The Java API for Amazon Web Services is provided by the AWS SDK.3

Create an S3 client. S3 access is handled by the class AmazonS3Client instantiated with the account

credentials of the AWS user:

AmazonS3Client s3 = new AmazonS3Client(

new BasicAWSCredentials("your\_access\_key", "your\_secret\_key"));

The access and the secret keys can be found on the user’s AWS account homepage, as mentioned in

Section 11.3.

Buckets. An S3 bucket is analogous to a file folder or directory, and it is used to store S3 objects. Bucket

names must be globally unique; hence, it is advisable to check first to see whether the name exists:

s3.doesBucketExist("bucket\_name");

This function returns “true” if the name exists and “false” otherwise. Buckets can be created and deleted

either directly from the AWS Management Console or programmatically as follows:

s3.createBucket("bucket\_name");

s3.deleteBucket("bucket\_name");

S3 objects. An S3 object stores the actual data and it is indexed by a key string. A single key points

to only one S3 object in one bucket. Key names do not have to be globally unique, but if an existing

key is assigned to a new object, the original object indexed by that key is lost. To upload an object in a

bucket, we can use the AWS Management Console or, programmatically, a file local\_ f ile\_name can

be uploaded from the local machine to the bucket bucket\_name under the key key using

File f = new File("local\_file\_name");

s3.putObject("bucket\_name", "key", f);

A versioning feature for the objects in S3 was made available recently; it allows us to preserve, retrieve,

and restore every version of an S3 object. To avoid problems in uploading large files, e.g., dropped

connections, use the .initiateMultipartUpload() with an API described at the AmazonS3Client. To access

this object with key key from the bucket bucket\_name use:

S3Object myFile = s3.getObject("bucket\_name", "key");

To read this file, you must use the S3Object’s InputStream:

InputStream in = myFile.getObjectContent();

The InputStream can be accessed using Scanner, BufferedReader, or any other supported method.

Amazon recommends closing the stream as early as possible, since the content is not buffered and it is

streamed directly from the S3. An open InputStream means an open connection to S3. For example, the

following code will read an entire object and print the contents to the screen:

AmazonS3Client s3 = new AmazonS3Client(

new BasicAWSCredentials("access\_key", "secret\_key"));

InputStream input = s3.getObject("bucket\_name", "key")

.getObjectContent();

Scanner in = new Scanner(input);

while (in.hasNextLine())

{

System.out.println(in.nextLine());

}

in.close();

input.close();

Batch upload/download. Batch upload requires repeated calls of s3.putObject() while iterating over

local files.

To view the keys of all objects in a specific bucket, use

ObjectListing listing = s3.listObjects("bucket\_name");

ObjectListing supports several useful methods, including getObjectSummaries(). S3ObjectSummary

encapsulates most of an S3 object properties (excluding the actual data), including the key to access the

object directly,

List<S3ObjectSummary> summaries = listing.getObjectSummaries();

For example, the following code will create a list of all keys used in a particular bucket and all of the

keys will be available in string form in List < String >allKeys:

AmazonS3Client s3 = new AmazonS3Client(

new BasicAWSCredentials("access\_key", "secret\_key"));

List<String> allKeys = new ArrayList<String>();

ObjectListing listing = s3.listObjects("bucket\_name");

for (S3ObjectSummary summary:listing.getObjectSummaries())

{

allKeys.add(summary.getKey());

}

Note that if the bucket contains a very large number of objects, then s3.listObjects() will return a

truncated list. To test if the list is truncated, we could use listing.isTruncated(); to get the next batch of

objects, use

s3.listNextBatchOfObjects(listing)};

To account for a large number of objects in the bucket, the previous example becomes

AmazonS3Client s3 = new AmazonS3Client(

new BasicAWSCredentials("access\_key", "secret\_key"));

List<String> allKeys = new ArrayList<String>();

ObjectListing listing = s3.listObjects("bucket\_name");

while (true)

{

for (S3ObjectSummary summary :

listing.getObjectSummaries())

{

allKeys.add(summary.getKey());

}

if (!listing.isTruncated())

{

break;

}

listing = s3.listNextBatchOfObjects(listing);

}