Using the Azure CLI with Azure Storage

06/02/2017 12 minutes to read Contributors all

In this article

The open-source, cross-platform Azure CLI provides a set of commands for working with the Azure platform. It provides much of the same functionality found in the Azure portal, including rich data access.

In this guide, we show you how to use the Azure CLI to perform several tasks working with resources in your Azure Storage account. We recommend that you download and install or upgrade to the latest version of the CLI before using this guide.

The examples in the guide assume the use of the Bash shell on Ubuntu, but other platforms should perform similarly.

Versions of the Azure CLI

There are two versions of the Azure Command-Line Interface (CLI) currently available:

- Azure CLI: The current version of the Azure CLI, for use with the Resource Manager deployment model.
- Azure classic CLI: An older version of the Azure CLI, intended for use only with the classic deployment model.

Prerequisites

This guide assumes that you understand the basic concepts of Azure Storage. It also assumes that you're able to satisfy the account creation requirements that are specified below for Azure and the Storage service.

Accounts

- **Azure account**: If you don't already have an Azure subscription, create a free Azure account.
- **Storage account**: See Create a storage account in About Azure storage accounts.

Install the Azure CLI

Download and install the Azure CLI by following the instructions outlined in Install the Azure CLI.

> **Tip**
>
> If you have trouble with the installation, check out the **Installation Troubleshooting** section of the article, and the **Install Troubleshooting** guide on GitHub.

Working with the CLI

Once you've installed the CLI, you can use the `az` command in your command-line interface (Bash, Terminal, Command Prompt) to access the Azure CLI commands. Type the `az` command to see a full list of the base commands (the following example output has been truncated):

Copy

```
     /\
    /  \    _____   _ _ __ ___
   / /\ \  |_  / | | | \'__/ _ \
  / ____ \  / /| |_| | | |  __/
 /_/    \_\/___|\__,_|_|  \___|


Welcome to the cool new Azure CLI!

Here are the base commands:

    account          : Manage subscriptions.
    acr              : Manage Azure container registries.
    acs              : Manage Azure Container Services.
    ad               : Synchronize on-premises directories and manage Azure Active
Directory
                       resources.
    ...
```

In your command-line interface, execute the command `az storage --help` to list the `storage` command subgroups. The descriptions of the subgroups provide an overview of the functionality the Azure CLI provides for working with your storage resources.

Copy

```
Group
    az storage: Durable, highly available, and massively scalable cloud storage.
```

```
Subgroups:
    account  : Manage storage accounts.
    blob     : Object storage for unstructured data.
    container: Manage blob storage containers.
    cors     : Manage Storage service Cross-Origin Resource Sharing (CORS).
    directory: Manage file storage directories.
    entity   : Manage table storage entities.
    file     : File shares that use the standard SMB 3.0 protocol.
    logging  : Manage Storage service logging information.
    message  : Manage queue storage messages.
    metrics  : Manage Storage service metrics.
    queue    : Use queues to effectively scale applications according to traffic.
    share    : Manage file shares.
    table    : NoSQL key-value storage using semi-structured datasets.
```

Connect the CLI to your Azure subscription

To work with the resources in your Azure subscription, you must first log in to your Azure account with `az login`. There are several ways you can log in:

- **Interactive login**: `az login`
- **Log in with user name and password**: `az login -u johndoe@contoso.com -p VerySecret`
  - This doesn't work with Microsoft accounts or accounts that use multi-factor authentication.
- **Log in with a service principal**:
  ```
  az login --service-principal -u http://azure-cli-2016-08-05-14-31-15 -p VerySecret --
  tenant contoso.onmicrosoft.com
  ```

Azure CLI sample script

Next, we'll work with a small shell script that issues a few basic Azure CLI commands to interact with Azure Storage resources. The script first creates a new container in your storage account, then uploads an existing file (as a blob) to that container. It then lists all blobs in the container, and finally, downloads the file to a destination on your local computer that you specify.

bash Copy

```bash
#!/bin/bash
# A simple Azure Storage example script

export AZURE_STORAGE_ACCOUNT=<storage_account_name>
export AZURE_STORAGE_KEY=<storage_account_key>

export container_name=<container_name>
export blob_name=<blob_name>
export file_to_upload=<file_to_upload>
export destination_file=<destination_file>
```

```
echo "Creating the container..."
az storage container create --name $container_name

echo "Uploading the file..."
az storage blob upload --container-name $container_name --file $file_to_upload --name
$blob_name

echo "Listing the blobs..."
az storage blob list --container-name $container_name --output table

echo "Downloading the file..."
az storage blob download --container-name $container_name --name $blob_name --file
$destination_file --output table

echo "Done"
```

Configure and run the script

1. Open your favorite text editor, then copy and paste the preceding script into the editor.
2. Next, update the script's variables to reflect your configuration settings. Replace the following values as specified:
   - **<storage_account_name>** The name of your storage account.
   - **<storage_account_key>** The primary or secondary access key for your storage account.
   - **<container_name>** A name the new container to create, such as "azure-cli-sample-container".
   - **<blob_name>** A name for the destination blob in the container.
   - **<file_to_upload>** The path to small file on your local computer, such as "~/images/HelloWorld.png".
   - **<destination_file>** The destination file path, such as "~/downloadedImage.png".
3. After you've updated the necessary variables, save the script and exit your editor. The next steps assume you've named your script **my_storage_sample.sh**.
4. Mark the script as executable, if necessary: `chmod +x my_storage_sample.sh`
5. Execute the script. For example, in Bash: `./my_storage_sample.sh`

You should see output similar to the following, and the **<destination_file>** you specified in the script should appear on your local computer.

Copy

```
Creating the container...
{
  "created": true
}
Uploading the file...
Percent complete: %100.0
Listing the blobs...
Name       Blob Type      Length  Content Type              Last Modified
```

```
---------  -----------  --------  ------------------------  ------------------------
README.md  BlockBlob        6700  application/octet-stream  2017-05-12T20:54:59+00:00
Downloading the file...
Name
---------
README.md
Done
```

Tip

The preceding output is in **table** format. You can specify which output format to use by specifying the `--output` argument in your CLI commands, or set it globally using `az configure`.

Manage storage accounts

Create a new storage account

To use Azure Storage, you need a storage account. You can create a new Azure Storage account after you've configured your computer to connect to your subscription.

Azure CLI | Copy

```
az storage account create \
    --location <location> \
    --name <account_name> \
    --resource-group <resource_group> \
    --sku <account_sku>
```

- `--location` [Required]: Location. For example, "West US".
- `--name` [Required]: The storage account name. The name must be 3 to 24 characters in length, and use only lowercase alphanumeric characters.
- `--resource-group` [Required]: Name of resource group.
- `--sku` [Required]: The storage account SKU. Allowed values:
  - `Premium_LRS`
  - `Standard_GRS`
  - `Standard_LRS`
  - `Standard_RAGRS`
  - `Standard_ZRS`

Set default Azure storage account environment variables

You can have multiple storage accounts in your Azure subscription. To select one of them to use for all subsequent storage commands, you can set these environment variables:

First, display your storage account keys by using the az storage account keys list command:

Azure CLI | Copy | Try It

```
az storage account keys list \
    --account-name <account_name> \
    --resource-group <resource_group> \
    --output table
```

Now that you have the key, you can define it and the account name as environment variables:

Azure CLI | Copy

```
export AZURE_STORAGE_ACCOUNT=<account_name>
export AZURE_STORAGE_KEY=<key>
```

Another way to set a default storage account is by using a connection string. First, get the connection string with the `show-connection-string` command:

Azure CLI | Copy

```
az storage account show-connection-string \
    --name <account_name> \
    --resource-group <resource_group>
```

Then copy the output connection string and set the `AZURE_STORAGE_CONNECTION_STRING` environment variable (you might need to enclose the connection string in quotes):

Azure CLI | Copy

```
export AZURE_STORAGE_CONNECTION_STRING="<connection_string>"
```

> **Note**
>
> All examples in the following sections of this article assume that you've set the `AZURE_STORAGE_ACCOUNT` and `AZURE_STORAGE_KEY` environment variables.

Create and manage blobs

Azure Blob storage is a service for storing large amounts of unstructured data, such as text or binary data, that can be accessed from anywhere in the world via HTTP or HTTPS. This section assumes that you are already familiar with Azure Blob storage concepts. For detailed information, see Get started with Azure Blob storage using .NET and Blob Service Concepts.

Create a container

Every blob in Azure storage must be in a container. You can create a container by using the `az storage container create` command:

Azure CLI | Copy

```
az storage container create --name <container_name>
```

You can set one of three levels of read access for a new container by specifying the optional `--public-access` argument:

- `off` (default): Container data is private to the account owner.
- `blob` : Public read access for blobs.
- `container` : Public read and list access to the entire container.

For more information, see Manage anonymous read access to containers and blobs.

Upload a blob to a container

Azure Blob storage supports block, append, and page blobs. Upload blobs to a container by using the `blob upload` command:

Azure CLI | Copy

```
az storage blob upload \
    --file <local_file_path> \
    --container-name <container_name> \
    --name <blob_name>
```

If you would like to upload directly into a folder inside the container in your storage account, replace `--name <blob_name>` with `--name <folder/blob_name>` .

By default, the `blob upload` command uploads *.vhd files to page blobs, or block blobs otherwise. To specify another type when you upload a blob, you can use the `--type` argument--allowed values are `append` , `block` , and `page` .

For more information on the different blob types, see Understanding Block Blobs, Append Blobs, and Page Blobs.

Download a blob from a container

This example demonstrates how to download a blob from a container:

Azure CLI | Copy

```
az storage blob download \
    --container-name mycontainer \
    --name myblob.png \
    --file ~/mydownloadedblob.png
```

List the blobs in a container

List the blobs in a container with the az storage blob list command.

Azure CLI | Copy

```
az storage blob list \
    --container-name mycontainer \
    --output table
```

Copy blobs

You can copy blobs within or across storage accounts and regions asynchronously.

The following example demonstrates how to copy blobs from one storage account to another. We first create a container in the source storage account, specifying public read-access for its blobs. Next, we upload a file to the container, and finally, copy the blob from that container into a container in the destination storage account.

Azure CLI | Copy

```
# Create container in source account
az storage container create \
    --account-name sourceaccountname \
    --account-key sourceaccountkey \
    --name sourcecontainer \
    --public-access blob

# Upload blob to container in source account
az storage blob upload \
    --account-name sourceaccountname \
    --account-key sourceaccountkey \
    --container-name sourcecontainer \
    --file ~/Pictures/sourcefile.png \
    --name sourcefile.png

# Copy blob from source account to destination account (destcontainer must exist)
az storage blob copy start \
    --account-name destaccountname \
    --account-key destaccountkey \
    --destination-blob destfile.png \
    --destination-container destcontainer \
    --source-uri
https://sourceaccountname.blob.core.windows.net/sourcecontainer/sourcefile.png
```

In the above example, the destination container must already exist in the destination storage account for the copy operation to succeed. Additionally, the source blob specified in the `--source-uri` argument must either include a shared access signature (SAS) token, or be publicly accessible, as in this example.

Delete a blob

To delete a blob, use the `blob delete` command:

Azure CLI | Copy

```
az storage blob delete --container-name <container_name> --name <blob_name>
```

Create and manage file shares

Azure Files offers shared storage for applications using the Server Message Block (SMB) protocol. Microsoft Azure virtual machines and cloud services, as well as on-premises applications, can share file data via mounted shares. You can manage file shares and file data via the Azure CLI. For more information on Azure Files, see Introduction to Azure Files.

Create a file share

An Azure file share is an SMB file share in Azure. All directories and files must be created in a file share. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the capacity limits of the storage account. The following example creates a file share named **myshare**.

Azure CLI | Copy

```
az storage share create --name myshare
```

Create a directory

A directory provides a hierarchical structure in an Azure file share. The following example creates a directory named **myDir** in the file share.

Azure CLI | Copy

```
az storage directory create --name myDir --share-name myshare
```

A directory path can include multiple levels, for example **dir1/dir2**. However, you must ensure that all parent directories exist before creating a subdirectory. For example, for path **dir1/dir2**, you must first create directory **dir1**, then create directory **dir2**.

Upload a local file to a share

The following example uploads a file from **~/temp/samplefile.txt** to root of the **myshare** file share. The `--source` argument specifies the existing local file to upload.

Azure CLI | Copy

```
az storage file upload --share-name myshare --source ~/temp/samplefile.txt
```

As with directory creation, you can specify a directory path within the share to upload the file to an existing directory within the share:

Azure CLI | Copy

```
az storage file upload --share-name myshare/myDir --source ~/temp/samplefile.txt
```

A file in the share can be up to 1 TB in size.

List the files in a share

You can list files and directories in a share by using the `az storage file list` command:

Azure CLI | Copy

```
# List the files in the root of a share
az storage file list --share-name myshare --output table

# List the files in a directory within a share
az storage file list --share-name myshare/myDir --output table

# List the files in a path within a share
az storage file list --share-name myshare --path myDir/mySubDir/MySubDir2 --output table
```

Copy files

You can copy a file to another file, a file to a blob, or a blob to a file. For example, to copy a file to a directory in a different share:

Azure CLI | Copy

```
az storage file copy start \
--source-share share1 --source-path dir1/file.txt \
--destination-share share2 --destination-path dir2/file.txt
```

Create share snapshot

You can create a share snapshot by using the `az storage share snapshot` command:

cli

```
az storage share snapshot -n <share name>
```

Sample Output

JSON

```
{
  "metadata": {},
  "name": "<share name>",
  "properties": {
    "etag": "\"0x8D50B7F9A8D7F30\"",
    "lastModified": "2017-10-04T23:28:22+00:00",
    "quota": null
  },
  "snapshot": "2017-10-04T23:28:35.0000000Z"
}
```

List share snapshots

You may list share snapshots of a particular share using `az storage share list --include-snapshots`

cli

```
az storage share list --include-snapshots
```

Sample Output

JSON

```
[
  {
    "metadata": null,
    "name": "sharesnapshotdefs",
    "properties": {
      "etag": "\"0x8D50B5F4005C975\"",
      "lastModified": "2017-10-04T19:36:46+00:00",
      "quota": 5120
    },
    "snapshot": "2017-10-04T19:44:13.0000000Z"
  },
  {
    "metadata": null,
    "name": "sharesnapshotdefs",
    "properties": {
      "etag": "\"0x8D50B5F4005C975\"",
      "lastModified": "2017-10-04T19:36:46+00:00",
```

```
      "quota": 5120
    },
    "snapshot": "2017-10-04T19:45:18.0000000Z"
  },
  {
    "metadata": null,
    "name": "sharesnapshotdefs",
    "properties": {
      "etag": "\"0x8D50B5F4005C975\"",
      "lastModified": "2017-10-04T19:36:46+00:00",
      "quota": 5120
    },
    "snapshot": null
  }
]
```


Browse share snapshots

You may also browse into a particular share snapshot to view its content using `az storage file list`. One has to specify the share name `--share-name <snare name>` and the timestamp `--snapshot '2017-10-04T19:45:18.0000000Z'`

Azure CLI | Copy | Try It

```
az storage file list --share-name sharesnapshotdefs --snapshot '2017-10-04T19:45:18.0000000Z' -otable
```

Sample Output

Copy

```
Name           Content Length    Type    Last Modified
-------------  ----------------  ------  ---------------
HelloWorldDir/                   dir
IMG_0966.JPG   533568            file
IMG_1105.JPG   717711            file
IMG_1341.JPG   608459            file
IMG_1405.JPG   652156            file
IMG_1611.JPG   442671            file
IMG_1634.JPG   1495999           file
IMG_1635.JPG   974058            file
```

Restore from share snapshots

You can restore a file by copying or downloading a file from a share snapshot using `az storage file download` command

Azure CLI | Copy | Try It

```
az storage file download --path IMG_0966.JPG --share-name sharesnapshotdefs --snapshot
'2017-10-04T19:45:18.0000000Z'
```

Sample Output

Copy


```
{
  "content": null,
  "metadata": {},
  "name": "IMG_0966.JPG",
  "properties": {
    "contentLength": 533568,
    "contentRange": "bytes 0-533567/533568",
    "contentSettings": {
      "cacheControl": null,
      "contentDisposition": null,
      "contentEncoding": null,
      "contentLanguage": null,
      "contentType": "application/octet-stream"
    },
    "copy": {
      "completionTime": null,
      "id": null,
      "progress": null,
      "source": null,
      "status": null,
      "statusDescription": null
    },
    "etag": "\"0x8D50B5F49F7ACDF\"",
    "lastModified": "2017-10-04T19:37:03+00:00",
    "serverEncrypted": true
  }
}
```


Delete share snapshot

You can delete a share snapshot by using the `az storage share delete` command by providing `--snapshot` parameter with share snapshot timestamp:

cli

Copy

```
az storage share delete -n <share name> --snapshot '2017-10-04T23:28:35.0000000Z'
```

Sample Output

JSON

Copy

```
{
  "deleted": true
}
```

Next steps

Here are some additional resources for learning more about working with the Azure CLI.

- Get started with Azure CLI
- Azure CLI command reference
- Azure CLI on GitHub