DS684
Cloud Computing
Week 03

Regarding Labs and Assignments

- Class participation means more than Zoom attendance. You must actively participate in the discussion and labs, and answer questions.

- Must hit Submit button, otherwise no grade
- If you need extension in time, must send written request (***email***). Otherwise no grade and no makeup. Requests sent over Zoom chat do not count.
- For any technical difficulty (installation, Azure access, etc), you must send written explanation (***email***) before the deadline. Otherwise no grade and no makeup.

Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

Importance of Storage

- Data are stored as files
- Data exchanges are (mostly) via files
- As a data engineer, your source data are likely files, and you normally will store your intermediate results as files.
- In recent years, more and more companies are storing their result data as files.

File Storage Solutions

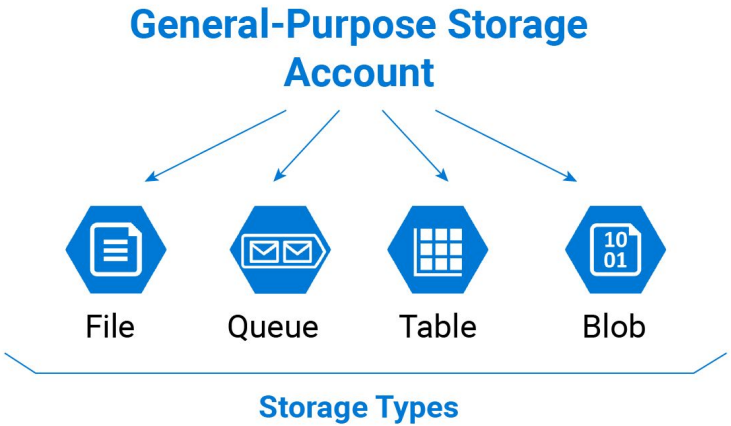
- Local hard disks: how to share with other users?
- File servers / share disk: limited volume, expensive
- Cloud file storage

Azure Storage Account

- An Azure storage account contains all of your Azure Storage data objects:
  - Storage account names must be between 3 and 24 characters in length and may contain numbers and lowercase letters only.
  - Your storage account name must be unique within Azure. No two storage accounts can have the same name.

Types of Azure Storage

- Types of Azure Storage:
  - Blobs: Cheaper, simpler, most widely used. What we will use in this class
  - Files: Usually for compatibility of business applications, such as serial read/write
  - Queues: Beyond the score of this class
  - Tables: Stores non-relational structured data. It is recommended to switch to Cosmos DB though



Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

Blob Storage

- Object, not file — Single Read/Write operation, no sequential operations such as append/update.

- Low-cost, tiered storage
- High availability
- Strong consistency
- Disaster recovery capabilities

Storage Container

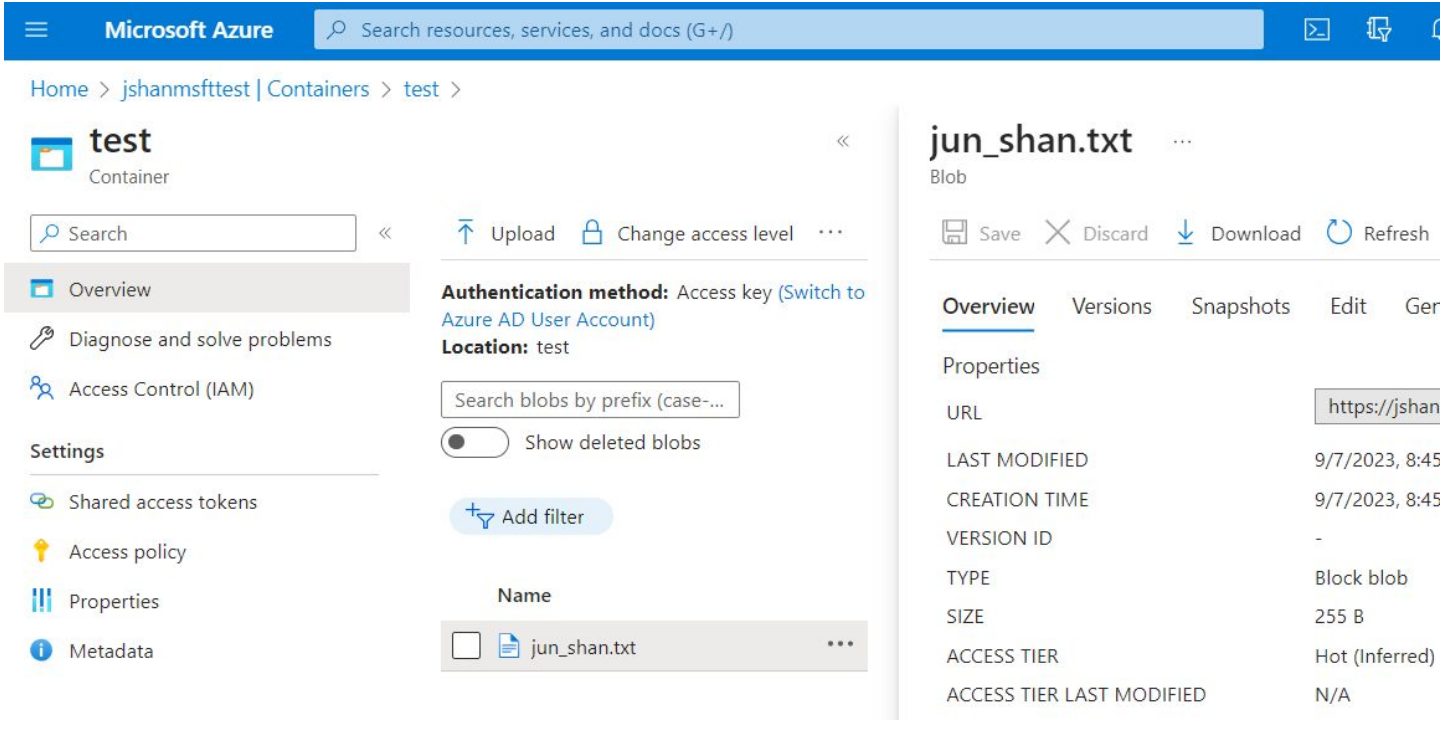
- Storage Account > Container > Blob
- Container: https://myaccount.blob.core.windows.net/mycontainer

  A container organizes a set of blobs, similar to a directory in a file system. A storage account can include an unlimited number of containers

  Not the container we discussed about in last session!

Blob

- Storage Account > Container > Blob



Blob Folder

- You can define folder for Blobs
- Storage Account - Container - folder - Blob forms the complete address of the blob

Blob Usage

Costs associated with blob storage

- Storage: charged by size
- Access: charged by # of access, plus special retrieval fee if applicable

General rule: The lower the storage cost, the higher the access cost

- That’s because the cheaper storage usually is more inconvenient to access

Tiered Storage

- Hot tier - accessed or modified frequently. The hot tier has the highest storage costs, but the lowest access costs.
- Cool tier - infrequently accessed or modified. Data in the cool tier should be stored for a minimum of 30 days. The cool tier has lower storage costs and higher access costs compared to the hot tier.
- Cold tier - infrequently accessed or modified. Data in the cold tier should be stored for a minimum of 90 days. The cold tier has lower storage costs and higher access costs compared to the cool tier.
- Archive tier - rarely accessed, and that has flexible latency requirements, on the order of hours. Data in the archive tier should be stored for a minimum of 180 days.

Lifecycle Management

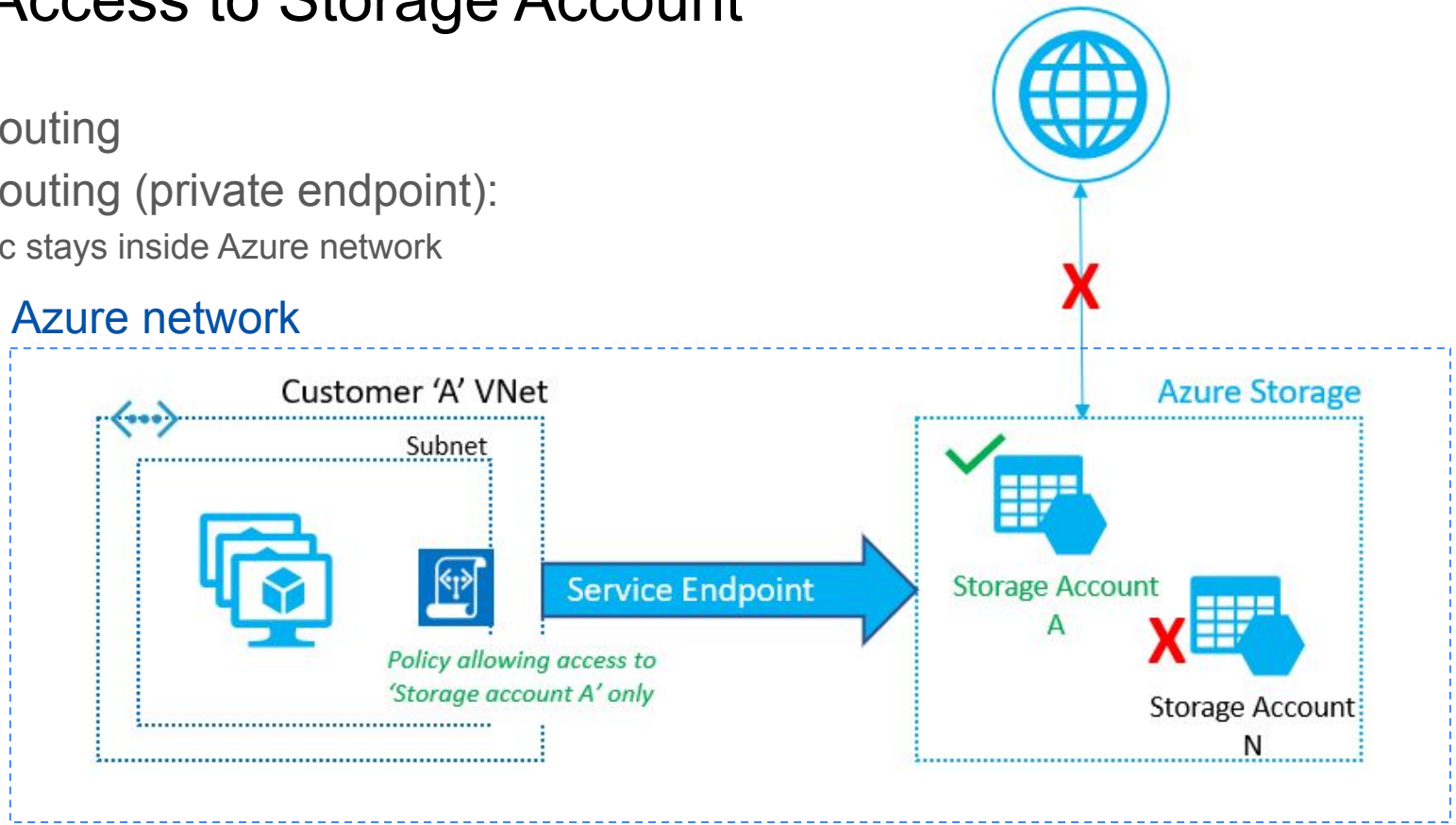
- Set on the storage account. Can be applied to containers or to a subset of blobs, using name prefixes or blob index tags as filters.
  - Transition current versions of a blob, previous versions of a blob, or blob snapshots to a cooler storage tier if these objects haven't been accessed or modified for a period of time, to optimize for cost.
  - Delete current versions of a blob, previous versions of a blob, or blob snapshots at the end of their lifecycles.
  - Define rules to be run once per day at the storage account level.

Privilege to Access Storage Account

- Role assignment of users/groups
- Role assignment of Managed Identity
  - Automatically managed identity in Azure Active Directory (Azure AD) for applications (virtual machines, services, etc.)
- Access key

Network Access to Storage Account

- Public Routing
- Private routing (private endpoint):
  - Traffic stays inside Azure network



Storage Demo

DEMO: Creating a container. Upload a blob.

Storage Account > Container > Blob

Tier, Lifecycle, Access Key, SAS, and IAM

Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

Blob Storage Lab

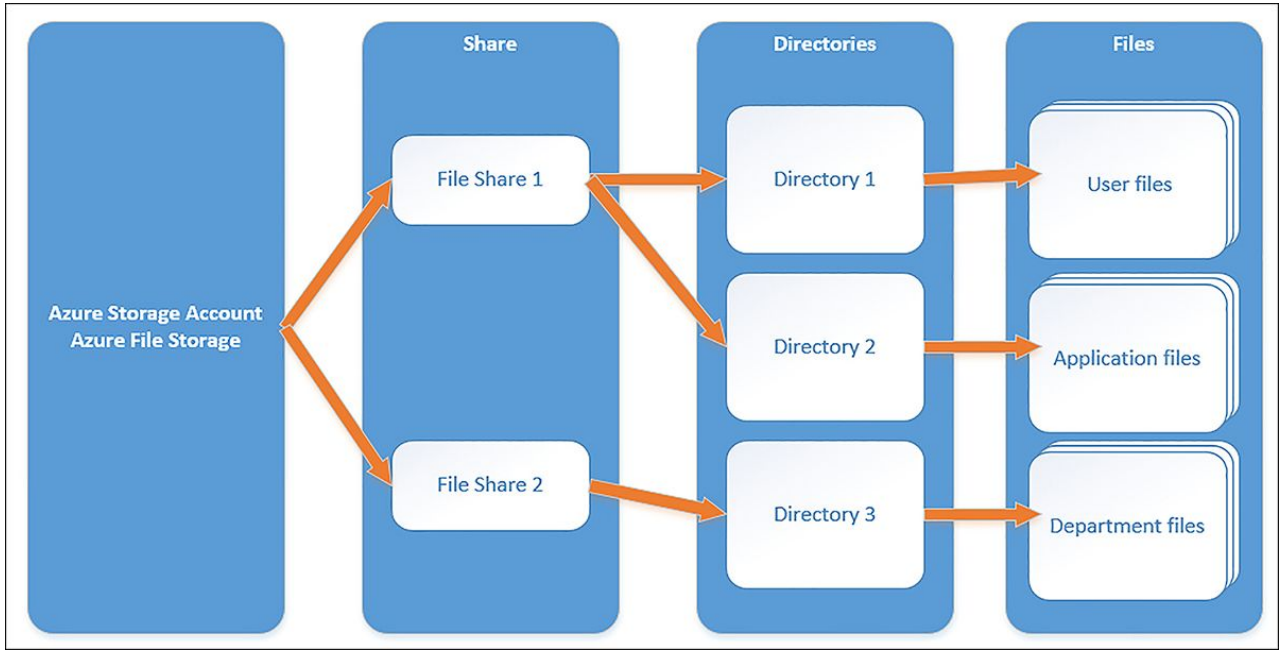
- Create Storage Account > Container > Blob in your own Azure account
- Upload a file

Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

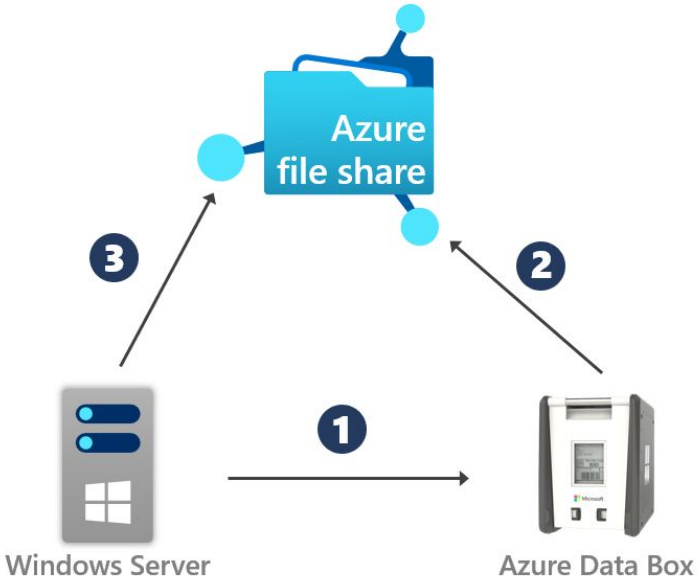
File Storage

- Cloud file server
- Can be mapped to virtual machine drives, and shared by multiple virtual machines



File Storage

- Can sync with on-premise file server
- Useful to serve as the central hub of enterprise file system



Blob Storage vs File Storage

- Why file storage: more convenient
  - Business files such as Word doc or Excel spreadsheet
- Why blob storage: It is cheaper
  - Large volumes such as data files, media files, etc.
  - Files requires long term archiving

Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

Accessing Storage Account

- Portal
- CLI/Powershell
- Programming code

Access Storage Account from Python

Install Azure packages in your Anaconda: azure-storage-blob

Access file on Azure storage from your Jupyter Notebook (or python console)

```
pip install azure-storage-blob
python

from azure.storage.blob import BlobClient

blob =
BlobClient(account_url="https://<mystorageaccount>.blob.core.windows.net",
                  container_name="<mycontainer>",
                  blob_name="<blob.txt>",
                  credential="myaccesskey1")

with open("example.csv", "wb") as f:
    data = blob.download_blob()
    data.readinto(f)
```

Agenda

- Azure Storage Account
  - Blob storage
  - Lab: Blob storage access via portal
  - File storage
  - Accessing storage account
  - Lab: Blob storage access via python

Lab

Install Azure packages in your Anaconda: azure-storage-blob

Access file on Azure storage from your Jupyter Notebook (or python console)

```
pip install azure-storage-blob
python

from azure.storage.blob import BlobClient

blob =
BlobClient(account_url="https://<mystorageaccount>.blob.core.windows.net",
                  container_name="<mycontainer>",
                  blob_name="<blob.txt>",
                  credential="myaccesskey1")

with open("example.csv", "wb") as f:
    data = blob.download_blob()
    data.readinto(f)
```

Assignment