In this lab you will learn how to upload a new text file to your Azure storage account using the Azure CLI.

Microsoft Azure is a cloud computing service offered and operated by Microsoft. You can use this service to host your data and applications in the cloud.

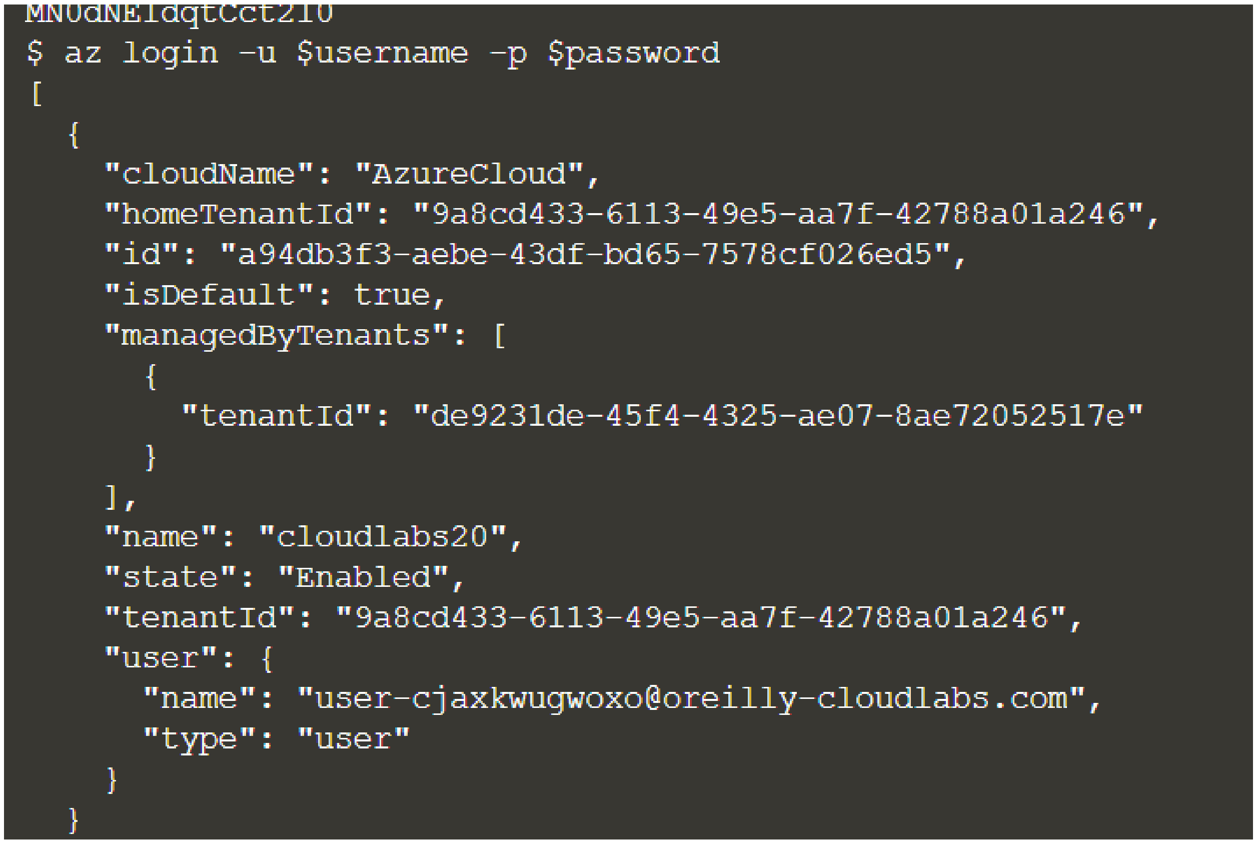
An Azure storage account is an Azure cloud service that contains all of your Azure storage data objects: blobs, file shares, queues, tables, and disks. You can host files, including images, videos, music and binary, as well as noSQL data and messages in the cloud using this service.

Azure Blob Storage is an Azure storage account service that enables you to host images, videos, music files, VM disks, and more in the cloud.

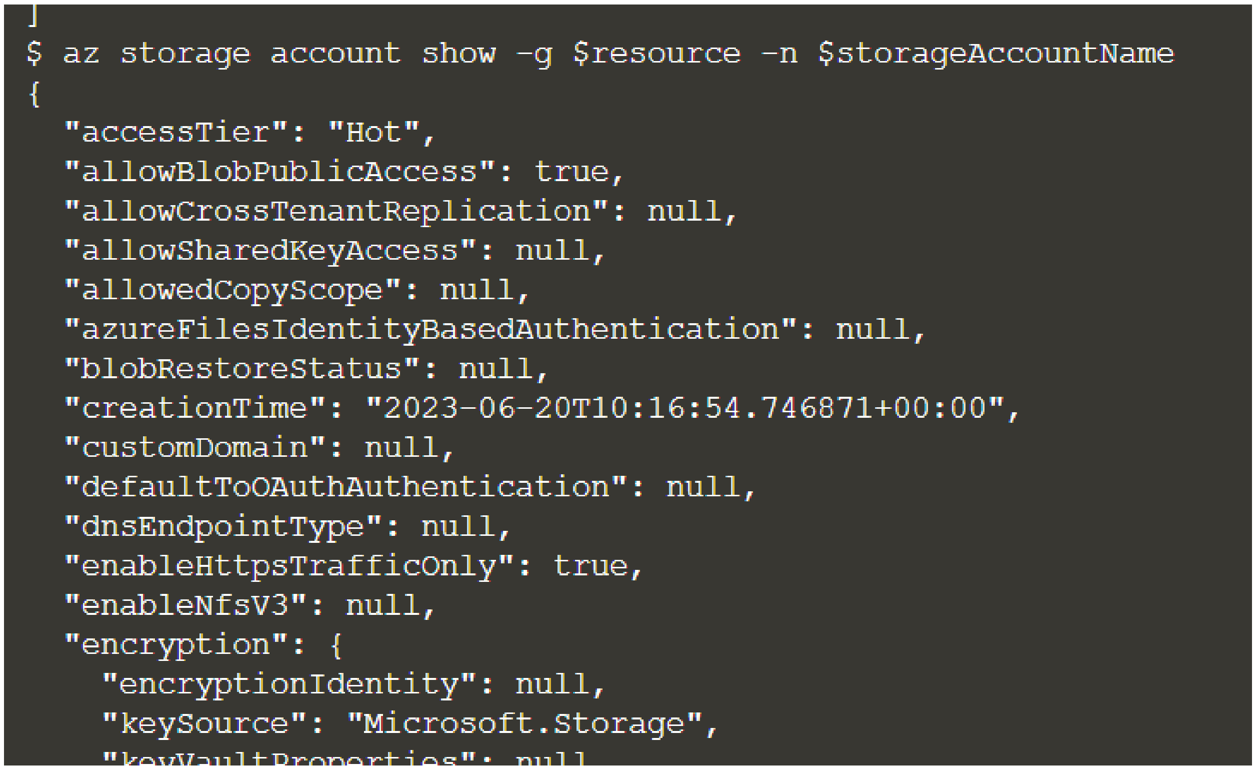
In this lab, you will learn how to do the following:

Create a new blob container in Azure storage account
Create a new blob
Clean up the blob and storage account

We have already created an Azure storage account for this lab. You'll need this account to work with Blob storage and upload your text file.

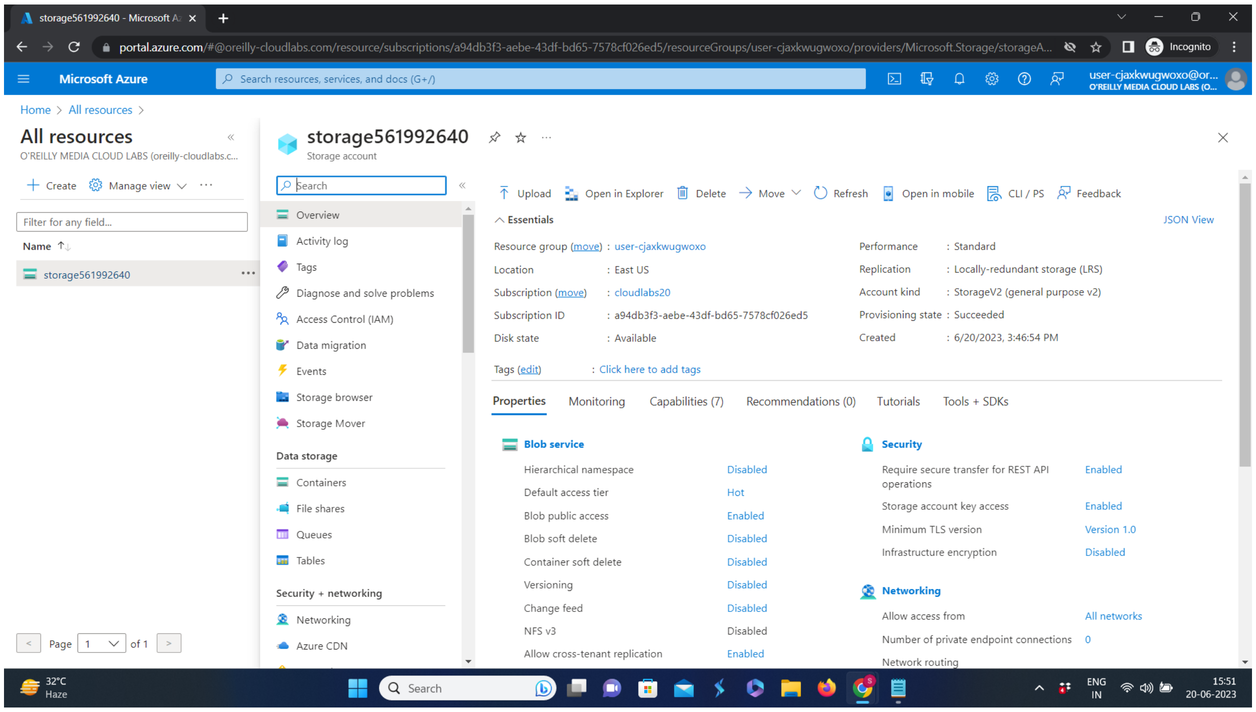


```
MN0dNE1dqtCct210
$ az login -u $username -p $password
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "9a8cd433-6113-49e5-aa7f-42788a01a246",
    "id": "a94db3f3-aebe-43df-bd65-7578cf026ed5",
    "isDefault": true,
    "managedByTenants": [
      {
        "tenantId": "de9231de-45f4-4325-ae07-8ae72052517e"
      }
    ],
    "name": "cloudlabs20",
    "state": "Enabled",
    "tenantId": "9a8cd433-6113-49e5-aa7f-42788a01a246",
    "user": {
      "name": "user-cjaxkwugwoxo@oreilly-cloudlabs.com",
      "type": "user"
    }
  }
```



```
]
$ az storage account show -g $resource -n $storageAccountName
{
  "accessTier": "Hot",
  "allowBlobPublicAccess": true,
  "allowCrossTenantReplication": null,
  "allowSharedKeyAccess": null,
  "allowedCopyScope": null,
  "azureFilesIdentityBasedAuthentication": null,
  "blobRestoreStatus": null,
  "creationTime": "2023-06-20T10:16:54.746871+00:00",
  "customDomain": null,
  "defaultToOAuthAuthentication": null,
  "dnsEndpointType": null,
  "enableHttpsTrafficOnly": true,
  "enableNfsV3": null,
  "encryption": {
    "encryptionIdentity": null,
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null
```

Now that we have our storage account ready, let's create a Blob storage container.



Resource group (move)	: user-cjaxkwugwoxo	Performance	: Standard
Location	: East US	Replication	: Locally-redundant storage (LRS)
Subscription (move)	: cloudlabs20	Account kind	: StorageV2 (general purpose v2)
Subscription ID	: a94db3f3-aebe-43df-bd65-7578cf026ed5	Provisioning state	: Succeeded
Disk state	: Available	Created	: 6/20/2023, 3:46:54 PM
Tags (edit)	: Click here to add tags		

Blob service

Hierarchical namespace	Disabled
Default access tier	Hot
Blob public access	Enabled
Blob soft delete	Disabled
Container soft delete	Disabled
Versioning	Disabled
Change feed	Disabled
NFS v3	Disabled
Allow cross-tenant replication	Enabled

Security

Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.0
Infrastructure encryption	Disabled

Networking

Allow access from	All networks
Number of private endpoint connections	0
Network routing	

Step 3: Create a New Container

In order to host Blob files in an Azure storage account, you need to first create a container. A container organizes a set of blobs, similar to the way a directory organizes files and other directories in a file system. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs.

One of the methods you can use to authenticate into a storage account is an account key. The following command fetches the primary mykatastorageaccount account key and stores it into the $storageKey environment variable:

storageKey=$(az storage account keys list -g $resource -n $storageAccountName --output json --query [0].value)

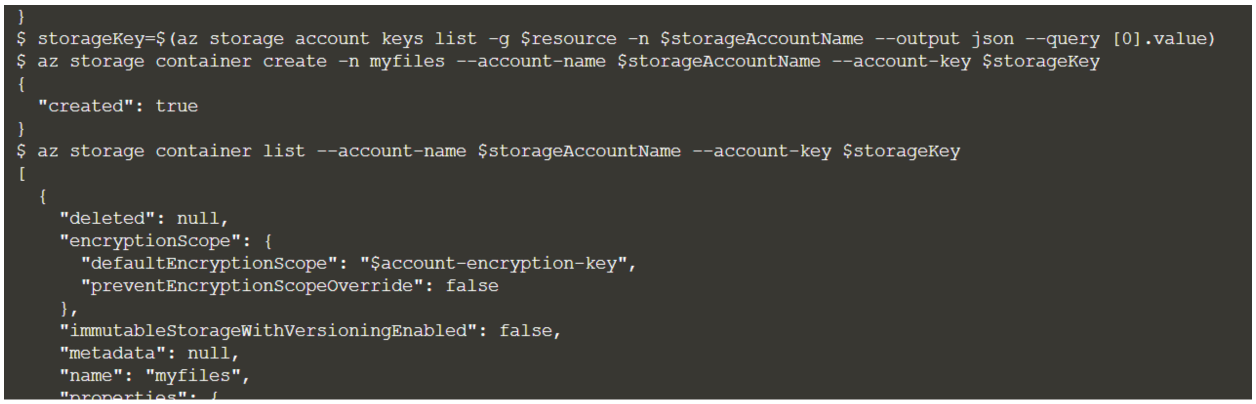
Now you can run the following command to create a new container, myfiles, in your storage account:

az storage container create -n myfiles --account-name $storageAccountName --account-key $storageKey

Use the following command to confirm that the new container appears in your storage account:

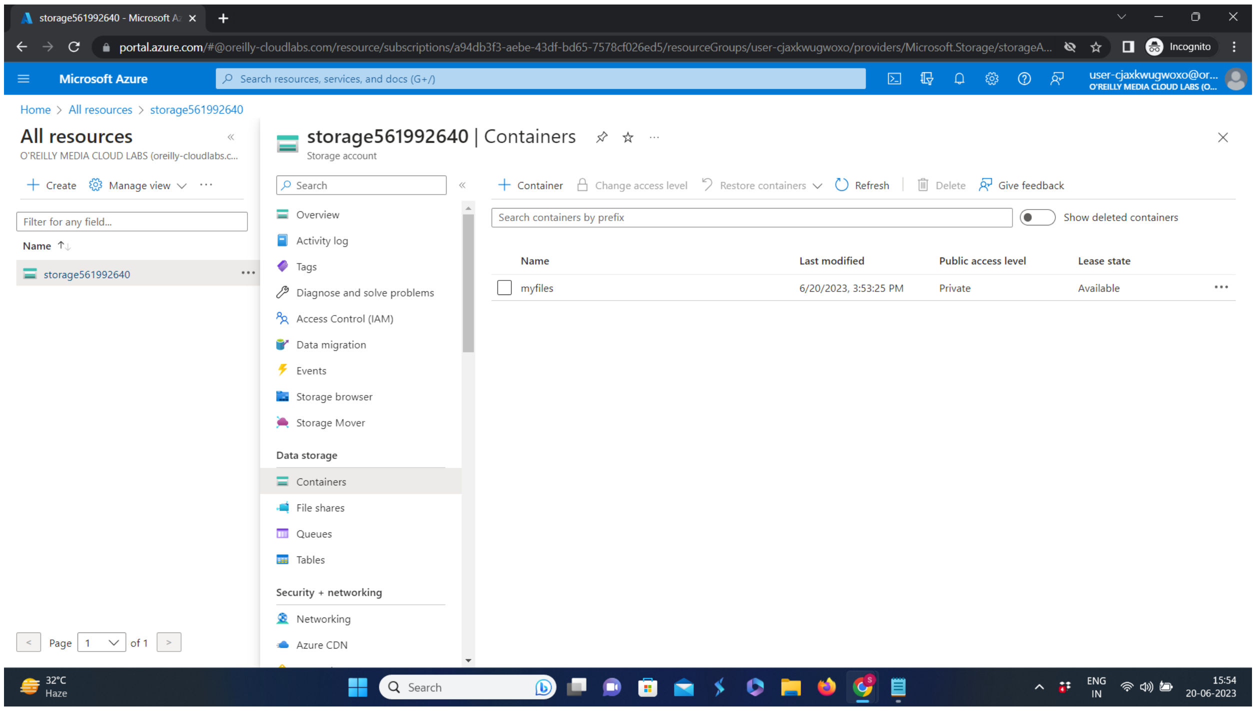
az storage container list --account-name $storageAccountName --account-key $storageKey

Make sure you can see the myfiles container in the list.

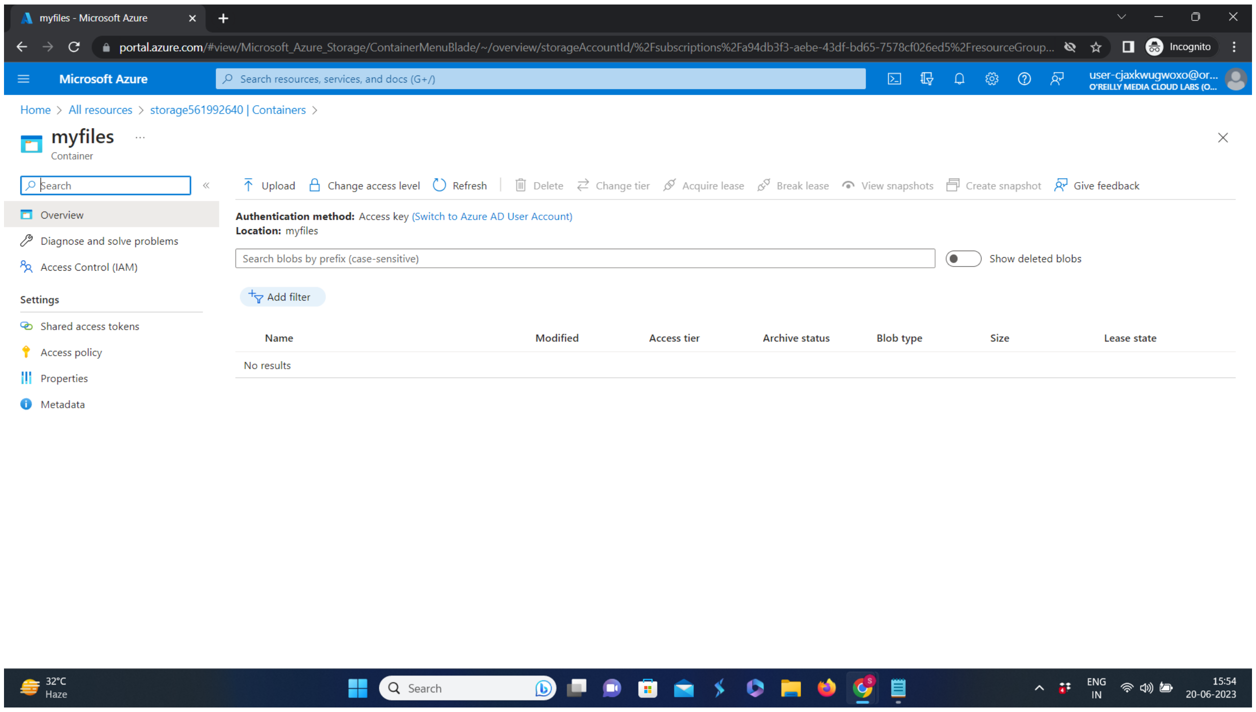


```
}
$ storageKey=$(az storage account keys list -g $resource -n $storageAccountName --output json --query [0].value)
$ az storage container create -n myfiles --account-name $storageAccountName --account-key $storageKey
{
  "created": true
}
$ az storage container list --account-name $storageAccountName --account-key $storageKey
[
  {
    "deleted": null,
    "encryptionScope": {
      "defaultEncryptionScope": "$account-encryption-key",
      "preventEncryptionScopeOverride": false
    },
    "immutableStorageWithVersioningEnabled": false,
    "metadata": null,
    "name": "myfiles",
```

Now we have our container ready. What's the next step? Let's upload a text file to it.


storage561992640 - Microsoft Azure
portal.azure.com/#@oreilly-cloudlabs.com/resource/subscriptions/a94db3f3-aebe-43df-bd65-7578cf026ed5/resourceGroups/user-cjaxkwugwoxo/providers/Microsoft.Storage/storageA...
Microsoft Azure
Home > All resources > storage561992640
All resources
storage561992640 | Containers
Storage account
+ Container
Change access level
Restore containers
Refresh
Delete
Give feedback
Search containers by prefix
Show deleted containers
Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Storage Mover
Data storage
Containers
File shares
Queues
Tables
Security + networking
Networking
Azure CDN

Name	Last modified	Public access level	Lease state
myfiles	6/20/2023, 3:53:25 PM	Private	Available


myfiles - Microsoft Azure
portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/~/overview/storageAccountId/%2Fsubscriptions%2Fa94db3f3-aebe-43df-bd65-7578cf026ed5%2FresourceGroup...
Microsoft Azure
Home > All resources > storage561992640 | Containers >
myfiles
Container
Upload
Change access level
Refresh
Delete
Change tier
Acquire lease
Break lease
View snapshots
Create snapshot
Give feedback
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Shared access tokens
Access policy
Properties
Metadata
Search blobs by prefix (case-sensitive)
Show deleted blobs
Add filter

Authentication method: Access key (Switch to Azure AD User Account)
Location: myfiles

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
No results						

Step 4: Create a Sample Text File
It's time finally upload a file to our blob container.

First, we need to create a new file, blob.txt. The following command creates this text file:

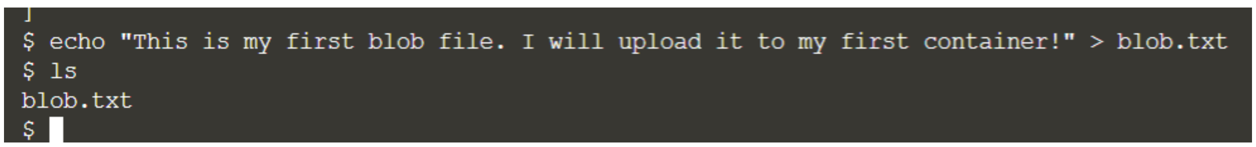
echo "This is my first blob file. I will upload it to my first container!" > blob.txt

You can confirm that the file was created by running the following command:

ls

Make sure you can see blob.txt in the list.

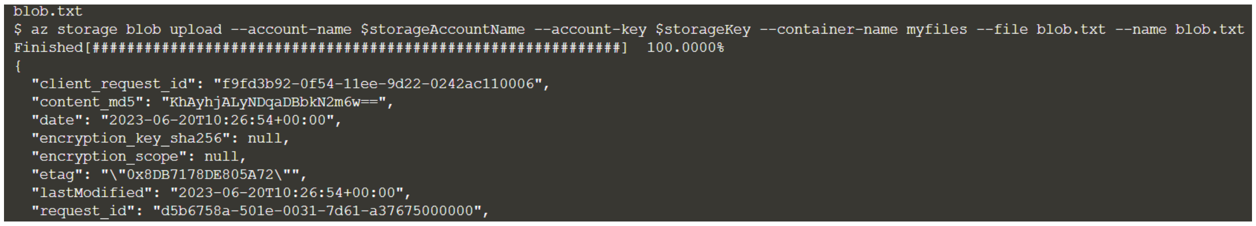
Now, let's upload this file to Azure Blob storage.



```
]
$ echo "This is my first blob file. I will upload it to my first container!" > blob.txt
$ ls
blob.txt
$ 
```

Step 5: Upload the Text File to the Storage Account
Let's upload blob.txt to our new container. Use the following command to do this:

az storage blob upload --account-name $storageAccountName --account-key $storageKey --container-name myfiles --file blob.txt --name blob.txt



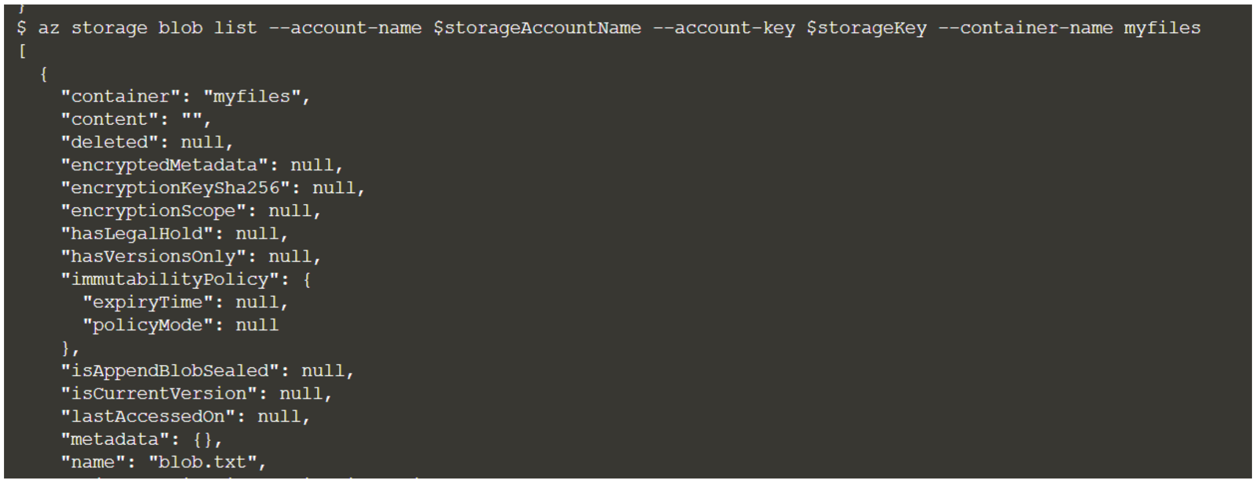
```
blob.txt
$ az storage blob upload --account-name $storageAccountName --account-key $storageKey --container-name myfiles --file blob.txt --name blob.txt
Finished[###############################################################]  100.0000%
{
  "client_request_id": "f9fd3b92-0f54-11ee-9d22-0242ac110006",
  "content_md5": "KhAyhjALyNDqaDBbkN2m6w==",
  "date": "2023-06-20T10:26:54+00:00",
  "encryption_key_sha256": null,
  "encryption_scope": null,
  "etag": "\"0x8DB7178DE805A72\"",
  "lastModified": "2023-06-20T10:26:54+00:00",
  "request_id": "d5b6758a-501e-0031-7d61-a37675000000",
```

Now, how do we confirm that the file was uploaded? We can use the following command:

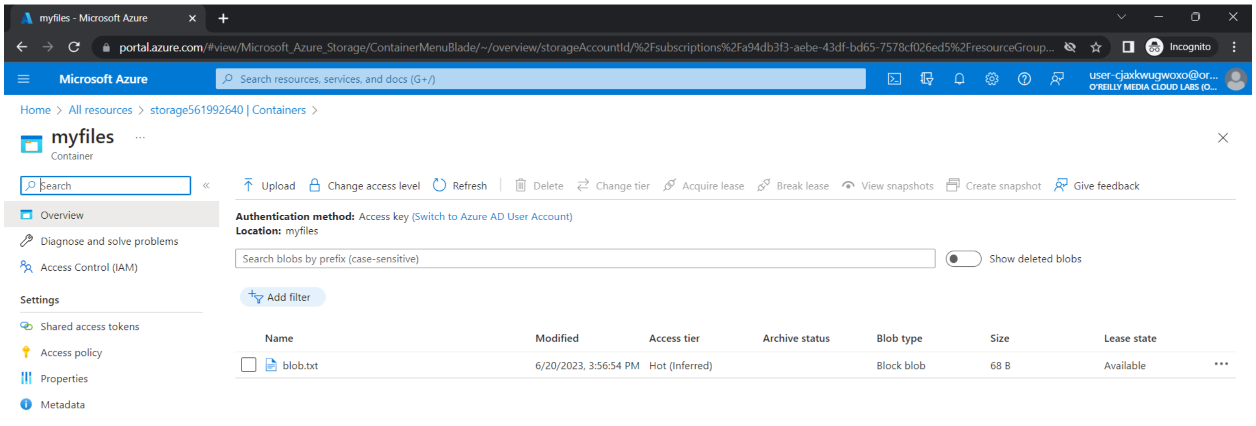
az storage blob list --account-name $storageAccountName --account-key $storageKey --container-name myfiles

Make sure you can find the blob.txt in the blob names.

Now that we've confirmed that the file was uploaded to Azure storage account, let's take a look at this file in the Azure portal as well.



```
$ az storage blob list --account-name $storageAccountName --account-key $storageKey --container-name myfiles
[
  {
    "container": "myfiles",
    "content": "",
    "deleted": null,
    "encryptedMetadata": null,
    "encryptionKeySha256": null,
    "encryptionScope": null,
    "hasLegalHold": null,
    "hasVersionsOnly": null,
    "immutabilityPolicy": {
      "expiryTime": null,
      "policyMode": null
    },
    "isAppendBlobSealed": null,
    "isCurrentVersion": null,
    "lastAccessedOn": null,
    "metadata": {},
    "name": "blob.txt",
```



myfiles - Microsoft Azure

portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/~/overview/storageAccountId/%2Fsubscriptions%2Fa94db3f3-aebe-43df-bd65-7578cf026ed5%2FresourceGroup...

Microsoft Azure

user-cjaxkwugwoxo@or... O'REILLY MEDIA CLOUD LABS (O...

Home > All resources > storage561992640 | Containers >

myfiles
Container

Authentication method: Access key (Switch to Azure AD User Account)
Location: myfiles

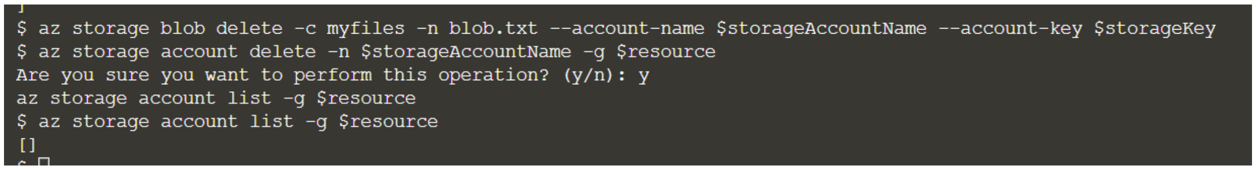
Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
blob.txt	6/20/2023, 3:56:54 PM	Hot (Inferred)		Block blob	68 B	Available

Step 7: Clean Up the Blob File
Before concluding this lab, use this command to clean up (i.e., delete) the Blob:

az storage blob delete -c myfiles -n blob.txt --account-name $storageAccountName --account-key $storageKey

We can also delete the parent container and the storage account, which we'll do in the next step.

Step 8: Delete the Storage Account



```
$ az storage blob delete -c myfiles -n blob.txt --account-name $storageAccountName --account-key $storageKey
$ az storage account delete -n $storageAccountName -g $resource
Are you sure you want to perform this operation? (y/n): y
az storage account list -g $resource
$ az storage account list -g $resource
[]
```