CSCI 4145/5409 Cloud Computing – Summer 2023
Compute & Storage Assignment

Reminder: This is an individual assignment. You are not allowed to collaborate with anyone else when completing this assignment. You can borrow code and configuration snippets from internet sources that are not from students in this class, however that code must be cited and include comments for how you have modified the original code.

Introduction

This assignment will measure your understanding of the main compute and storage mechanisms of our cloud provider AWS. This assignment assures us that you have attended the tutorials and learned about AWS EC2 and S3, or that you have found some other way to learn these services. In addition, you will have to do some self-learning to study how to use the gRPC framework.

Learning Outcomes

- You understand how to launch AWS Elastic Compute (EC2) instances.
- You understand how to connect to an EC2 instance and provision it to support your web applications.
- You understand the AWS Simple Storage Service (S3) and how to create a bucket.
- Experience working with AWS libraries, like boto3, that allow you to perform AWS operations such as creating a file on S3.
- More experience building APIs.

Requirements

You will build a gRPC server with any language or framework you like, **deployed on an EC2 instance**. You can use the quick start guide from the official gRPC documentation, and build gRPC methods based on that code. Here's a Python quick start and Node.Js quick start, you can select other compatible programming languages if you prefer. gRPC uses Protocol Buffers, for serializing structured data, you will learn more on how to use these with gRPC in the quick start links mentioned previously, however, here is protobuf code that can be used by your gRPC server (use the file name *computeandstorage.proto* for this file),

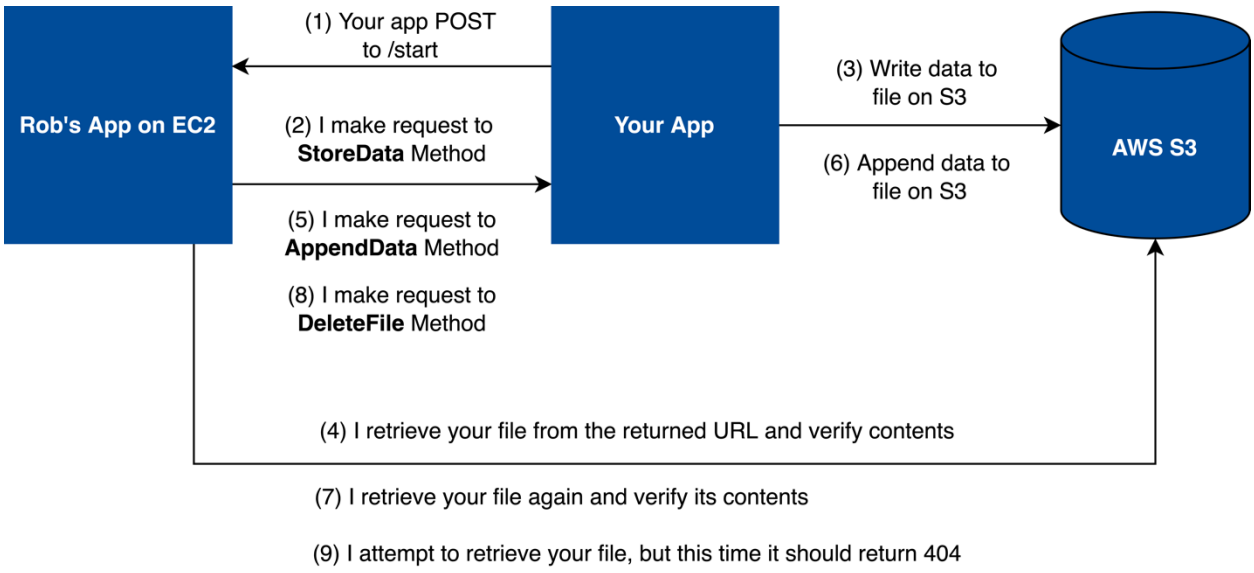
```
syntax = "proto3";

package computeandstorage;

service EC2Operations {
  rpc StoreData (StoreRequest) returns (StoreReply) {}
  rpc AppendData (AppendRequest) returns (AppendReply) {}
```

```
  rpc DeleteFile (DeleteRequest) returns (DeleteReply) {}
}

message StoreRequest {
  string data = 1;
}

message StoreReply {
  string s3uri = 1;
}

message AppendRequest {
  string data = 1;
}

message AppendReply {}

message DeleteRequest {
  string s3uri = 1;
}

message DeleteReply {}
```

I will build a web application to test your gRPC server's methods. Your server will "introduce" itself to my application by sending a POST request with some JSON to a URL I provide to you that begins a chain of events:

1. My app will make a request to your app's **StoreData** method with a request message that gives you data to store in a file on S3.
2. Your app will retrieve the data from the request message and use an AWS library to programmatically store the data in a file you create on S3.
3. Your app will return the **publicly readable** URL for the file you created on S3.
4. My app will download your file and verify that it contains the data I sent you.
5. My app will make a request to your app's **AppendData** method with request message that gives you data to **append** to the same file on S3 that you created in step 1.
6. Your app will retrieve the data from the request message and again use an AWS library to append the data to the file you created in step 2 on S3.
7. My app will again download the file and verify that it contains the correct data.
8. My app will make a request to your app's **DeleteFile** method with some request message that gives you the public URL you gave to me, your app will use that URL to figure out which file to delete on S3 and will delete the file from S3.
9. My app will attempt to download your file, it expects to get a 404 status code response indicating the file has been deleted.

When you are finished the system will look and function like this:



Please keep in mind that when I use <>'s I am trying to convey to you that you need to replace the information with the real value, don't include the <>'s or you'll break our code that checks your responses.

JSON your gRPC server sends to my app's /start

Your app will send me the following JSON in your POST to /start

```
{
   "banner": "<Replace with your Banner ID, e.g. B00123456>",
   "ip": "<Replace with the IPv4 of your EC2 instance, e.g.
74.23.154.12>"
}
```

Request message my app sends to your app's StoreData

When you POST to my app's /start endpoint with valid JSON my app will immediately interact with yours by sending a POST with the following JSON to your app's /storedata endpoint:

```
{
   "data": "<A string you must store in a file on S3>"
}
```

Your app's response message to StoreData method:

When my app sends the request message above to your StoreData method, after you create the file on S3 you must return the following response message:

```
{
   "s3uri": "<Public URL of the S3 file you created>"
}
```

Request message my app sends to your app's AppendData

```
{
   "data": "<A string to append to the existing file on S3>"
}
```

Request message my app sends to your app's DeleteFile

```
{
   "s3uri": "<Public URL of the S3 file you created>"
}
```

Marking Rubric

In this class I'm not very concerned about the quality of the code you write, if you write bad quality code it is you that will suffer in maintaining and supporting it (especially on your term assignment). I care that you can meet the learning objectives defined at the top of this document, and I can verify this by simply verifying the correct behaviour of your app's interaction with mine.

Your submission will be marked by the app that I will write, my app will:

- Listen for requests to /start, and initiate the check process:
    1. Records the IP you send to /start in DynamoDB.
    2. Sends a request to your IP's StoreData method.
    3. Retrieves the file from the URL you returned.
    4. Verifies that the file contains the correct string.
    5. Sends a request to your IP's AppendData method.
    6. Retrieves the file from the URL you returned in step 3.
    7. Verifies that the file contains the correct string.
    8. Sends a request to your IP's DeleteFile endpoint.
    9. Verifies that the file can no longer be downloaded from the URL you returned in step 3.

Your mark is entirely based on the success of steps 1 - 9:

- **Your app posts to /start from an AWS IP address: 40%**
- **Your app stores a file on S3 and returns the URL when StoreData is called: 20%**
- **The file from S3 retrieved via the returned URL contains the string sent to your app: 10%**
- **Your app appends to the same file on S3 when AppendData is called: 10%**
- **The file from S3 retrieved via the original URL contains the string sent to your app to append: 10%**
- **Your app properly deletes the file when DeleteFile is called: 10%**

I care about learning. If it takes you multiple attempts to learn the outcomes of this assignment, that's fine with me because in the end you learned! **Therefore, I will build my app such that only the performance of your highest scoring call to /start counts towards grading.**

Because your mark is entirely results based it makes sense for you to spend time testing to ensure it works! I again recommend that you use a tool like POSTman to test your app's behaviour. Apart from POSTman, to test gRPC methods there are more tools like gRPCurl.

How To Submit

Create a folder in your repository labeled **A2**. Put all your source code in this folder and push it to your individual repository on gitlab **before the assignment deadline.**

I will publish the IP of my running app a few days before the assignment deadline, I will leave it running. You must build, provision, and execute your app such that it makes its call to /start **before the assignment deadline.**