M7024E Laboratory 2

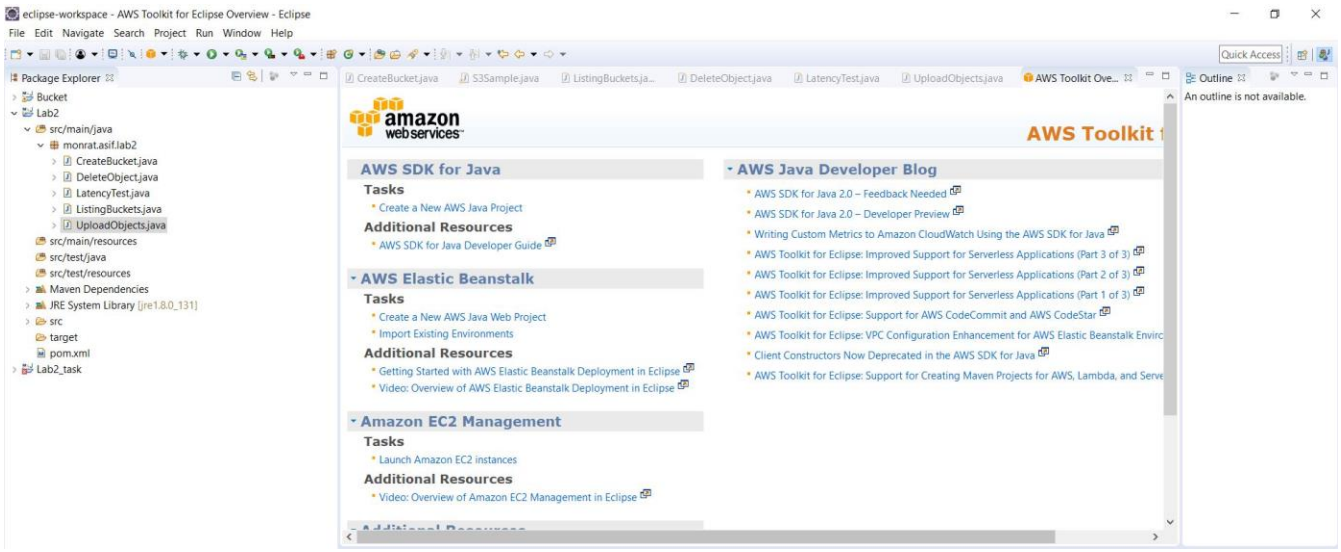
SERVICES - STORAGE SERVICES

Ahmed Afif Monrat, Syed Asif Iqbal

Objectives

- Setup a programming environment for building (using programming tools and languages) Cloud services for a major Cloud provider, for example, Amazon Web services (AWS);
- Develop Cloud services for file storage, listing and retrieval using the APIs provided by the AWS.

Exercise a: In this exercise, we setup the programming environment to create Cloud services using the APIs provided by AWS. Here we used Eclipse IDE for setting up AWS SDK for java.



Exercise b: In this exercise, we have learned how to use storage service provided by AWS.

1. Identify ways of creating the Amazon S3 service clients.

(a) Explain in detail how the Amazon S3 service clients are created by providing details of the packages and classes involved. Create a diagram of the dependencies involved.

There are two ways to creating Amazon S3 service clients. One of them is to create them manually from the AWS management console. Every object in Amazon S3 is stored in a bucket. Before we can store data in Amazon S3, we must create a bucket. Another way of creating the Amazon S3 service client is to use the service client builder. For instance, AmazonS3ClientBuilder is used with Credentials and region of user to create Amazon S3 service client.

In order to obtain an instance of client builder, the following code is used:

AmazonS3 s3 = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials).withRegion("Region Name, for example: us-west-2").build();

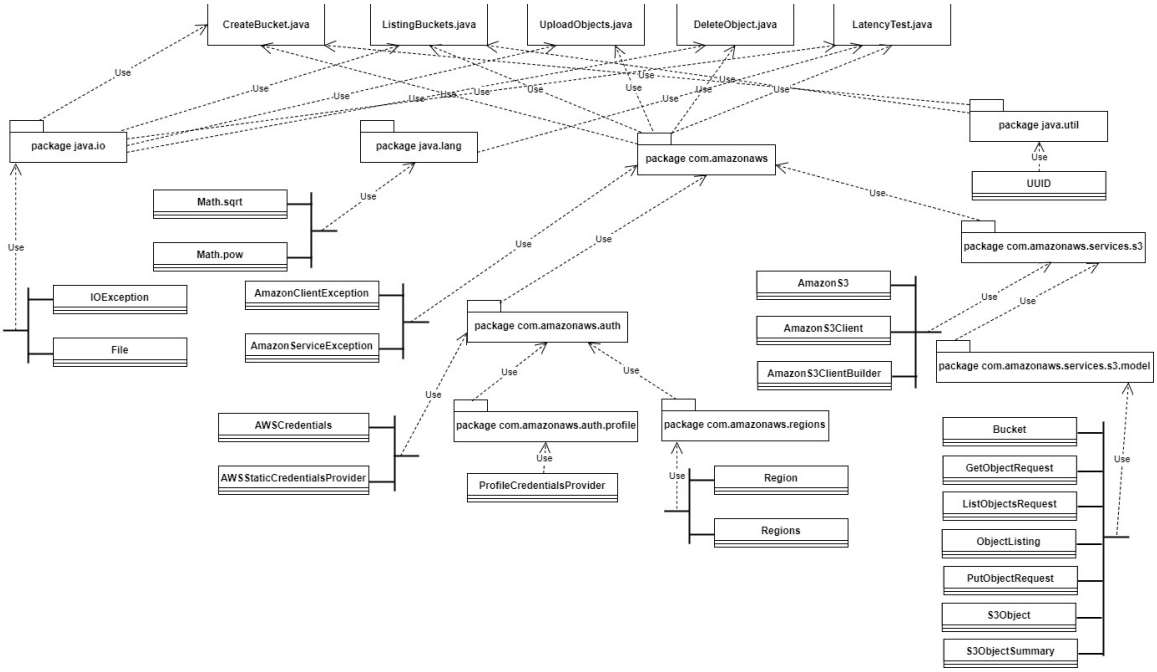


Figure 1. Dependency diagram of Amazon S3 client's packages and classes

S3Sample.java file uses three main packages:

1.Package java.io

2. Package java.util

3. Package com.amazonaws

- Package java.io is used to upload and download file using java, delete objects from that and throws I/O exceptions.
- Package.java.util contains the collection of classes, date and time facilities, handles many utility classes.
- Package com.amazonaws mainly handles two functionalities: Client exception and Service Exception in Amazon. Moreover, it invokes three packages such as,
- package.com.amazonaws.auth checks authentication by linking Amazon credential providers and Amazon user. This also includes to sub packages -

com.amazonaws.auth.regions which manages region based content and com.amazonaws.auth.profile for user profile authentication.

- Package com.amazonaws.services.s3 which is also included in package.com.amazonaws provides to build Amazon S3Client and manages client etc.
- Package com.amazonaws.services.s3.model use to create bucket, managing buckets in specific region, upload, download objects to or from buckets etc[1].

2. Create a Java program to create a bucket in three regions of your choice.

(a) Explain in detail the steps involved, and explain the output.

Step 1: First we need to import all the packages needed to use different functionalities of java and AWS S3 client.

```
1  package monrat.asif.lab2;
2  import java.io.IOException;
3  import com.amazonaws.AmazonClientException;
4  import com.amazonaws.AmazonServiceException;
5  import com.amazonaws.auth.AWSCredentials;
6  import com.amazonaws.auth.AWSStaticCredentialsProvider;
7  import com.amazonaws.auth.profile.ProfileCredentialsProvider;
8  import com.amazonaws.services.s3.AmazonS3;
9  import com.amazonaws.services.s3.AmazonS3ClientBuilder;
10 import java.util.UUID;
```

Step 2: Then we have to create a class and provide credentials to get access in AWS management console using our codes.

```
12 public class CreateBucket {
13
14     public static void main(String[] args) throws IOException {
15         // TODO Auto-generated method stub
16         AWSCredentials credentials = null;
17         try {
18                 credentials = new ProfileCredentialsProvider("default").getCredentials();
19            } catch (Exception e)
20         {
21                 throw new AmazonClientException("Cannot load the credentials from the credential profiles file. ",e);
22         }
```

Step 3: Next, we have to set the unique bucket as well as region's name and bind the credentials in each region to build the environment for the S3 object.

```
24          String bucket1 = "munrat-bucket-" + UUID.randomUUID();
25          String bucket2 = "munrat-bucket-" + UUID.randomUUID();
26          String bucket3 = "munrat-bucket-" + UUID.randomUUID();
27          String region1 = "us-west-2";
28          String region2 = "ca-central-1";
29          String region3 = "eu-west-2";
30
31          AmazonS3 s3_region1 = AmazonS3ClientBuilder.standard()
32                  .withCredentials(new AWSStaticCredentialsProvider(credentials))
33                  .withRegion(region1)
34                  .build();
35          AmazonS3 s3_region2 = AmazonS3ClientBuilder.standard()
36                  .withCredentials(new AWSStaticCredentialsProvider(credentials))
37                  .withRegion(region2)
38                  .build();
39          AmazonS3 s3_region3 = AmazonS3ClientBuilder.standard()
40                  .withCredentials(new AWSStaticCredentialsProvider(credentials))
41                  .withRegion(region3)
42                  .build();
```

Step 4: Finally, we will create bucket using createBucket method under S3 client object. This segment will create bucket after one another in the specified regions. However, if the S3 API has more than 100 buckets it might cause error while creating bucket as that's the default limit. We needed to clear some in order to maintain the threshold while creating these buckets.

```
43          try {
44          System.out.println("Creating bucket : " + bucket1 + "\n");
45          s3_region1.createBucket(bucket1);
46          System.out.println("Creating bucket : " + bucket2 + "\n");
47          s3_region2.createBucket(bucket2);
48          System.out.println("Creating bucket : " + bucket3 + "\n");
49          s3_region3.createBucket(bucket3);
50          }
```

3. Create a Java program that lists your buckets in the region of your choice.

```
28          try {
29          int count=0;
30          ArrayList<String> bucNames= new ArrayList<String>();
31          String region="ap-south-1";
32          System.out.println("Total Number of Buckets: " + s3.listBuckets().size());
33          for (Bucket bucket : s3.listBuckets()) { //listing buckets in bucket variable
34              String BucketRegion = s3.getBucketLocation(bucket.getName()); // getting the location for each bucket
35                     if(BucketRegion.equals(region))
36                       {
37                          //System.out.println("-" + bucket.getName());
38                           bucNames.add(bucket.getName());
39                           count++;
40                       }
41                              }
42
43              System.out.println("Total Number of Buckets for: Asia Pacific (Mumbai): "+ count);
44              System.out.println("Listing Buckets for: Asia Pacific (Mumbai)");
45              System.out.println(bucNames);
46          }
```

```
Problems  Javadoc  Declaration  Console
<terminated> ListingBuckets [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Nov 18, 2017, 5:28:37 AM)
Total Number of Buckets for: Asia Pacific (Mumbai): 3
Listing Buckets for: Asia Pacific (Mumbai)
[furkatubucket3, karanbucket1, plmbucket1]
```

4. Create a Java program to upload objects in your newly created bucket.

```
13 public class UploadObjects {
14
15     public static void main(String[] args) throws IOException {
16         // TODO Auto-generated method stub
17         AWSCredentials credentials = null;
18          try {
19                 credentials = new ProfileCredentialsProvider("default").getCredentials();
20             } catch (Exception e)
21          {
22                  throw new AmazonClientException("Cannot load the credentials from the credential profiles file. ",e);
23          }
24
25          String region = "us-west-2";
26          String bucket= "munrat-bucket-fe884488-4470-405c-9b4e-823e48dab10e";
27          String keyname = "MyObjectKey";
28          String filename = "C://Users//ahmed//eclipse-workspace//Lab2//src//main//java//monrat//asif//lab2//home1.jpg";
29
30          AmazonS3 s3 = AmazonS3ClientBuilder.standard()
31                  .withCredentials(new AWSStaticCredentialsProvider(credentials))
32                  .withRegion(region)
33                  .build();
34         try {
```

Problems @ Javadoc Declaration Console

```
<terminated> UploadObjects [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Nov 18, 2017, 5:32:17 AM)
Uploading a new object to S3:

Done uploading!
```

5. Create a Java program to delete a particular objects from your newly created bucket.

```
22
23          String region = "us-west-2";
24          String bucket= "munrat-bucket-fe884488-4470-405c-9b4e-823e48dab10e";
25          String keyname = "MyObjectKey";
26
27          AmazonS3 s3 = AmazonS3ClientBuilder.standard()
28                  .withCredentials(new AWSStaticCredentialsProvider(credentials))
29                  .withRegion(region)
30                  .build();
31
32          try {
33              System.out.println("Deleting an object\n");
34               s3.deleteObject(bucket, keyname);
35               System.out.println("Object Deleted Successfully");
36              }
37          catch (AmazonServiceException ase) {
38              System.out.println("Caught an AmazonServiceException, the request was made "
39                      + "to Amazon S3, but was rejected with an error response for some reason.");
40              System.out.println("Error Message:    " + ase.getMessage());
41              System.out.println("HTTP Status Code: " + ase.getStatusCode());
42              System.out.println("AWS Error Code:   " + ase.getErrorCode());
43              System.out.println("Error Type:       " + ase.getErrorType());
44              System.out.println("Request ID:       " + ase.getRequestId());
45          }
46          catch (AmazonClientException ace) {
47              System.out.println("Caught an AmazonClientException, The client has encountered "
48                      + "a serious internal problem while trying to communicate with S3, "
49                      + "such as not being able to access the network.");
50              System.out.println("Error Message: " + ace.getMessage());
```

Problems @ Javadoc Declaration Console

```
<terminated> DeleteObject [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (Nov 18, 2017, 5:33:44 AM)
Deleting an object

Object Deleted Successfully
```

Exercise c: Create a Java program to upload and download objects (at least 1 MB) from the three regions used in the above exercise. Measure the object upload and download latency from these regions. Plot and explain the results in your report.

Table 1: Comparison of latency of same image in different regions.

US West (Oregon)		Canada (Central)		EU (London)	
Upload Latency (Milliseconds)	Download Latency (Milliseconds)	Upload Latency (Milliseconds)	Download Latency (Milliseconds)	Upload Latency (Milliseconds)	Download Latency (Milliseconds)
3026	234	4092	369	892	74
3127	237	4145	334	953	89
3043	244	3958	309	859	72
3115	239	4396	1014	969	72
3182	237	4164	326	929	79
3112	214	4097	324	989	75
3171	239	4243	325	1018	95
3136	256	4203	343	853	70
3085	220	4251	340	929	96
3466	238	4218	333	956	77
3126	251	4038	333	969	79
3122	229	4168	316	1000	88
3079	321	4169	374	1453	71
3072	240	4230	334	1047	68
3053	245	4128	321	928	78
3067	298	4234	356	1526	67
3022	234	4567	334	1678	77
3015	222	4321	333	976	79
3056	245	4098	323	956	89
3021	215	4532	321	943	84

In the above table we can see the values of different regions in the upload and download latency of a 1 MB image file. The reason might be the geographical locations where the values differ from one place to another. In our case, we are uploading and downloading in Europe. So, the eu-west-2, EU (London) region has the shortest latency.

Table 2: Standard Deviation of upload-download latency.

Observation No.	Standard Deviation of Upload Latency (ms)	Standard Deviation of Download Latency (ms)
1	2011.5468	1192.6487
2	2050.8633	1151.8681
3	1967.9158	1084.4076
4	2015.1183	1125.6343
5	1972.7371	1100.3958
6	1971.2693	1130.7417
7	2036.1851	1131.1564
8	2055.1999	1112.4372
9	2037.9984	1170.7138
10	1981.1335	1135.3962
11	1986.4644	1119.9812
12	2067.5381	1136.4051
13	1985.1651	1107.2977
14	2064.6577	1137.6997
15	2060.0732	1153.0899
16	2025.5203	1157.0354
17	2006.1561	1076.9394
18	2077.1234	1082.8271
19	2179.4661	1343.3576
20	1994.1352	1125.3089

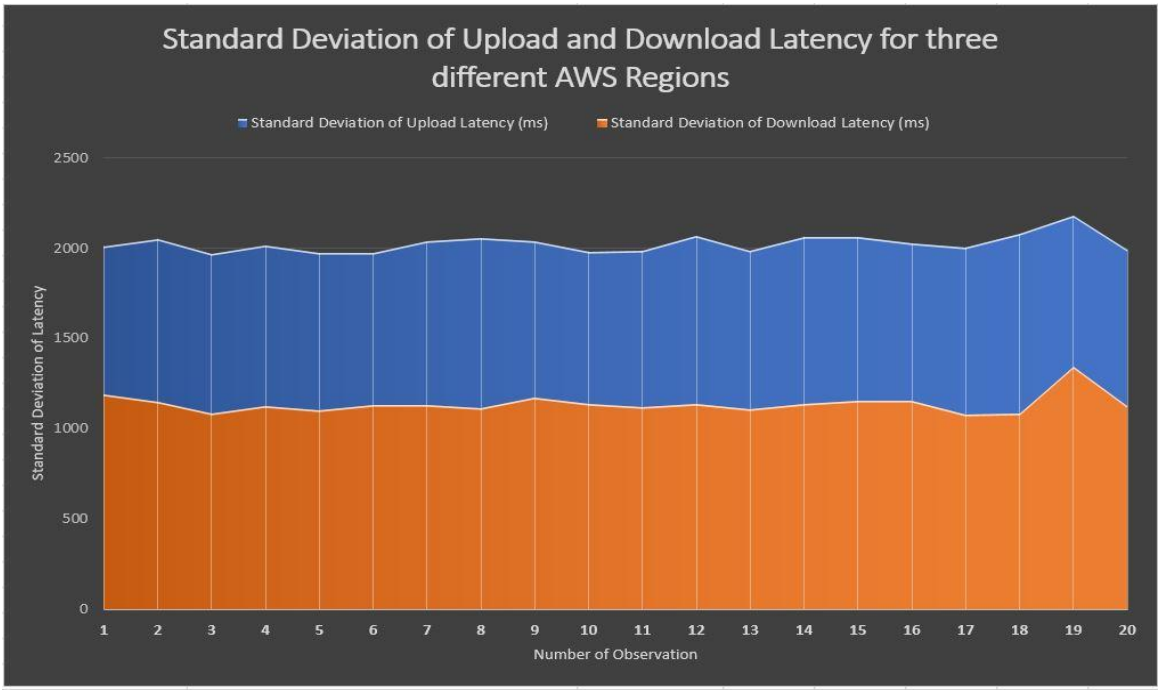


Figure 2. Standard Deviation of Upload & Download latency

From table 2 and Figure 2, we can observe that the latency is similar from the standard deviation of three different regions. It explains the network consistency of the AWS regions and their performance.

Reference

[1] "AmazonS3Client (AWS SDK for Java - 1.11.231)", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/AmazonS3Client.html. [Accessed: 12- Nov- 2017].

[2] "Creating, Listing, and Deleting Amazon S3 Buckets - AWS SDK for Java", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/sdkfor-java/v1/developer-guide/examples-s3-buckets.html. [Accessed: 13- Nov- 2017].

[3] "AWS Regions and Endpoints - Amazon Web Services", Docs.aws.amazon.com, 2017. [Online]. Available: http://docs.aws.amazon.com/general/latest/gr/rande.html. [Accessed:16- Nov- 2017].