

Versioning on Azure Storage

Name : Thejdeep Gudivada

IRC : thej @ freenode

In this document, I am going to walk through my understanding and experiences with versioning on Azure's Storage.

Getting Started

- Login to your Azure Portal, create an account if you do not have one.
- Create a Storage Account inside the portal by selecting your preferred Replication methods, deployment models and so on.
- Click on Access Keys to know your project name and also the key which you will be needing.

Using the Java SDK

Although there are SDK's available for most of the major languages, I decided to use Java for testing out the waters.

Download the Source SDK from [here](#) and plug it into your IDE as a library dependency.

Creating an Application

Lets create a sample application that will help us to demonstrate the versioning on Azure's Storage.

Below are a few `imports` classes that I will be needing for the application.

```
import com.microsoft.azure.storage.*;
import com.microsoft.azure.storage.blob.*;
```

In order to make a connection to the Storage Service, the request string has to be built up in the following manner :

```
public static final String storageConnectionString =  
    "DefaultEndpointsProtocol=http;"  
    + "AccountName=thejdeep;"  
    + "<Insert key here>";
```

Now, let us establish a connection to the service and create a client worker for us.

```
CloudStorageAccount account =  
CloudStorageAccount.parse(storageConnectionString);  
CloudBlobClient serviceClient = account.createCloudBlobClient();
```

Each of the blob that exists on Azure's Storage must belong to a container. So let's create a container first.

```
CloudBlobContainer container =  
serviceClient.getContainerReference("myimages");  
container.createIfNotExists();
```

Now that we have a container called `myimages` ready, we can go ahead and store blobs on it.

```
CloudBlockBlob blob = container.getBlockBlobReference("test");  
    File sourceFile = new  
File("/home/thejdeep/Documents/storage_test/dp.jpg");  
  
    // Uploading the file  
    blob.upload(new FileInputStream(sourceFile),  
sourceFile.length());  
    HashMap<String,String> new_map = new HashMap<String,  
String>();  
    new_map.put("OriginalFilename","dp.jpg");  
    blob.setMetadata(new_map);  
    HashMap<String,String> temp = blob.getMetadata();  
    System.out.println("Before snapshot");  
    System.out.println(temp.get("OriginalFilename"));  
  
    // Create snapshot  
    CloudBlockBlob newblob;  
    newblob = (CloudBlockBlob) blob.createSnapshot();
```

```

        // Create another Blob and upload a file
        CloudBlockBlob blobi =
container.getBlockBlobReference("test1");
        File sourceFile1 = new
File("/home/thejdeep/Documents/storage_test/dp1.jpg");
        blobi.upload(new FileInputStream(sourceFile1),
sourceFile1.length());
        HashMap<String,String> teste = new
HashMap<String,String>();
        teste.put("OriginalFilename","dp1.jpg");
        blobi.setMetadata(teste);
        HashMap<String,String> wat = blobi.getMetadata();
        System.out.println("After change file");
        System.out.println(wat.get("OriginalFilename"));

```

Promoting a Snapshot

Versioning in Azure boils down to copying a snapshot over the base blob in order to revert back to the old version which was stored as a snapshot. All the metadata of the base blob is overwritten with the information from the snapshot.

Now we can try to start the copy of a blob over another :

```

System.out.println("Now trying to replace");
String temper = blobi.startCopy(newblob);

HashMap<String,String> teste1 = blobi.getMetadata();
System.out.println(teste1.get("OriginalFilename"));

```