Jessica Nguyen

CSSE 436: Cloud Computing

Program 3: AWS S3

Table of Contents

[\*\*\* SETUP INSTRUCTIONS \*\*\* 1](#_Toc63966600)

[\*\*\* COMPILATION INSTRUCTIONS \*\*\* 1](#_Toc63966601)

[\*\*\*IMPLEMENTATION DESIGN\*\*\* 3](#_Toc63966602)

[\*\*\*EXAMPLE OUTPUTS\*\*\* 5](#_Toc63966603)

This program recursively traverses the files of a directory and makes a backup to

the cloud. The program should also be able to restore from the cloud as well.

## \*\*\* SETUP INSTRUCTIONS \*\*\*

1. If you don’t already have Maven, install Apache Maven:
2. Open up Mac Terminal and unzip my submission, you should see a folder named “apache-maven-3.6.3”. I have included the maven package that I use for your convenience.
3. From your terminal, add the bin directory apache-maven-3.6.3/bin to your $PATH variable by typing: “ sudo nano /etc/paths ”
4. Enter your password, when prompted.
5. Go to the bottom of the file, and enter “ /Users/jessicanguyen/Downloads/apache-maven-3.6.3/bin ”
6. Replace the red part with your path file to the maven “apache-maven-3.6.3/bin” directory
7. Hit ^+X key to exit and “Y” to save what you have edited
8. In a **NEW Tab** in Terminal, enter “mvn -v” to verify you have downloaded maven
9. If there is a error pop up that said *“ “libjansi.jnilib” cannot be opened because the developer cannot be verified”,* hit cancel and proceed to see maven verified on your MacOS
10. More info can be found on this [site](https://maven.apache.org/install.html) for backup scenario when you can’t download maven.
11. You also have to have AWS Configuration with credentials on your computer with access\_key, secret\_key, and region all set up.

## \*\*\* COMPILATION INSTRUCTIONS \*\*\*

All necessary files for assembly references are included in the project submission:

- HW3 folder

I used **MacOS and** **Linux** to compile test my program. I have **java** openjdk version "15" 2020-09-15. I have **Apache Maven** version 3.6.3.

1. Compile java file using Maven:
2. Open a terminal or command prompt window and navigate to project directory “HW3”

**Note**: there are many more subdirectories in this HW3 folder (like src, target, root) – they are a part of the Maven project. Make sure you stay at the folder HW3 only.

1. Use the following 2 commands to build your project, wait until everything is finished building (might take a while to finish downloading all the dependencies):

mvn package

mvn clean compile assembly:single

1. By now, you should see that there is a “HW3-0.0.1-SNAPSHOT-jar-with-dependencies.jar” file in the HW3/target folder and it has been compiled successfully
2. Run java file using Maven by following these commands:

**NOTE**: please allow for at most 60 seconds from updating/changing/deleting your local files before running “backup/restore” command. It takes a while for S3 to reflect the new changes from the last command, so backup might not recognize that there has been a modification in data in order to run the algorithm correctly.

1. For backup: replace the part in red with any local directory on your computer you want to upload to S3; replace the part in blue with your cloud bucket and directory

java -cp target/HW3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.test.HW3.backup /Users/jessicanguyen/Documents/CSS436/prog3/sampleTestFolder bucket070112312342::hello

1. For restore: replace the part in red with any local directory on your computer you want to restore from S3; replace the part in blue with your cloud bucket and directory

java -cp target/HW3-0.0.1-SNAPSHOT-jar-with-dependencies.jar com.test.HW3.restore backup1612734166490::hello /Users/jessicanguyen/Documents/CSS436/prog3/sampleTestFolder

## \*\*\*IMPLEMENTATION DESIGN\*\*\*

1. Libraries used: [com.amazonaws](https://aws.amazon.com/sdk-for-java/) SDK for Java
2. BackUp.java design:

* void backUp(AmazonS3 s3Client, String dirPath, String name);
  + Purpose: backup files specified in the local directory dirPath and uploads it to the S3 bucket “name” of the AWS User “s3Client”
  + Will handle dirPath with either relative path or absolute path
  + Preconditions: dirPath and name are initialized String objects, s3Client exists in AWS with credentials set up in your environment
  + Postconditions: returns ERROR if the input name isn’t in the form bucket::directory.

Create a new bucket named “name” if the input user doesn’t have that bucket in S3. The new bucket name will be input “name” plus the current timestamp in milliseconds.

Returns without running the backup algorithm if there are no changes locally to be uploaded onto cloud to save bandwidth.

Succesfully upload new files onto cloud if there are new changes locally.

* + Exceptions: throws AmazonServiceException e;

throws AmazonClientException e;

throws InterruptedException e;

* boolean bucketExists(AmazonS3 user, String name);
  + Purpose: checks if the bucket “name” exists on S3 for the input AWS user “user”
  + Preconditions: name is an initialized String object, user exists in AWS with credentials set up in your environment
  + Postconditions: return true if bucket exists

Return false if bucket doesn’t exist

* boolean compare(AmazonS3 user, String localPath, String bucket, String key);
  + Purpose: recursively compare if the local directory “localPath” has the same structure and file timestamps as the cloud bucket “bucket” does
  + Preconditions: localPath, bucket and key are initialized String objects, user exists in AWS with credentials set up in your environment
  + Postconditions: returns false if the cloud doesn’t have a file that exists locally in given directory “localPath”.

Returns false if the timestamp of the local file is later than the cloud’s timestamp (meaning it has been modified locally but not yet reflected onto the cloud).

Returns true otherwise.

1. Restore.java design:

* restore(AmazonS3 s3, String name, String dirPath);
  + Purpose: restore files specified in the S3 bucket “name” of the AWS User “s3Client” into the local directory “dirPath”
  + Will handle dirPath with either relative path or absolute path
  + Preconditions: dirPath and name are initialized String objects, s3Client exists in AWS with credentials set up in your environment
  + Postconditions: returns ERROR if the input name isn’t in the form bucket::directory.

Create a new local directory named “dirPath” if the input user doesn’t have that local directory.

Succesfully restore cloud files onto local directory

* + Exceptions: throws AmazonServiceException e;

throws AmazonClientException e;

throws InterruptedException e;

**NOTE**: please allow for at most 60 seconds from updating/changing/deleting your cloud files before running “restore” command. It takes a while for S3 to reflect the new changes from the last command, so restore might not recognize that there has been a modification in data in order to run the algorithm correctly.

## \*\*\*EXAMPLE OUTPUTS\*\*\*

1. For backup:

Text

Description automatically generated

In my local root folder, it looks like: Background pattern

Description automatically generated

In my cloud S3, it looks like:

Graphical user interface, text, application, email

Description automatically generated

If all file structures match and there is no need for a backup:

Text

Description automatically generated

If the local folder is empty, will generate an empty key on S3:

Text

Description automatically generated

1. For restore:

Text

Description automatically generated