Module 6 - Lab 1: Implementing and configuring Azure Storage File and Blob Services

Scenario

Adatum Corporation hosts large amounts of unstructured and semi-structured data in its on-premises storage. Its maintenance becomes increasingly complex and costly. Some of the data is preserved for extensive amount of time in order to address data retention requirements. Adatum Enterprise Architecture team is looking for inexpensive alternatives that would support tiered storage, while, at the same time allow for secure access that minimizes the possibility of data exfiltration. While the team is aware of practically unlimited capacity offered by Azure Storage, it is concerned about the usage of account keys, which grant unlimited access to the entire content of the corresponding storage accounts. While keys can be rotated in an orderly manner, such operation needs to be carried out with proper planning. In addition, access keys constitute exclusively an authorization mechanism, which limits the ability to properly audit their usage.

To address these shortcomings, the Architecture team decided to explore the use of shared access signatures. A shared access signature (SAS) provides secure delegated access to resources in a storage account while minimizing the possiblity of unintended data exposure. SAS offers granular control over data access, including the ability to limit access to an individual storage object, such as a blob, restricting such access to a custom time window, as well as filtering network access to a designated IP address range. In addition, the Architecture team wants to evaluate the level of integration between Azure Storage and Azure Active Directory, hoping to address its audit requirements. The Architecture team also decided to determine suitability of Azure Files as an alernative to some of its on-premises file shares.

To accomplish these objectives, Adatum Corporation will test a range of authentication and authorization mechanisms for Azure Storage resources, including:

- Using shared access signatures on the account, container, and object-level
- Configuring access level for blobs
- Implementing Azure Active Directory based authorization
- Using storage account access keys

After completing this lab, you will be able to:

- Implement authorization of Azure Storage blobs by leveraging shared access signatures
- Implement authorization of Azure Storage blobs by leveraging Azure Active Directory
- Implement authorization of Azure Storage file shares by leveraging access keys

Exercise 1: Configure Azure Storage account authorization by using shared access signature.

The main tasks for this exercise are as follows:

1. Create an Azure Storage account
2. Install Storage Explorer
3. Generate an account-level shared access signature
4. Create a blob container by using Azure Storage Explorer
5. Upload a file to a blob container by using AzCopy
6. Access a blob by using a blob-level shared access signature

Task 1: Create an Azure Storage account

- [ ] 1. Login to the Azure Portal **https://portal.azure.com** with the username: **sheikhnasirGOU8K@gdcs2.com** and password **C7wbSz2HH0XJbe5t**
- [ ] 2. In the Azure portal, search for and select **Storage accounts** and, on the **Storage accounts** blade, select **+ Create**.
- [ ] 3. On the **Basics** tab of the **Create storage account** blade, specify the following settings (leave others with their default values):

Setting	Value
Subscription	the name of the Azure subscription you are using in this lab
Resource group	select **unrecognised token ($gd.com(azure).resourceGroups(az30306a-labRG))**
Storage account name	any globally unique name between 3 and 24 in length consisting of letters and digits
Region	**East US**
Performance	**Standard**
Redundancy	**Locally redundant storage (LRS)**

- [ ] 4. Select **Next: Advanced >** review the options.
- [ ] 5. Select **Next: Networking >**, on the **Networking** tab of the **Create storage account** blade, review the available options, accept the default option **Public endpoint (all networks}** and select **Next: Data protection >**.

- [ ] 6. On the **Data protection** tab of the **Create storage account** blade, review the available options, accept the defaults, select **Review + Create**, wait for the validation process to complete and select **Create**.

Task 2: Install Storage Explorer

- [ ] 1. On your lab VM open a new browser tab and navigate to the download page of Azure Storage Explorer **https://azure.microsoft.com/en-us/features/storage-explorer/**

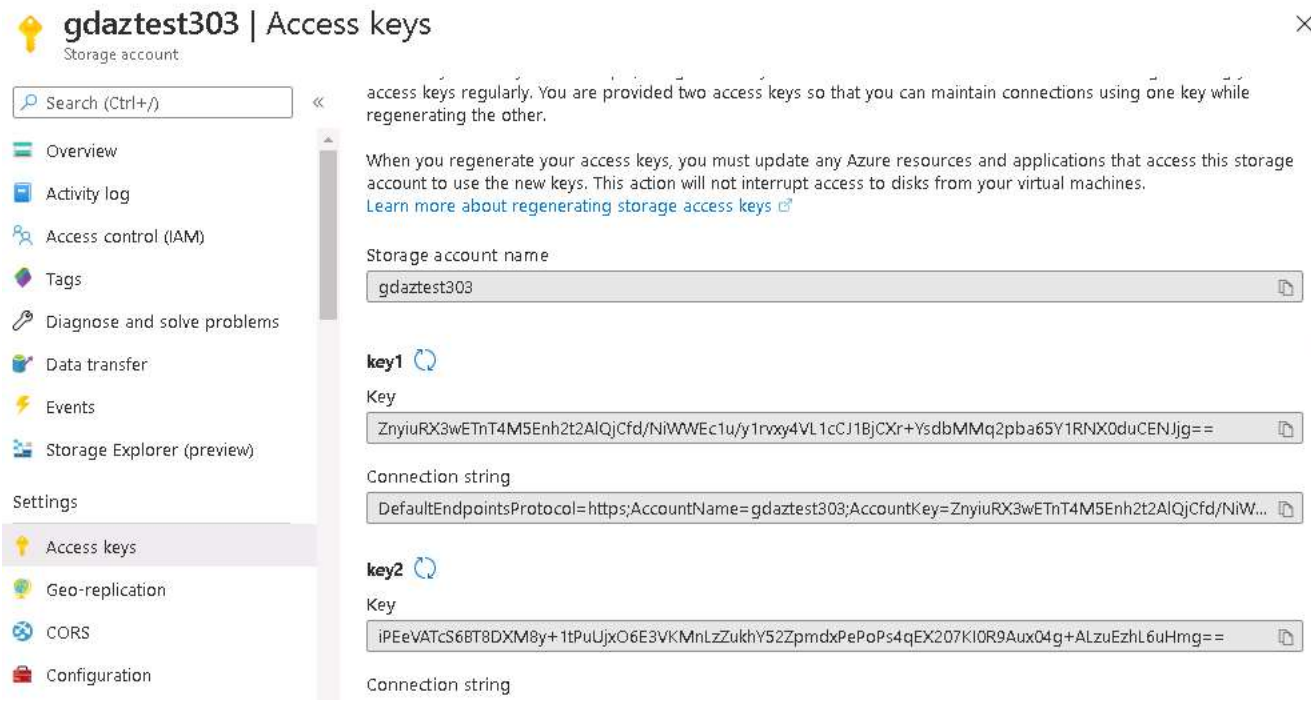
- [ ] 2. Download and install Azure Storage Explorer with the default settings.

Task 3: Generate an account-level shared access signature

- [ ] 1. Return back to the browser with the Azure Portal open.

- [ ] 2. Navigate to the blade of the newly created storage account, select **Access keys** and review the settings of the target blade.

> ⚠ **Note**: Each storage account has two keys which you can independently regenerate. Knowledge of the storage account name and either of the two keys provides full access to the entire storage account.



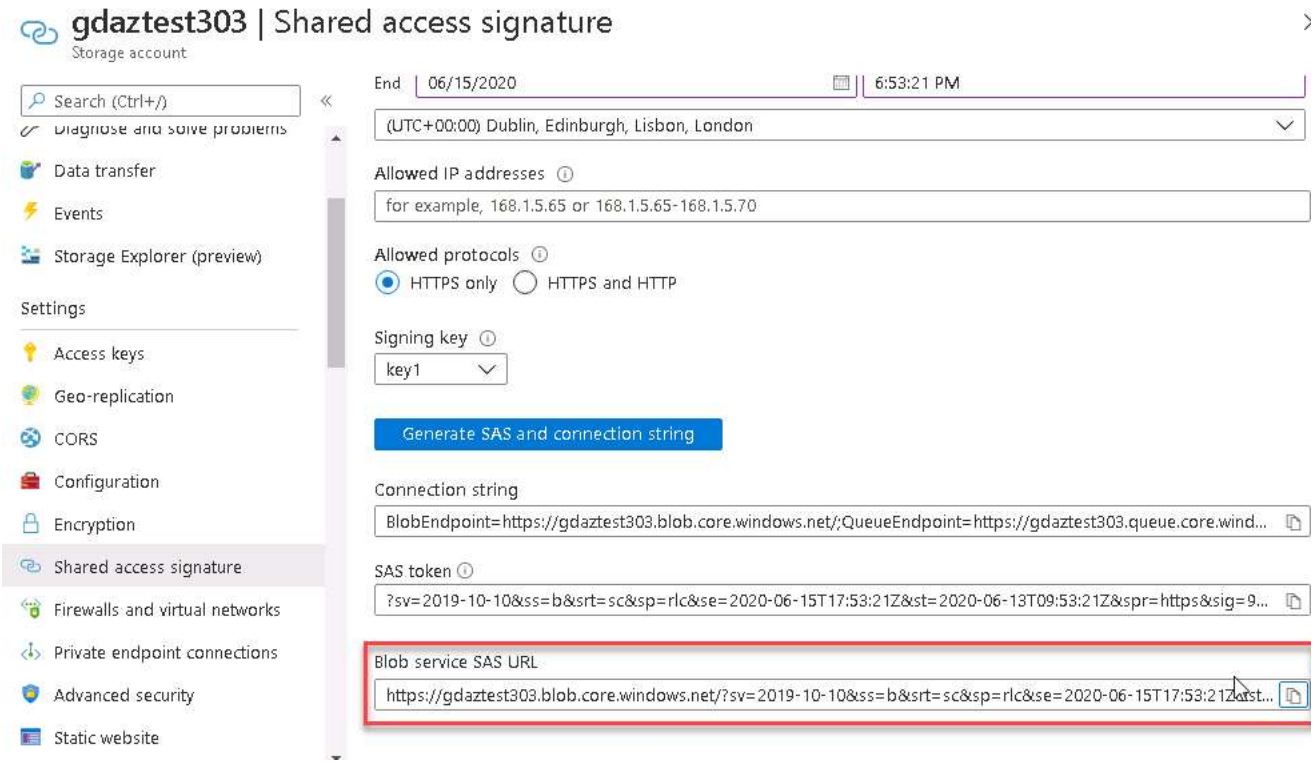
- [ ] 3. On the storage account blade, select **Shared access signature** and review the settings of the target blade.

- [ ] 4. On the resulting blade, specify the following settings (leave others with their default values):

Setting	Value
Allowed services	**Blob**
Allowed service types	**Service** and **Container**
Allowed permissions	**Read**, **List** and **Create**
Blob versioning permissions	disabled
Start	24 hours before the current time in your current time zone
End	24 hours after the current time in your current time zone
Allowed protocols	**HTTPS only**
Signing key	**key1**

- [ ] 5. Select **Generate SAS and connection string**.

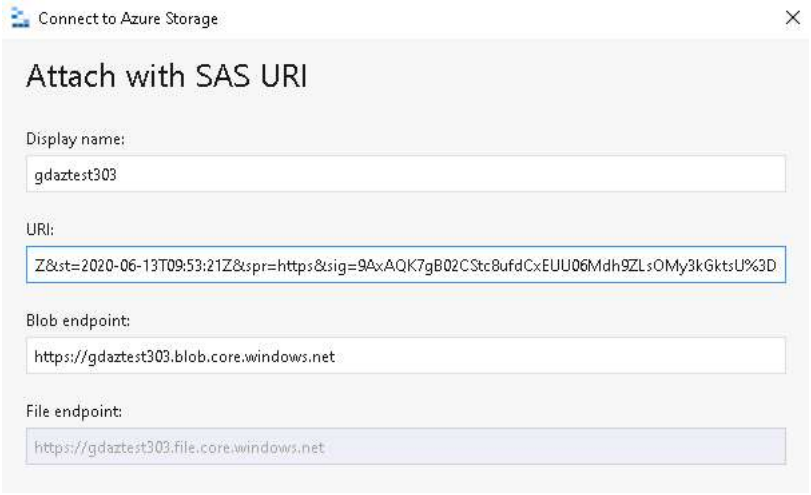
- [ ] 6. Copy the value of **Blob service SAS URL** into Clipboard.



Task 4: Create a blob container by using Azure Storage Explorer

- [ ] 1. In the Lab VM, start Azure Storage Explorer.
- [ ] 2. In the Azure Storage Explorer window, select **Storage account or service**.
- [ ] 3. In the Azure Storage Explorer window, on the **Select Connection Method** window, select **Shared access signature URL (SAS)** and select **Next**.
- [ ] 4. In the **Attach with SAS URI** window, in the **Display name** text box, type az30306a-blobs, in the **Block container SAS URL** text box, paste the value you copied into Clipboard, and select **Next**.

> ⚠ **Note**: This should automatically populate the value of **Blob endpoint** text box.



- [ ] 5. In the **Summary** window, select **Connect**.
- [ ] 6. In the Azure Storage Explorer window, in the **EXPLORER** pane, navigate to the **az30306a-blobs** entry, expand it and note that you have access to **Blob Container** endpoint only.
- [ ] 7. Right select the **az30306a-blobs** entry, in the right-click menu, select **Create Blob Container**, and use the empty text box to set the container name to **container1**.
- [ ] 8. Select **container1**, in the **container1** pane, select **Upload**, and in the drop-down list, select **Upload Files**.
- [ ] 9. In the **Upload Files** window, select the ellipsis button next to the **Selected files** label, in the **Choose files to upload** window, select **C:\Windows\system.ini**, and select **Open**.
- [ ] 10. Back in the **Upload Files** window, select **Upload** and note the error message displayed in the **Activities** list.

> ⚠ **Note**: This is expected, since the shared access signature does not provide object-level permissions.

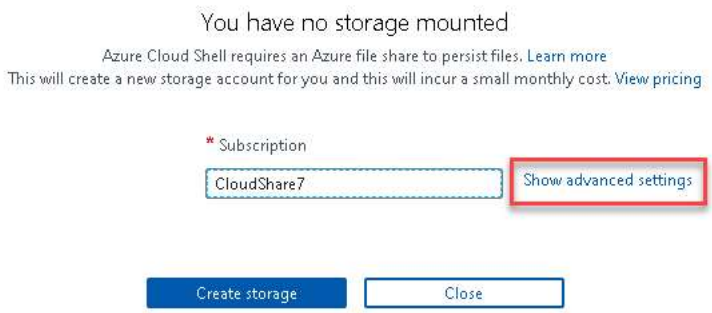
- [ ] 11. Leave the Azure Storage Explorer window open.

Task 5: Upload a file to a blob container by using AzCopy

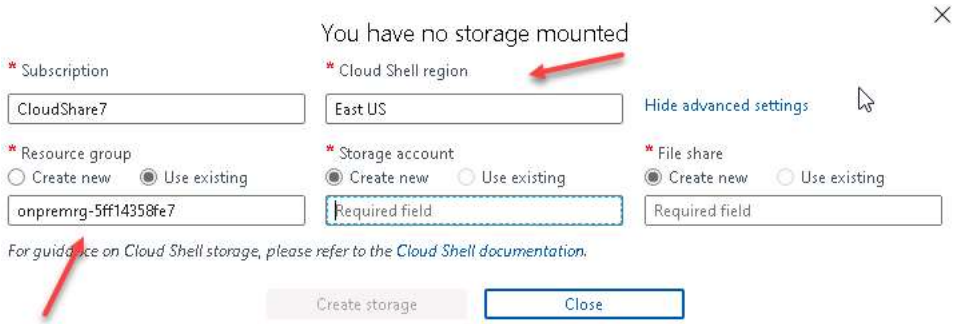
- [ ] 1. In the browser window, on the **Shared access signature** blade, specify the following settings (leave others with their default values):

Setting	Value
Allowed services	**Blob**
Allowed service types	**Object**
Allowed permissions	**Read**, **Create**
Blob versioning permissions	disabled
Start	24 hours before the current time in your current time zone
End	24 hours after the current time in your current time zone
Allowed protocols	**HTTPS only**
Signing key	**key1**

- [ ] 2. Select **Generate SAS and connection string**.
- [ ] 3. Copy the value of **Blob service SAS URL** into Clipboard.
- [ ] 4. In the Azure portal, open **Cloud Shell** pane by selecting on the toolbar icon directly to the right of the search textbox.
- [ ] 5. If prompted to select either **Bash** or **PowerShell**, select **PowerShell**.
- [ ] 6. Click **Show advanced settings**.



- [ ] 7. Select the **East US** region. Select **Use existing** Resource group and select the pre-provisioned resource group for the lab.



- [ ] 8. Enter a name for the storage account (this must be unique) and type **clouldshell** as the name of the File share then click **Create Storage.**

Your Cloud Shell will now launch.

- [ ] 9. From the Cloud Shell pane, run the following to create a file and add a line of text into it:

```
New-Item -Path './az30306ablob.html'

Set-Content './az30306ablob.html' '<h3>Hello from az30306ablob via SAS</h3>'
```

- [ ] 10. From the Cloud Shell pane, run the following to upload the newly created file as a blob into container1 of the Azure Storage account you created earlier in this exercise (replace the <sas_token> placeholder with the value of the shared access signature you copied to Clipboard earlier in this task:

```
$storageAccountName = (Get-AzStorageAccount -ResourceGroupName '<span class="text-danger">unrecognised token ($gd.com(azure).resourceGro

azcopy cp './az30306ablob.html' "https://$storageAccountName.blob.core.windows.net/container1/az30306ablob.html<sas_token>"
```

- [ ] 11. Review the output generated by azcopy and verify that the job completed successfully.
- [ ] 12. Close the Cloud Shell pane.
- [ ] 13. In the browser window, on the storage account blade, in the **Blob service** section, select **Containers**.
- [ ] 14. In the list of containers, select **container1**.
- [ ] 15. On the **container1** blade, verify that **az30306ablob.html** appears in the list of blobs.

Task 6: Access a blob by using a blob-level shared access signature

- [ ] 1. In the browser window, on the **container1** blade, select **Change access level**, verify that is set to **Private (no anonymous access)**, and select **Cancel**.

> ⚠ **Note**: If you want to allow anonymous access, you can set the public access level to **Blob (anonymous read access for blobs only)** or **Container (anonymous read access for containers and blobs)**.

- [ ] 2. On the **container1** blade, select **az30306ablob.html**.
- [ ] 3. On the **az30306ablob.html** blade, select **Generate SAS**, review the available options without modifying them, and then select **Generate SAS token and URL**.
- [ ] 4. Copy the value of the **Blob SAS URL** into Clipboard.
- [ ] 5. Open a new tab in the browser window and navigate to the URL you copied into Clipboard in the previous step.
- [ ] 6. Verify that the message **Hello from az30306ablob via SAS** appears in the browser window.

Exercise 2: Configure Azure Storage blob service authorization by using Azure Active Directory

> ❓ The main tasks for this exercise are as follows:
>
> 1. Enable Azure Active Directory authorization for Azure Storage blob service
> 2. Upload a file to a blob container by using AzCopy

Task 1: Locate your additional Azure AD user

- [ ] 1. Within the lab environment, click on the Home tab and locate the additional Azure AD user that has been created for you.

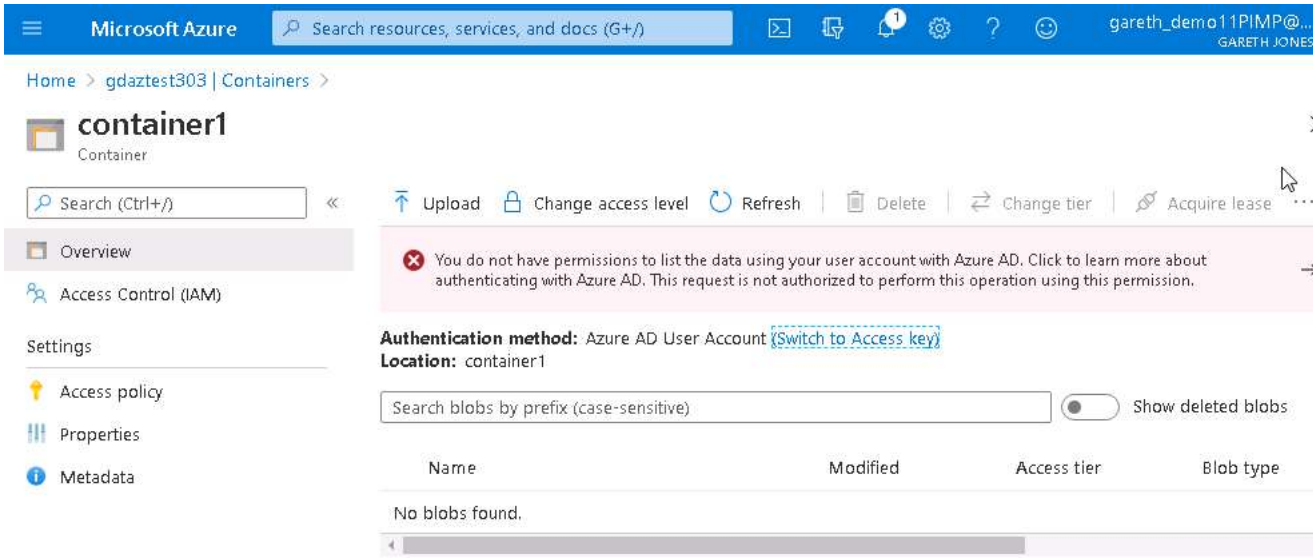
> ⚠ **Note:** This account has now permissions to any resources by default.

Task 2: Enable Azure Active Directory authorization for Azure Storage blob service

- [ ] 1. Within your LAB-VM, in the browser window displaying the Azure portal, navigate back to the **container1** blade.
- [ ] 2. On the **container1** blade, select **Switch to Azure AD User Account**.

Note: the error message indicating that you no longer have permissions to list data in the blob container. This is expected.

Note: Despite having the **Owner** role in the subscription, you also need to be assigned either built-in or a custom role that provides access to the blob content of the storage account, such as **Storage Blob Data Owner**, **Storage Blob Data Contributor**, or **Storage Blob Data Reader**.



3. In the Azure portal, navigate back to the blade of the storage account hosting **container1**, select **Access control (IAM)**, select **+ Add**, and, in the drop-down list, select **Add role assignment**.

   Note: Write down the name of the storage account. You will need it in the next task.

4. On the **Add role assignment** blade, in the **Role** drop-down list, select **Storage Blob Data Owner**, ensure that the **Assign access to** drop-down list entry is set to **User, group, or service principal**, select both your user account and the user account you created in the previous task from the list displayed below the **Select** text box, and select **Save**.

5. Navigate back to the **container1** blade and verify that you can see the content of the container.

Task 3: Upload a file to a blob container by using AzCopy

1. Within the LAB-VM, in the browser window, navigate to the following URL: **https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10**

2. Download the azcopy.zip file and extract azcopy.exe into the **C:\Labfiles** folder (create the folder if needed).

3. Within the LAB-VM, start Windows PowerShell.

4. From the Windows PowerShell prompt, run the following to download the **azcopy.zip** archive, extract its content, and switch to the location containing **azcopy.exe**:

```
$url = 'https://aka.ms/downloadazcopy-v10-windows'
$zipFile = '.\azcopy.zip'

Invoke-WebRequest -Uri $Url -OutFile $zipFile

Expand-Archive -Path $zipFile -DestinationPath '.\'

Set-Location -Path 'azcopy*'
```

5. From the Windows PowerShell prompt, run the following to authenticate AzCopy by using the Azure AD user account you created in the first task of this exercise.

```
.\azcopy.exe login
```

   Note: You cannot use for this purpose a Microsoft account, which is the reason that Azure AD user account had to be created first.

6. Follow instructions provided in the message generated by the command you run in the previous step to authenticate as the second user account created for you on the Home tab of the lab environment.

7. Once you successfully authenticated, from the Windows PowerShell prompt, run the following to create a file you 4will upload to **container1**:

```
New-Item -Path './az30306bblob.html'

Set-Content './az30306bblob.html' '<h3>Hello from az30306bblob via Azure AD</h3>'
```

- [ ] 8. From the the Windows PowerShell prompt, run the following to upload the newly created file as a blob into **container1** of the Azure Storage account you created in the previous exercise (replace the <storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
.\azcopy cp './az30306bblob.html' 'https://<storage_account_name>.blob.core.windows.net/container1/az30306bblob.html'
```

- [ ] 9. Review the output generated by azcopy and verify that the job completed successfully.
- [ ] 10. From the Windows PowerShell prompt and run the following to verify that you do not have access to the uploaded blob outside of the security context provided by the AzCopy utility (replace the <storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
Invoke-WebRequest -Uri 'https://<storage_account_name>.blob.core.windows.net/container1/az30306bblob.html'
```

- [ ] 11. In the LAB-VM, in the browser window, navigate back to **container1**.
- [ ] 12. On the **container1** blade, verify that **az30306bblob.html** appears in the list of blobs.
- [ ] 13. On the **container1** blade, select **Change access level**, set the public access level to **Blob (anonymous read access for blobs only)** and select **OK**.
- [ ] 14. Switch back to the Windows PowerShell prompt and re-run the following command to verify that now you can access the uploaded blob anonymously (replace the <storage_account_name> placeholder with the value of the storage account you noted in the previous task):

```
Invoke-WebRequest -Uri 'https://<storage_account_name>.blob.core.windows.net/container1/az30306bblob.html'
```

Exercise 3: Implement Azure Files.

The main tasks for this exercise are as follows:

1. Create an Azure Storage file share
2. Map a drive to an Azure Storage file share from Windows
3. Remove Azure resources deployed in the lab

Task 1: Create an Azure Storage file share

- [ ] 1. In the LAB-VM, in the browser window displaying the Azure portal, navigate back to the blade of the storage account you created in the first exercise of this lab and, in the **File service** section, select **File shares**.
- [ ] 2. Select **+ File share** and create a file share with the following settings:

Setting	Value
Name	**az30306a-share**
Quota	**1024**

Task 2: Map a drive to an Azure Storage file share from Windows

- [ ] 1. Select the newly created file share and select **Connect**.
- [ ] 2. On the **Connect** blade, ensure that the **Windows** tab is selected, and select **Copy to clipboard**.

> **Note**: Azure Storage file share mapping uses the storage account name and one of two storage account keys as the equivalents of user name and password, respectively in order to gain access to the target share.

- [ ] 3. In the LAB-VM, at the PowerShell prompt, paste and execute the script you copied.
- [ ] 4. Verify that the script completed successfully.
- [ ] 5. Start File Explorer, navigate to **Z:** drive and verify that the mapping was successful.
- [ ] 6. In File Explorer, create a folder named **Folder1** and a text file inside the folder named **File1.txt**.
- [ ] 7. Switch back to the browser window displaying the Azure portal, on the **az30306a-share** blade, select **Refresh**, and verify that **Folder1** appears in the list of folders.
- [ ] 8. Select **Folder1** and verify that **File1.txt** appears in the list of files.