

Tutorial on how to create and manage storage account on Microsoft Azure

Mujtaba Ahmed Abbasi

February 2017

Contents

- Requirements
- Creating an Azure Blob Storage account
- Creating a container in that storage account
- Seeing what containers exist within the storage account
- Uploading data to that container
- Seeing items are inside of that container
- Deleting an item in that container
- Deleting the container
- Deleting the storage account
- Installing Java Development Kit (JDK)

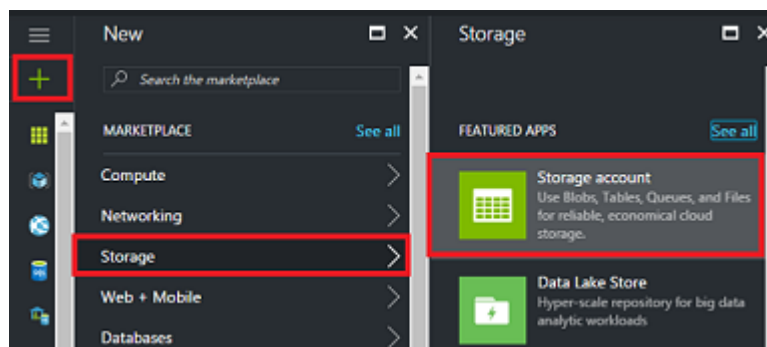
Requirements

- Sign in for 30 day free trial account from Microsoft Azure subscription.
- Install the Java Development Kit (JDK)
- Text editor, I will be using sublime 3

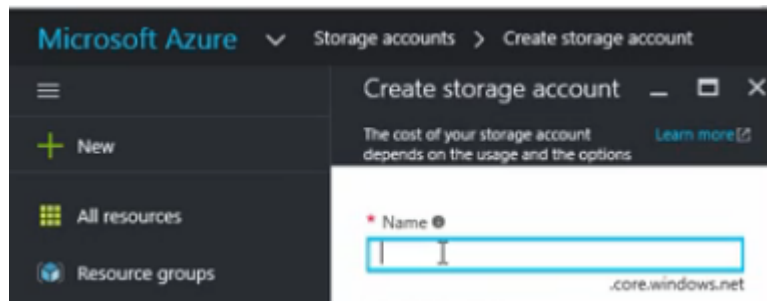
i. Creating an Azure Blob Storage account

We will be creating blob storage account in this step to store our unstructured data as objects in Azure storage.

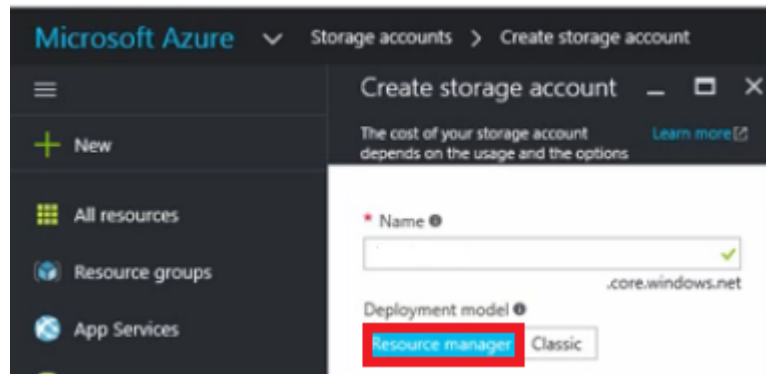
1. Sign in to the Azure portal.
2. On the Hub menu, select **New**, **Storage** and then **Storage account**.



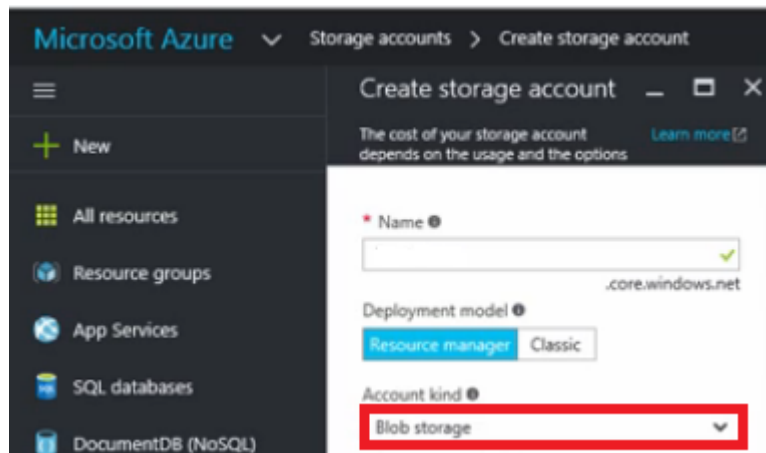
3. Enter a unique name for your storage account. Name must be of 3-24 characters in length and may contain numbers and lowercase letters only.



4. Select the deployment model option as **Resource Manager** . Blob storage accounts can only be created using this particular option.



5. Change the type of storage account from General purpose to **Blob Storage**. By default it is General purpose.



6. Now for **Performance** and **Replication** wise we will leverage the default options for this demo.

Microsoft Azure Storage accounts > Create storage account

Create storage account

The cost of your storage account depends on the usage and the options [Learn more](#)

* Name

hanselman

.core.windows.net

Deployment model

Resource manager Classic

Account kind

General purpose

Performance

Standard Premium

Replication

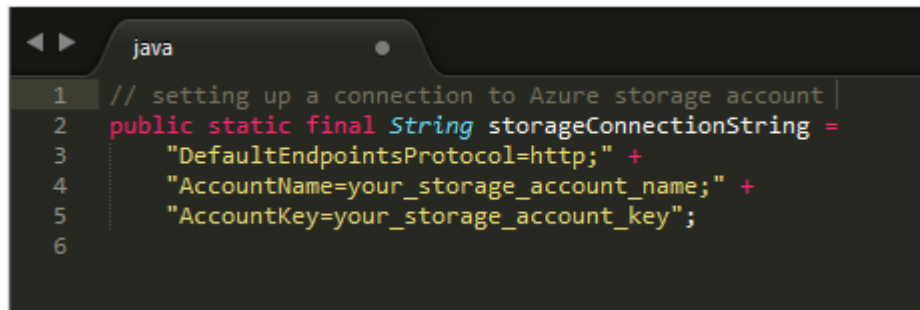
Read-access geo-redundant storage (RA...)

7. Leave the remaining fields default and hit the pin to dashboard button at the bottom of your screen. Finally click the **Create** button to create the storage account.

ii. Creating a container in that storage account

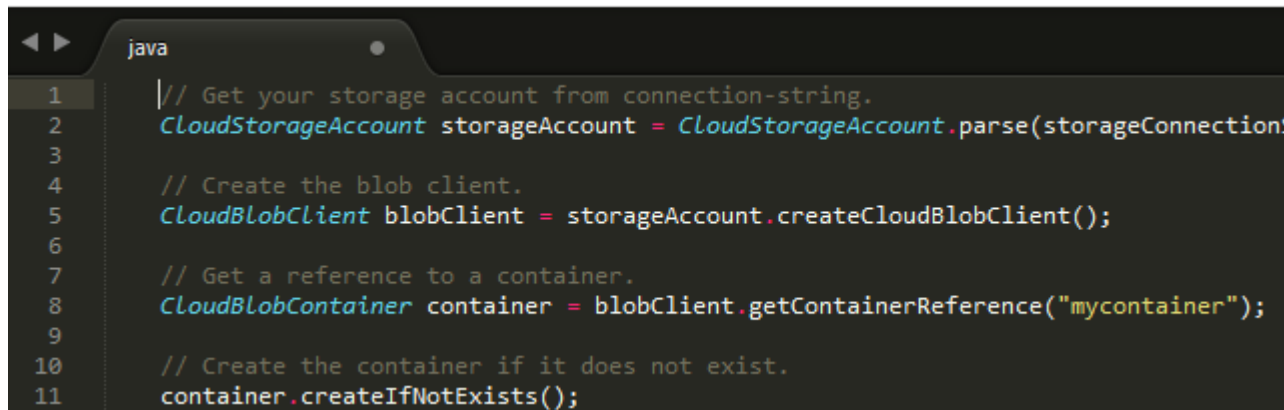
In this step we will create a container (CloudBlobClient) in the storage account we created in the previous section.

1. Set a connection string to the Azure storage account.



```
1 // setting up a connection to Azure storage account |
2 public static final String storageConnectionString =
3     "DefaultEndpointsProtocol=http;" +
4     "AccountName=your_storage_account_name;" +
5     "AccountKey=your_storage_account_key";
6
```

2. Create a container: Using **CloudBlobClient** we can get reference objects for containers. You can create the container if it doesn't exist with the **createIfNotExists** method, which will otherwise return the existing container.



```
1 // Get your storage account from connection-string.
2 CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnection
3
4 // Create the blob client.
5 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
6
7 // Get a reference to a container.
8 CloudBlobContainer container = blobClient.getContainerReference("mycontainer");
9
10 // Create the container if it does not exist.
11 container.createIfNotExists();
```

iii. Seeing what containers exist within the storage account

In order to list the containers, we can use `ListContainers()` to list all containers in one storage account, consider the java code:

```
java
1 CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnectionString);
2 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
3 CloudBlobContainer container = blobClient.getContainerReference("container");
4 container.createIfNotExists();
5
6 for(CloudBlobContainer item: blobClient.listContainers())
7 {
8     System.out.println(item.getName());
9 }
```

iv. Uploading data to that container

Now we have created a container, its time to upload data in it. Consider the code:

```
java
1 // Source of the file to upload.
2 File source = new File("path/to/your/file");
3
4 // In which container to upload the data.
5 String uri = "your-container-name/data";
6
7 // Change this to your container name.
8 String containerName = "your-container-name";
9
10 // Create the client.
11 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
12
13 // Retrieve reference of the container we created previously.
14 CloudBlobContainer container = blobClient.getContainerReference(uri);
15
16 // Create the container if it does not exist.
17 container.createIfNotExist();
18
19 // Finally upload the data into container.
20 CloudBlockBlob blob = container.getBlockBlobReference(source.getName());
21 blob.upload(new FileInputStream(source), source.length());
```


The above example will upload the data file into the data directory of your container.

v. Seeing items are inside of that container

To see the blobs in a container, we need to get reference of the container first like we did while uploading a blob. Using **listBlobs** method with a for loop, we can print the uri of each item in a container. Consider the java code:

```
java
1 // Get your storage account from connection-string.
2   CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnection
3
4   // Create the blob client.
5   CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
6
7   // Get a reference to a container.
8   CloudBlobContainer container = blobClient.getContainerReference("container");
9
10  // Loop over blobs within the container and output the URI to each of them.
11  for (ListBlobItem blobItem : container.listBlobs()) {
12      System.out.println(blobItem.getUri());
13  }
```

vi. Deleting an item in that container

In order to delete an item, get a reference of that item and call `deleteIfExists`.

```
1 // Get your storage account from connection-string.
2 CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnectionString);
3
4 // Create the blob client.
5 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
6
7 // Get a reference to a container.
8 CloudBlobContainer container = blobClient.getContainerReference("container");
9
10 // Retrieve reference of the item by giving your_item_name.
11 CloudBlockBlob blob = container.getBlockBlobReference("your_item_name");
12
13 // Delete the blob.
14 blob.deleteIfExists();
```

vii. Deleting the container

To delete a blob container, consider the following code:

```
1 // Get your storage account from connection-string.
2 CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnectionString);
3
4 // Create the blob client.
5 CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
6
7 // Get a reference to a container.
8 CloudBlobContainer container = blobClient.getContainerReference("container");
9
10 // Delete the blob container.
11 container.deleteIfExists();
```

viii. Deleting the storage account

To delete storage account click the **Delete** button on top of your storage account in Azure portal. Deleting a storage account deletes the entire account, including all data in the account.

Installing Java Development Kit (JDK)

1. Go to <http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html> and select the highlighted box.



2. Select your preferred option and check the Accept license agreement, I am going for windows x84 depending upon my system specs.

