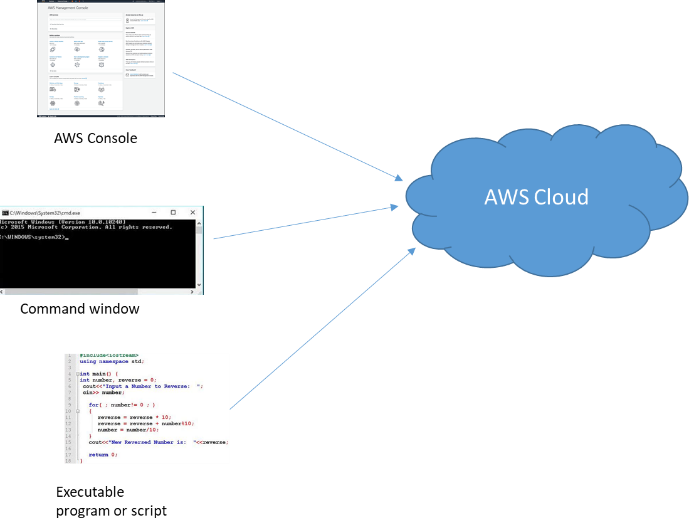
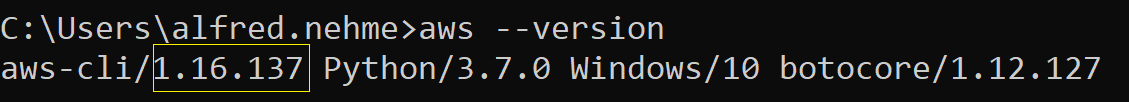
Module 5  
Install the AWS CLI and Configure your Security Credentials

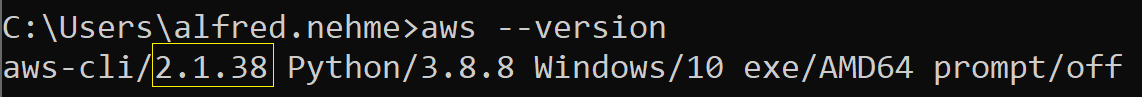
Overview

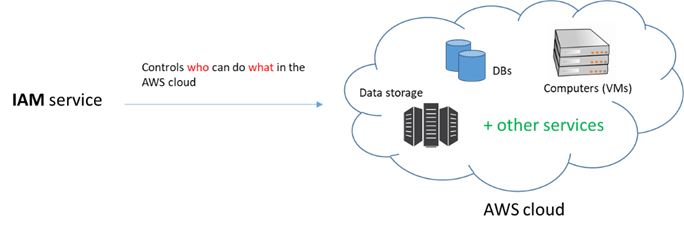
There are three ways to interact with AWS (Figure 1):

1. Using the **AWS Console** (that is, the AWS website UI). You already did that in Module 5 – Identity Access Management.
2. Using the **AWS CLI** (Command Line Interface). That is, by typing commands in a command or terminal window. This is the subject of this module.
3. Using a programming language-specific **AWS SDK**. There are AWS SDKs for most popular languages such as Python, C#, Java, Go, etc.

  
**Figure 1**: Three ways to interact with AWS: the AWS Console (that is, the AWS website), CLI, and APIs.  
  
  
The AWS CLI gives you command line access to AWS Services. The CLI allows you to do all the functionalities that you can do from the AWS Console. Knowledge of the CLI is indispensable since most developers automate operations, something that cannot be done using the AWS console UI.  
  
There are two versions of the AWS CLI: version 1 and version 2. We will use the newer version 2.  
If you already have the AWS CLI installed on your computer, then first check its version. You can do this by opening a terminal window and running the command: aws --version. The snapshot below shows that the AWS CLI is installed and it is version 1.

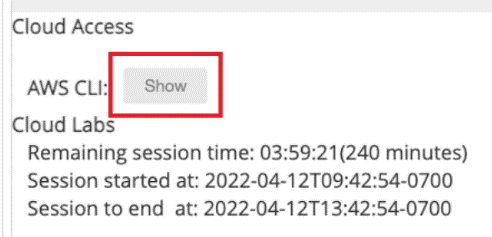
  
  
If you have version 1, then uninstall it first. If you don’t have the AWS CLI, then you can proceed with installing version 2.  
  
Install the AWS CLI Version 2  
  
For Windows, install the AWS CLI v2 from [here](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html#cliv2-windows-install).  
For Mac, install the AWS CLI v2 from [here](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-mac.html#cliv2-mac-install-gui).  
  
After finishing the installation, check that you have version 2 by running command: aws –version  
(the version you have might be different than mine – which is fine).

  
  
  
Configuring the Credentials File  
  
Recall from Module 4 that any time you interact with AWS you need to first authenticate yourself. And IAM has many primitives for doing that (Figure 2).



**Figure 2**: The IAM service is a collection of security primitives used to control who can do what in the AWS cloud.

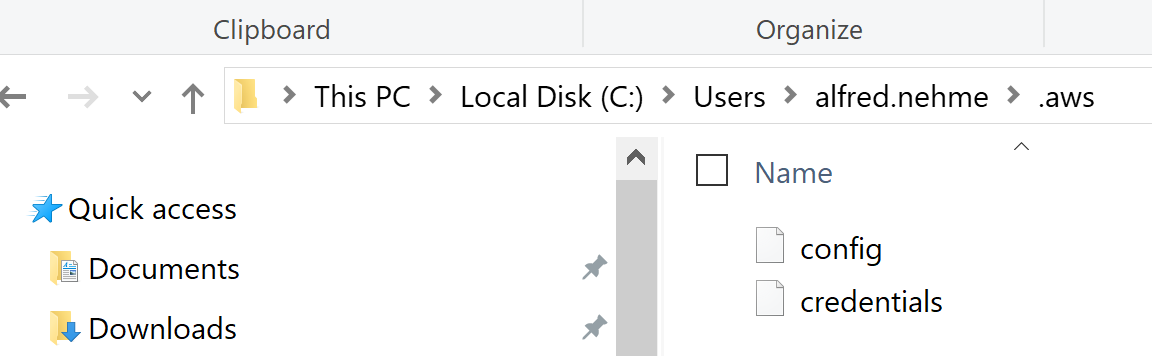
When you access the AWS Console website you usually use some email/password or account number/username/password to authenticate yourself. When you use the CLI (or SDKs) you are also doing things inside AWS and, therefore, there is a need to control access to who can do what. When you use the AWS CLI (or the AWS APIs), you don’t specify a username/password the way you do when you login to the web console because it is not practical to pass a username/password with every command or function call. Instead, you specify an **Access Key ID** and a **Secret Access Key** that are associated with your IAM user. Because the Access Key ID and Secret Access Key are associated with your IAM user account, they are pieces of information that say who you are. Therefore, think of the Access Key ID and Secret Access Key as the equivalents of your username/password credentials, but geared for programmatic use.  
  
Locate your Access Key ID and Secret Access Key:

1. Go to the AWS Academy Canvas site.
2. Click the AWS Details link on the menu in the upper-right side of the window.  
     
     
     
   (If you don’t see the AWS Details button, try starting the lab first).
3. Click the **Show** button:  
     
   

You should now see your IAM user Access Key and Secret Access Key. You should also see an AWS session token.

1. Copy the entire content of the box (including the [default] label). You will need to paste all of it shortly.

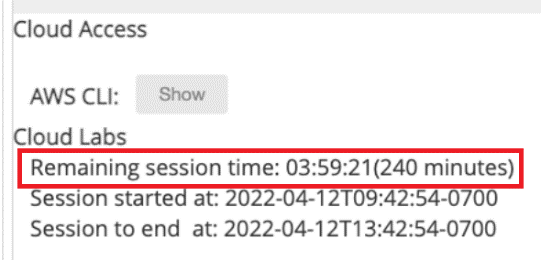
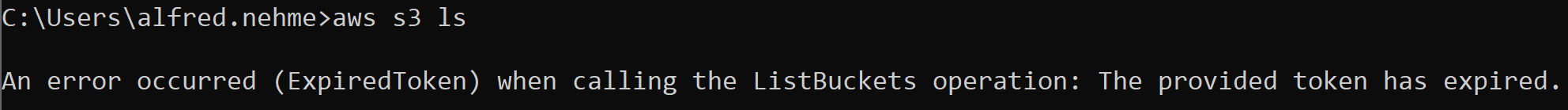
Configure the credentials File:

1. Open a command window and type this command: aws configure
2. For AWS Access Key ID, don’t type anything. Press Enter.
3. For AWS Secret Access Key, don’t type anything. Press Enter.
4. Take the default us-east-1 region (nothing to type). Press the Enter key.
5. Take the default json output format (nothing to type). Press the Enter key.
6. This should create a **.aws** folder in your home directory with 2 files in it: **config** and **credentials**.  
   (In Windows you can go to your home directory by typing %UserProfile% in the address bar of File Explorer).  
     
   

Note that if you do not see the .aws folder, you need to enable showing hidden files and folders in Windows. You do this from File Explorer: Click the **View** tab, click the **Options** button and select menu **Change folder and search options**. In the Folder Options window, click the **View** tab. Then check the **Show hidden files, folders, and drives** option button.

In Linux, you see hidden files and folders by using the -a switch with the ls command. So the following command shows you hidden files and folders:  
  
ls -al

1. Open the credentials file in an editor like Notepad. Delete the first 4 lines (and their values):  
     
   [default]  
   aws\_access\_key\_id=…  
   aws\_secret\_access\_key=…  
   aws\_session\_token=…
2. Paste the content you copied in step (d) of “Locate your Access Key ID and Secret Access Key”. Your credentials file should now look like this:

Save the credentials file.  
  
We said that the **aws\_access\_key\_id** and **aws\_secret\_access\_key** are used as the equivalents of username/password – credentials to authenticate you to the AWS cloud. How about the **aws\_session\_token**? The session token makes these credentials temporary by expiring them after a set amount of time. This adds an additional layer of security: by making the credentials keys temporary and expire after a set period of time, we limit the damage in case of accidental exposure or loss of the keys to an unauthorized user. You can see the expiration time of the keys in the AWS Details box (it is set to 240 minutes or 4 hours).  
  
  
  
After the credentials expire, any calls that you make with those credentials will fail (Figure 3). To correct this issue you must get a new set of credentials (by again clicking the Show button, and pasting them in the credentials file).   
  
  
**Figure 3**: Error indicating that the keys have expired. To correct this problem, get a new set of keys.  
  
Note that you do NOT have to re-run the aws configure command. When your credentials expire, just open the credentials file in a text editor and paste the new credentials.  
  
What appears in between the [ ] (default in this case) is a profile name. You can have more than one sets of keys in the credentials file and you can give each set some name. For example, the credentials file can look like this:  
  
[default]  
aws\_access\_key\_id=…  
aws\_secret\_access\_key=…  
aws\_session\_token=…  
  
[test]  
aws\_access\_key\_id=…  
aws\_secret\_access\_key=…  
aws\_session\_token=…  
  
[dev]  
aws\_access\_key\_id=…  
aws\_secret\_access\_key=…  
aws\_session\_token=…

In general, keys don’t expire and the session token is not used. Your AWS Educate account, however, is set up in this way.

In the above example, the credentials file has 3 profiles: “default”, “test”, and “dev”. We will see a bit later how to specify which profile to use when executing a CLI command.

Just like any credentials such as username/password, you should not share your keys with someone else.

1. Open the config file (also in the .aws folder) and notice that there is a **region** variable associated with a profile (e.g., region = us-east-1). We will talk about regions and availability zones later in the course. For now, just think of it as a geographical location that maps to one or more data centers. In this same file you will also see an **output** variable (with default value of json). This means that for some commands the output will be shown to you in json format.

What you did in steps (a) to (i) now allows you to use the CLI without having to type a username/password with every command. The AWS CLI is going to use data from your .aws/credentials file to authenticate you.  
  
In the next section we will test a few CLI commands to see how they work. We won’t test every single command because there are thousands of them. But you will see the general pattern utilized and how to get documentation help to correctly type a command and figure out its options/arguments.

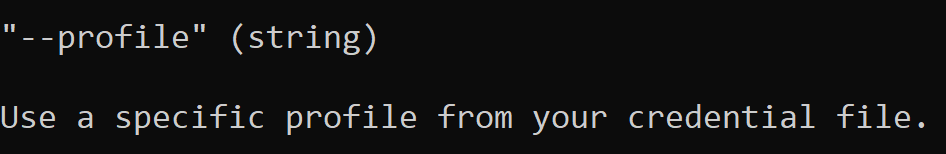
Using the AWS CLI

1. Open the command prompt window and type the following command (followed by the Enter key): aws s3 ls

If you don’t have any buckets in S3 (most likely you don’t) the command should return nothing.   
If you do, the command should list the buckets you have (we will discuss S3 and S3 buckets in a later module).

1. Type this command: aws s3 help  
     
   This should give you help about the AWS S3 service. If you scroll to the bottom (or keep clicking the space bar) you can see the available aws S3 commands.
2. The above syntax is the same for all services. The format is:  
     
   aws <service\_name> help  
     
   To see the help for the IAM service, try this: aws iam help  
     
   Press the space bar to scroll down until you start seeing some commands that might sound familiar to you after completing the IAM module. For example you should see commands like create-user, create-role, create-policy, delete-policy, etc.
3. Most of the time we are interested in a given functionality in a service. For example, let’s assume we want to find out how to add a user to a group in IAM (the command is called add-user-to-group). To get help on add-user-to-group, run this command:  
     
   aws iam add-user-to-group help  
     
   Notice that there is also a pattern here:  
     
   aws <service\_name> <command\_name> help
4. Here is the general format of an AWS command:  
     
   aws <service> <command> [options and parameters]  
     
   aws: An executable program (called aws)  
   <service>: The service name (required)  
   <command>: A specific command of the service

[options and parameters]: optional parameters

1. We will now see a command example with parameters. We said that the credentials file can have more than one profile. How does the command know which one to use? When no profile name is specified in the command, the [default] profile is always used. If you need to use another profile, you need to specify the --profile switch. Type the following command and press the Enter key: aws help  
     
   Scroll down and find the --profile option:  
     
     
     
   You use this parameter to specify a profile other than default.  
     
   aws s3 ls // No profile specified: uses default  
   aws s3 ls --profile test // Profile explicitly specified (test)

The --profile is a **global** parameter and can be used with any AWS command. That is, it is not specific to specific commands. To view all AWS global parameters, type: aws help  
(If you type aws s3 ls help, you will not find --profile because --profile is not specific to the ls command).

1. Another keyword you see in aws commands is the wait keyword   
     
   aws <service> wait <command> [options and parameters]  
     
   Look at the 2 examples [here](https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-commandstructure.html#cli-usage-commandstructure-wait) to see how this is useful.

## Exercise to do:

1. Login to the AWS console and go to the S3 service.
2. If you haven’t created any S3 bucket yet (we will talk about these in a future module), you should see that you have 0 buckets.
3. Use the AWS CLI to create a bucket. To start, get help on what are the available command in the s3api service. That is, type:  
     
   aws s3api help  
     
   Once you find the command you want (remember we want to create a bucket), get help on that command:  
     
   aws s3api <command> help  
     
   NOTE: bucket names should be globally unique. So if you try to use easy names like “mybucket” or “testbucket”, your command is likely to fail because someone else has used these before you ☹.
4. After you create the bucket, go back to the AWS Console. You should see the bucket there.
5. Now delete the bucket using the appropriate command (again use help to find out the name of the appropriate command, and help again to learn about the parameters it expects)
6. After you delete the bucket, go back to the AWS Console. You should see that the bucket you created in step (d) was deleted.
7. Create another bucket (same as you did in part c).
8. Go to the AWS Console and upload 2 test files to it.
9. Use the CLI to list the content of the bucket (again use the aws s3api help to look for a command that lists the content of the bucket.
10. Try to use the CLI (not the AWS Console) to upload a 3rth file to the bucket (again use the aws s3api help command and look for a command that allows you to upload objects/files to the bucket
11. Go to the AWS Console, and verify that the file you uploaded using the CLI is there.

**What to Submit**  
  
Nothing to submit for this module (no quiz)