Tutorial on how to create and manage storage account on Azure

Mujtaba Ahmed Abbasi

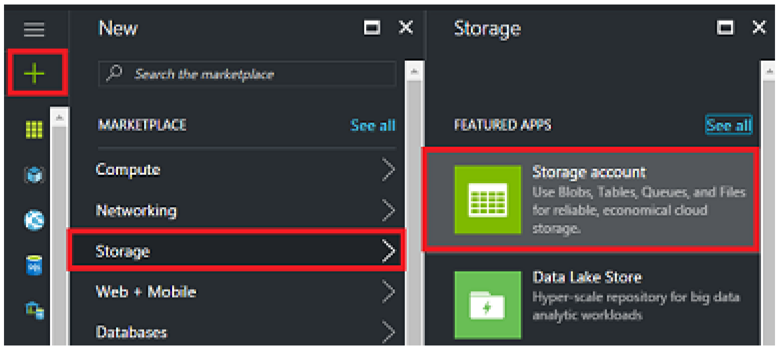
February 2017

i. Creating an Azure Blob Storage account

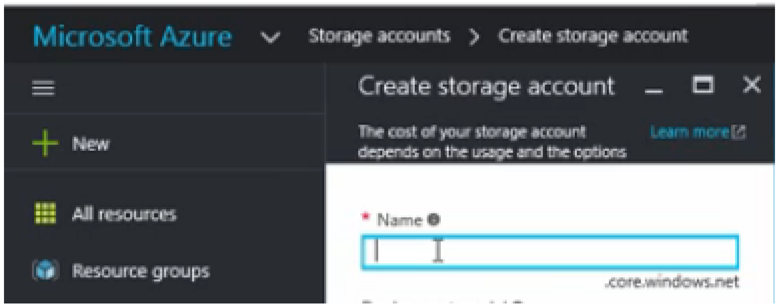
We will be creating blob storage account in this step to store our unstructured data as objects in Azure storage.

1. Sign in to the Azure portal.

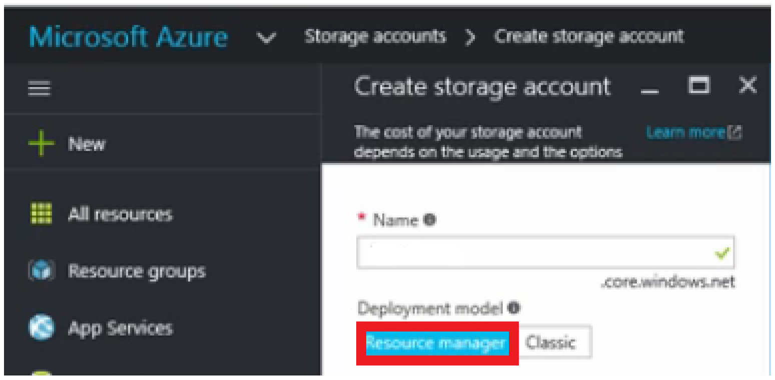
2. On the Hub menu, select **New**, **Storage** and then **Storage account**.



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.


Microsoft Azure
Storage accounts > Create storage account
New
All resources
Resource groups
Create storage account
The cost of your storage account depends on the usage and the options
Learn more
* Name
.core.windows.net

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



Create storage account

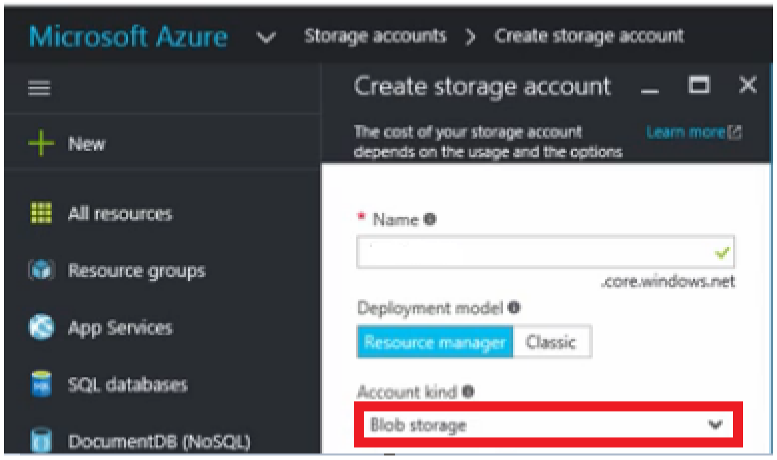
The cost of your storage account depends on the usage and the options

* Name

.core.windows.net

Deployment model: Resource manager | Classic

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



Create storage account

The cost of your storage account depends on the usage and the options

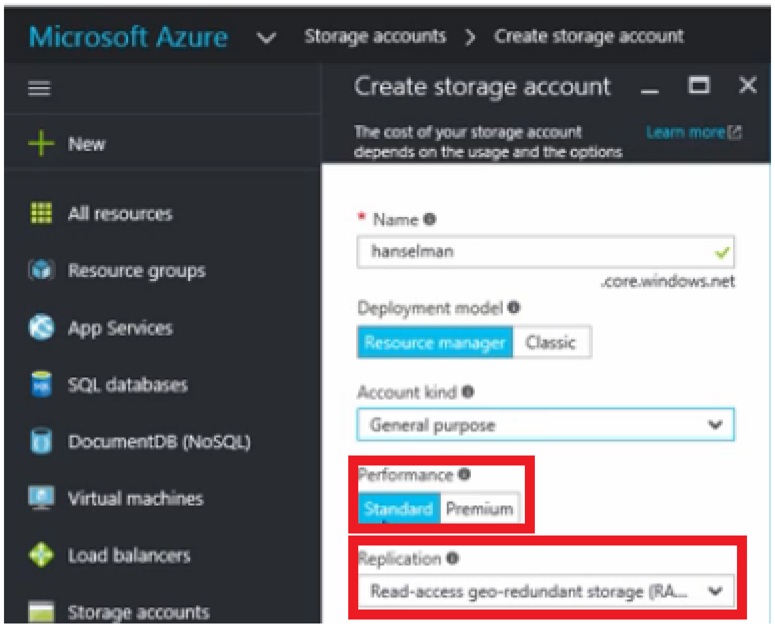
* Name

.core.windows.net

Deployment model: Resource manager | Classic

Account kind: Blob storage

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



The cost of your storage account depends on the usage and the options

* Name: hanselman .core.windows.net

Deployment model: [✓ Resource manager] Classic

Account kind: General purpose

Performance: [✓ Standard] Premium

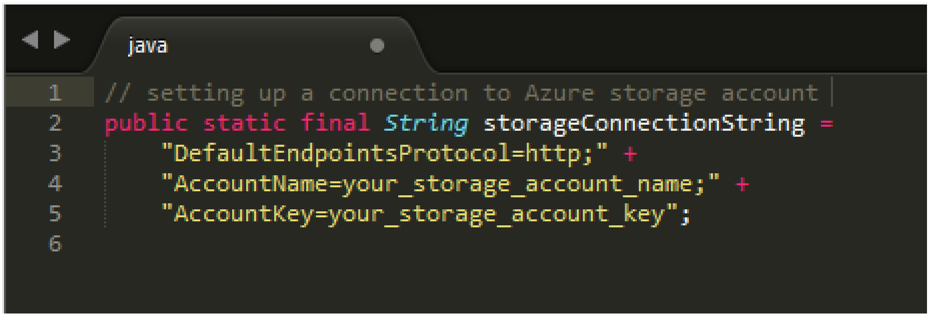
Replication: Read-access geo-redundant storage (RA...

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

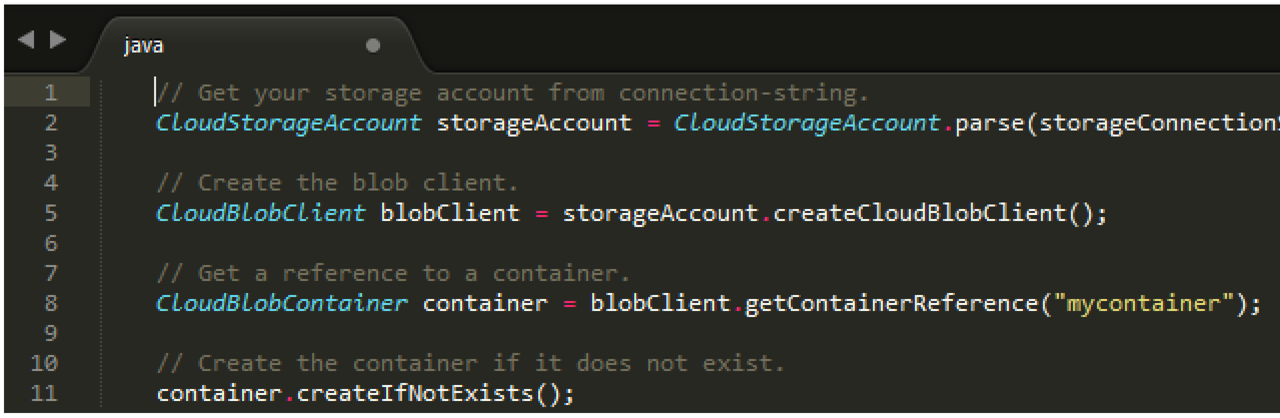
In this step we will create a container (CloudBlobClient) in the storage account we created in the previous section.

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



```java
// setting up a connection to Azure storage account
public static final String storageConnectionString =
    "DefaultEndpointsProtocol=http;" +
    "AccountName=your_storage_account_name;" +
    "AccountKey=your_storage_account_key";

```

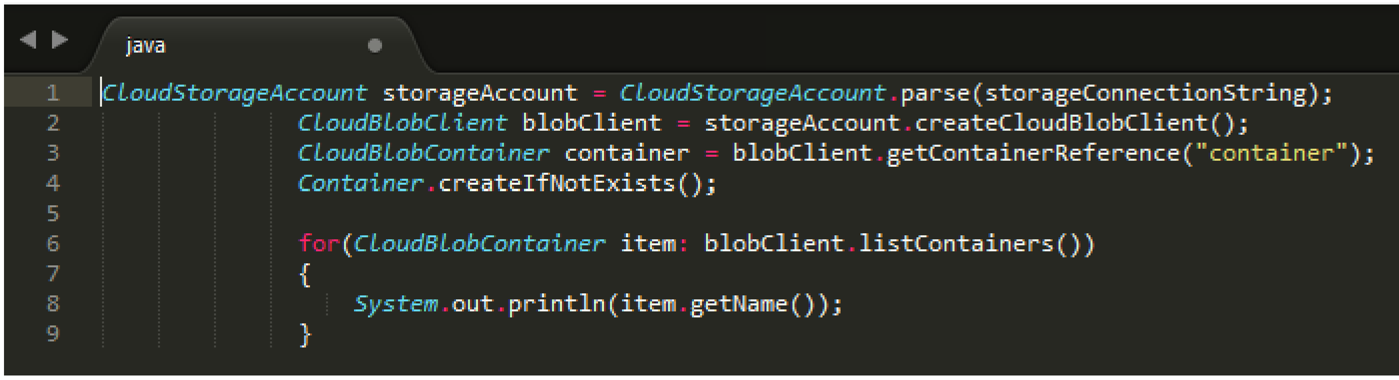
2. Create a container: Using **CloudBlobClient** we can get reference objects for containers. You can create the container if it doesn't exist with the **createIfNotExists** method, which will otherwise return the existing container.



```java
// Get your storage account from connection-string.
CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnection

// Create the blob client.
CloudBlobClient blobClient = storageAccount.createCloudBlobClient();

// Get a reference to a container.
CloudBlobContainer container = blobClient.getContainerReference("mycontainer");

// Create the container if it does not exist.
container.createIfNotExists();
```

iii. Seeing what containers exist within the storage account

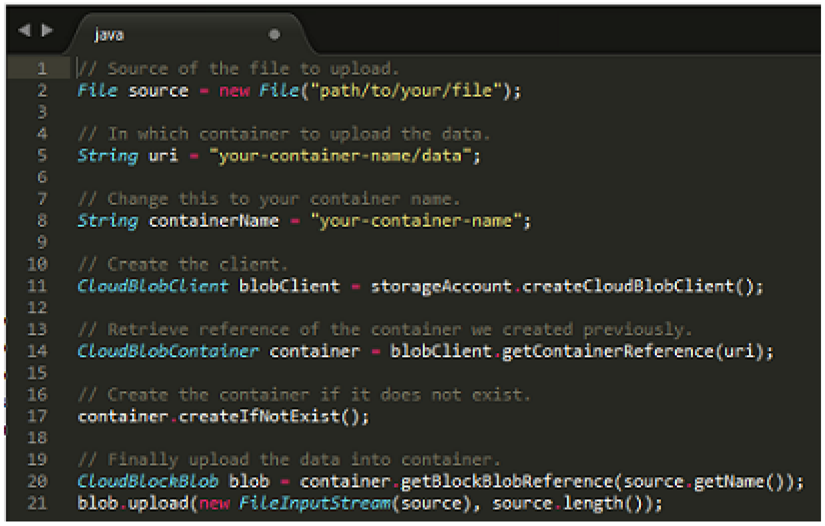
In order to list the containers, we can use **ListContainers()** to list all containers in one storage account, consider the java code:



```java
CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnectionString);
                CloudBlobClient blobClient = storageAccount.createCloudBlobClient();
                CloudBlobContainer container = blobClient.getContainerReference("container");
                Container.createIfNotExists();

                for(CloudBlobContainer item: blobClient.listContainers())
                {
                    System.out.println(item.getName());
                }
```

iv. Uploading data to that container

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

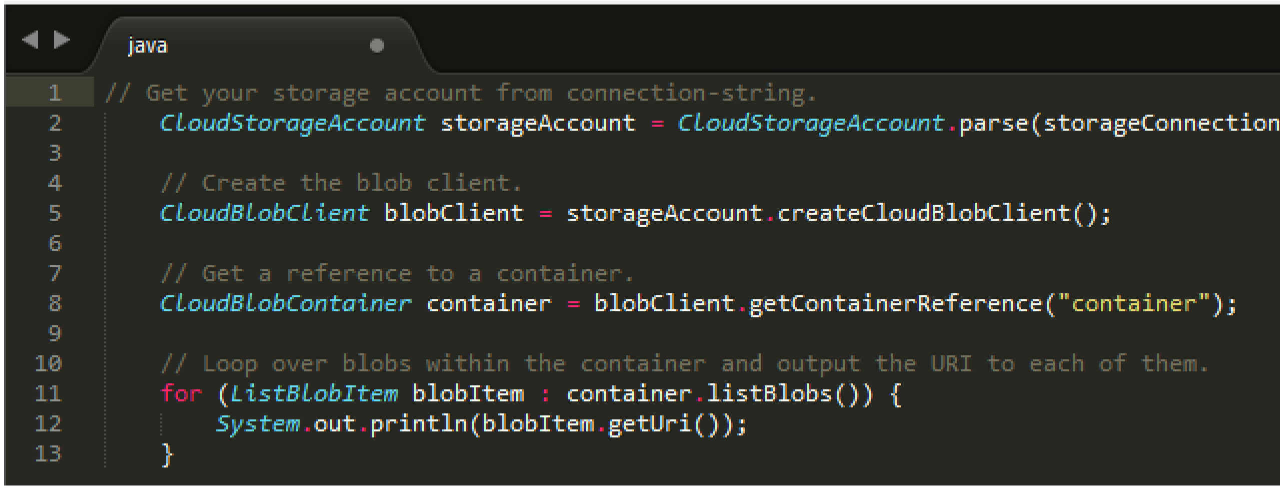


```java
// Source of the file to upload.
File source = new File("path/to/your/file");

// In which container to upload the data.
String uri = "your-container-name/data";

// Change this to your container name.
String containerName = "your-container-name";

// Create the client.
CloudBlobClient blobClient = storageAccount.createCloudBlobClient();

// Retrieve reference of the container we created previously.
CloudBlobContainer container = blobClient.getContainerReference(uri);

// Create the container if it does not exist.
container.createIfNotExist();

// Finally upload the data into container.
CloudBlockBlob blob = container.getBlockBlobReference(source.getName());
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

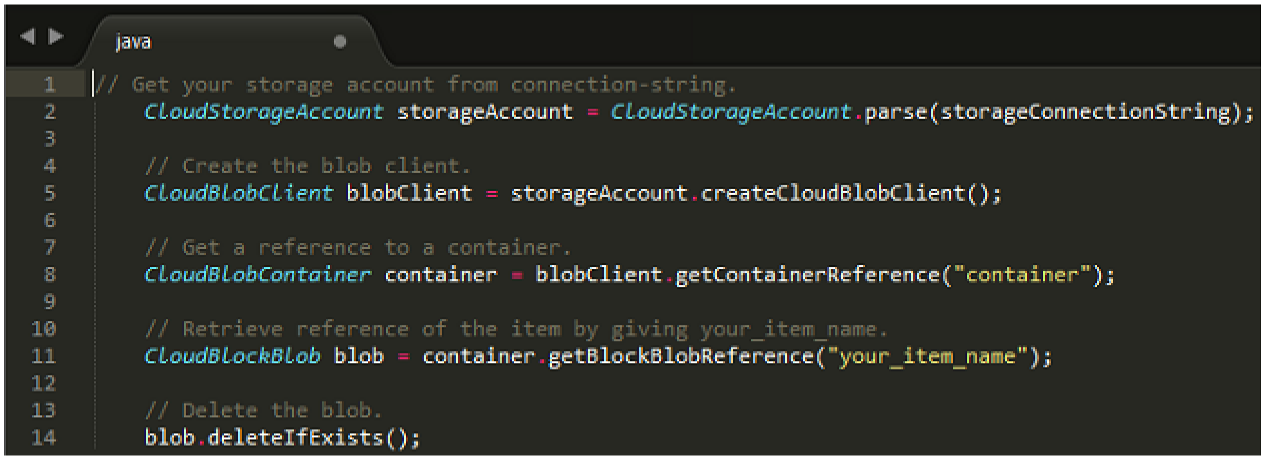
To see the blobs in a container, we need to get reference of the container first like we did while uploading a blob. Using **listBlobs** method with a for loop, we can print the uri of each item in a container. Consider the java code:



```java
// Get your storage account from connection-string.
    CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnection

    // Create the blob client.
    CloudBlobClient blobClient = storageAccount.createCloudBlobClient();

    // Get a reference to a container.
    CloudBlobContainer container = blobClient.getContainerReference("container");

    // Loop over blobs within the container and output the URI to each of them.
    for (ListBlobItem blobItem : container.listBlobs()) {
        System.out.println(blobItem.getUri());
    }
```

vi. Deleting an item in that container

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



```java
// Get your storage account from connection-string.
    CloudStorageAccount storageAccount = CloudStorageAccount.parse(storageConnectionString);

    // Create the blob client.
    CloudBlobClient blobClient = storageAccount.createCloudBlobClient();

    // Get a reference to a container.
    CloudBlobContainer container = blobClient.getContainerReference("container");

    // Retrieve reference of the item by giving your_item_name.
    CloudBlockBlob blob = container.getBlockBlobReference("your_item_name");

    // Delete the blob.
    blob.deleteIfExists();
```