

# Microsoft Azure Cloud Services



Authorized & published by Summitworks Technologies Inc

# Agenda

- **Azure services**
  - Data services
    - Azure Storage
      - Azure Blob
      - Files

# **Azure Data Services - Azure Storage**

# Azure Storage

## Introduction to Azure Storage

Azure Storage is Microsoft's cloud storage solution for modern data storage scenarios. Azure Storage offers a massively scalable object store for data objects, a file system service for the cloud, a messaging store for reliable messaging, and a NoSQL store. Azure Storage is:

- **Durable and highly available.** Redundancy ensures that your data is safe in the event of transient hardware failures. You can also opt to replicate data across datacenters or geographical regions for additional protection from local catastrophe or natural disaster. Data replicated in this way remains highly available in the event of an unexpected outage.
- **Secure.** All data written to Azure Storage is encrypted by the service. Azure Storage provides you with fine-grained control over who has access to your data.

# Azure Storage

- **Scalable.** Azure Storage is designed to be massively scalable to meet the data storage and performance needs of today's applications.
- **Managed.** Microsoft Azure handles maintenance and any critical problems for you.
- **Accessible.** Data in Azure Storage is accessible from anywhere in the world over HTTP or HTTPS. Microsoft provides SDKs for Azure Storage in a variety of languages -- .NET, Java, Node.js, Python, PHP, Ruby, Go, and others -- as well as a mature REST API. Azure Storage supports scripting in Azure PowerShell or Azure CLI. And the Azure portal and Azure Storage Explorer offer easy visual solutions for working with your data.



# Azure Storage Services

Azure Storage includes these data services:

- **Azure Blobs:** A massively scalable object store for text and binary data.
- **Azure Files:** Managed file shares for cloud or on-premises deployments.
- **Azure Queues:** A messaging store for reliable messaging between application components.
- **Azure Tables:** A NoSQL store for schemaless storage of structured data.

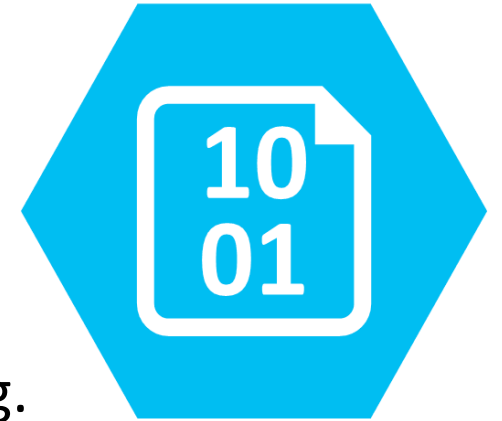
# Azure Blob Storage

# Blob Storage

Azure Blob storage is Microsoft's object storage solution for the cloud. Blob storage is optimized for storing massive amounts of unstructured data, such as text or binary data.

Blob storage is ideal for:

- Serving images or documents directly to a browser.
- Storing files for distributed access.
- Streaming video and audio.
- Storing data for backup and restore, disaster recovery, and archiving.
- Storing data for analysis by an on-premises or Azure-hosted service.

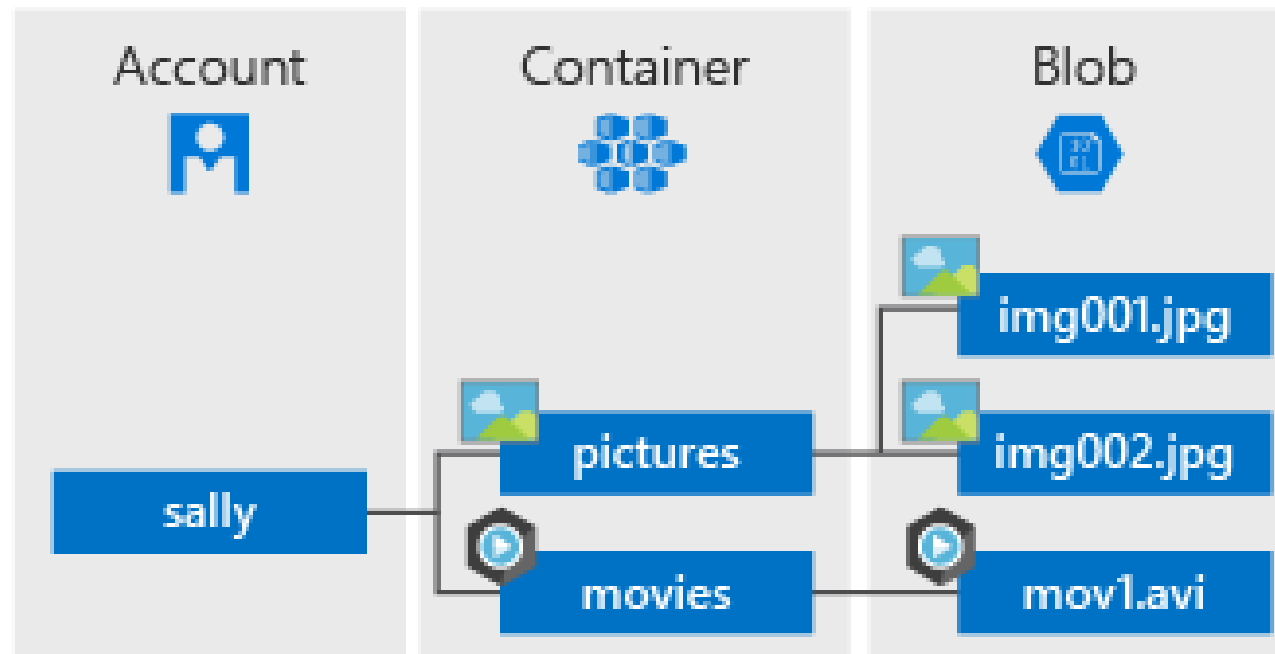


Objects in Blob storage can be accessed from anywhere in the world via HTTP or HTTPS. Users or client applications can access blobs via URLs, the Azure Storage REST API, Azure PowerShell, Azure CLI, or an Azure Storage client library. The storage client libraries are available for multiple languages, including .NET, Java, Node.js, Python, PHP, and Ruby.



# Blob Service Concepts

Blob storage exposes three resources: your storage account, the containers in the account, and the blobs in a container. The following diagram shows the relationship between these resources.



# Blob Service Concepts

## Storage account

All access to data objects in Azure Storage happens through a storage account.

## Container

A container organizes a set of blobs, similar to a folder in a file system. All blobs reside within a container. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.

## Blob

Azure Storage offers three types of blob:

- Block blobs
- Append blobs
- Page blobs

# Blob Service Concepts

## ***Block blobs***

Store text and binary data, up to about 4.7 TB. Block blobs are made up of blocks of data that can be managed individually.

## ***Append blobs***

Are made up of blocks like block blobs, but are optimized for append operations. Append blobs are ideal for scenarios such as logging data from virtual machines.

## ***Page blobs***

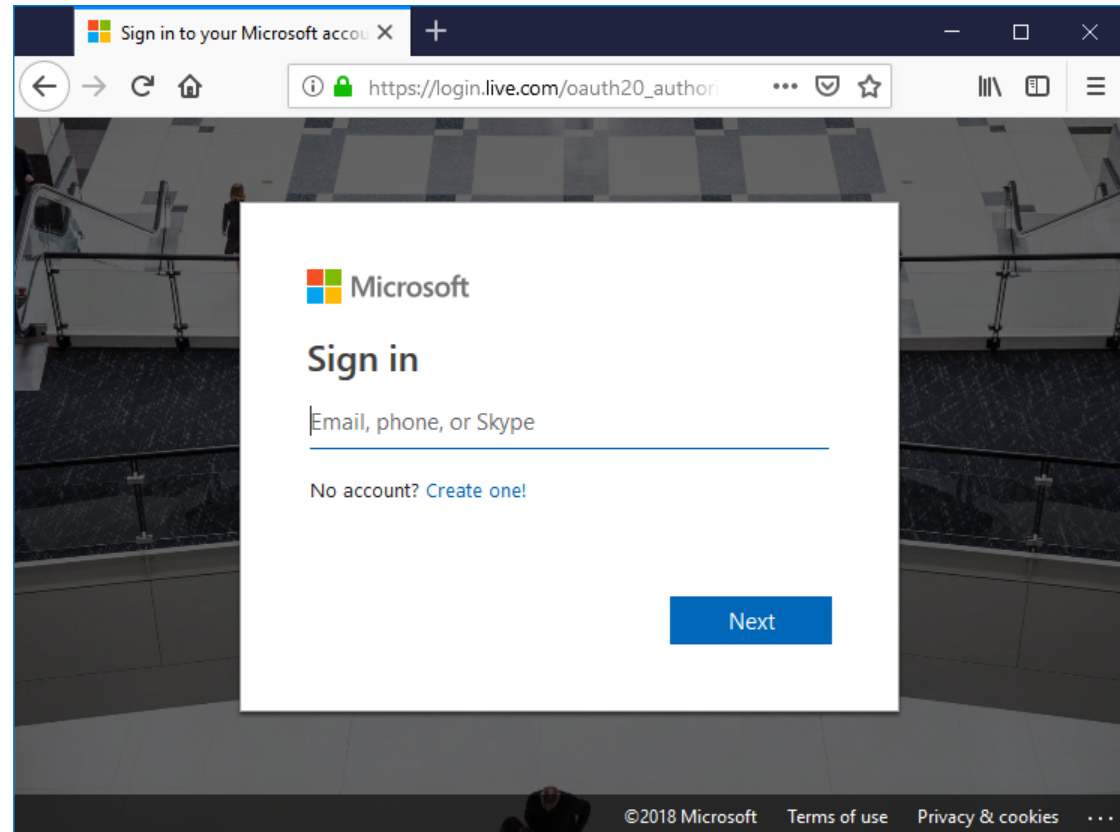
Store random access files up to 8 TB in size. Page blobs store the VHD files that back VMs.

All blobs reside within a container. A container is similar to a folder in a file system. You can further organize blobs into virtual directories and navigate them as you would a file system.

# Create a storage account (portal)

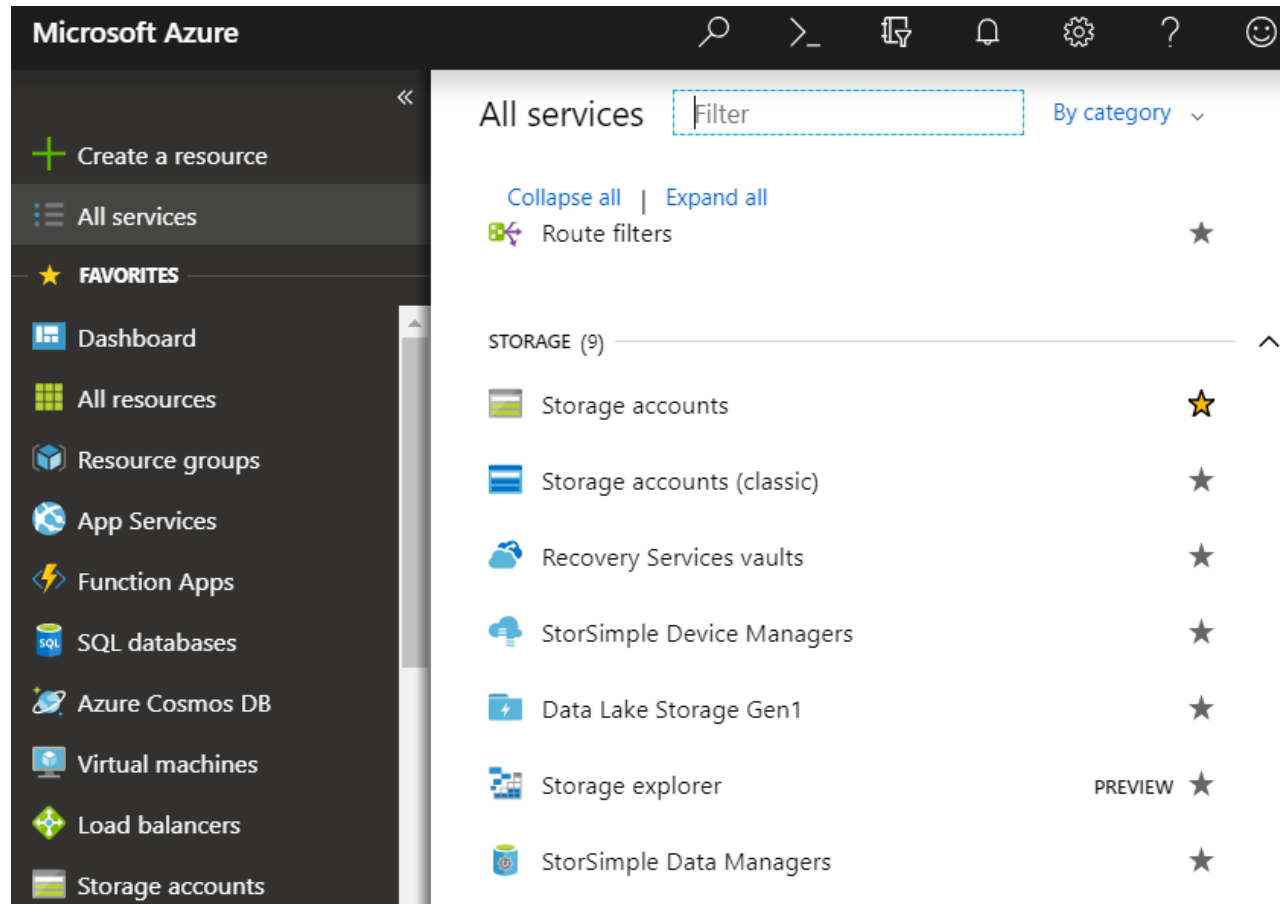
Sign in to the Azure portal at

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



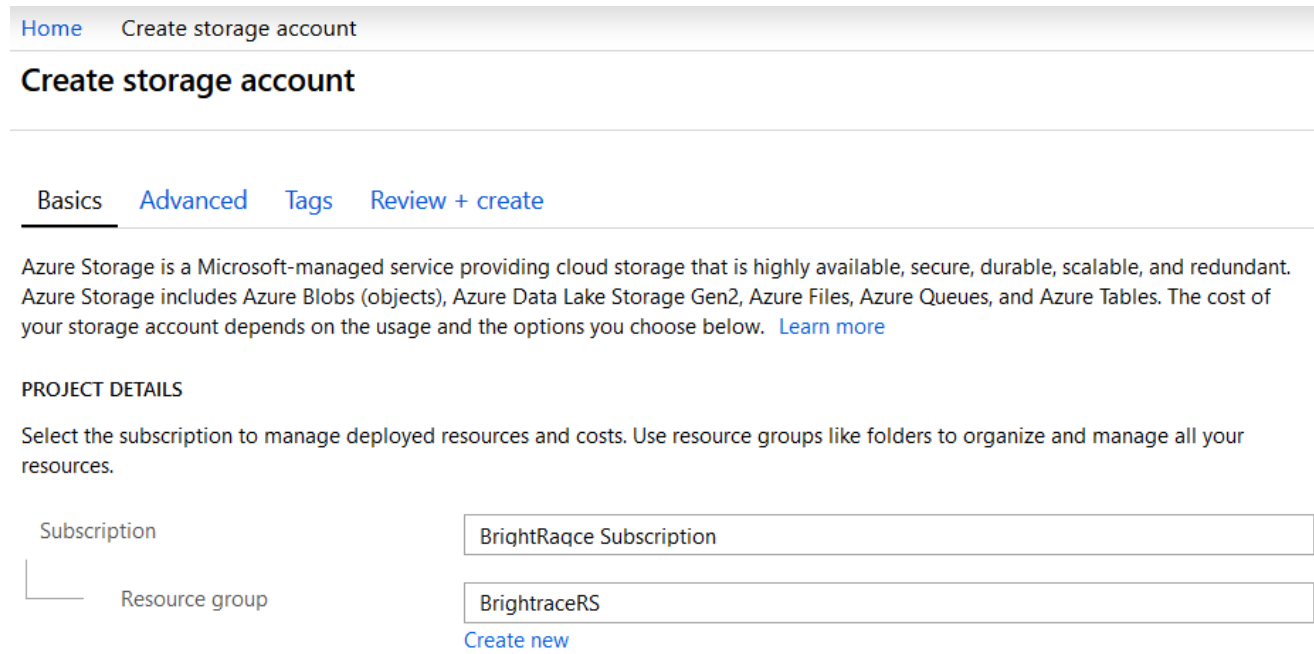
# Create a storage account (portal)

1. In the Azure portal, select **All services**. In the list of resources, type **Storage Accounts**. As you begin typing, the list filters based on your input. Select **Storage Accounts**.



# Create a storage account (portal)

2. On the **Storage Accounts** window that appears, choose **Add**.
3. Select the subscription in which to create the storage account.
4. Under the **Resource group** field, select **Create new**. Enter a name for your new resource group.



The screenshot shows the 'Create storage account' page in the Azure portal. At the top, there are navigation links for 'Home' and 'Create storage account'. The main heading is 'Create storage account'. Below this, there are tabs for 'Basics', 'Advanced', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The page contains a paragraph about Azure Storage, followed by a section titled 'PROJECT DETAILS'. This section includes instructions to select a subscription and resource group. There are two input fields: 'Subscription' with the value 'BrightRaqce Subscription' and 'Resource group' with the value 'BrightraceRS'. A 'Create new' link is visible below the resource group field.

[Home](#) [Create storage account](#)

## Create storage account

[Basics](#) [Advanced](#) [Tags](#) [Review + create](#)

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [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

Resource group

[Create new](#)

# Create a storage account (portal)

5. Next, enter a name for your storage account.
6. Select a location for your storage account, or use the default location.

## INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name	<input type="text" value="brightrace"/>
Location	<input type="text" value="Central US"/>
Performance	<input checked="" type="radio"/> Standard <input type="radio"/> Premium
Account kind	<input type="text" value="StorageV2 (general purpose v2)"/>
Replication	<input type="text" value="Locally-redundant storage (LRS)"/>
Access tier (default)	<input type="radio"/> Cool <input checked="" type="radio"/> Hot

[Review + create](#)[Previous](#)[Next : Advanced >](#)

7. Select **Review + Create** to review your storage account settings and create the account.
8. Select **Create**.

# Upload, download, and list blobs (portal)

Now lets use the Azure portal to create a container in Azure Storage, and to upload and download block blobs in that container.

## Create a container

1. In the left menu for the storage account, scroll to the **Blob service** section, then select **Blobs**.
2. Select the **+ Container** button.

### Services



#### Blobs

REST-based object storage for unstructured data

[Explore data using OAuth preview](#)

[Learn more](#)



Container



Refresh




Delete


Storage account: brighttrace




# Upload, download, and list blobs (portal)

4. Type a name for your new container.
5. Set the level of public access to the container.
6. Select **OK** to create the container.


 Container


 Refresh


 Delete

New container

\* Name

mycontainer

Public access level 

Private (no anonymous access)

OK

Cancel

# Upload, download, and list blobs (portal)

## Upload a block blob

Block blobs consist of blocks of data assembled to make a blob. Most scenarios using Blob storage employ block blobs. Block blobs are ideal for storing text and binary data in the cloud, like files, images, and videos.

To upload a block blob to your new container in the Azure portal, follow these steps:

1. In the Azure portal, navigate to the container you created in the previous section.
2. Select the container to show a list of blobs it contains. Since this container is new, it won't yet contain any blobs.
3. Select the **Upload** button to upload a blob to the container.




# Upload, download, and list blobs (portal)

4. Browse your local file system to find a file to upload as a block blob, and select **Upload**.
5. Select the **Authentication type**. The default is **SAS**.
6. Upload as many blobs as you like in this way. You'll see that the new blobs are now listed within the container.

Upload blob

mycontainer/

Files ⓘ  
"blockBlob.txt" 

Authentication type ⓘ  

OAuth (preview) SAS

☐ Overwrite if files already exist

▼ Advanced

Upload

## Current uploads

Dismiss: [Completed](#) [All](#)

12-Azure VM Monitoring Options-3...	✓ 2 MiB / 2 MiB	...
blockBlob.txt	✓ 37 B / 37 B	...

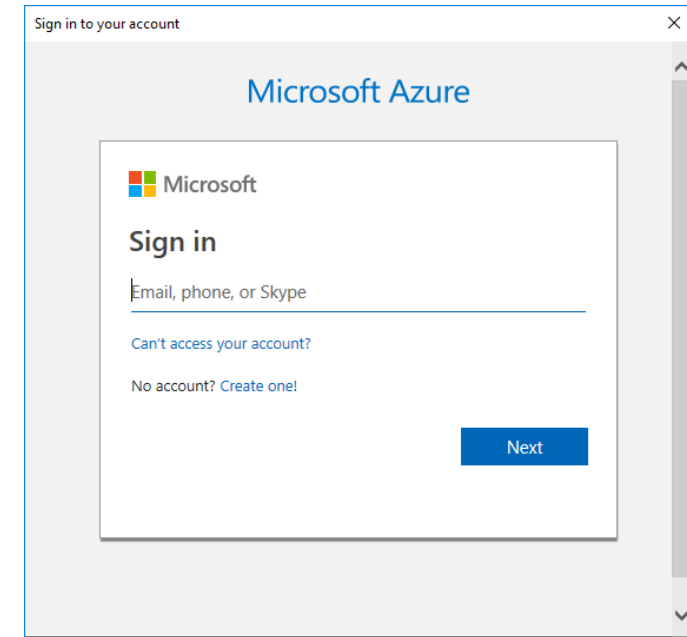
# Upload, download, and list blobs (PowerShell)

## Connect to Your Azure Account

Enter the following cmdlet in PowerShell.

**Connect-AzureRmAccount**

The screen will pop up and ask for credentials of your account. Enter the credentials and sign in.



# Upload, download, and list blobs (PowerShell)

## Create a storage account

The following script shows how to create a general-purpose storage account using [New-AzureRmStorageAccount](#). After you create the account, retrieve its context, which can be used in subsequent commands rather than specifying the authentication with each call.

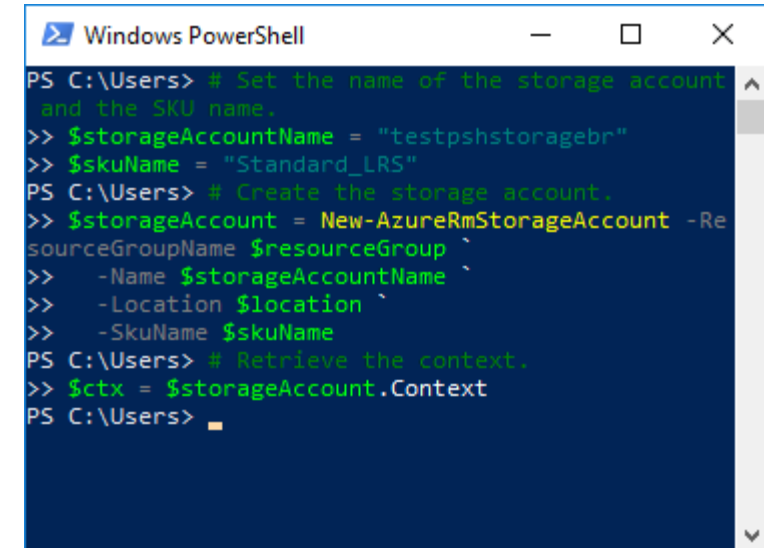
```
# Get list of locations and select one.
Get-AzureRmLocation | select Location
$location = "eastus"

# Create a new resource group.
$resourceGroup = "teststoragerg"
New-AzureRmResourceGroup -Name $resourceGroup -Location $location

# Set the name of the storage account and the SKU name.
$storageAccountName = "testpshstorage"
$skuName = "Standard_LRS"

# Create the storage account.
$storageAccount = New-AzureRmStorageAccount -ResourceGroupName $resourceGroup `
    -Name $storageAccountName `
    -Location $location `
    -SkuName $skuName

# Retrieve the context.
$ctx = $storageAccount.Context
```



```
Windows PowerShell
PS C:\Users> # Set the name of the storage account and the SKU name.
>> $storageAccountName = "testpshstoragebr"
>> $skuName = "Standard_LRS"
PS C:\Users> # Create the storage account.
>> $storageAccount = New-AzureRmStorageAccount -ResourceGroupName $resourceGroup `
>> -Name $storageAccountName `
>> -Location $location `
>> -SkuName $skuName
PS C:\Users> # Retrieve the context.
>> $ctx = $storageAccount.Context
PS C:\Users>
```

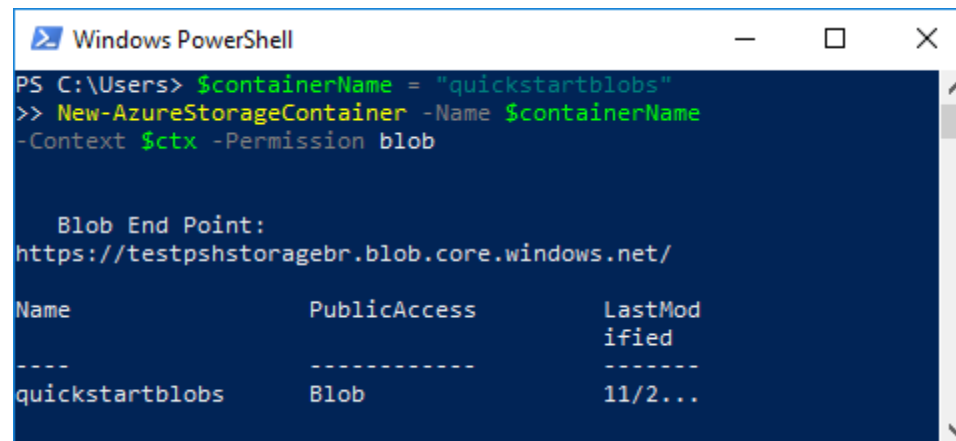
# Upload, download, and list blobs (PowerShell)

## Create a container

Blobs are always uploaded into a container. You can organize groups of blobs like the way you organize your files on your computer in folders.

Set the container name, then create the container by using **New-AzureStorageContainer**. Set the permissions to blob to allow public access of the files.

- `$containerName = "quickstartblobs" New-AzureStorageContainer -Name $containerName -Context $ctx -Permission blob`



```
Windows PowerShell
PS C:\Users> $containerName = "quickstartblobs"
>> New-AzureStorageContainer -Name $containerName
-Context $ctx -Permission blob

Blob End Point:
https://testpshstoragebr.blob.core.windows.net/

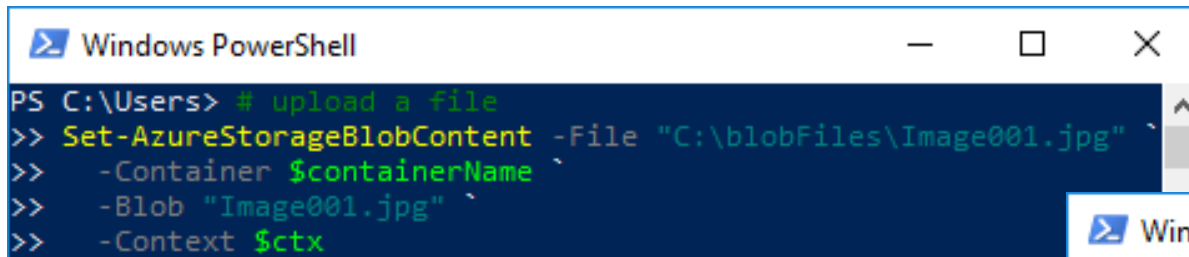
Name                PublicAccess        LastMod
----                -
quickstartblobs     Blob                11/2...
```

# Upload, download, and list blobs (PowerShell)

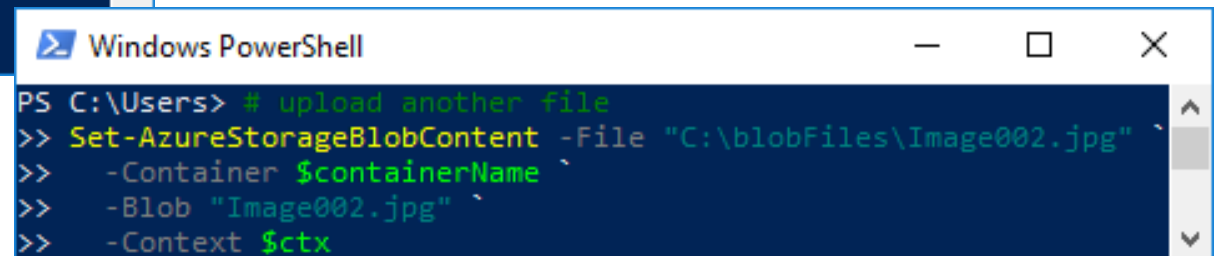
## Upload blobs to the container

To upload a file to a block blob, get a container reference, then get a reference to the block blob in that container. Once you have the blob reference, you can upload data to it by using **Set-AzureStorageBlobContent**. This operation creates the blob if it doesn't exist, or overwrites the blob if it exists.

The following examples upload *Image001.jpg* and *Image002.jpg* from the *C:\blobFiles* folder on the local disk to the container you created.



```
Windows PowerShell
PS C:\Users> # upload a file
>> Set-AzureStorageBlobContent -File "C:\blobFiles\Image001.jpg" `
>> -Container $containerName `
>> -Blob "Image001.jpg" `
>> -Context $ctx
```

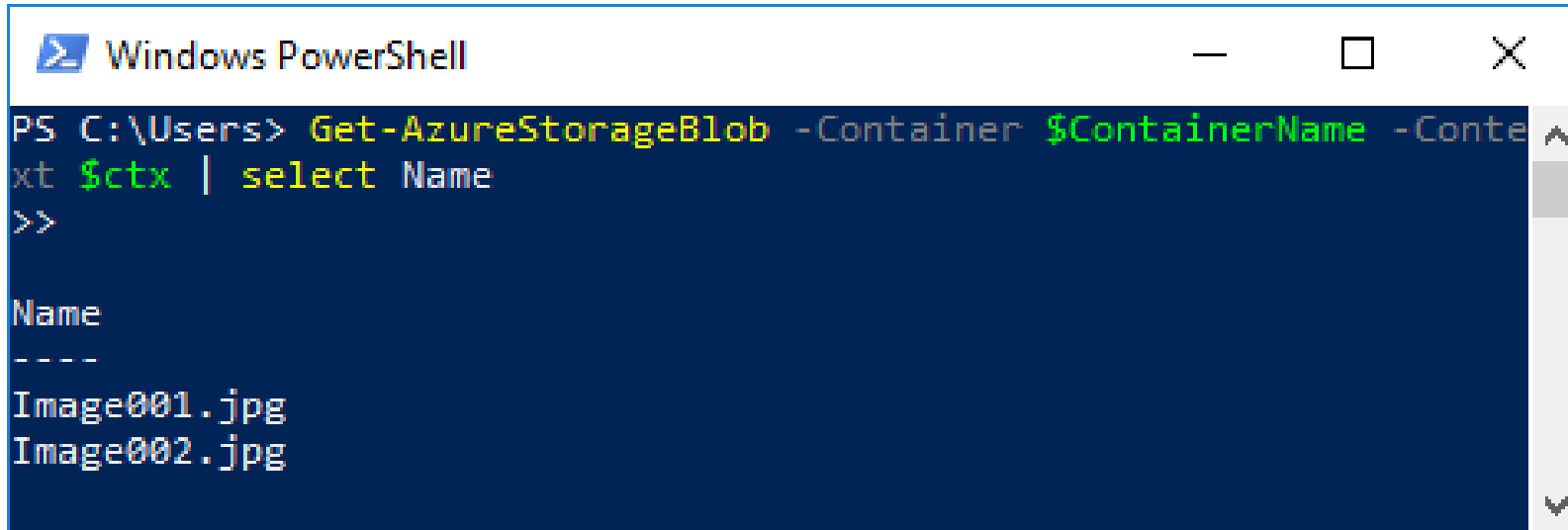


```
Windows PowerShell
PS C:\Users> # upload another file
>> Set-AzureStorageBlobContent -File "C:\blobFiles\Image002.jpg" `
>> -Container $containerName `
>> -Blob "Image002.jpg" `
>> -Context $ctx
```

# Upload, download, and list blobs (PowerShell)

## List the blobs in a container

Get a list of blobs in the container by using `Get-AzureStorageBlob`. This example shows just the names of the blobs uploaded.

A screenshot of a Windows PowerShell window. The title bar reads "Windows PowerShell". The command prompt shows the command `Get-AzureStorageBlob -Container $ContainerName -Context $ctx | select Name` being executed. The output displays the names of the blobs: `Image001.jpg` and `Image002.jpg`.

```
PS C:\Users> Get-AzureStorageBlob -Container $ContainerName -Context $ctx | select Name
Name
----
Image001.jpg
Image002.jpg
```



# Upload, download, and list blobs (PowerShell)

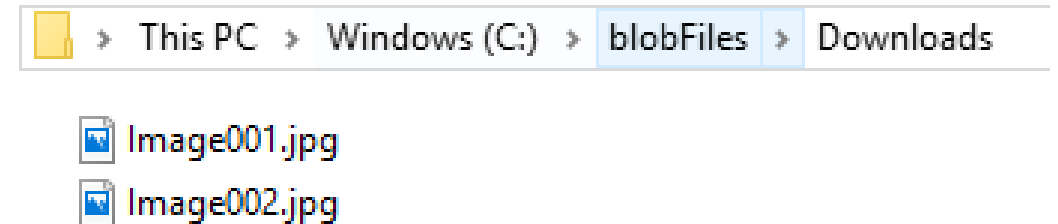
## Download blobs

Download the blobs to your local disk. For each blob you want to download, set the name and call **Get-AzureStorageBlobContent** to download the blob.

This example downloads the blobs to *C:\blobFiles\Downloads* on the local disk.

```
Windows PowerShell
PS C:\Users> # download first blob
>> Get-AzureStorageBlobContent -Blob "Image001.jpg" `
>>   -Container $containerName `
>>   -Destination "C:\blobFiles\Downloads\" `
>>   -Context $ctx
```

```
Windows PowerShell
PS C:\Users> # download first blob
>> Get-AzureStorageBlobContent -Blob "Image002.jpg" `
>>   -Container $containerName `
>>   -Destination "C:\blobFiles\Downloads\" `
>>   -Context $ctx
```



# Azure Files

# What is Azure Files?

Azure Files offers fully managed file shares in the cloud that are accessible via the industry standard **Server Message Block (SMB) protocol**. Azure file shares can be mounted concurrently by cloud or on-premises deployments of Windows, Linux, and macOS. Additionally, Azure file shares can be cached on Windows Servers with Azure File Sync for fast access near where the data is being used.

The difference between the azure files and the files on a corporate file share is that we can access the azure files from anywhere in the world using a URL that points to the file.

We can generate a Shared Access Signature SAS Tokens to allow specific access to a private asset for a specific amount of time

# Why Azure Files is useful

## **Replace or supplement on-premises file servers**

Azure Files can be used to completely replace or supplement traditional on-premises file servers or NAS devices.

## **"Lift and shift" applications**

Azure Files makes it easy to "lift and shift" applications to the cloud that expect a file share to store file application or user data.

## **Simplify cloud development**

Azure Files can also be used in numerous ways to simplify new cloud development projects. For example:

- **Shared application settings**
- **Diagnostic share**
- **Dev/Test/Debug**

# Key benefits

## Shared access

Azure file shares support the industry standard SMB protocol, meaning you can seamlessly replace your on-premises file shares with Azure file shares without worrying about application compatibility.

## Fully managed

Azure file shares can be created without the need to manage hardware or an OS. This means you don't have to deal with patching the server OS with critical security upgrades or replacing faulty hard disks.

## Scripting and tooling

PowerShell cmdlets and Azure CLI can be used to create, mount, and manage Azure file shares as part of the administration of Azure applications. You can create and manage Azure file shares using Azure portal and Azure Storage Explorer.

# Key benefits

## **Resiliency**

Azure Files has been built from the ground up to be always available. Replacing on-premises file shares with Azure Files means you no longer have to wake up to deal with local power outages or network issues.

## **Familiar programmability.**

Applications running in Azure can access data in the share via file system I/O APIs. Developers can therefore leverage their existing code and skills to migrate existing applications. In addition to System IO APIs, you can use Azure Storage Client Libraries or the Azure Storage REST API.

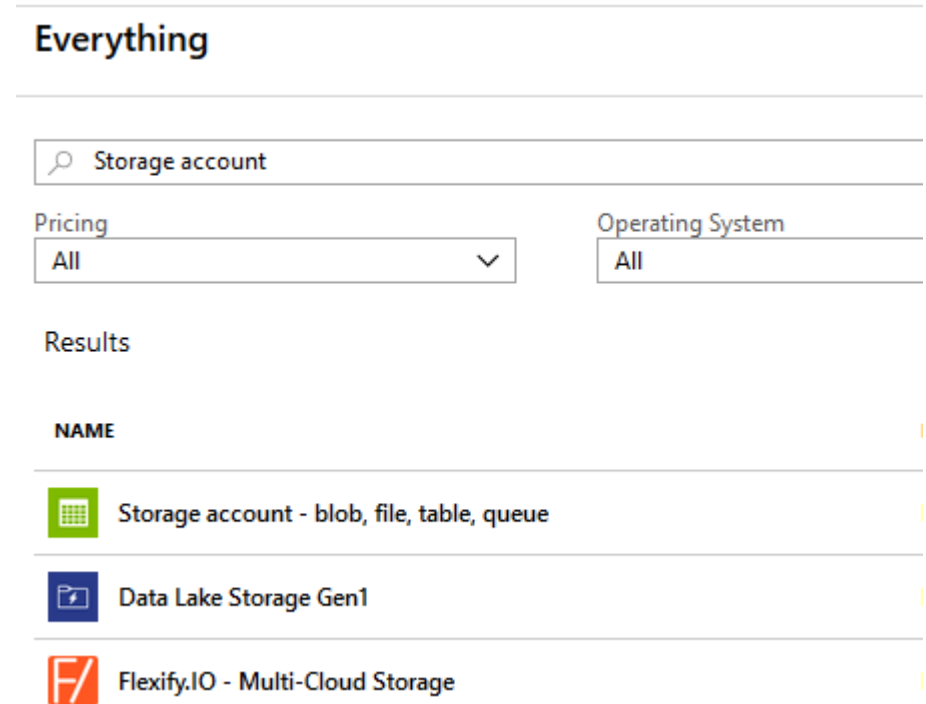
# Create and manage Azure file (portal)

## Create a storage account

A storage account is a shared pool of storage in which you can deploy an Azure file share or other storage resources, such as blobs or queues. A storage account can contain an unlimited number of shares. A share can store an unlimited number of files, up to the capacity limits of the storage account.

To create a storage account:

1. In the left menu, select **+** to create a resource.
2. In the search box, enter **storage account**, select **Storage account - blob, file, table, queue**, and then select **Create**.



# Create and manage Azure file (portal)

## Create an Azure file share

To create an Azure file share:

1. Select the storage account from your dashboard.
2. On the storage account page, in the **Services** section, select **Files**.

### Services



#### Blobs

REST-based object storage for unstructured data

[Explore data using OAuth preview](#)

[Learn more](#)



#### Files

File shares that use the standard SMB 3.0 protocol

[Learn more](#)



#### Tables

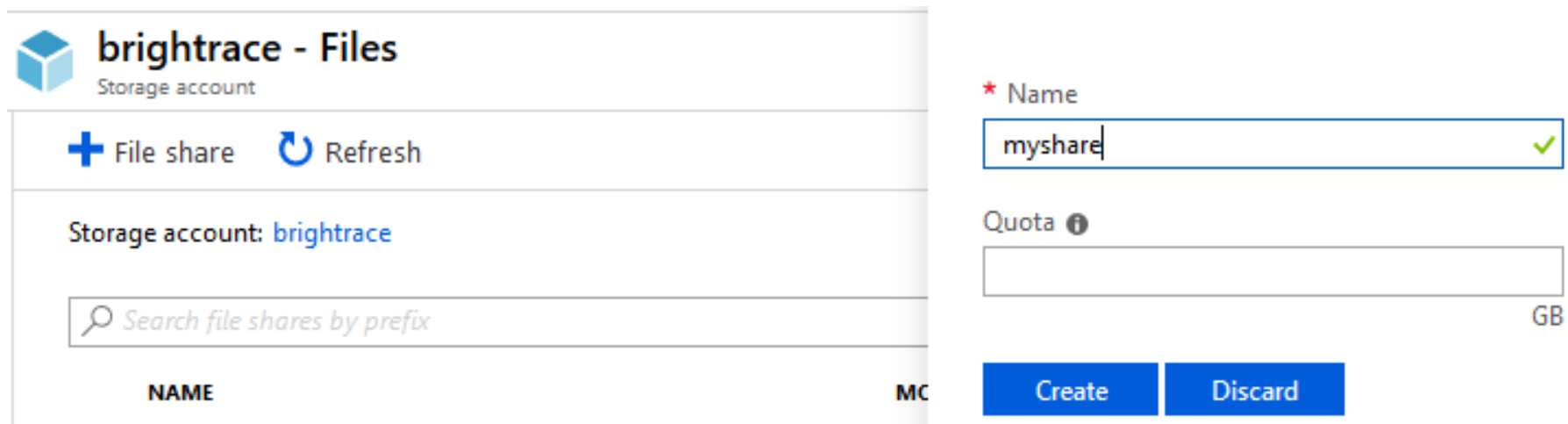
Tabular data storage

[Learn more](#)



# Create and manage Azure file (portal)

3. On the menu at the top of the **File service** page, click **+ File share**. The **New file share** page drops down.
4. In **Name** type *myshare*.
5. Click **OK** to create the Azure file share.



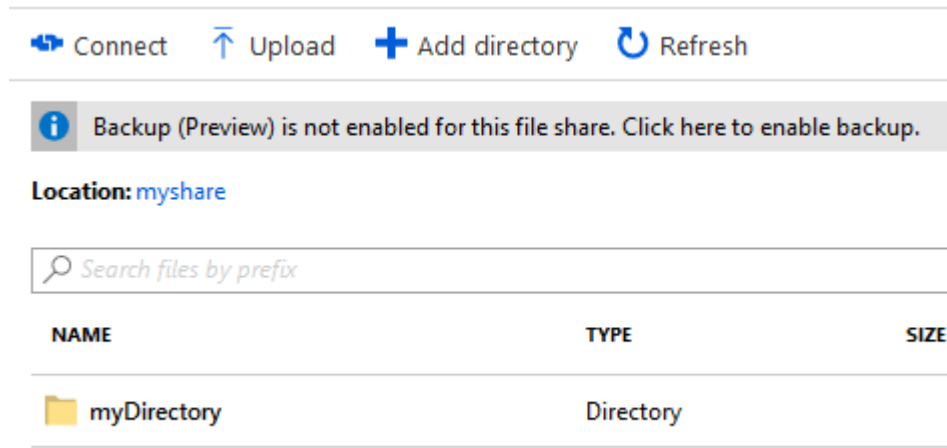
The screenshot shows the 'brightrace - Files' page in the Azure portal. On the left, there's a sidebar with a 'File share' button (a plus sign) and a 'Refresh' button (a circular arrow). Below these, it says 'Storage account: brightrace' and a search bar with the placeholder 'Search file shares by prefix'. The main area on the right is titled 'New file share'. It has a 'Name' field with a red asterisk, containing the text 'myshare' and a green checkmark. Below that is a 'Quota' field with an information icon, which is currently empty. At the bottom right, there are two buttons: 'Create' and 'Discard'.

# Use your Azure file share (portal)

## Create a directory

To create a new directory named *myDirectory* at the root of your Azure file share:

1. On the **File Service** page, select the **myshare** file share. The page for your file share opens.
2. On the menu at the top of the page, select **+ Add directory**. The **New directory** page drops down.
3. Type *myDirectory* and then click **OK**.

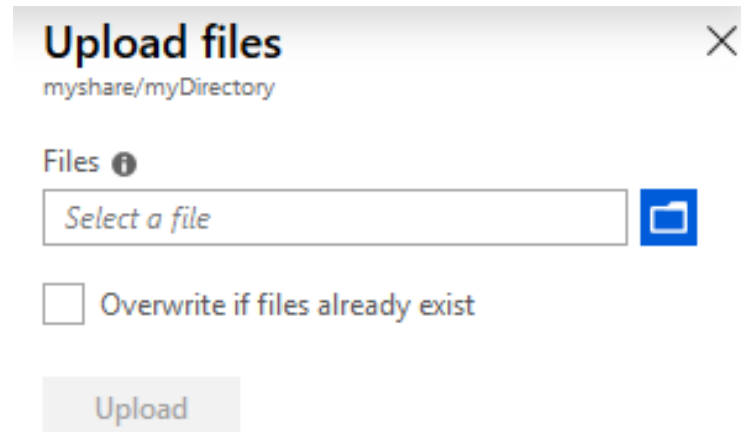


# Use your Azure file share (portal)

## Upload a file

To demonstrate uploading a file, you first need to create or select a file to be uploaded. You may do this by whatever means you see fit. Once you've selected the file you would like to upload:

1. Click on the **myDirectory** directory. The **myDirectory** panel opens.
2. In the menu at the top, click **Upload**. The **Upload files** panel opens.



The screenshot shows the 'Upload files' panel in the Azure portal. The panel has a title bar with 'Upload files' and a close button (X). Below the title bar, the path 'myshare/myDirectory' is displayed. The main area is labeled 'Files' with an information icon. There is a text input field with the placeholder 'Select a file' and a blue folder icon to its right. Below the input field, there is a checkbox labeled 'Overwrite if files already exist'. At the bottom of the panel, there is a grey 'Upload' button.

# Use your Azure file share (portal)

3. Click on the folder icon to open a window to browse your local files.
4. Select a file and then click **Open**.
5. In the **Upload files** page, verify the file name and then click **Upload**.
6. When finished, the file should appear in the list on the **myDirectory** page.

## Current uploads




Dismiss: [Completed](#) [All](#)

Image001.jpg	✓ 16 KiB / 16 KiB	...
12-Azure VM Monitoring Options-3...	✓ 2 MiB / 2 MiB	...

# Use your Azure file share (portal)

## Download a file

You can download a copy of the file you uploaded by right-clicking on the file. After clicking the download button, the exact experience will depend on the operating system and browser you're using.

NAME	TYPE	SIZE	
 [..]			...
 12-Azure VM Monitoring Options-314d43a0-07b3-41ed...	File	1.69 MiB	...
 Image001.jpg	File	15.65 KiB	...

Edit

Download

Properties

Edit metadata

Delete

# Create and manage Azure file (PowerShell)

## Create a storage account

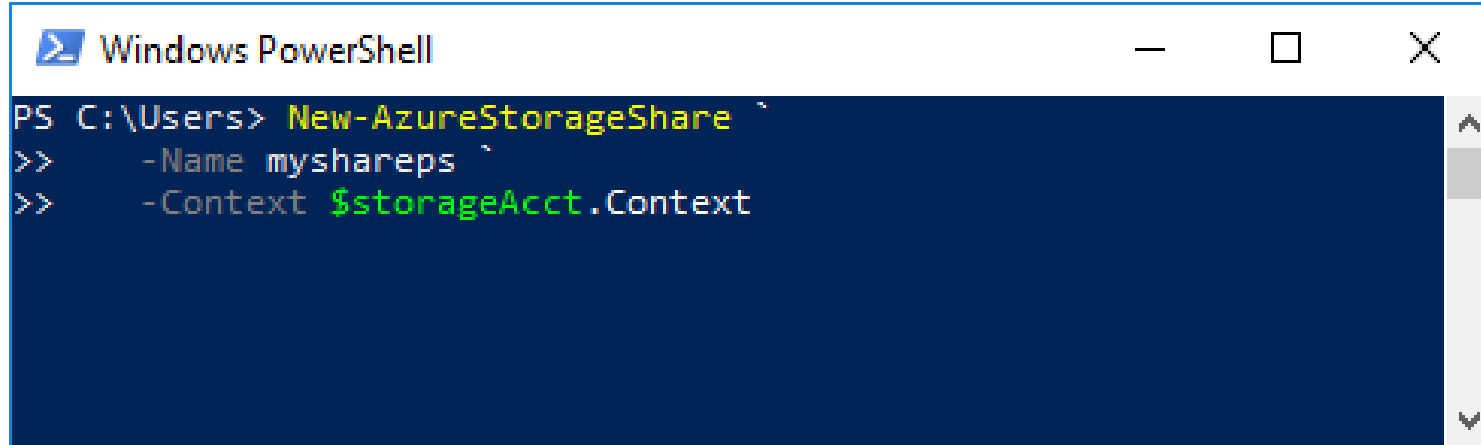
This example creates a storage account using the **New-AzureRmStorageAccount** cmdlet. The storage account is named brightrace and a reference to that storage account is stored in the variable \$storageAcct. Storage account names must be unique, so use Get-Random to append a number to the name to make it unique.

```
$storageAcct = New-AzureRmStorageAccount `
  -ResourceGroupName "BrightraceRS" `
  -Name "brightrace$(Get-Random)" `
  -Location eastus `
  -SkuName Standard_LRS
```

# Create and manage Azure file (PowerShell)

## Create an Azure file share

Now you can create your first Azure file share. You can create a file share using the **New-AzureStorageShare** cmdlet. This example creates a share named myshareps.

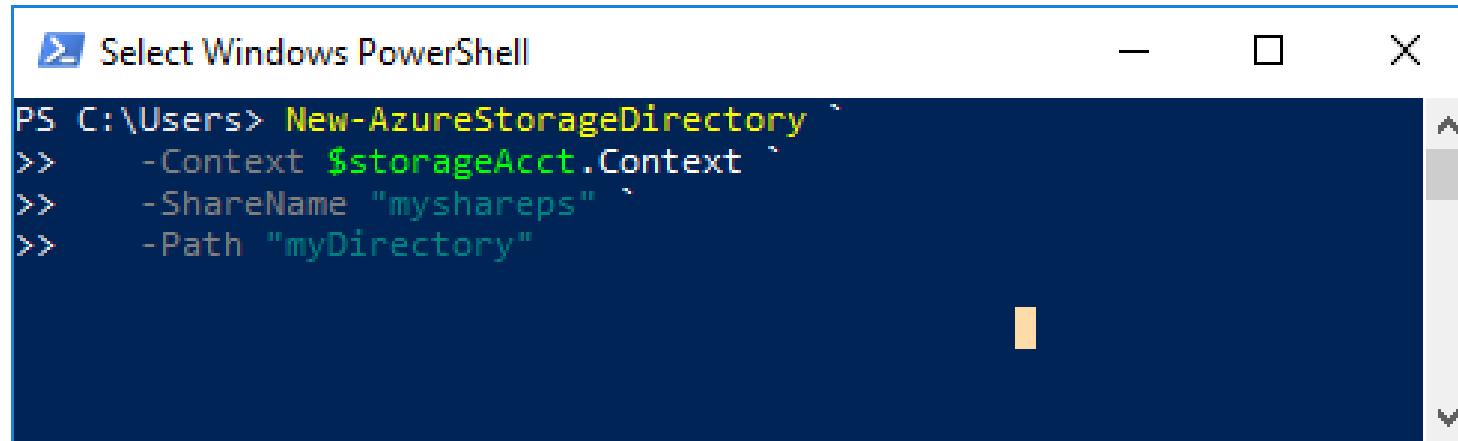
A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell" with standard minimize, maximize, and close buttons. The command prompt shows the user at the C:\Users directory. The command entered is "New-AzureStorageShare" followed by two parameters: "-Name myshareps" and "-Context \$storageAcct.Context". The command is currently on the third line of the prompt, with the first two lines being the command name and the first parameter.

```
PS C:\Users> New-AzureStorageShare `
>> -Name myshareps `
>> -Context $storageAcct.Context
```

# Use your Azure file share (PowerShell)

## Create directory

To create a new directory named *myDirectory* at the root of your Azure file share, use the **New-AzureStorageDirectory** cmdlet.

A screenshot of a Windows PowerShell window titled "Select Windows PowerShell". The window has a dark blue background. The command prompt shows the following text:

```
PS C:\Users> New-AzureStorageDirectory`  
>> -Context $storageAcct.Context`  
>> -ShareName "myshareps"`  
>> -Path "myDirectory"
```

The cursor is at the end of the last line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

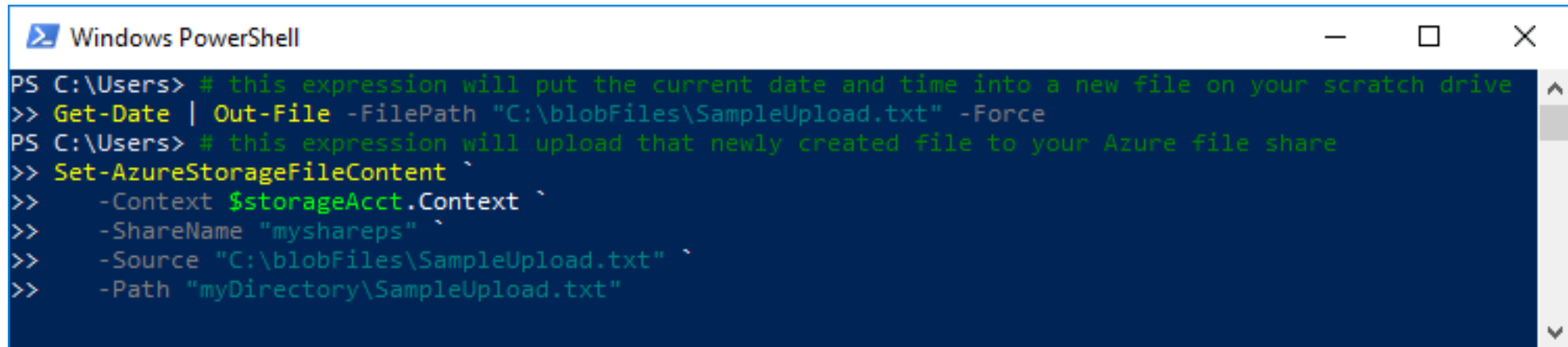


# Use your Azure file share (PowerShell)

## Upload a file

To demonstrate how to upload a file using the **Set-AzureStorageFileContent** cmdlet, we first need to create a file inside your PowerShell Cloud Shell's scratch drive to upload.

This example puts the current date and time into a new file on your C drive, then uploads the file to the file share.

A screenshot of a Windows PowerShell terminal window. The title bar reads "Windows PowerShell". The terminal shows two commands being executed. The first command is a comment followed by a PowerShell command: "PS C:\Users> # this expression will put the current date and time into a new file on your scratch drive" followed by ">> Get-Date | Out-File -FilePath 'C:\blobFiles\SampleUpload.txt' -Force". The second command is another comment followed by a PowerShell command: "PS C:\Users> # this expression will upload that newly created file to your Azure file share" followed by ">> Set-AzureStorageFileContent", ">> -Context \$storageAcct.Context", ">> -ShareName 'myshareps'", ">> -Source 'C:\blobFiles\SampleUpload.txt'", and ">> -Path 'myDirectory\SampleUpload.txt'".

```
Windows PowerShell
PS C:\Users> # this expression will put the current date and time into a new file on your scratch drive
>> Get-Date | Out-File -FilePath "C:\blobFiles\SampleUpload.txt" -Force
PS C:\Users> # this expression will upload that newly created file to your Azure file share
>> Set-AzureStorageFileContent `
>>   -Context $storageAcct.Context `
>>   -ShareName "myshareps" `
>>   -Source "C:\blobFiles\SampleUpload.txt" `
>>   -Path "myDirectory\SampleUpload.txt"
```

# Use your Azure file share (PowerShell)

After uploading the file, you can use **Get-AzureStorageFile** cmdlet to check to make sure that the file was uploaded to your Azure file share.

```
Windows PowerShell
PS C:\Users> Get-AzureStorageFile -Context $storageAcct.Context -ShareName
"myshareps" -Path "myDirectory"
>>


Directory: https://brightrace1022101620.file.core.windows.net/myshareps


Type            Length Name
----            -
1 myDirectory
```

Location: [myshareps / myDirectory](#)

Search files by prefix

NAME

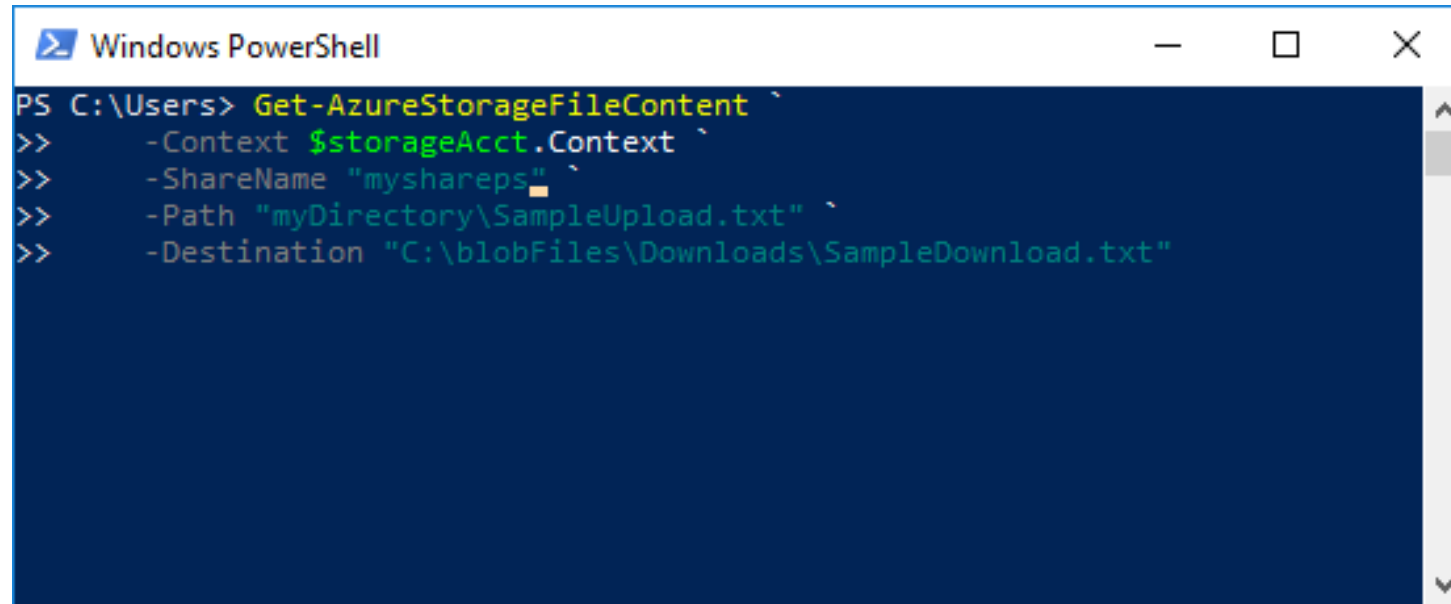
 [..]

 SampleUpload.txt

# Use your Azure file share (PowerShell)

## Download a file

You can use the **Get-AzureStorageFileContent** cmdlet to download a copy of the file you just uploaded.



```
Windows PowerShell
PS C:\Users> Get-AzureStorageFileContent `
>>   -Context $storageAcct.Context `
>>   -ShareName "myshareps" `
>>   -Path "myDirectory\SampleUpload.txt" `
>>   -Destination "C:\blobFiles\Downloads\SampleDownload.txt"
```

# Use your Azure file share (PowerShell)

After downloading the file, you can use the **Get-ChildItem** to see that the file has been downloaded to your C drive.

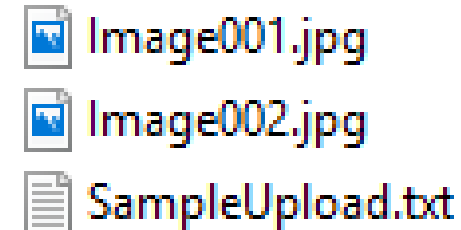
```
Windows PowerShell
PS C:\blobFiles\downloads> Get-ChildItem -Path "C:\blobFiles\downloads"
>>

Directory: C:\blobFiles\downloads

Mode                LastWriteTime         Length Name
----                -
-a----          11/29/2018   3:16 PM         16025 Image001.jpg
-a----          11/29/2018   3:17 PM          6393 Image002.jpg
-a----          11/30/2018  11:43 AM           92 SampleUpload.txt

PS C:\blobFiles\downloads>
```

blobFiles > Downloads



# When to use Azure Blobs / Azure Files

Microsoft Azure provides several features in Azure Storage for storing and accessing your data in the cloud. This article covers Azure Files and Blobs, and is designed to help you choose between these features.

## Scenarios

Feature	Description	When to use
<b>Azure Files</b>	Provides an SMB interface, client libraries, and a REST interface that allows access from anywhere to stored files.	<p>You want to "lift and shift" an application to the cloud which already uses the native file system APIs to share data between it and other applications running in Azure.</p> <p>You want to store development and debugging tools that need to be accessed from many virtual machines.</p>
<b>Azure Blobs</b>	Provides client libraries and a REST interface that allows unstructured data to be stored and accessed at a massive scale in block blobs.	<p>You want your application to support streaming and random access scenarios.</p> <p>You want to be able to access application data from anywhere.</p>