NASUNI

Technical white paper

How to use AzCopy in Azure AI pipelines with Nasuni

Table of contents

Introduction 3

Prerequisites 4

Recommended reading 4

Examples 4

Copying PDF files from Nasuni to an Azure Blob container using AzCopy 4

Recursively copying PDF files from Nasuni to an Azure Blob container using AzCopy 8

Scheduling the use of AzCopy 10

Architecture 11

1. Azure Virtual Machines 11

2. Azure Container Instances (ACI) 11

3. Azure Functions 11

4. Azure Batch 11

5. Azure DevOps pipelines 11

Nasuni edge considerations 12

Next steps 12

Introduction

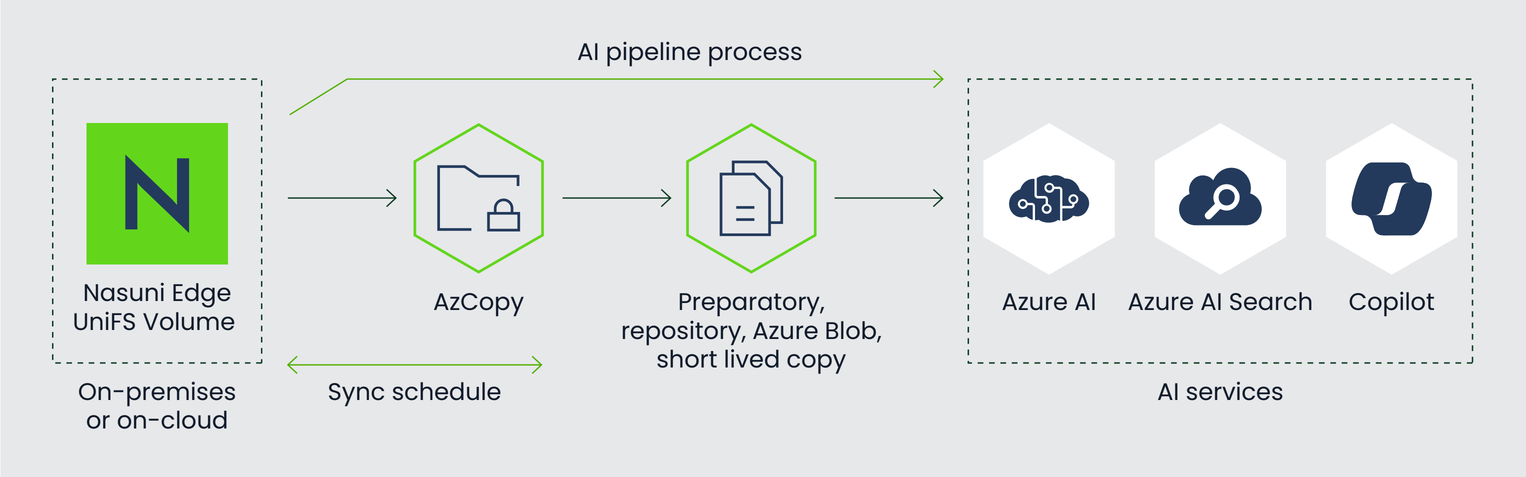
AzCopy is a powerful command-line utility provided and fully supported by Microsoft that enables efficient and secure data transfer between file systems and Azure Blob Storage, as well as between different Azure Storage accounts.

It can play a crucial role in AI pipelines by facilitating the movement of data, which is a fundamental aspect of building and deploying AI solutions.

AI pipelines often require large volumes of data to be ingested from various sources. AzCopy enables the easy transfer of data from Nasuni to Azure Blob Storage, which can serve as a central repository for an AI pipeline for any customer looking to leverage Azure AI tools.

Before feeding data into AI models, it often needs to be preprocessed and transformed. AzCopy can be used by companies to transfer data from Nasuni to a preprocessing container, enabling application of transformations, data cleanup, and format conversions before passing the data to the next stage of the AI pipeline. Any Nasuni volume is available to be accessed via the file interface from AzCopy.

AzCopy provides a command-line interface which can be easily integrated into automation scripts and CI/CD pipelines. AzCopy commands can be built into PowerShell scripts, Azure Functions, or other automation tools to automate data movement tasks in AI pipelines. This allows for the building of fully automated and repeatable AI workflows, ensuring consistency and reducing manual intervention.



Prerequisites

- For the purposes of the examples provided in this document, install AzCopy on a Windows machine (https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10)[1]
- An Azure Storage account with a container for storing PDF files
- Download Azure Storage Explorer for creating the SAS authentication token to access the storage resource (https://azure.microsoft.com/en-us/products/storage/storage-explorer/)

Recommended reading

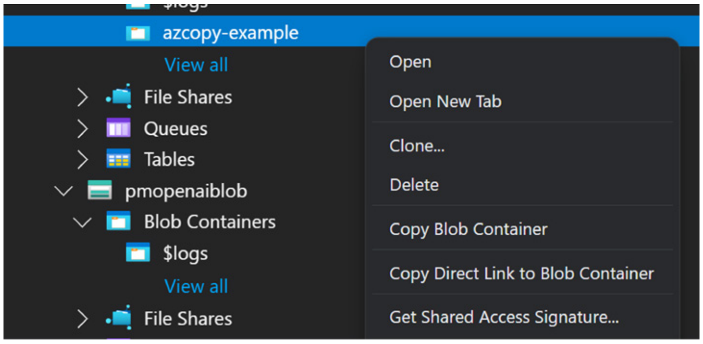
- Getting started with AzCopy
- Transfer data with AzCopy and file storage
- Optimize the performance of AzCopy v10 with Azure Storage | Microsoft Learn
- Troubleshoot problems with AzCopy (Azure Storage) - Azure | Microsoft Learn

Examples

Copying PDF files from Nasuni to an Azure Blob container using AzCopy

1. Ensure the Azure storage container is created for use as the centralized repository for files to be leveraged from an AI perspective.
2. Ensure installation of the Azure Storage Explorer and utilization of it to obtain a SAS token for accessing the container from an AzCopy command.[2]

To do this, login to Azure Storage Explorer and locate the storage container. Right click and choose **'Get shared access signature.'**



[1] This document focuses on Windows, but AzCopy can also be installed on Mac and Linux

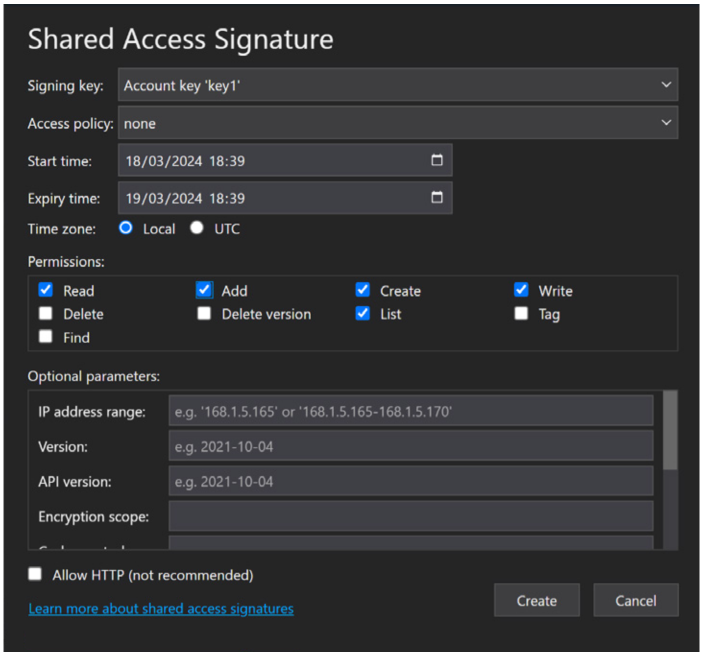
[2] SAS tokens can also be obtained via the Azure Portal

Ensure that the **'Read', 'Add', 'Create', 'Write' and 'List'** options are checked. Leave the signing key as is. Amend the start and expiration time as necessary and click **'Create'**.

3. A SAS Token and a URL will be provided, copy the URL.[3]

4. At this point, all necessary components are available to run AzCopy for copying files to the AI blob container:

```
.\azcopy copy 'N:\Nasuni Files\
Home Directories\jliddle\demo data\*.
pdf''https://yourstorageaccount.blob.
core.windows.net/yourcontainer?SAS_TOKEN
' --recursive=false --log-level=INFO
```



Shared Access Signature

Signing key: Account key 'key1'

Access policy: none

Start time: 18/03/2024 18:39

Expiry time: 19/03/2024 18:39

Time zone: ● Local ○ UTC

Permissions:

☑ Read ☑ Add ☑ Create ☑ Write
□ Delete □ Delete version ☑ List □ Tag
□ Find

Optional parameters:

IP address range: e.g. '168.1.5.165' or '168.1.5.165-168.1.5.170'

Version: e.g. 2021-10-04

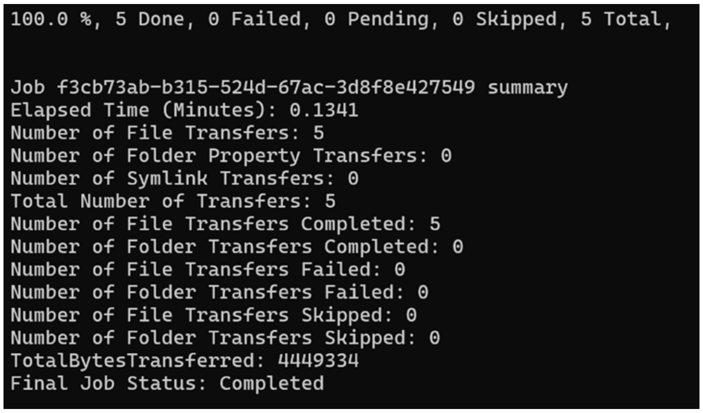
API version: e.g. 2021-10-04

Encryption scope:

□ Allow HTTP (not recommended)

Learn more about shared access signatures

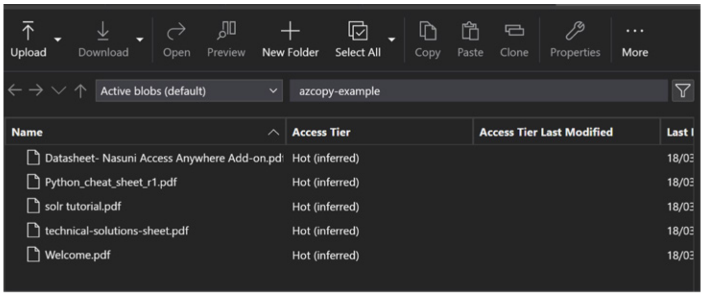
The log level enables the ability to watch and validate the process:



```
100.0 %, 5 Done, 0 Failed, 0 Pending, 0 Skipped, 5 Total,


Job f3cb73ab-b315-524d-67ac-3d8f8e427549 summary
Elapsed Time (Minutes): 0.1341
Number of File Transfers: 5
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 5
Number of File Transfers Completed: 5
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 0
Number of Folder Transfers Skipped: 0
TotalBytesTransferred: 4449334
Final Job Status: Completed
```

5. Checking the Azure Storage Explorer, it is possible to validate that the files have been transferred to the container, ready for pre-processing or use with AI services:



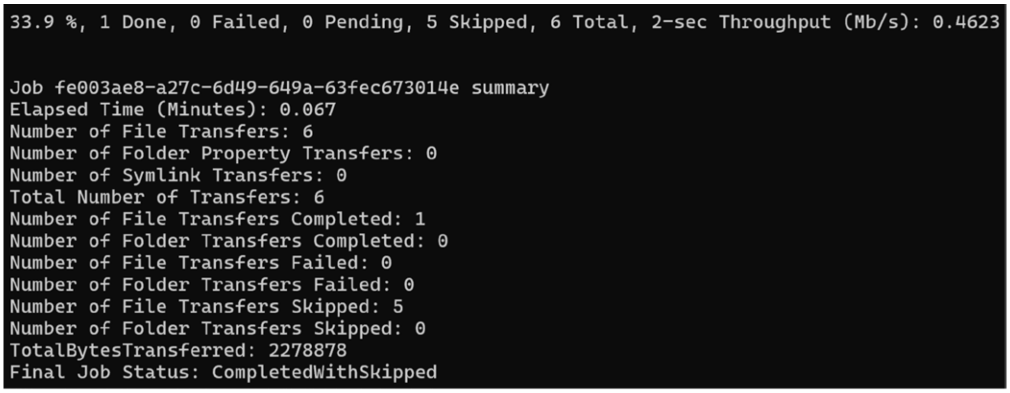
Name	Access Tier	Access Tier Last Modified	Last I
Datasheet- Nasuni Access Anywhere Add-on.pd	Hot (inferred)		18/03
Python_cheat_sheet_r1.pdf	Hot (inferred)		18/03
solr tutorial.pdf	Hot (inferred)		18/03
technical-solutions-sheet.pdf	Hot (inferred)		18/03
Welcome.pdf	Hot (inferred)		18/03

[3] An Account SAS can grant access to resources in one or more of the storage services (Blob, File, Queue, Table). It is signed with the storage account's key. A User Delegation SAS provides a way to delegate access to resources in Azure Blob Storage and Azure Files using Azure Active Directory (Azure AD) credentials. It is signed with Azure AD credentials through a user delegation key. It allows for similar permissions as an Account SAS but is scoped more securely since it uses Azure AD for authentication.

If new files are added, there is no need to re-add files that have not changed. The command can be adjusted slightly to ignore unchanged files and instruct AzCopy to only overwrite files if the source is newer:

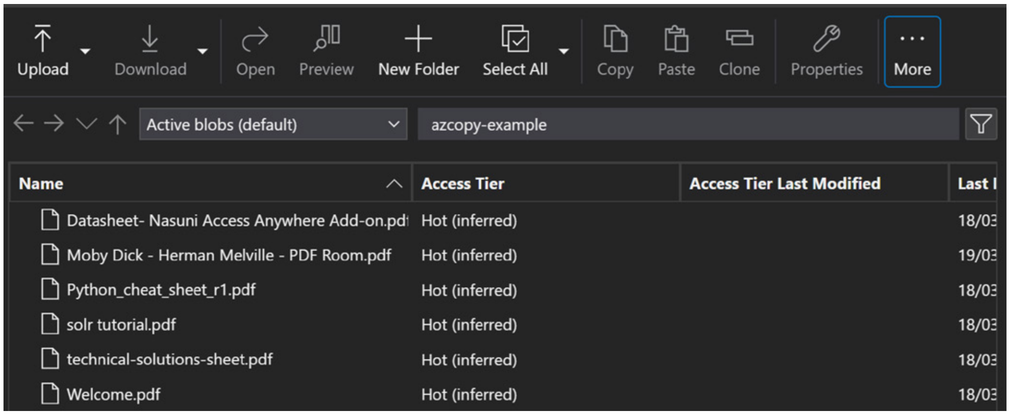
```
.\azcopy copy ‘N:\Nasuni Files\Home Directories\jliddle\
demo data\*.pdf’’https://yourstorageaccount.blob.core.
windows.net/yourcontainer?SAS_TOKEN ‘ --recursive=false –
overwrite=ifSourceNewer --log-level=INFO
```

If the AzCopy command is re-run with the additional flag, one new file that was added is picked up, while the rest are ignored:

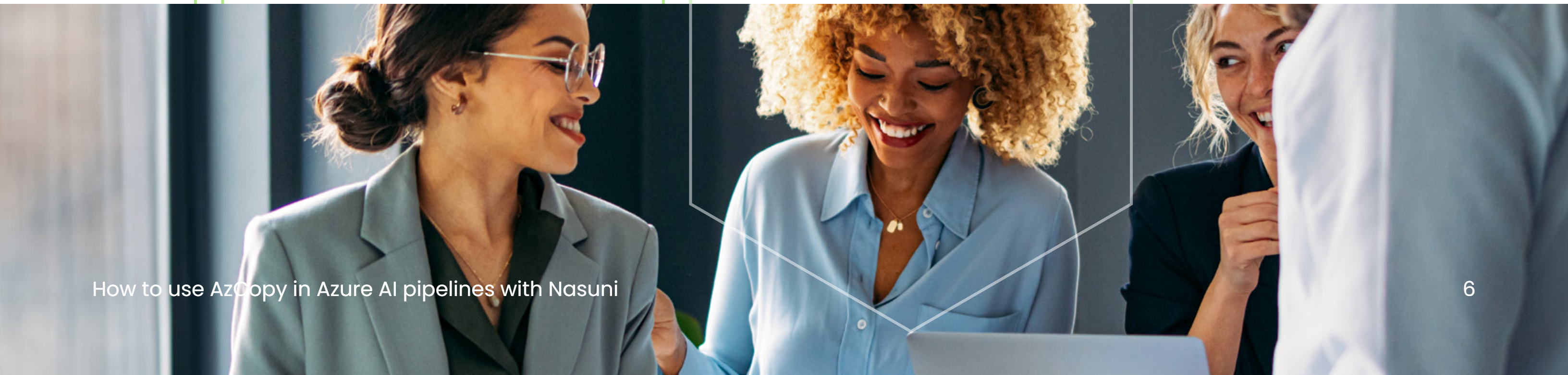


```
33.9 %, 1 Done, 0 Failed, 0 Pending, 5 Skipped, 6 Total, 2-sec Throughput (Mb/s): 0.4623


Job fe003ae8-a27c-6d49-649a-63fec673014e summary
Elapsed Time (Minutes): 0.067
Number of File Transfers: 6
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 6
Number of File Transfers Completed: 1
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 5
Number of Folder Transfers Skipped: 0
TotalBytesTransferred: 2278878
Final Job Status: CompletedWithSkipped
```



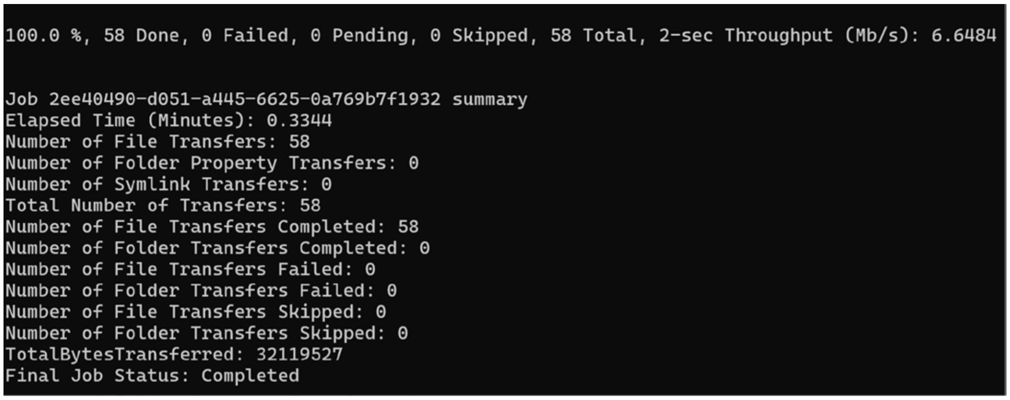
Name	Access Tier	Access Tier Last Modified	Last I
Datasheet- Nasuni Access Anywhere Add-on.pdf	Hot (inferred)		18/03
Moby Dick - Herman Melville - PDF Room.pdf	Hot (inferred)		19/03
Python_cheat_sheet_r1.pdf	Hot (inferred)		18/03
solr tutorial.pdf	Hot (inferred)		18/03
technical-solutions-sheet.pdf	Hot (inferred)		18/03
Welcome.pdf	Hot (inferred)		18/03



Other PDF files might be of interest in the sub directories of the source directory. If those are to be targeted, the AzCopy command can be modified:

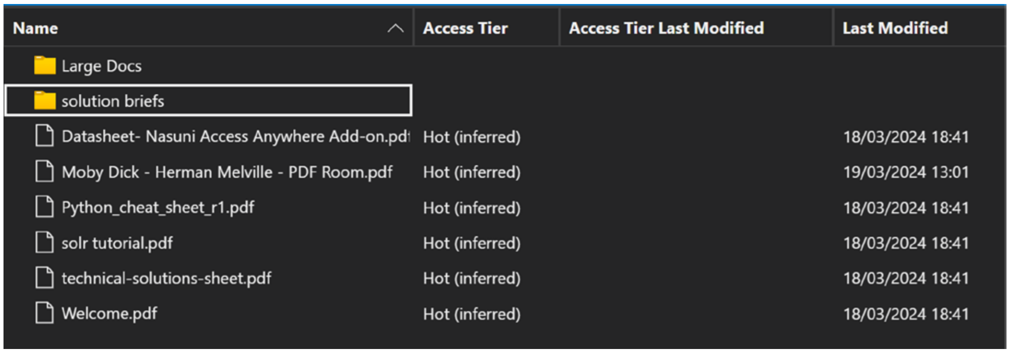
```
.\azcopy copy ‘N:\Nasuni Files\Home Directories\jliddle\
demo data\**\*.pdf’’https://yourstorageaccount.blob.core.
windows.net/yourcontainer?SAS_TOKEN ‘ --recursive=false –
overwrite=ifSourceNewer --log-level=INFO
```

The double asterisk (**) acts as a wildcard representing any number of subdirectories linked from the source directory.



```
100.0 %, 58 Done, 0 Failed, 0 Pending, 0 Skipped, 58 Total, 2-sec Throughput (Mb/s): 6.6484


Job 2ee40490-d051-a445-6625-0a769b7f1932 summary
Elapsed Time (Minutes): 0.3344
Number of File Transfers: 58
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 58
Number of File Transfers Completed: 58
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 0
Number of Folder Transfers Skipped: 0
TotalBytesTransferred: 32119527
Final Job Status: Completed
```



Name	Access Tier	Access Tier Last Modified	Last Modified
Large Docs			
solution briefs			
Datasheet- Nasuni Access Anywhere Add-on.pdf	Hot (inferred)		18/03/2024 18:41
Moby Dick - Herman Melville - PDF Room.pdf	Hot (inferred)		19/03/2024 13:01
Python_cheat_sheet_r1.pdf	Hot (inferred)		18/03/2024 18:41
solr tutorial.pdf	Hot (inferred)		18/03/2024 18:41
technical-solutions-sheet.pdf	Hot (inferred)		18/03/2024 18:41
Welcome.pdf	Hot (inferred)		18/03/2024 18:41

AzCopy can be used to fetch files that were modified after a particular date and time. This can be done by using the **‘include-after’** flag:

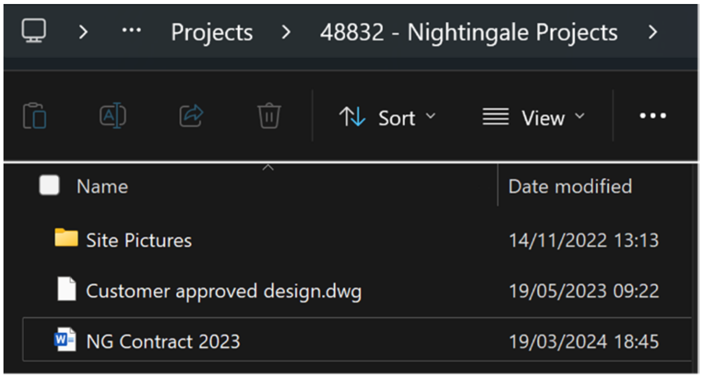
--include-after **<Date-Time-in-ISO-8601-format>**

This is another useful command to know for prepping the AI data repository with Nasuni data.

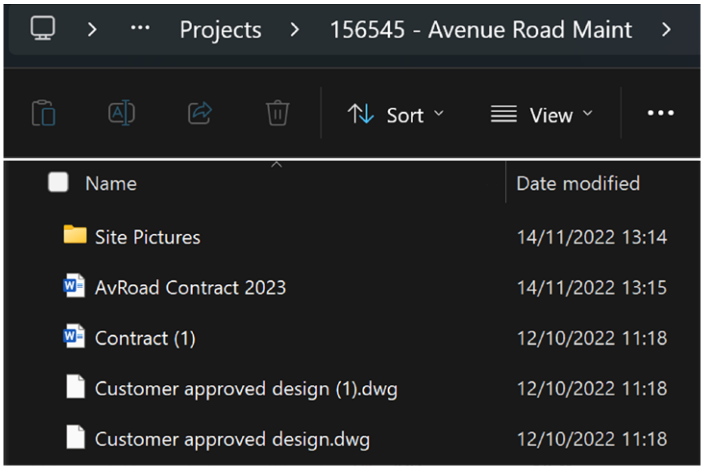
Recursively copying PDF files from Nasuni to an Azure Blob container using AzCopy

Prior examples only scanned PDF files in the nominated directory, which was subsequently extended to also scan sub directories. For AI datasets, what if there's a need to pattern match to only pull files that match a particular pattern? This is likely to be common when looking for specific data to use with AI or GenAI.

The example below recursively scans for files that look for word document files that contain both **'contract'** and **'2023'**. The example below will check recursively from the source directory for such files, of which there are two:



Name	Date modified
Site Pictures	14/11/2022 13:13
Customer approved design.dwg	19/05/2023 09:22
NG Contract 2023	19/03/2024 18:45



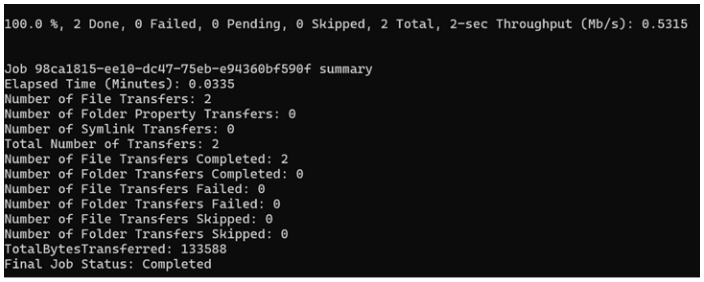
Name	Date modified
Site Pictures	14/11/2022 13:14
AvRoad Contract 2023	14/11/2022 13:15
Contract (1)	12/10/2022 11:18
Customer approved design (1).dwg	12/10/2022 11:18
Customer approved design.dwg	12/10/2022 11:18

No other contracts are of interest other than those labeled **'contract'** from **'2023'**.

AzCopy enables pattern match using the regex flag:

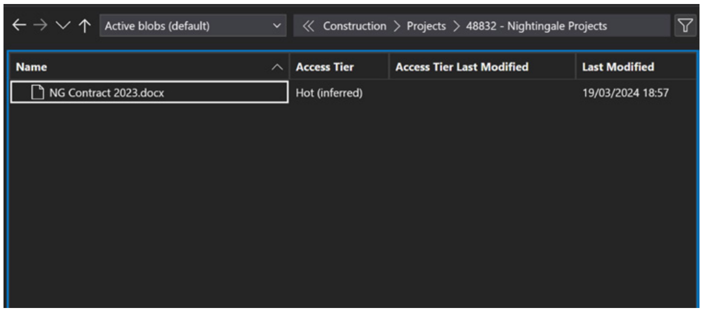
```
.\azcopy copy ‘N:\Nasuni Files\Home
Directories\jliddle\demo data’’https://
yourstorageaccount.blob.core.windows.net/
yourcontainer?SAS_TOKEN ‘ --recursive=true
--overwrite=ifSourceNewer --log-level=INFO
--include-regex “.*[Cc]ontract\s+2023\.
(docx|doc)$”
```

Running this command picks up the two contracts and transfers them across to the AI data repository:

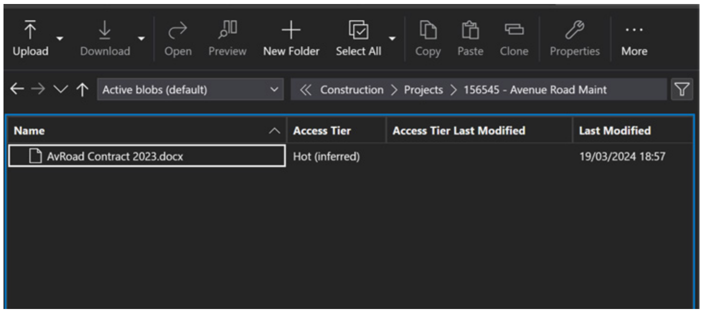


```
100.0 %, 2 Done, 0 Failed, 0 Pending, 0 Skipped, 2 Total, 2-sec Throughput (Mb/s): 0.5315


Job 98ca1815-ee10-dc47-75eb-e94360bf590f summary
Elapsed Time (Minutes): 0.0335
Number of File Transfers: 2
Number of Folder Property Transfers: 0
Number of Symlink Transfers: 0
Total Number of Transfers: 2
Number of File Transfers Completed: 2
Number of Folder Transfers Completed: 0
Number of File Transfers Failed: 0
Number of Folder Transfers Failed: 0
Number of File Transfers Skipped: 0
Number of Folder Transfers Skipped: 0
TotalBytesTransferred: 133588
Final Job Status: Completed
```



Name	Access Tier	Access Tier Last Modified	Last Modified
NG Contract 2023.docx	Hot (inferred)		19/03/2024 18:57



Name	Access Tier	Access Tier Last Modified	Last Modified
AvRoad Contract 2023.docx	Hot (inferred)		19/03/2024 18:57

At this point, the most common command for creating an AI pipeline with Nasuni data has been explored. Another AzCopy flag that may be useful is one that excludes a directory from being checked or processed.

In the example above, the **'old contracts'** directory could be excluded as follows:

```
.\azcopy copy ‘N:\Nasuni Files\Home Directories\jliddle\
demo data’’https://yourstorageaccount.blob.core.
windows.net/yourcontainer?SAS_TOKEN ‘ --recursive=true
--overwrite=ifSourceNewer --log-level=INFO --include-regex
“.*[Cc]ontract\s+2023\.(docx|doc)$” --exclude-path
“old contracts”
```

In the above example, the assumption is that the **'old contracts'** directory exists directly within the source directory (which it does). If it's in a subfolder, adjust the path (e.g., "subfolder/old contracts").

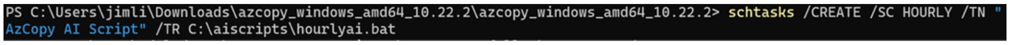
Scheduling the use of AzCopy

The final aspect to review is the scheduling of AzCopy so that an AI data pipeline can run unattended. This can be accomplished by encapsulating the desired AzCopy command in a batch file and saving it.

Then, use the SchTasks command from the command line to schedule the command to run.

For example:

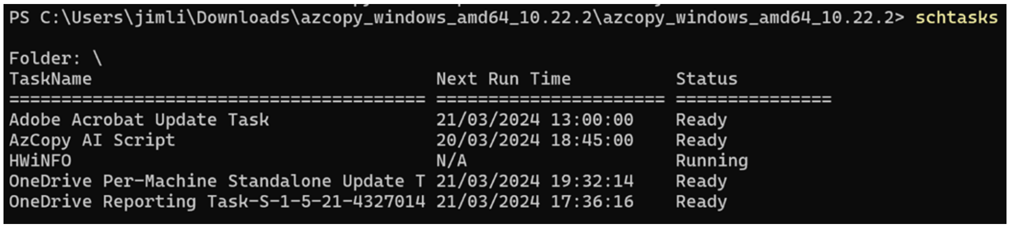
schtasks /CREATE /SC HOURLY /TN “AzCopy AI Script” /TR C:\aiscripts\hourlyai.bat



```
PS C:\Users\jimli\Downloads\azcopy_windows_amd64_10.22.2\azcopy_windows_amd64_10.22.2> schtasks /CREATE /SC HOURLY /TN "
AzCopy AI Script" /TR C:\aiscripts\hourlyai.bat
```

This will add the schedule to the scheduled task list and execute it hourly.

Check if it has been added by typing **’schtasks’** in the terminal window.



```
PS C:\Users\jimli\Downloads\azcopy_windows_amd64_10.22.2\azcopy_windows_amd64_10.22.2> schtasks

Folder: \
TaskName                                 Next Run Time          Status
======================================== ====================== ===============
Adobe Acrobat Update Task                21/03/2024 13:00:00    Ready
AzCopy AI Script                         20/03/2024 18:45:00    Ready
HWiNFO                                   N/A                    Running
OneDrive Per-Machine Standalone Update T 21/03/2024 19:32:14    Ready
OneDrive Reporting Task-S-1-5-21-4327014 21/03/2024 17:36:16    Ready
```

To delete the task, use:

schtasks /Delete /TN “AzCopy AI Script “ /F

** Note that if AzCopy is not being used from a desktop or an Azure-deployed virtual machine, the built-in scheduler would most likely be used for the deployment of infrastructure in use. For example: Azure functions may use time triggers or external triggers; Azure batch would attach a schedule; Azure Apps would use an external orchestrator to schedule the creation and the instance etc.

Architecture

To demonstrate AzCopy's use, examples have been shown running from a desktop against a Nasuni shared drive. However, for automated production use, deployment on the cloud is probable. Several architectures could be implemented to work with an Azure-deployed Nasuni edge. The choice of deployment method will depend on the use case and expertise with each of the technologies.

1 Azure Virtual Machines

This is the most similar environment to the examples shown in this document. Deploying AzCopy on an Azure Virtual Machine (VM) is a straightforward way to run AzCopy, against an Azure deployed Nasuni edge, within the Azure ecosystem. This approach gives full control over the environment and allows AzCopy to be used just as one would on a local machine but with the added benefits of Azure's scalability and the benefit of the virtual machine running in the Azure network.

2 Azure Container Instances (ACI)

For a lighter solution, Azure Container Instances (ACI) provides a serverless way to run containers without managing servers. One can containerize AzCopy, upload the container image to a registry (such as Azure Container Registry), and run it as an instance in ACI.

3 Azure Functions

For automated or event-driven tasks, Azure Functions can be used to execute AzCopy commands. Since Azure Functions supports custom containers, one can deploy a containerized version of AzCopy as a Function. This is ideal for integrating AzCopy operations into automated AI workflows or applications.

4 Azure Batch

Azure Batch is designed for running large-scale parallel and high-performance computing applications efficiently in the cloud. Azure Batch can be used to run AzCopy commands on multiple VMs for large AI data transfer jobs, managing resource allocation and job scheduling automatically.

5 Azure DevOps pipelines

If Azure DevOps is already in use for CI/CD, AzCopy can be integrated into the pipelines as part of deployment or AI data management tasks. This method is useful for automating data transfer as part of the build or release processes.

> This approach gives **full control over the environment** and allows AzCopy to be used just as one would on a local machine but with the **added benefits of Azure's scalability**

Nasuni edge considerations

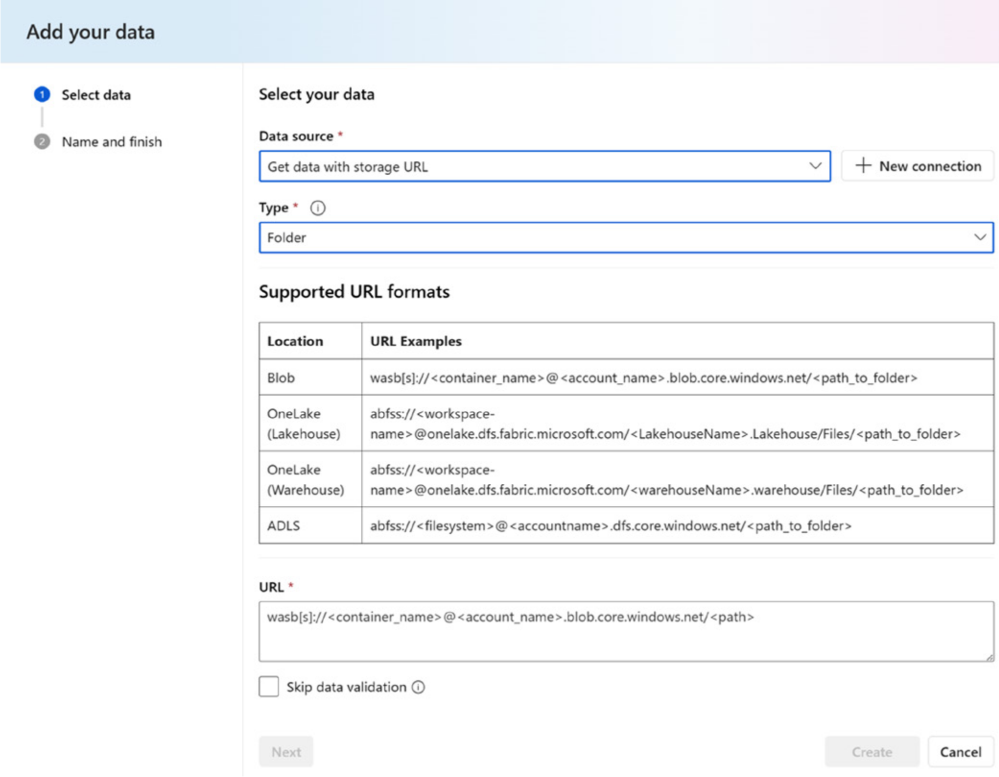
When creating an AI pipeline for use with an Azure-deployed Nasuni Edge, one may consider deploying a Nasuni Edge specifically for this purpose. Additionally, consideration should be given to pinning the file metadata to be used in the AI pipeline to the cache for speed and efficiency.

If further advice is required on the best way to configure the edge for specific AI data pipeline, reach out to the Nasuni team.

Next steps

Now that the AI dataset is in a blob container, consideration may be given to whether there is any other data to add to the AI data repository.

At this point, further curation on the dataset can be chosen, or indexing it for visibility using Azure AI Search. If satisfied that the dataset contains the required data, it is available for Azure AI Studio.



Add your data

Select your data

Data source *
Get data with storage URL

Type *
Folder

Supported URL formats

Location	URL Examples
Blob	wasb[s]://<container_name>@<account_name>.blob.core.windows.net/<path_to_folder>
OneLake (Lakehouse)	abfss://<workspace-name>@onelake.dfs.fabric.microsoft.com/<LakehouseName>.Lakehouse/Files/<path_to_folder>
OneLake (Warehouse)	abfss://<workspace-name>@onelake.dfs.fabric.microsoft.com/<warehouseName>.warehouse/Files/<path_to_folder>
ADLS	abfss://<filesystem>@<accountname>.dfs.core.windows.net/<path_to_folder>

URL *
wasb[s]://<container_name>@<account_name>.blob.core.windows.net/<path>

□ Skip data validation

Let’s talk

Want to find out more about how Nasuni can provide your business with a fluid data infrastructure designed for the hybrid cloud world?

Nasuni’s hybrid cloud platform unifies file and object data storage to deliver effortless scale and control at the network edge.

Learn more

Nasuni is a hybrid cloud storage solution designed specifically for scalability, security, and performance. The Nasuni File Data Platform is a cloud-native replacement for traditional network-attached storage (NAS) and file server infrastructure but with many more advanced capabilities. Nasuni consolidates file data in easily expandable, highly durable object storage such as Amazon S3, Azure Blob, and Google Cloud object storage at a fraction of the cost of on-premises or other cloud solutions. The Nasuni File Data Platform delivers effortless scalability, built-in security, and fast edge performance that increases business productivity, all within a single, unified administrative experience.

NASUNI