Intro. to Network Programming 2020 Spring

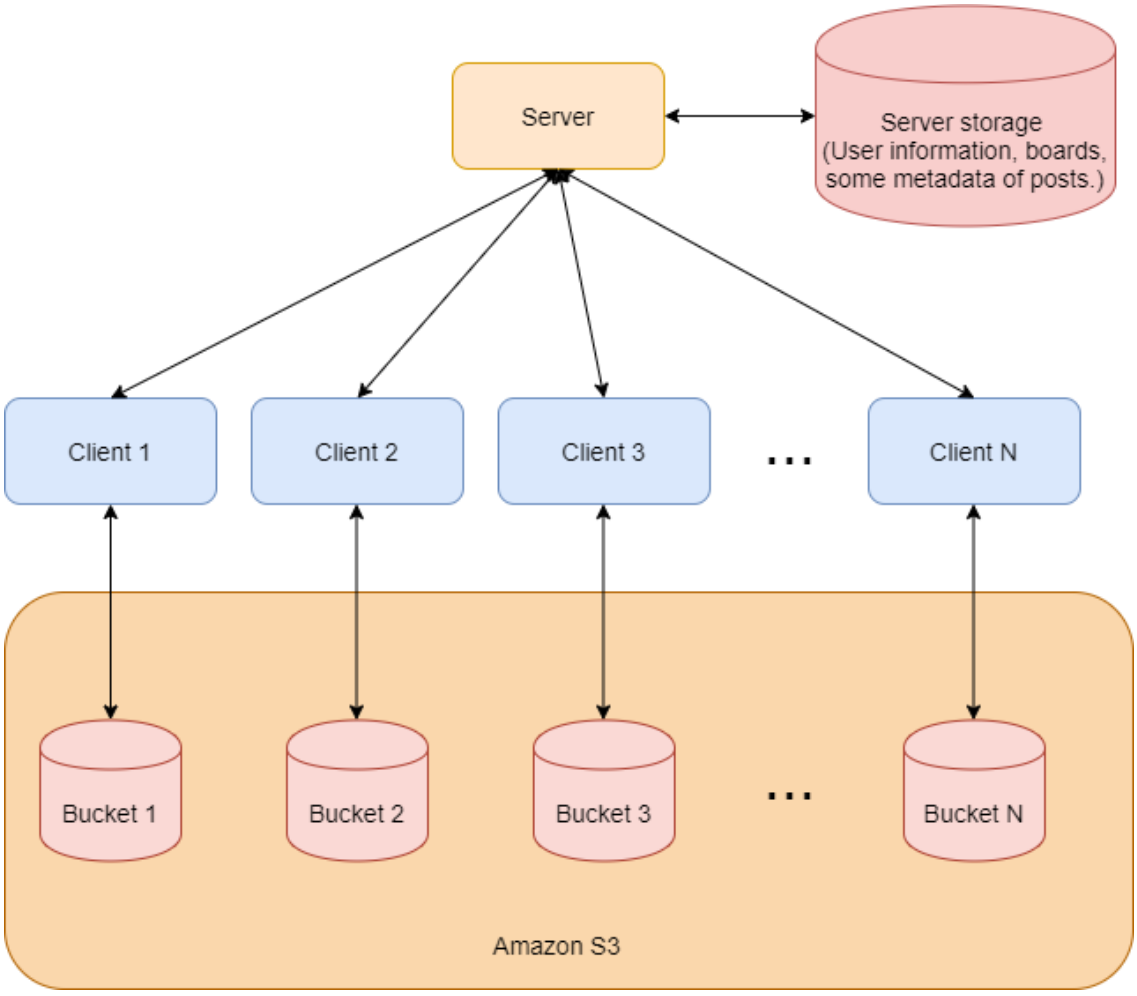
Homework 3 - Bulletin Board System: Part 3

Due on Sunday, May 24, 2020 by 11:55pm

Description

In this part, you are going to write **a client program for BBS service**. However, in this part, **the content of the post** will **store on the client-side (Amazon S3)** instead of the server-side. Take the post function of the BBS service, for example. The server in this part only stores the metadata of posts (e.g., **post id**, **post title**, **author** and **date**.) **except for the content of the post**. That is to say, each client has its storage to store the content of their posts, and we use **Amazon S3** for the storage service. Besides, you are also going to implement new features such as **a simple mail service**.

System Architecture



The server here only stores **user information, boards and some metadata of posts.** (e.g., post id, post title, etc). The real **content of the post** will **store on the client-side (Amazon S3)**. Each bucket of Amazon S3 represents the storage of each client. So, each client can **store the content of its posts in its bucket,** and this bucket is also an **incoming mailbox for the mail service of this client.** Note that **only your client program** can get access to Amazon S3 using **Amazon S3 API**.

Requirements

The service can serve **at least 10 clients.** Your **server** and **client program** must be able to **handle all commands in the previous part (output results must be the same as the previous part)**. For some commands such as **whoami**, **exit**, **logout**, **create-board**, **list-board ##<key>** and **list-post <board-name> ##<key>**, your client program only sends the command to the server and gets the corresponding result from the server. However, there are some commands that your **client program** will interact with **Amazon S3**. These commands are described as follows:

Command format	Description	Result	
register <username> <email> <password>	If successful execution, your **client program** will **create a new bucket in Amazon S3 for this new user**, and your server program will store the bucket name of this new user so that the user will be able to log in with its bucket in the future. If failed, only print the error message.	Success	Register successfully.
		Fail	Username is already used.
login <username> <password>	If successful execution, the user will **log in with its Amazon S3 bucket**, so that it can manipulate (e.g., upload posts, delete posts, etc) its bucket in the future. If failed, only print the error message.	Success	Welcome, <username>.
		Fail (1)	Please logout first.
		Fail (2)	Login failed.
create-post <board-name> --title <title> --content <content> (command is in the same line)	If successful execution, the **user (client)** will **upload the content of this post to its S3 bucket using Amazon S3 API**, and the **server** will **keep the metadata of this post**. If failed, only print the error message.	Success	Create post successfully.
		Fail (1)	Board does not exist.
		Fail (2)	Please login first.
read <post-id>	Show the post whose ID is <post-id>. If successful execution, the **client** will **get the content of this post from the post owner's bucket using Amazon S3 API** and print the result. If failed, only print the error message.	Success	Author :<Author1> Title :<Title1> Date :<Date1> -- <content> -- <User1>:<Comment1>
		Fail	Post does not exist.
delete-post <post-id>	Delete the post whose ID is <post-id>. If successful execution, **the user (client)** will **delete this post from its bucket**, and the **server** will **delete the metadata of this post**. If failed, only print the error message.	Success	Delete successfully.
		Fail (1)	Please login first.
		Fail (2)	Post does not exist.
		Fail (3)	Not the post owner.

update-post <post-id> --title/content <new>	Update the post whose ID is <post-id>. If a successful update on the title, the **server** will **update the title of this post**. If a successful update on the content, the **user (client)** will **update the content of this post in Amazon S3 using Amazon S3 API.** If failed, only print the error message.	Success	Update successfully.
		Fail (1)	Please login first.
		Fail (2)	Post does not exist.
		Fail (3)	Not the post owner.
comment <post-id> <comment>	Add a comment < comment > to the post whose ID is <post-id>. If successful execution, the **user (client)** will **append the comment to the content of this post in the post owner's bucket**. That is to say, the **comment stores in the post owner's bucket**. If failed, only print the error message.	Success	Comment successfully.
		Fail (1)	Please login first.
		Fail (2)	Post does not exist.

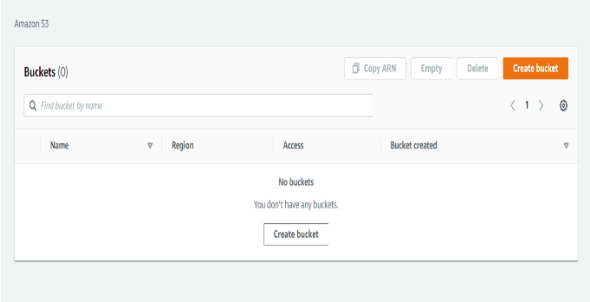
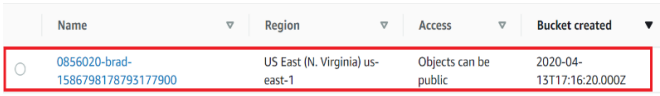
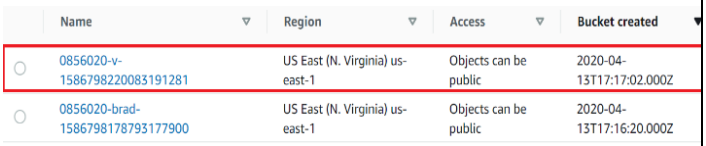
Also, there are some **new commands** you have to implement for **simple mail service**. These commands are described as follows:

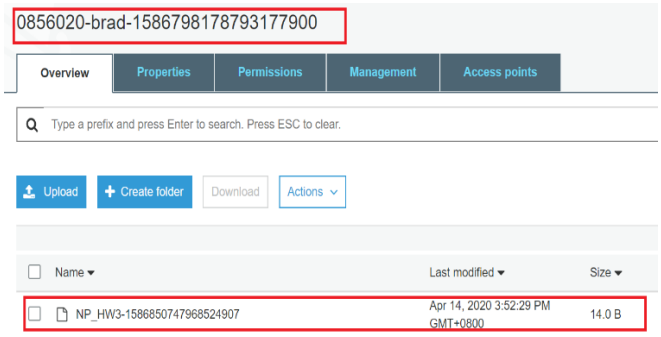
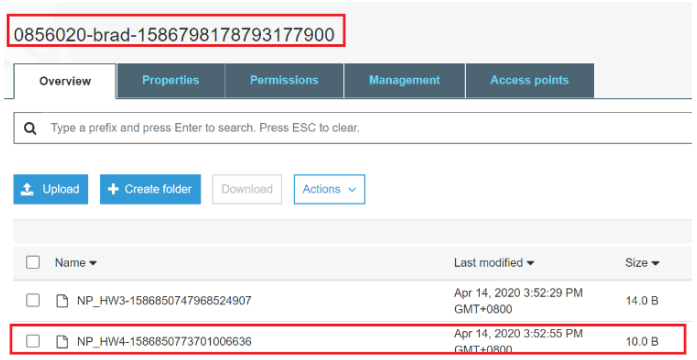
Command format	Description	Result	
mail-to <username> --subject <subject> --content <content> (command is in the same line)	Send a mail whose subject is <subject> and content is <content> to user <username>. Use --subject and --content to separate subject and content. <subject> has the same format as <title> of the post. <content> has the same format as <content> of the post. If successful execution, the **user (client**) will **create an object with <content> in user <username>'s bucket using Amazon S3 API.** Failed execution: Fail (1): No user logged in. Fail (2): User <username> doesn't exist.	Success	Sent successfully.
		Fail (1)	Please login first.
		Fail (2)	<username> does not exist.
list-mail	List all incoming mails of the current logged in user. Success: Note that **each user has its mail id sequence numbers**. **It should start at 1**.	Success	ID Subject From Date 1 <Subject1><From_user1> <Date1> 2 <Subject2><From_user2> <Date2>
		Fail (1)	Please login first.

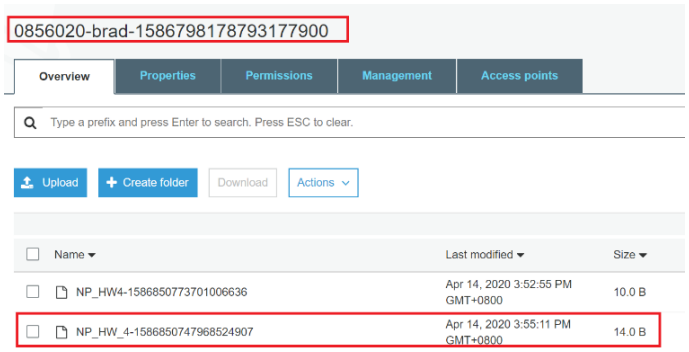
	<Subject1> represents the subject of this mail. <From_user1> represents that this mail is sent by user <From_user1>. <Date1> represents the sent date of this mail. **There is a \t between each column.** Failed execution: Fail (1): No user logged in.		
retr-mail <mail#>	Retrieve the content of the mail <mail#> If successful execution, the **user (client**) will **get the content of the mail from its bucket using Amazon S3 API** and print the result. <Subject1> represents the subject of this mail. <From_user1> represents that this mail is sent by user <From_user1>. <Date1> represents the sent date of this mail. **There is a \t before ':'.** Failed execution: Fail (1): No user logged in. Fail (2): Mail <mail#> is not in your mailbox.	Success	Subject :<Subject1> From :<From_user1> Date :<Date1> -- <content>
		Fail (1)	Please login first.
		Fail (2)	No such mail.
delete-mail <mail#>	Delete mail <mail#> from your mailbox. If successful execution, the **user (client**) will **delete the content of the mail from its bucket using Amazon S3 API**. Failed execution: Fail (1): No user logged in. Fail (2): Mail <mail#> is not in your mailbox.	Success	Mail deleted.
		Fail (1)	Please login first.
		Fail (2)	No such mail.

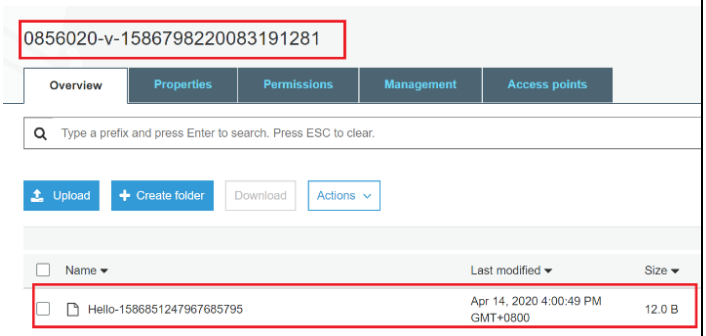
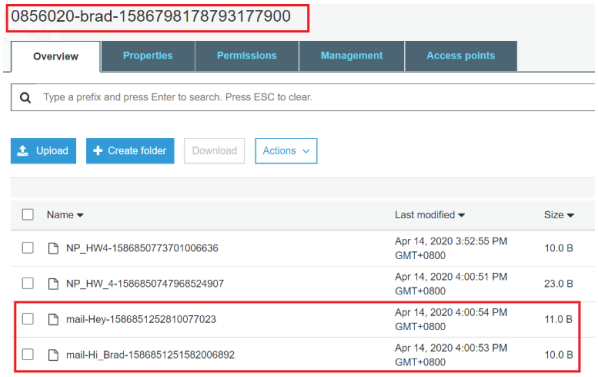
Scenario

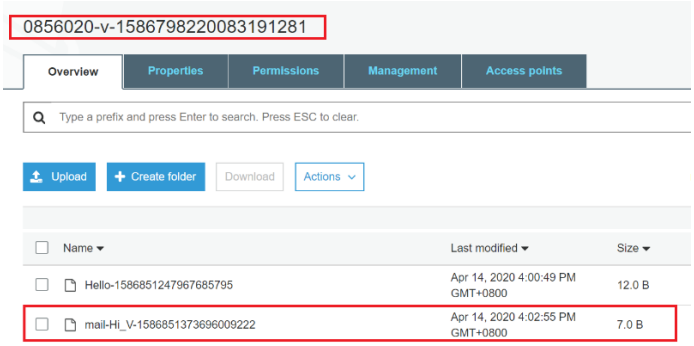
Run your server first, and run your client program to connect to your server. If the line only shows "% ", that means we type <Enter> in our client program. It is just for the height alignment of two columns here. You can ignore that. The sample outputs of the client program are listed as follows:

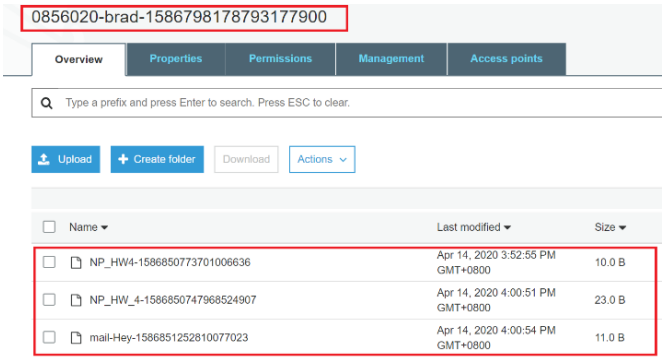
Terminal output	Description and Amazon S3 Console State
	Start with nothing in your Amazon S3. 
bash$ ./client 127.0.0.1 7890 ********** ** Welcome to the BBS server. ** ********** % register Brad bb@cs.nctu.edu.tw 12345 Register successfully. %	Create a bucket for user Brad. You can name the bucket name of each user by yourself, but there are some limitations of the bucket name. Please refer to the **Note** part. 
% register Brad bb@cs.nctu.edu.tw 12345 Username is already used.	Just print the error message. Nothing changes in Amazon S3.
% register V v@cs.nctu.edu.tw bt21 Register successfully. % % % %	Create a bucket for user V. 
% login Brad 12345 Welcome, Brad.	The client program will log in with Brad's bucket.
% whoami Brad	Nothing changes in Amazon S3.
% create-board NP_HW Create board successfully.	Nothing changes in Amazon S3.
% create-board NP_HW Board already exist.	Nothing changes in Amazon S3.
% list-board Index Name Moderator 1 NP_HW Brad % %	Nothing changes in Amazon S3.

% list-board ##HW Index Name Moderator 1 NP_HW Brad	Nothing changes in Amazon S3.
% create-post NP_HW --title NP_HW3 --content Err... Ha! Create post successfully. % % % % % % % % %	Create an object to store the content of this post in Brad's bucket. By the way, you can name the object by yourself, but there are some restrictions for object naming. Please refer to the **Note** part. 
% create-post NCTU --title NP_HW3 --content Uh... Board does not exist.	Nothing changes in Amazon S3.
% create-post NP_HW --title NP_HW4 --content Wow... Create post successfully. % % % % % % % %	Create a new post. 
% list-post NP Board does not exist. % %	Nothing changes in Amazon S3.
% list-post NP_HW ID Title Author Date 1 NP_HW3 Brad 04/14 2 NP_HW4 Brad 04/14	Nothing changes in Amazon S3.
% list-post NP_HW ##HW3 ID Title Author Date 1 NP_HW3 Brad 04/14	Nothing changes in Amazon S3.
% read 888 Post does not exist. % % %	Nothing changes in Amazon S3.

% read **1** Author :Brad Title :NP_HW3 Date :2020-04-14 -- Err... Ha! --	Get the content of post 1 from the post owner's bucket.
% update-post **888 --title NP HW_4** Post does not exist.	Nothing changes in Amazon S3.
% update-post **1 --title NP HW_4** Update successfully. % % % % % % % % % % %	The state of Amazon S3 may be unchanged. It depends on your design. But for my design, I changed the object key of that post. 
% read **1** Author :Brad Title :NP HW_4 Date :2020-04-14 -- Err... Ha! --	Get the content of post 1 from the post owner's bucket.
% update-post **1 --content Yeah!** Update successfully.	Update the content of this post in Amazon S3.
% read **1** Author :Brad Title :NP HW_4 Date :2020-04-14 -- Yeah! --	Get the content of post 1 from the post owner's bucket.
% logout Bye, Brad.	Nothing changes in Amazon S3.
% whoami	Nothing changes in Amazon S3.

Please login first.	
% logout Please login first.	Nothing changes in Amazon S3.
% login **V** bt21 Welcome, V.	The client program will login with V's bucket.
% create-post NP_HW --title Hello --content I am V. Create post successfully. **%** **%** **%** **%** **%** **%** **%**	
% update-post 1 --content Ha! ha! Not the post owner.	Nothing changes in Amazon S3.
% delete-post 1 Not the post owner.	Nothing changes in Amazon S3.
% comment 888 Ha ha! Post does not exist.	Nothing changes in Amazon S3.
% comment 1 Ha ha! Comment successfully. **%**	The client program will append the comment to the corresponding post object (e.g., NP_HW_4-1586850747968524907 object in Brad's bucket) in Amazon S3.
% read 1 Author :Brad Title :NP HW_4 Date :2020-04-14 -- Yeah! -- V:Ha ha!	Get the content of post 1 from the post owner's bucket.
% mail-to TT --subject Hi TT --content Hi TT! TT does not exist.	Nothing changes in Amazon S3.
% mail-to Brad --subject Hi Brad --content Hi Brad! Sent successfully. **% mail-to Brad --subject Hey --content Hey Brad!** Sent successfully. **%** **%** **%** **%** **%**	User V creates two email objects in Brad's bucket. 

% list-mail ID Subject From Date	Nothing changes in Amazon S3.
% delete-mail **1** No such mail.	Nothing changes in Amazon S3.
% retr-mail **1** No such mail.	Nothing changes in Amazon S3.
% logout Bye, V.	Nothing changes in Amazon S3.
% login **Brad** 12345 Welcome, Brad.	The client program will login with Brad's bucket.
% list-mail ID Subject From Date 1 Hi Brad V 04/14 2 Hey V 04/14	Nothing changes in Amazon S3.
% retr-mail **1** Subject :Hi Brad From :V Date :2020-04-14 -- Hi Brad!	Get the content of the mail from his bucket.
% mail-to V --subject Hi V --content WoW V! Sent successfully. % % % % % % %	User Brad creates a mail object in V's bucket. 
% logout Bye, Brad.	Nothing changes in Amazon S3.
% login **V** bt21 Welcome, V.	The client program will login with V's bucket.
% list-mail ID Subject From Date 1 Hi V Brad 04/14	Nothing changes in Amazon S3. But note that user V has its mail id sequence number. It starts at 1.
% logout Bye, V.	Nothing changes in Amazon S3.
% login **Brad** 12345 Welcome, Brad.	The client program will login with Brad's bucket.

% list-mail ID Subject From Date 1 Hi Brad V 04/14 2 Hey V 04/14	Nothing changes in Amazon S3.
% delete-mail 1 Mail deleted. % % % % % % % % %	User Brad deletes the object corresponding to his mail 1 from his bucket. 
% list-mail ID Subject From Date 1 Hey V 04/14	Nothing changes in Amazon S3. But note that ID of the mail with the subject “Hey” is 1.
% logout Bye, Brad.	Nothing changes in Amazon S3.
% exit	Nothing changes in Amazon S3.

Note

1. **About bucket naming**:
   - Bucket names must be **unique** across all existing bucket names in Amazon S3.
   - Bucket names must be **at least 3 and no more than 63 characters long.**
   - Bucket names must **not contain uppercase characters or underscores.**
   - Bucket names must **start with a lowercase letter or number**.
2. **About object key name**:
   - Unique identifier within a bucket. If you upload the same key name object without versioning-enabled, it will overwrite the original one.
   - The following character sets are generally safe for use in key names.

Alphanumeric characters	• 0-9 • a-z • A-Z
Special characters	• ! • - • _ • . • * • ' • (•)

For more details about buckets and objects, you can refer to reference [5][6]

3. **About AWS API access key**:

   If you want to use AWS SDK to make Amazon S3 API calls, you have to provide your AWS credential first.

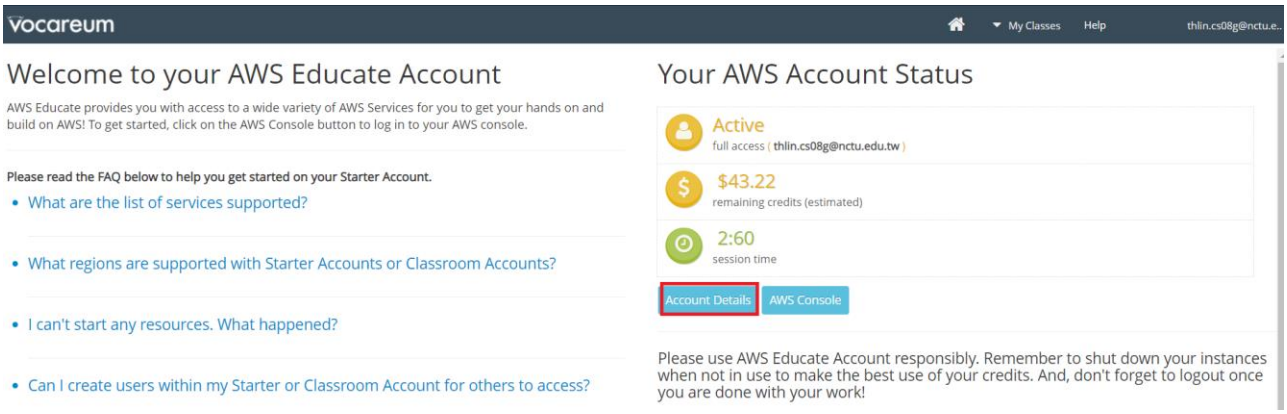
   How to set up authentication credential:

   Create a credential file at ~/.aws/credentials. The content of this file is described as follows:

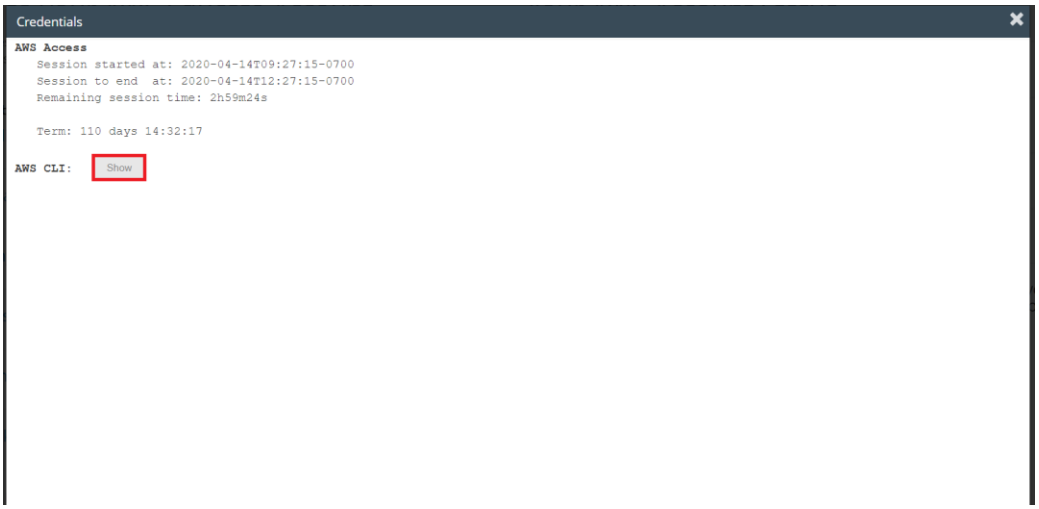
```
[default]
aws_access_key_id=<your access key>
aws_secret_access_key=<your secret access key>
aws_session_token=<your session token>
```

   You can get these key from your AWS Educate account. Log in your account and go to Intro. to Network Programming classroom. Then, you will see the following page.

   Click Account Details



Click Show and copy those keys into ~/.aws/credentials



The credential we use here is temporary, so you have to copy and paste again when the credential expiration.

4. **About C++ AWS SDK**:

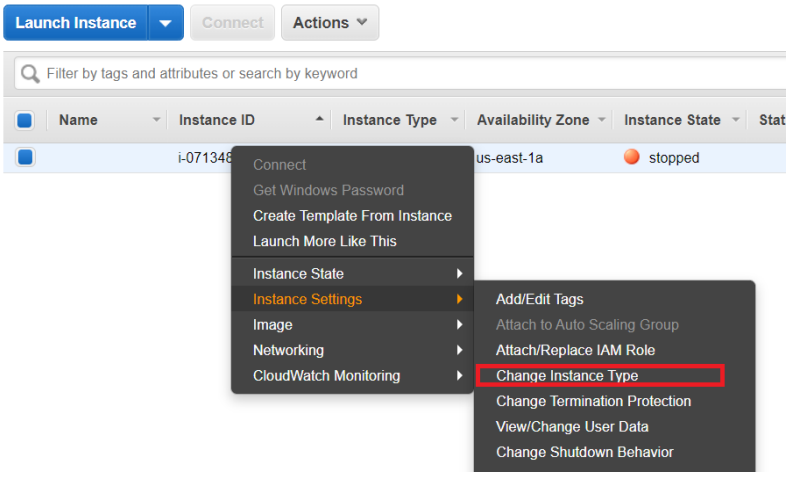
To use the AWS SDK for C++, you need:

- Visual Studio 2015 or later
- *or* GNU Compiler Collection (GCC) 4.9 or later
- *or* Clang 3.3 or later
- A minimum of 4 GB of RAM

So, if you currently use Amazon EC2 with instance type of t2.micro, please change to at least t2.medium. Moreover, please backup all your environment when you are running out of classroom credits. We suggest you write a script to set up your environment. It will save a lot of time.

How to change instance type:

Right-click your original instance -> Instance Settings -> Change Instance Type



5. **You can use any code examples in the Amazon S3 API documentation to manipulate your Amazon S3.**

Grade (100%)

For those commands that interact with Amazon S3, we will check your Amazon S3 console. If the state of your Amazon S3 is wrong, we will deduct some scores of that command.

- register command – (8%)
- login command – (8%)
- create-post command – (8%)
- read command – (8%)
- delete-post command – (8%)
- update-post command – (8%)
- comment command – (8%)
- mail-to command – (8%)
- list-mail command – (8%)
- retr-mail command – (8%)
- delete-mail command – (8%)
- list-post command – (5%)
- logout command – (2%)
- exit command – (2%)
- whoami command – (1%)
- create-board command – (1%)
- list-board command – (1%)

Submission

Please upload a zip file called “hw3_{$student_id}.zip” (e.g., hw3_0856020.zip) that includes your source code. It must include at least your **server source code** and **client source code**. Submission that doesn’t follow the rule will **get 20% punishment** on the grade.

You will get **0 points** on this project for **plagiarism**. Please don’t copy-paste other students’ code!

Reference

1. C/C++ Socket
2. SQLite C/C++ Interface
3. Linux socket SELECT
4. AWS SDK supported languages

5. Bucket Restrictions and Limitations
6. Object Key and Metadata
7. AWS Command Line Interface
8. AWS SDK for C++
9. AWS SDK for Python
10. Sequence diagram reference