

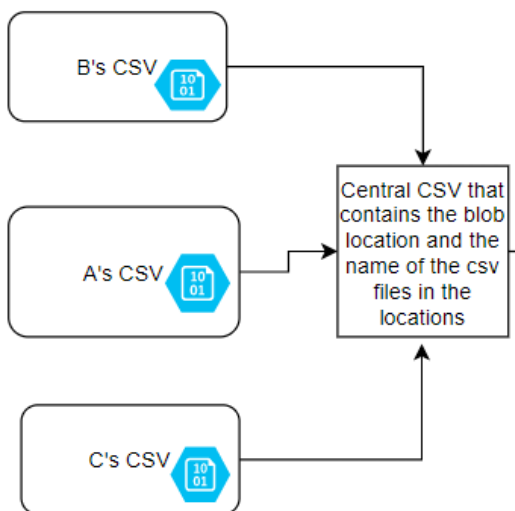
## How to Create Blob API using Python

### Problem Statement-

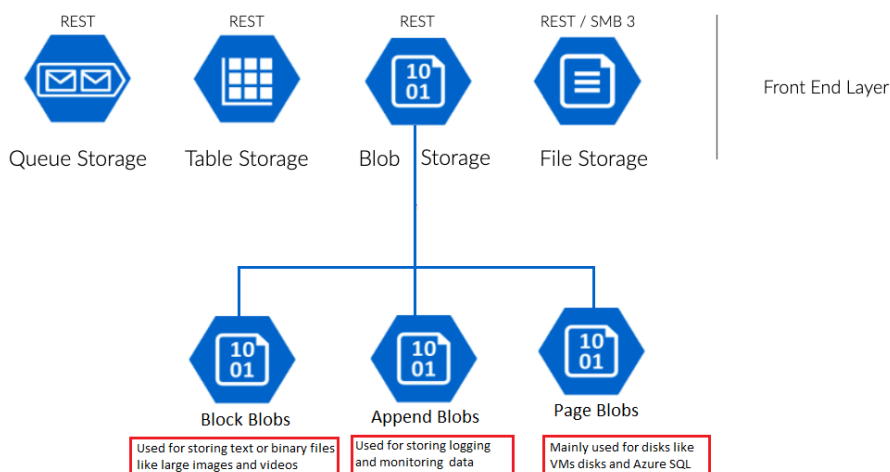
Since we were facing more code and management issues to handle the blob storage we decided to do the automation by creating API using python. So whenever we need any Blob storage operations we can call the API

My client works a lot with trainers, and we have a lot of train the trainer videos, that various trainers associated with the organisation contribute to us. The challenge assigned for me was to setup a system that would function as a repository of all the video objects that one could upload, and then output to a browser for someone to watch a train the trainer video on demand.

### Solution



## Azure Storage Architecture



This was a fairly simple thing to do, with a little bit of Python and Azure magic, put together.

Firstly, I wish to introduce you to the concept of a Blob. Specifically, Azure Blob

A Blob or BLOB means “Binary Large Object”, it is a collection of binary data stored as a single entity. Blobs can be of images, audio or any other multimedia objects. It is ideal to store heavy files or multimedia or unstructured data for the web.

Azure Blob storage is a service by Microsoft’s cloud Azure and a one stop solution to store high density and volume of Blobs on the cloud service in an optimal fashion. Blob storage provides users with strong data consistency, storage and access flexibility that adapts to the user’s needs, and it also provides high availability by implementing geo-replication. Objects in Blob storage can be accessed using Azure Storage REST API, Azure CLI or an Azure Storage Client Library. Azure provides Blob library support for the following languages :-

1. .NET
2. Java
3. Node.js
4. Python
5. Go
6. PHP
7. Ruby

Let us now see how, using a python script, we tap into the power of Microsoft Azure to create containers, blobs and fetch them for different projects. How to work with blobs in Microsoft Azure using python script and learn the basic commands to interact with Azure like creating blob container, uploading , downloading blobs and so on.

### **Tech Details and Implementation of Solution**

Make sure you have the following going for you, before you start this journey.

1. Active Microsoft Azure Account
2. Python 3.6 or Later
3. Have pip installed
4. Visual Studio Code or your preferred text editor
5. Azure CLI installed

Once these are in place, step into this journey I have shared. It’s fairly simple, and straightforward.

### **Steps-**

1. Create a folder for your project or select the destination where you want to use BLOB service by Azure for
2. If not setup yet download and install Azure CLI for your operating system
3. Open a new terminal or command line and install Azure Storage Blobs client library for python using pip installation. Run the following command:

\$ pip install azure-storage-blob

```
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> pip install azure-storage-blob
Defaulting to user installation because normal site-packages is not writeable
Collecting azure-storage-blob
  Downloading azure_storage_blob-12.13.1-py3-none-any.whl (377 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 377.4/377.4 KB 7.8 MB/s eta 0:00:00
Collecting cryptography>=2.1.4
  Downloading cryptography-38.0.1-cp36-abi3-win_amd64.whl (2.4 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.4 MB 1.1 MB/s eta 0:00:01
```

4. You should get the following output on successful installation

```
2.1.7)
Installing collected packages: typing-extensions, isodate, cryptography, azure-core, msrest, a
zure-storage-blob
Successfully installed azure-core-1.25.1 azure-storage-blob-12.13.1 cryptography-38.0.1 isodat
e-0.6.1 msrest-0.7.1 typing-extensions-4.3.0
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Program Files\Python310\python.exe -m pip install --
upgrade pip' command.
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython>
```

5. Now that the azure storage blob library is installed for python lets create a new storage account to use in project using terminal or Azure CLI.

a. Run the following command to create a new resource group for this project to hold the storage account:

\$ az group create --name azureBlobTest --location westus2

```
read more about the command in reference docs
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> az group create --name azureBlobTest --location westus2
{
  "id": "/subscriptions/b867379a-8972-4468-aaaa-d91009d63503/resourceGroups/azureBlobTest",
  "location": "westus2",
  "managedBy": null,
  "name": "azureBlobTest",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

b. Run the following command to create the storage account in that resource group:

\$ az storage account create -n storageblobtestnewazure -g azureBlobTest

```
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> az storage account create -n storageblobtestnewazure -g azureBlobTest
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
  "allowCrossTenantReplication": null,
  "allowSharedKeyAccess": null,
  "allowedCopyScope": null,
  "azureFilesIdentityBasedAuthentication": null,
  "blobRestoreStatus": null,
  "creationTime": "2022-09-29T14:58:24.623034+00:00",
  "customDomain": null,
  "defaultToOAuthAuthentication": null,
  "dnsEndpointType": null,
  "enableHttpsTrafficOnly": true,
  "enableNfsV3": null,
  "encryption": {
    "encryptionIdentity": null,
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "requireInfrastructureEncryption": null,
    "services": {
      "blob": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-09-29T14:58:25.326135+00:00"
      },
      "file": {
        "enabled": true,
        "keyType": "Account",
        "lastEnabledTime": "2022-09-29T14:58:25.326135+00:00"
      },
      "queue": null,
      "table": null
    }
  }
}
```

NOTE – Make sure that the storage account name is unique and lowercase letters

We have successfully created a storage account on Azure for BLOB

6. Let's move onto the python side of the project.

To Create a client using Azure Storage Blobs client library for python we will need the storage accounts blob service account URL and a credential which will allow you to access the storage resource.

- a. Create a client.py file in the root of the project directory and enter the following code

```
from azure.storage.blob import BlobServiceClient

service = BlobServiceClient(account_url="https://<storage-account-name>.blob.core.windows.net/", credential=credential)
```

- b. To Look up the blob service account URL for the storage run the following command in terminal:  
\$ az storage account show -n my-storage-account-name -g my-resource-group --query "primaryEndpoints.blob"

```
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> az storage account show -n storageblobtestnewazure -g
azureBlobTest --query "primaryEndpoints.blob"
"https://storageblobtestnewazure.blob.core.windows.net/"
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython>
```

Now that we have the account URL, we need the credential parameter

- c. We can use a storage account shared key or access key to authenticate our client. To get the key as string , run the following command-

```
$ az storage account keys list -g MyResourceGroup -n MyStorageAccount
```

```
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> az storage account keys list -g azureBlobTest -n stor
ageblobtestnewazure
[
  {
    "creationTime": "2022-09-29T14:58:24.716733+00:00",
    "keyName": "key1",
    "permissions": "FULL",
    "value": "qGMgf8tEcWpYn0tavitMjsH8sBQv50Ly09egB80vT91zJC7cpYtjWvn2nANFxb2oFBiV00jjsDPc+AS
tyXzK8w=="
  },
  {
    "creationTime": "2022-09-29T14:58:24.716733+00:00",
    "keyName": "key2",
    "permissions": "FULL",
    "value": "Lfn0xuJjcYw10P+oEPrFIAXwSsmEawj21v0o5WEVw82gxcwp31pPzTwqjHB0IRXqBDHJSxXA1N8R+AS
tn7eyEw=="
  }
]
```

We can use this Key received to authenticate our client.

- d. Replace the “credential” parameter with the key. Replace the code in client.py like this:

```
from azure.storage.blob import BlobServiceClient

service =
BlobServiceClient(account_url="https://<my_account_name>.blob.core.windo
ws.net", credential="<account_access_key>")
```

We have successfully created the client to handle the authentication

- 7. Get the connection string to the Azure blob storage account. Run the following command

```
$ az storage account show-connection-string -g MyResourceGroup -n
MyStorageAccount
```

```
Message: Resource group 'MyResourceGroup' could not be found.
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> az storage account show-connection-string -g azureBlob
Test -n storageblobtestnewazure
{
  "connectionString": "DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=stor
ageblobtestnewazure;AccountKey=qGMgf8tEcWpYn0tavitMjsH8sBQv50Ly09egB80vT91zJC7cpYtjWvn2nANFxb2oFBiV00j
jsDPc+AStyXzK8w==;BlobEndpoint=https://storageblobtestnewazure.blob.core.windows.net/;FileEndpoint=htt
ps://storageblobtestnewazure.file.core.windows.net/;QueueEndpoint=https://storageblobtestnewazure.queu
e.core.windows.net/;TableEndpoint=https://storageblobtestnewazure.table.core.windows.net/"
}
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython>
```

Save the output string received. This string is the connection string which will be used in the other commands to connect and authenticate the service.

8. We can now start using the BLOB storage account in our code using the connection string and azure blob python library.
9. Let start by creating a container where we can upload or download blobs.  
To create a container for blob storage, create a container.py file and enter the following code in it.

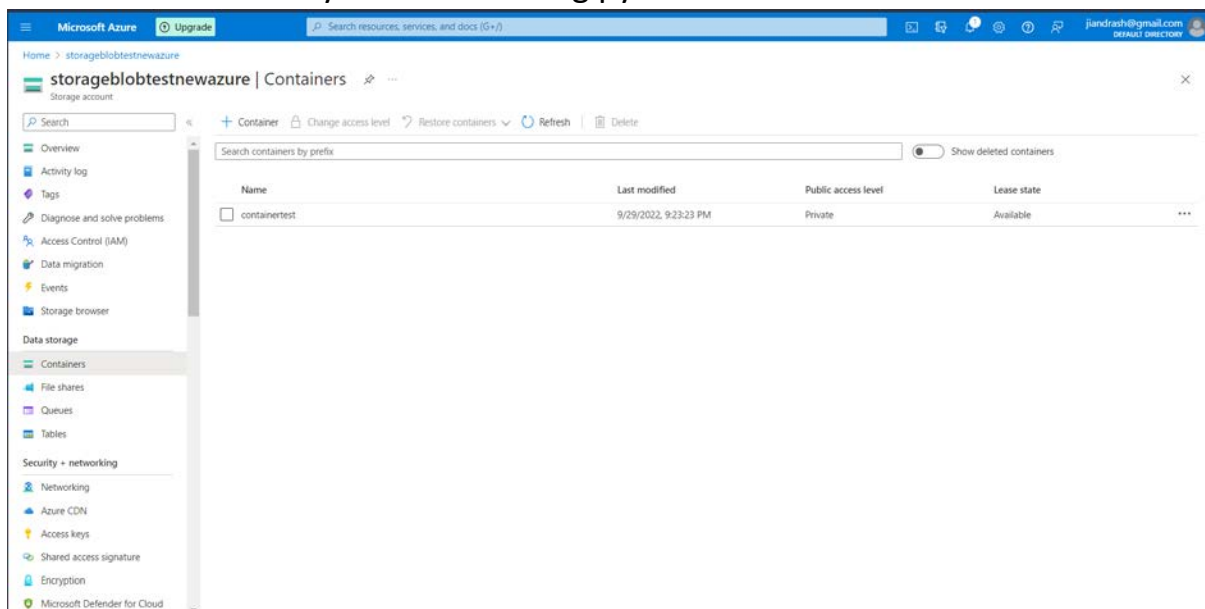
```
from azure.storage.blob import ContainerClient

container_client =
ContainerClient.from_connection_string(conn_str="<connection_string>",
container_name="my_container")

container_client.create_container()
```

10. Run the python code using the following command to create the container on storage account in Azure  
\$ python container.py

11. To check the container, navigate to Microsoft azure portal and go to storage accounts><created storage account> and then open the containers tab from the left to see the container you created using python



12. Now that we have created a container for our blob on Azure lets move on by uploading a blob to the container using python  
To upload a blob using python , create a uploadBlob.py file and enter the following code :

```
from azure.storage.blob import BlobClient

blob =
```

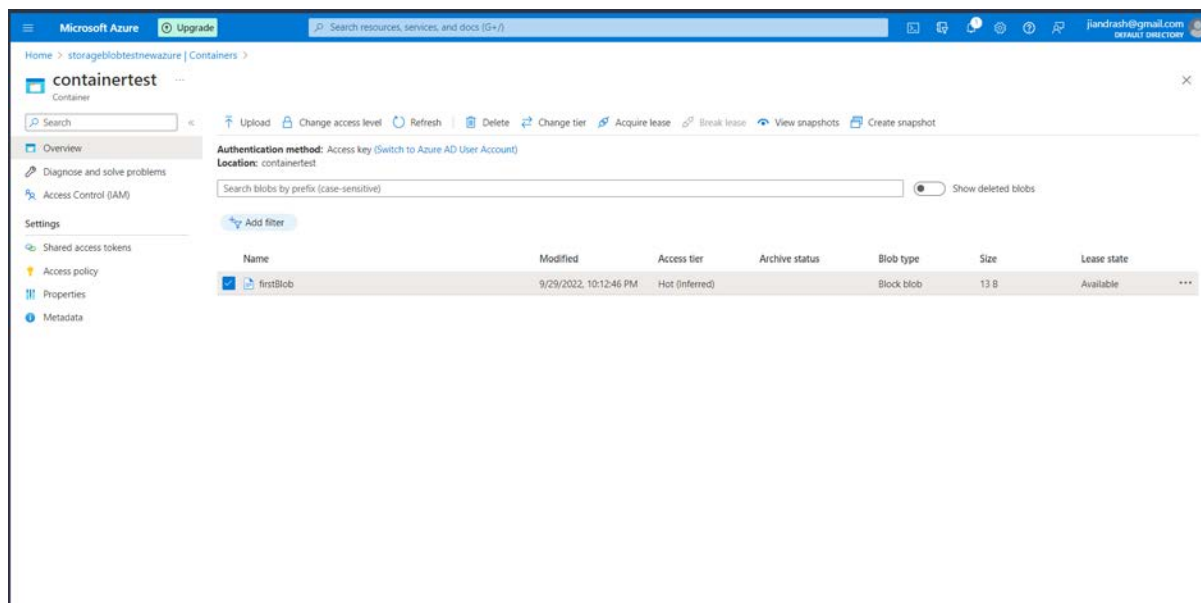
```
BlobClient.from_connection_string(conn_str="<connection_string>",
container_name="my_container", blob_name="my_blob")
```

```
with open("./SampleSource.txt", "rb") as data:
    blob.upload_blob(data)
```

13. Run the uploadBlob.py using the following command:

```
$ python uploadBlob.py
```

14. To check the uploaded file, click on the container we created in Microsoft Azure Portal. You will be able to see the BLOB uploaded and its details such as timestamp and access tier etc.



15. Now that we have successfully uploaded a blob to the container, let's try to fetch or download the blob using python.

To download a blob using python, create a python file named "downloadBlob.py" and enter the following code:

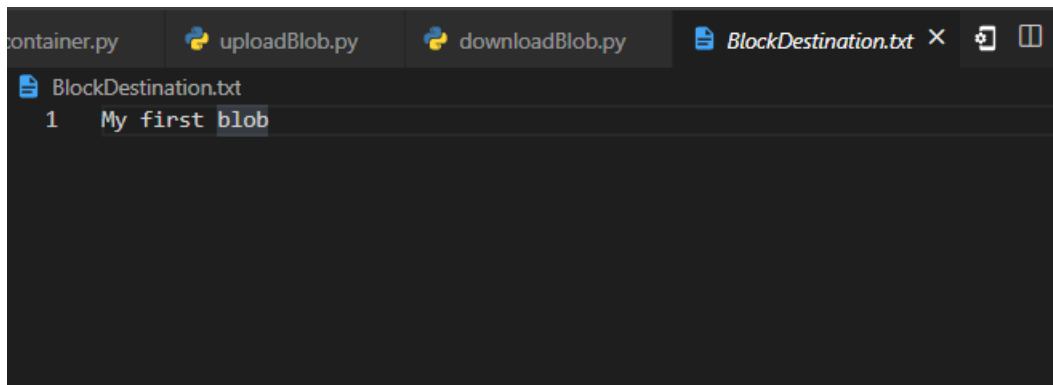
```
from azure.storage.blob import BlobClient

blob =
BlobClient.from_connection_string(conn_str="my_connection_string",
container_name="my_container", blob_name="my_blob")

with open("./BlockDestination.txt", "wb") as my_blob:
    blob_data = blob.download_blob()
    blob_data.readinto(my_blob)
```

16. Run the downloadBlob.py using the following command:

```
$ python downloadBlob.py
```



```
BlockDestination.txt
1 My first blob
```

As you can see the blob we uploaded previously has been downloaded as well successfully

Let's get a list of all the blobs available in the container using python.

To get a list of blobs in the container using python, create a python file named "listBlobs.py" and enter the following code:

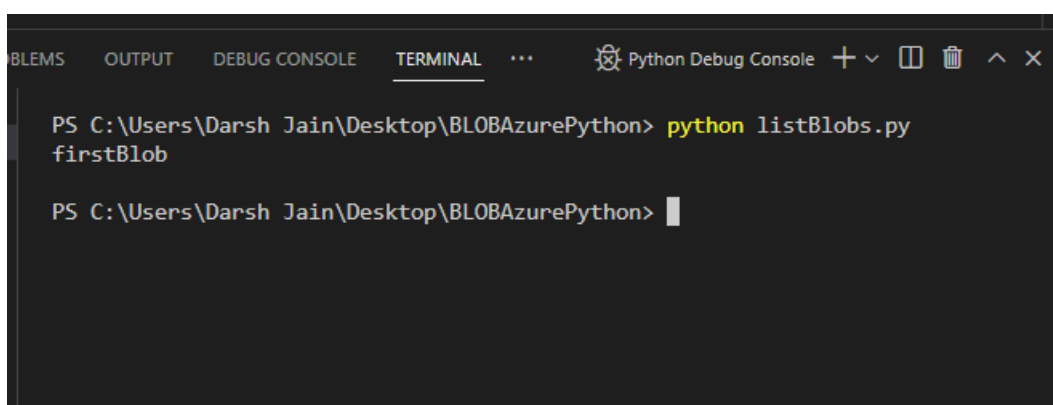
```
from azure.storage.blob import ContainerClient

container =
ContainerClient.from_connection_string(conn_str="my_connection_string",
container_name="my_container")

blob_list = container.list_blobs()
for blob in blob_list:
    print(blob.name + '\n')
```

17. Run the listBlob.py using the following command:

```
$ python listBlob.py
```



```
PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython> python listBlobs.py
firstBlob

PS C:\Users\Darsh Jain\Desktop\BLOBAzurePython>
```

As you can see, we received the uploaded blobs from Azure successfully

## Knowledge Sharing:

Troubleshooting

Storage Blob clients raise exceptions defined in Azure Core.



This list can be used for reference to catch thrown exceptions. To get the specific error code of the exception, use the `error_code` attribute, i.e, `exception.error_code`.

## Logging

This library uses the standard logging library for logging. Basic information about HTTP sessions (URLs, headers, etc.) is logged at INFO level.

Detailed DEBUG level logging, including request/response bodies and unredacted headers, can be enabled on a client with the `logging_enable` argument:

Similarly, `logging_enable` can enable detailed logging for a single operation, even when it isn't enabled for the client:

```
service_client.get_service_stats(logging_enable=True)
```

## Challenges

Applications with complex System Design and heterogeneous Language and framework may face integration issue of the API, Such systems will end up as a huge mess, if you don't plan for a content structure, or rules for content aggregation to come together, before executing an assignment like – say -an OTT. It can quickly get messy if you don't anticipate requirements and plan for them, right from the time you think up how and why content is going to be consumed and served.

## Business Benefits-

For most of you who are asked to do multimedia projects like maybe an OTT, or work with a system where you handle a huge volume of documents, don't look any further.

Blob storage is ideal for:

- Serving images or documents directly to the website and hence browser without the need of server
- Storing files for distributed access optimally
- Streaming heavy video and audio
- Automation of Workflows
- Less Management
- Cost Efficient
- Storing data backups and restore, disaster recovery, and archiving

In the internet world most companies rely on data and usually have enormous amounts of unstructured data including logs, files, images, videos. Hence with Microsoft Azure's Blob

storage service these companies can store their data and use the service with other services for various needs such as data mining or warehousing or serving the customer needs, the possibilities are limitless with Azure.