azure Storage". Below this, it asks "How do you want to connect to your Storage Account or service?". There are five radio button options: "Add an Azure Account" (which is selected), "Use a connection string", "Use a shared access signature URI", "Use a storage account name and key", and "Attach to a local emulator". Under the "Add an Azure Account" option, there is a label "Azure environment:" followed by a dropdown menu currently showing "Azure". At the bottom of the dialog, there are four buttons: "Back", "Next", "Sign in...", and "Cancel".

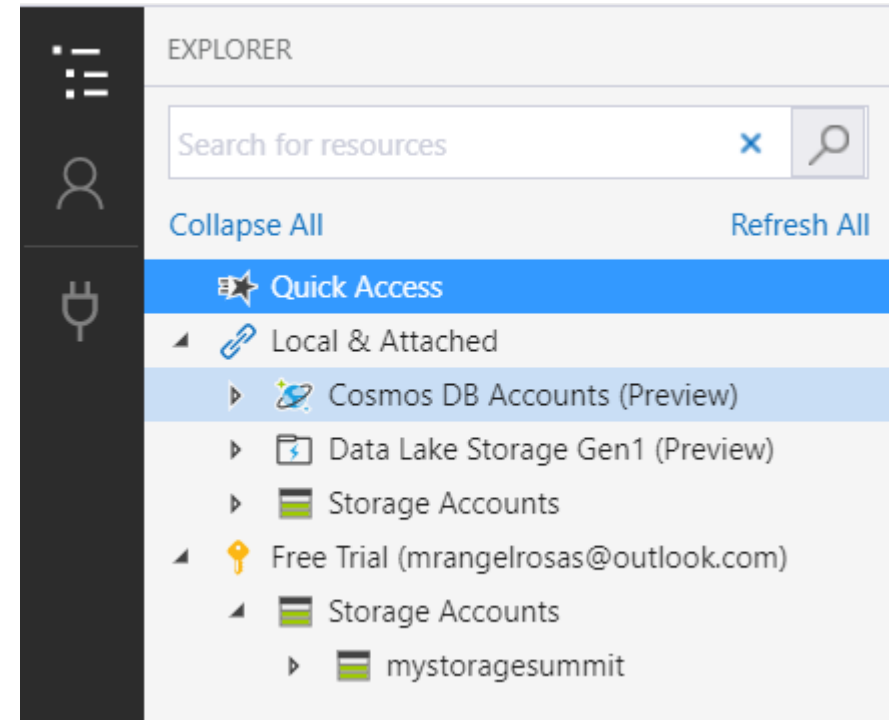
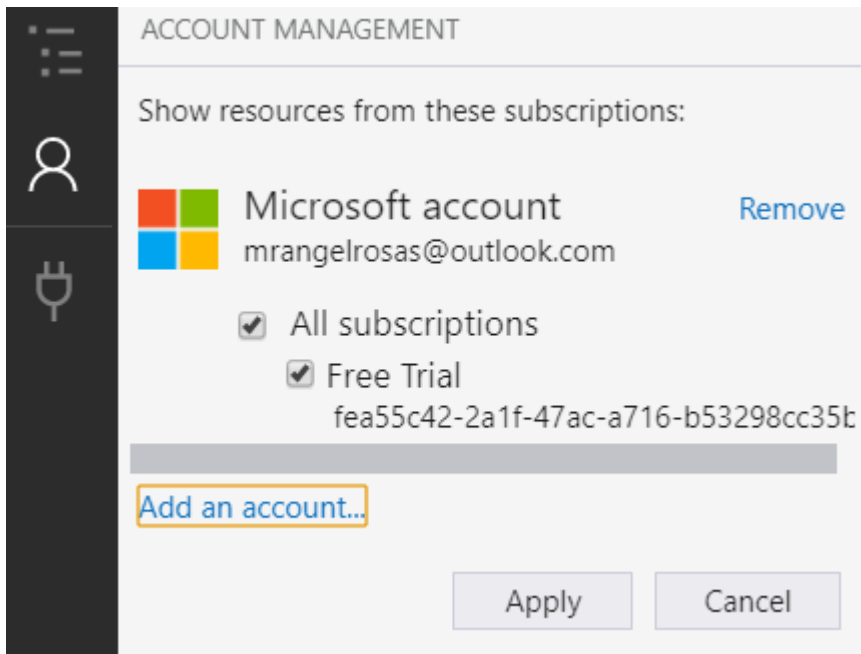
Connect to a storage account or service

3. Click “Add an Azure Account”, then click “Sign in”.



Connect to a storage account or service

4. After you successfully sign in with an Azure account, the account and the Azure subscriptions associated with that account are added to the left pane. Select the Azure subscriptions that you want to work with, and then select **Apply**.



Connect to a storage account or service

The left pane displays the storage accounts associated with the selected Azure subscriptions.

The screenshot shows the Microsoft Azure Storage Explorer interface. The left pane displays the Explorer view with a search bar and a tree structure. The tree shows the following hierarchy:

- Local & Attached
- Free Trial (mrangelrosas@outlook.com)
 - Storage Accounts
 - mystoragesummit
 - Blob Containers
 - File Shares
 - Queues
 - Tables
 - \$MetricsHourPrimaryTransactionsBlob
 - \$MetricsHourPrimaryTransactionsFile
 - \$MetricsHourPrimaryTransactionsQueue
 - \$MetricsHourPrimaryTransactionsTable
 - \$MetricsHourSecondaryTransactionsTable
 - \$MetricsMinutePrimaryTransactionsTable
 - \$MetricsMinuteSecondaryTransactionsTable
 - demo7019a

The right pane shows the selected table, \$MetricsHourPrimaryTransactionsTable, with the following data:

PartitionKey	RowKey	Timestamp	Email	PhoneNun
Smith	0050	2019-01-09T01:17:41.429Z	0050@contoso.co...	425-555-
Smith	0000	2019-01-09T01:17:41.420Z	0000@contoso.co...	425-555-
Smith	0002	2019-01-09T01:17:41.421Z	0002@contoso.co...	425-555-
Smith	0003	2019-01-09T01:17:41.421Z	0003@contoso.co...	425-555-
Smith	0004	2019-01-09T01:17:41.421Z	0004@contoso.co...	425-555-
Smith	0005	2019-01-09T01:17:41.421Z	0005@contoso.co...	425-555-
Smith	0006	2019-01-09T01:17:41.421Z	0006@contoso.co...	425-555-
Smith	0007	2019-01-09T01:17:41.421Z	0007@contoso.co...	425-555-
Smith	0008	2019-01-09T01:17:41.422Z	0008@contoso.co...	425-555-
Smith	0009	2019-01-09T01:17:41.422Z	0009@contoso.co...	425-555-
Smith	0010	2019-01-09T01:17:41.422Z	0010@contoso.co...	425-555-

Showing 1 to 100 of 100 cached items

The bottom pane shows the Properties tab for the selected table, with the following information:

- URL: https://mystoragesummit.table.core.windows.net/demo7019a
- Type: Table

The Activities pane shows the following status:

- Clear completed
- Clear successful

Connect to a storage account or service

The left pane displays the storage accounts associated with the selected Azure subscriptions.

The screenshot shows the Microsoft Azure Storage Explorer interface. The left pane displays the Explorer view with a search bar and a tree structure. The tree shows the following hierarchy:

- Local & Attached
- Free Trial (mrangelrosas@outlook.com)
 - Storage Accounts
 - mystoragesummit
 - Blob Containers
 - File Shares
 - Queues
 - Tables
 - \$MetricsHourPrimaryTransactionsBlob
 - \$MetricsHourPrimaryTransactionsFile
 - \$MetricsHourPrimaryTransactionsQueue
 - \$MetricsHourPrimaryTransactionsTable
 - \$MetricsHourSecondaryTransactionsTable
 - \$MetricsMinutePrimaryTransactionsTable
 - \$MetricsMinuteSecondaryTransactionsTable
 - demo7019a

The right pane shows the selected table, \$MetricsHourPrimaryTransactionsTable, with the following data:

PartitionKey	RowKey	Timestamp	Email	PhoneNun
Smith	0050	2019-01-09T01:17:41.429Z	0050@contoso.co...	425-555-
Smith	0000	2019-01-09T01:17:41.420Z	0000@contoso.co...	425-555-
Smith	0002	2019-01-09T01:17:41.421Z	0002@contoso.co...	425-555-
Smith	0003	2019-01-09T01:17:41.421Z	0003@contoso.co...	425-555-
Smith	0004	2019-01-09T01:17:41.421Z	0004@contoso.co...	425-555-
Smith	0005	2019-01-09T01:17:41.421Z	0005@contoso.co...	425-555-
Smith	0006	2019-01-09T01:17:41.421Z	0006@contoso.co...	425-555-
Smith	0007	2019-01-09T01:17:41.421Z	0007@contoso.co...	425-555-
Smith	0008	2019-01-09T01:17:41.422Z	0008@contoso.co...	425-555-
Smith	0009	2019-01-09T01:17:41.422Z	0009@contoso.co...	425-555-
Smith	0010	2019-01-09T01:17:41.422Z	0010@contoso.co...	425-555-

Showing 1 to 100 of 100 cached items

The bottom pane shows the Properties tab for the selected table, with the following information:

- URL: https://mystoragesummit.table.core.windows.net/demo7019a
- Type: Table

The Activities pane shows the following status:

- Clear completed
- Clear successful

Azure Queue Storage

What is Queue Storage?

Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size, and a queue can contain millions of messages, up to the total capacity limit of a storage account.

Common uses of Queue storage include:

- Creating a backlog of work to process asynchronously
- Passing messages from an Azure web role to an Azure worker role



What is Queue Storage?

Queue Service Concepts

The Queue service contains the following components:

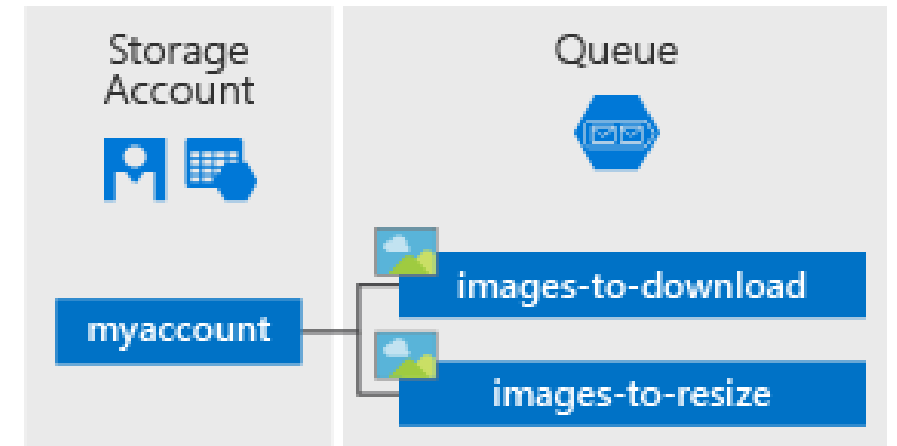
- **URL format**

Queues are addressable using the following URL format:

<http://<storage account>.queue.core.windows.net/<queue>>

The following URL addresses a queue in the diagram:

<http://myaccount.queue.core.windows.net/images-to-download>



What is Queue Storage?

- **Storage Account**

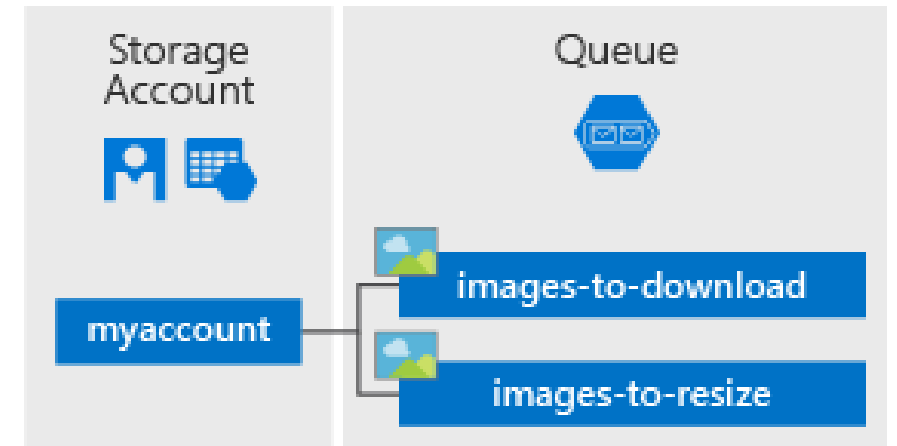
All access to Azure Storage is done through a storage account. See [Azure Storage Scalability and Performance Targets](#) for details about storage account capacity.

- **Queue**

A queue contains a set of messages. All messages must be in a queue. Note that the queue name must be all lowercase. For information on naming queues, see [Naming Queues and Metadata](#).

- **Message**

A message, in any format, of up to 64 KB. The maximum time that a message can remain in the queue is 7 days.

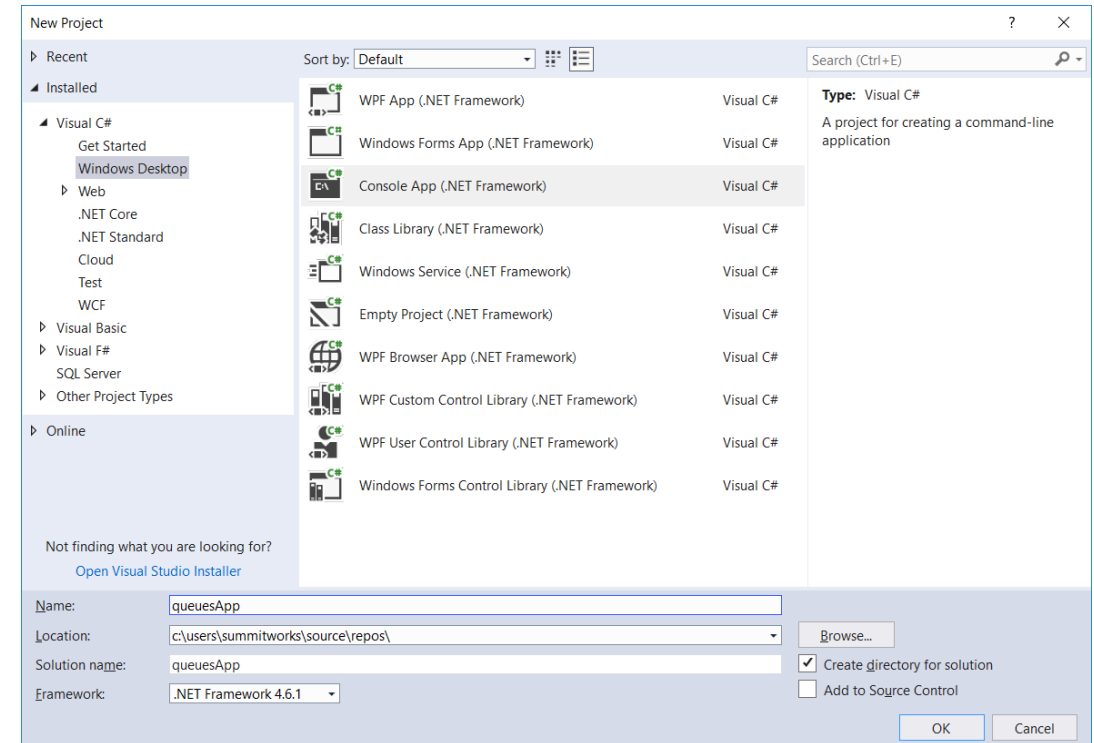


Queue Storage using .NET

Create a Windows console application project

In Visual Studio, create a new Windows console application. The following steps show you how to create a console application in Visual Studio 2017. The steps are similar in other versions of Visual Studio.

1. Select **File > New > Project**.
2. Select **Installed > Templates > Visual C# > Windows Desktop**.
3. Select **Console App (.NET Framework)**.
4. In the **Name** field, enter a name for your application.
5. Select **OK**.



Queue Storage using .NET

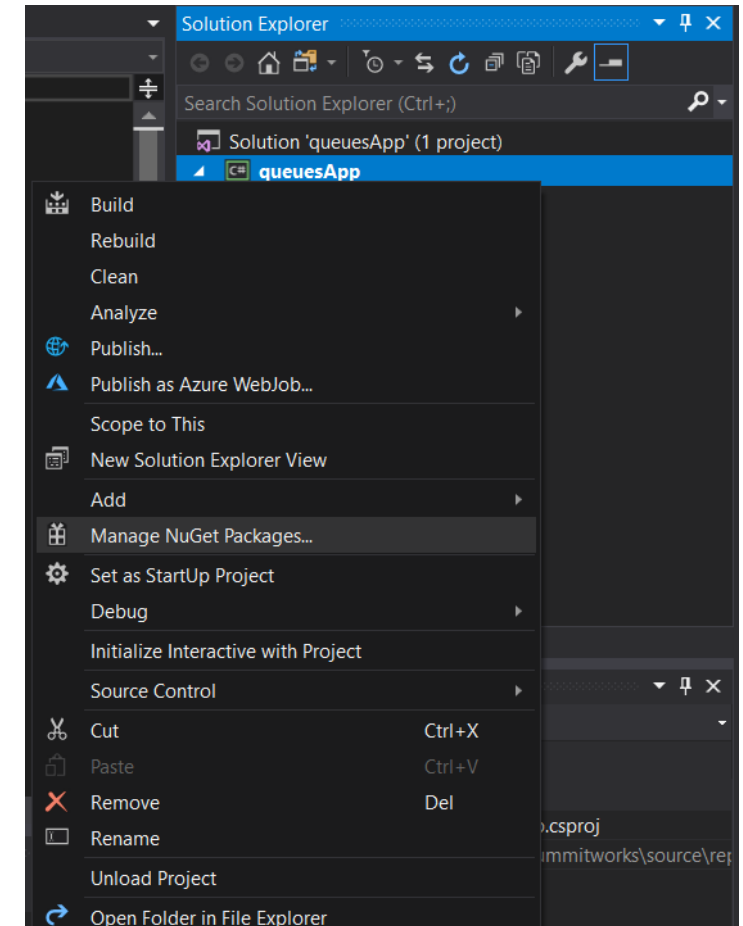
Use NuGet to install the required packages

You need to reference this two packages in your project:

- *Microsoft Azure Storage Client Library* for .NET
- *Microsoft Azure Configuration Manager library* for .NET

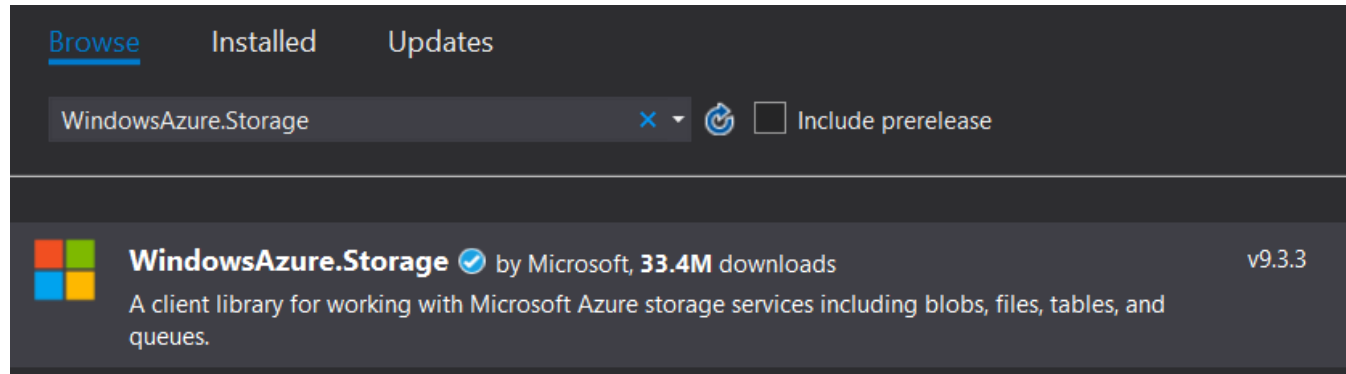
You can use NuGet to obtain both packages.

1. Right-click your project in **Solution Explorer**, and choose **Manage NuGet Packages**.

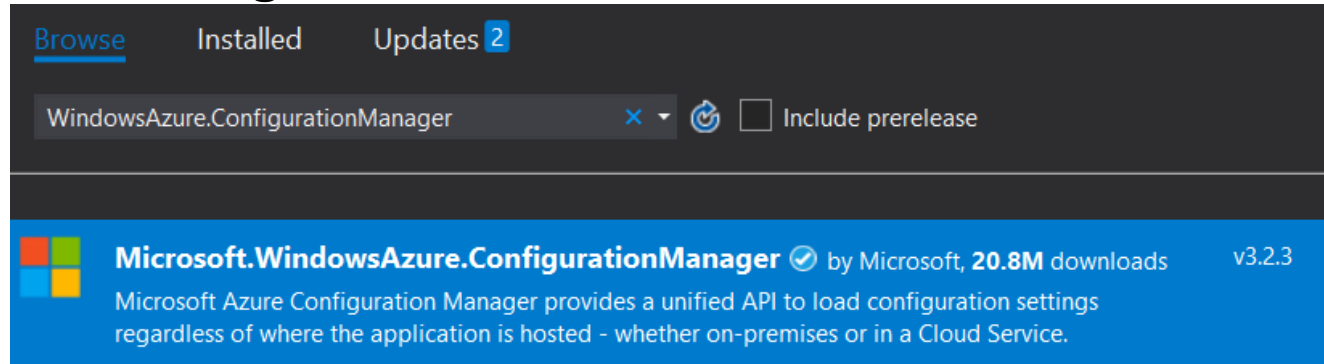


Queue Storage using .NET

2. Search online for "WindowsAzure.Storage", and select **Install** to install the Storage Client Library and its dependencies.



3. Search online for "WindowsAzure.ConfigurationManager", and select **Install** to install the Azure Configuration Manager.



Queue Storage using .NET

Determine your target environment

You have two environment options for running the examples in this guide:

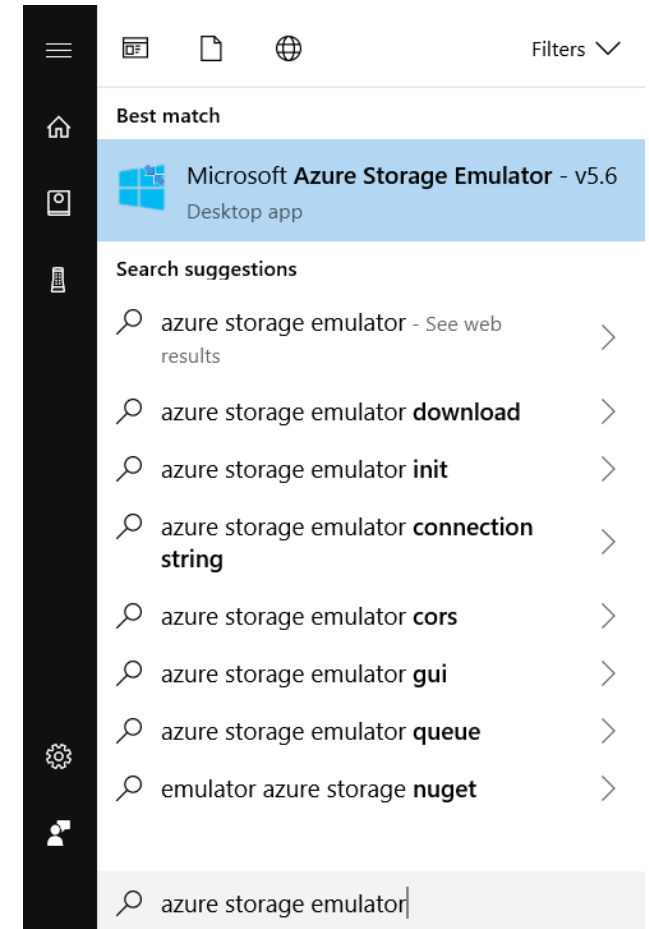
- You can run your code against an Azure Storage account in the cloud.
- You can run your code against the Azure storage emulator. The storage emulator is a local environment that emulates an Azure Storage account in the cloud. The emulator is a free option for testing and debugging your code while your application is under development. The emulator uses a well-known account and key. For more information.

Queue Storage using .NET

Start and initialize the storage emulator

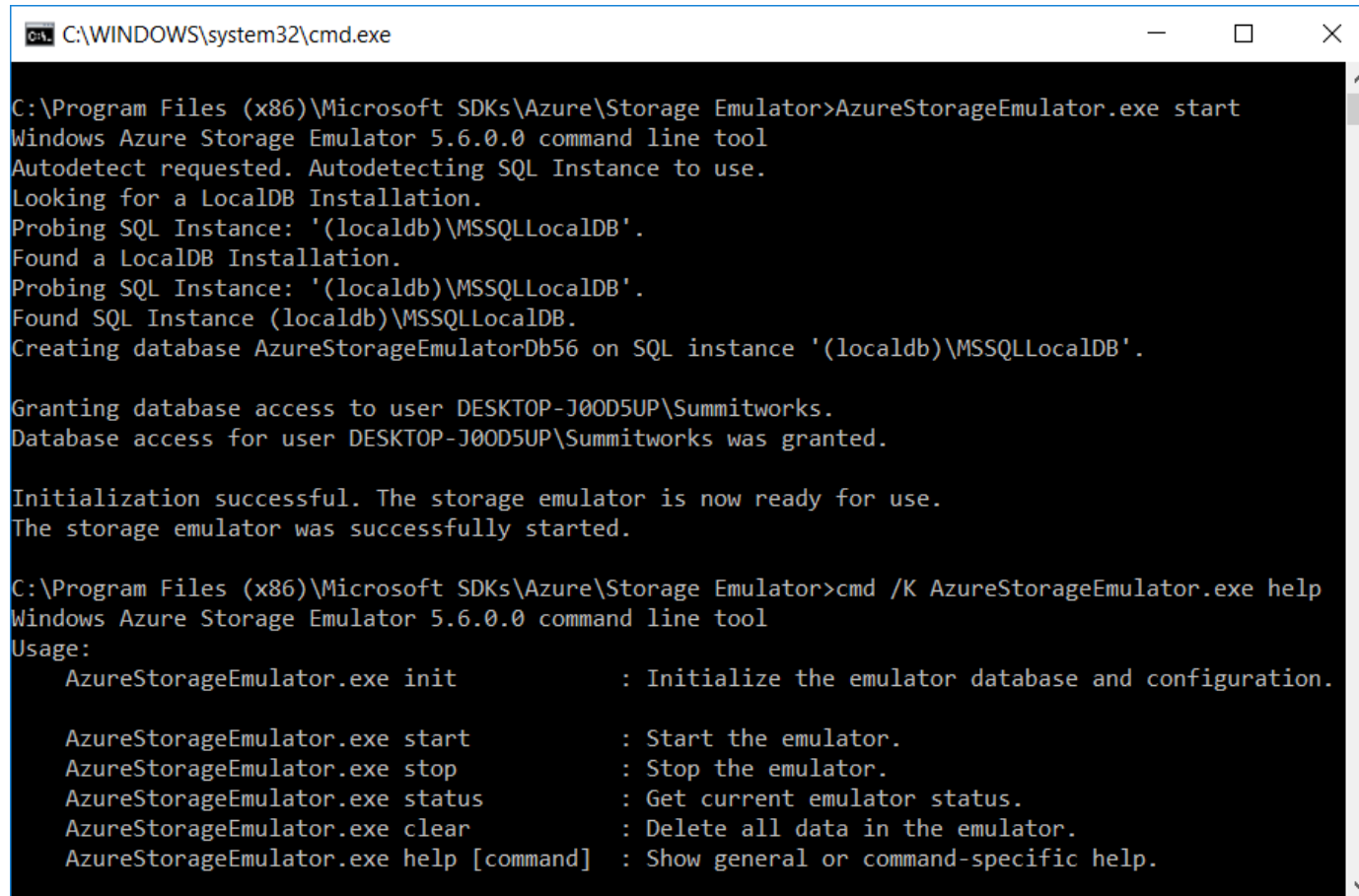
To start the Azure storage emulator:

1. Select the Start button or press the Windows key.
2. Begin typing Azure Storage Emulator.
3. Select the emulator from the list of displayed applications.



Queue Storage using .NET

When the storage emulator starts, a Command Prompt window will appear. You can use this console window to start and stop the storage emulator, clear data, get status, and initialize the emulator.

A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the execution of the "AzureStorageEmulator.exe start" command. The output text is as follows:

```
C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>AzureStorageEmulator.exe start
Windows Azure Storage Emulator 5.6.0.0 command line tool
Autodetect requested. Autodetecting SQL Instance to use.
Looking for a LocalDB Installation.
Probing SQL Instance: '(localdb)\MSSQLLocalDB'.
Found a LocalDB Installation.
Probing SQL Instance: '(localdb)\MSSQLLocalDB'.
Found SQL Instance (localdb)\MSSQLLocalDB.
Creating database AzureStorageEmulatorDb56 on SQL instance '(localdb)\MSSQLLocalDB'.

Granting database access to user DESKTOP-J00D5UP\Summitworks.
Database access for user DESKTOP-J00D5UP\Summitworks was granted.

Initialization successful. The storage emulator is now ready for use.
The storage emulator was successfully started.

C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>cmd /K AzureStorageEmulator.exe help
Windows Azure Storage Emulator 5.6.0.0 command line tool
Usage:
    AzureStorageEmulator.exe init           : Initialize the emulator database and configuration.

    AzureStorageEmulator.exe start          : Start the emulator.
    AzureStorageEmulator.exe stop           : Stop the emulator.
    AzureStorageEmulator.exe status         : Get current emulator status.
    AzureStorageEmulator.exe clear          : Delete all data in the emulator.
    AzureStorageEmulator.exe help [command] : Show general or command-specific help.
```

Queue Storage using .NET

Configure your storage connection string

The Azure Storage Client Library for .NET supports using a storage connection string to configure endpoints and credentials for accessing storage services. The best way to maintain your storage connection string is in a configuration file.

To configure your connection string, open the app.config file from Solution Explorer in Visual Studio. Add the contents of the <appSettings> element shown below. Replace account-name with the name of your storage account, and account-key with your account access key:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
  </startup>
  <appSettings>
    <add key="StorageConnectionString" value="UseDevelopmentStorage=true" />
  </appSettings>
</configuration>
```

Queue Storage using .NET

Add using directives

Add the following using directives to the top of the *Program.cs* file:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure; // Namespace for CloudConfigurationManager
using Microsoft.WindowsAzure.Storage; // Namespace for CloudStorageAccount
using Microsoft.WindowsAzure.Storage.Queue; // Namespace for Queue storage types

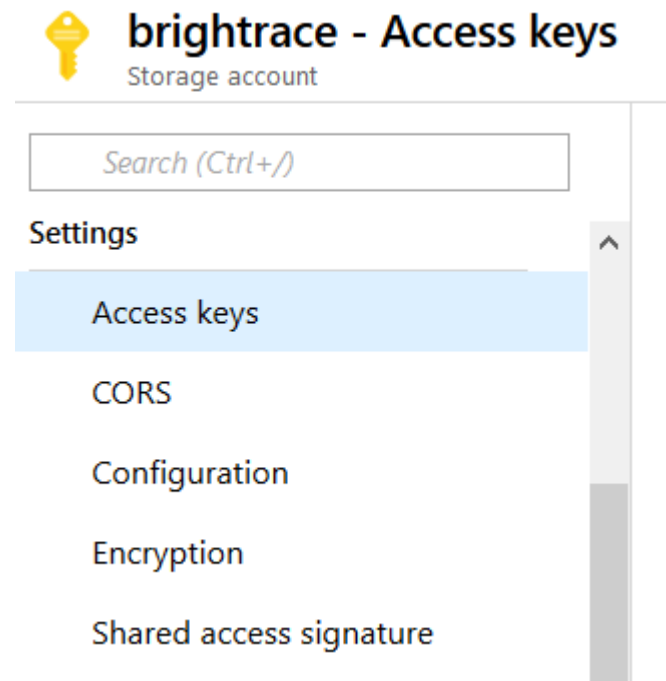
namespace queuesApp
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Queue Storage using .NET

Copy your credentials from the Azure portal

The sample code needs to authorize access to your storage account. To authorize, you provide the application with your storage account credentials in the form of a connection string. To view your storage account credentials:

1. Navigate to the Azure portal.
2. Locate your storage account.
3. In the **Settings** section of the storage account overview, select **Access keys**. Your account access keys appear, as well as the complete connection string for each key.
4. Find the **Connection string** value under **key1**, and click the **Copy** button to copy the connection string. You will add the connection string value to an environment variable in the next step.




Queue Storage using .NET

Copy your credentials from the Azure portal

The sample code needs to authorize access to your storage account. To authorize, you provide the application with your storage account credentials in the form of a connection string. To view your storage account credentials:

1. Navigate to the Azure portal.
2. Locate your storage account.
3. In the **Settings** section of the storage account overview, select **Access keys**. Your account access keys appear, as well as the complete connection string for each key.
4. Find the **Connection string** value under **key1**, and click the **Copy** button to copy the connection string. You will add the connection string value to an environment variable in the next step.

key1 

Key

uiBYn7ctP0Bkn6HcSUZqj8g6u8p9ty3cHvVOz1bYuNx0Yuc5PYP83ADgxsyF4DStV4fFGi0oLv8TIEyhS3wulw==



Connection string

DefaultEndpointsProtocol=https;AccountName=brighttrace;AccountKey=uiBYn7ctP0Bkn6HcSUZqj8g6u8p9ty3cHvVOz1...



Queue Storage using .NET

Parse the connection string

The Microsoft Azure Configuration Manager Library for .NET provides a class for parsing a connection string from a configuration file. The `CloudConfigurationManager` class parses configuration settings regardless of whether the client application is running on the desktop, on a mobile device, in an Azure virtual machine, or in an Azure cloud service.

To reference the `CloudConfigurationManager` package, add the following using directive:

```
using Microsoft.Azure; // Namespace for CloudConfigurationManager
```

Here's an example that shows how to retrieve a connection string from a configuration file:

```
static void Main(string[] args)
{
    // Parse the connection string and return a reference to the storage account.
    CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
        CloudConfigurationManager.GetSetting("StorageConnectionString"));
}
```

Queue Storage using .NET

Create the Queue service client

The **CloudQueueClient** class enables you to retrieve queues stored in Queue storage. Here's one way to create the service client:

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

Now you are ready to write code that reads data from and writes data to Queue storage.

Queue Storage using .NET

Create a queue

This example shows how to create a queue if it does not already exist:

```
static void Main(string[] args)
{
    // Retrieve storage account from connection string.
    CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
        CloudConfigurationManager.GetSetting("StorageConnectionString"));

    // Create the queue client.
    CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

    // Retrieve a reference to a container.
    CloudQueue queue = queueClient.GetQueueReference("myqueue");

    // Create the queue if it doesn't already exist
    queue.CreateIfNotExists();
}
```

Queue Storage using .NET

Insert a message into a queue

To insert a message into an existing queue, first create a new **CloudQueueMessage**. Next, call the **AddMessage** method. A **CloudQueueMessage** can be created from either a **string** (in UTF-8 format) or a **byte** array. Here is code which creates a queue (if it doesn't exist) and inserts the message 'Hello, World':

```
// Retrieve storage account from connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the queue client.
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

// Retrieve a reference to a container.
CloudQueue queue = queueClient.GetQueueReference("myqueue");

// Create the queue if it doesn't already exist
queue.CreateIfNotExists();

// Create a message and add it to the queue.
CloudQueueMessage message = new CloudQueueMessage("Hello, World");
queue.AddMessage(message);
```

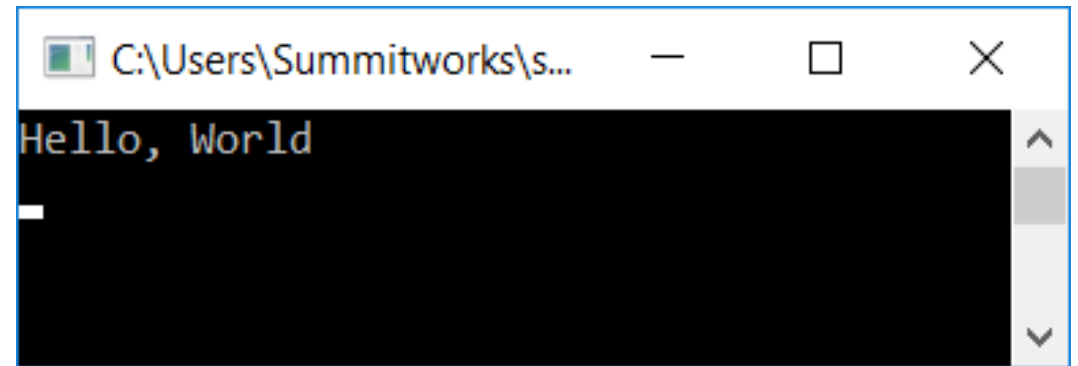
Queue Storage using .NET

Peek at the next message

You can peek at the message in the front of a queue without removing it from the queue by calling the **PeekMessage** method.

```
// Peek at the next message
CloudQueueMessage peekedMessage = queue.PeekMessage();

// Display message.
Console.WriteLine(peekedMessage.AsString);
```



Queue Storage using .NET

Change the contents of a queued message

You can change the contents of a message in-place in the queue. If the message represents a work task, you could use this feature to update the status of the work task. The following code updates the queue message with new contents, and sets the visibility timeout to extend another 5 seconds

```
// Retrieve a reference to a container.  
CloudQueue queue = queueClient.GetQueueReference("myqueue");  
  
// Get the message from the queue and update the message contents.  
CloudQueueMessage message = queue.GetMessage();  
message.SetMessageContent("Updated contents.");  
queue.UpdateMessage(message,  
    TimeSpan.FromSeconds(5.0), // Make it invisible for another 60 seconds.  
    MessageUpdateFields.Content | MessageUpdateFields.Visibility);
```

Queue Storage using .NET

De-queue the next message

Your code de-queues a message from a queue in two steps. When you call **GetMessage**, you get the next message in a queue. A message returned from **GetMessage** becomes invisible to any other code reading messages from this queue. By default, this message stays invisible for 30 seconds. To finish removing the message from the queue, you must also call **DeleteMessage**. This two-step process of removing a message assures that if your code fails to process a message due to hardware or software failure, another instance of your code can get the same message and try again. Your code calls **DeleteMessage** right after the message has been processed.

```
// Retrieve a reference to a container.  
CloudQueue queue = queueClient.GetQueueReference("myqueue");  
  
// Get the next message  
CloudQueueMessage retrievedMessage = queue.GetMessage();  
  
//Process the message in less than 30 seconds, and then delete the message  
queue.DeleteMessage(retrievedMessage);
```

Queue Storage using .NET

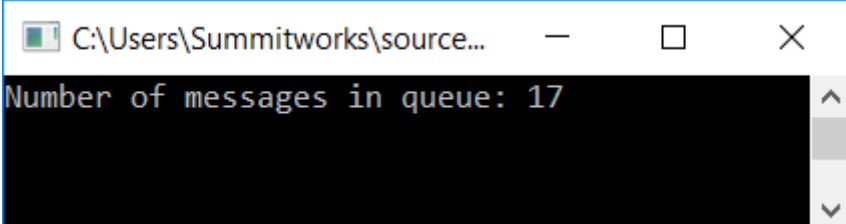
Get the queue length

You can get an estimate of the number of messages in a queue. The **FetchAttributes** method asks the Queue service to retrieve the queue attributes, including the message count. The **ApproximateMessageCount** property returns the last value retrieved by the **FetchAttributes** method, without calling the Queue service.

```
// Fetch the queue attributes.
queue.FetchAttributes();

// Retrieve the cached approximate message count.
int? cachedMessageCount = queue.ApproximateMessageCount;

// Display number of messages.
Console.WriteLine("Number of messages in queue: " + cachedMessageCount);
```



A screenshot of a Windows console window. The title bar shows the file path 'C:\Users\Summitworks\source...'. The console output displays the text 'Number of messages in queue: 17'.

Queue Storage using .NET

Delete a queue

To delete a queue and all the messages contained in it, call the **Delete** method on the queue object.

```
// Delete the queue.  
queue.Delete();  
  
// Display number of messages.  
Console.WriteLine("Number of messages in queue: " + cachedMessageCount);
```