Homework Assignment #1

Estimated time: 120 – 240 minutes

Objectives

- Gain some familiarity setting up environments for cloud development using AWS
- Start to be familiar with at least one AWS SDK.
- Gain some familiarity with the AWS Command-Line Interface

Overview

In this assignment, you will set up two different development environments for writing programs that use AWS, along with the AWS Command-Line Interface (AWS CLI). The first development environment will be a Cloud 9 environment in your AWS Educate workspace. The second environment will be on your own machine and use a development stack of your choice. You will also set up the AWS CLI on your own machine. Finally, you will write two very simple programs, one in both environments, that list your S3 buckets.

Instructions

Step 0 – Preliminaries

Be sure you use the Learner Lab associated with "Cloud Developing" in your AWS Academy's Canvas account. If you use some other AWS account, we will not be able to grade your homework submission and you will receive a zero.

Start up your CS5260 Learner Lab environment and access AWS. Then, create a VPC, EC2 instances, two S3 buckets, and a DynamoDB table, like you did in CS5250. Call the VPC, cs5260-vpc, and its subnets, cs5260-public-1 and cs5260-public-2. Choose new names for your S3 buckets that following the pattern:

usu-cs5260-<a unique name or word>-<suffix>

Appendix A contains a summary of the steps that you'll needed to complete to re-create these resources in your CS5260 Learner Lab.

Step 1 – Setup a Cloud 9 Environment

Using what you learned from Lab 1 and the AWS documentation for Cloud 9, set up your own Cloud 9 interactive development environment (IDE). Call the environment cs5250-dev and give it description like "Development environment for CS5260."

Once you have it is setup, your Cloud 9 IDE will remain in your account even if you log out of the AWS Management Console, because it is hosted on a VM instance. You can see that instance if you go the EC2 Console and list your VMs. If this VM is idle for 30 minutes, if will automatically stop (but not terminate). Accessing Cloud 9 again will automatically start the VM.

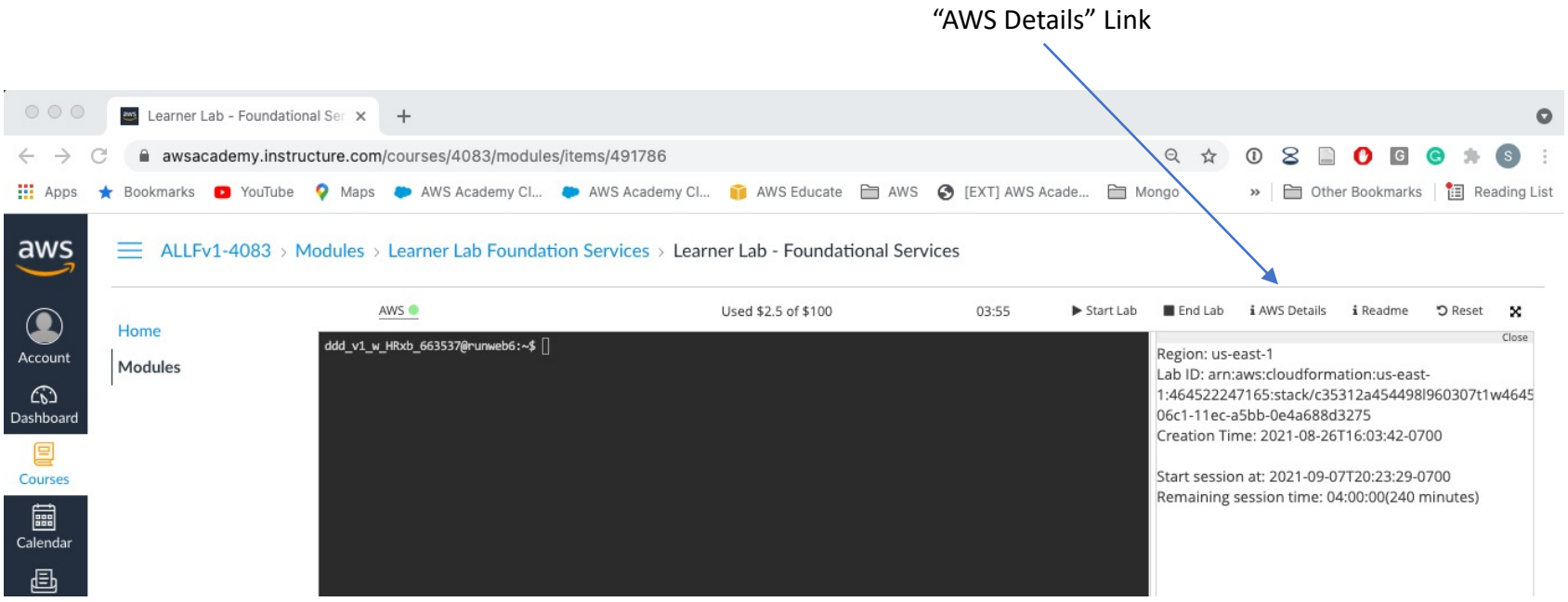
Once you have the Cloud 9 IDE set up, go to the *Project Settings* under *Preferences* and take a snapshot of your screen. You will submit this snapshot with your assignment.

Step 2 – Install the AWS Command-line Interface

Look up instructions for installing the AWS CLI on your own machine, based on your operating systems. Then, download the installation package and install it.

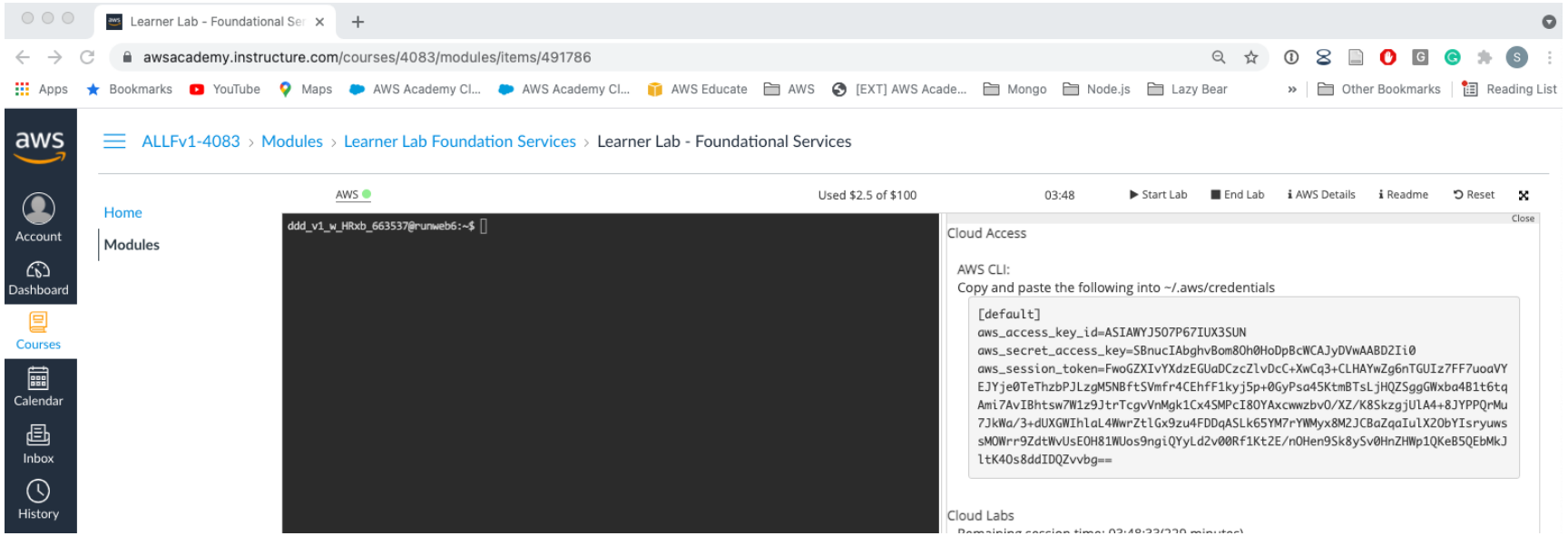
To configure the AWS CLI to access your AWS Account, you will need the credentials that can be found on the Learner Lab launch page. Click on the AWS Details link. See Figure 1.

Figure 1 – "AWS Details" link on the Learner Lab launch page



Then, on the "Show" button next to "AWS CLI" and copy the aws_access_key_id, aws_secret_access_key, and aws_session_token into your paste buffer. See Figure 2.

Figure 2 – Accessing the AWS CLI credentials.



The AWS CLI configuration wizard can create an initial credentials file. Unfortunately, the configuration wizard does not allow you to enter session token. Therefore, use a text editor to paste it into your credentials file. On Linux, this file typically is *~/.aws/credentials* and, on Windows, it is *%USERPROFILE%\.aws\credentials*.

Test your installation of AWS CLI by executing a command that list all your S3 buckets. It is your responsibility to figure out what the actually command. If you don't have at least three S3 buckets create some using the S3 Console.

Take a snapshot of your command-line window showing the execution of a command that lists your S3 buckets and the resulting output. You will submit this snapshot with your assignment.

Step 3 – Setup a local IDE for Cloud Developing

Pick your favorite IDE and development stack and then install the appropriate AWS SDK. You will need to search for the necessary download(s) and installation instructions.

Step 4 – Using your Local IDE, write a S3 Bucket Listing Program

Using your local IDE and chosen language, write a simple program that lists your S3 buckets. You will need to look for appropriate instruction for how to do this using your chosen language and the SDK that you installed in Step 3. Your program should use the credentials that you setup for the AWS CLI. DO NOT put any credentials in your code.

Run your program and take of snapshot of the screen that show its execution and output. You will submit this snapshot with the assignment.

Step 5 – Using Cloud 9, Write a S3 Bucket Listing Program

Using Cloud 9, write another program that lists your S3 buckets. Depending on the language you choose for Steps 3 and 4, you may be able to reuse the same code as before. Otherwise, write this program using one of the languages supported by Cloud 9.

Run your program and take of snapshot of the screen that show its execution and output. You will submit this snapshot with the assignment.

Step 6 – Complete a code review

To complete this assignment, you must do a code review with the instructor or TA. This does not have to before the due date. It can be done within 1 week after the due date.

You will sign up for code using *SignupGenius* and a link provided by the instructor. Just before you assigned time, get everything ready so you can walk through your programs in an efficient manner. For example, start your local IDE, your Learner Lab, and Cloud 9 prior to your schedule meeting time. For this assignment, the code review should take less than 10 minutes.

Submission

Your submission should consist of the screen snapshots from Steps 1, 2, 4, and 5.

Grading Criteria

Criteria	Points
Successfully completed Step 1, illustrated by a screen snapshot	20
Successfully completed Step 2, illustrated by a screen snapshot	20
Successfully completed Step 3 and 4, illustrated by a screen snapshot	20
Successfully completed Step 5, illustrated by a screen snapshot	20
Quality of S3 Bucket listing programs	20
Deductions (applied on top of scores given for the above): • No code review (-50 points) • Late (-25 points per day up to -100 points) • Sloppy presentation of the comparison and questions/answers (up to -25) • Cheating, e.g., copying someone else's work. This will be an automatic -200 (i.e., a negative twice the maximum points).	-200 to 0
Max Points	**100**

Appendix A

Step 1 – Create a VPC with two public subnets

VPC IPv4 CIDR block:	172.33.0.0/24
VPC IPv6 CIDR block:	No IPv6 CIDR Block
VPC Name:	cs5260-vpc
Subnet:	cs5250-public-1
IPv4 CIDR:	172.33.0.0/27
Availability zone:	us-east-1a
Subnet:	cs5250-public-2
IPv4 CIDR:	172.33.0.32/27
Availability zone:	us-east-1b

Step 2 – Setup routing

For each public subnet, make sure that traffic destine for

- 172.33.0.0/24 goes to “local”
- 0.0.0.0/0 goes to the gateway associated with the VPC.

Step 3 – Define Security Groups

Define the following security group:

Security group name:	ssh-sg
Description:	This group contains rules that will limit inbound traffic to port 22 (ssh).
VPC:	cs5260-vpc
Inbound Rules:	

Type: SSH (TCP, port 22)
Source: 0.0.0.0/0

Outbound Rules:

Keep the default outbound rules that allow all outbound traffic

Note: you will not need the widget-sg for the assignments in this class.

Step 4 – Launch EC2 Instances

Create the following EC2 instances:

Instance 1

Name:	Host A
AMI:	aws-elasticbeanstalk-amzn-2018.03.20.x86_64-java8-hvm-202011090147
Instance Type:	t2.micro
Network:	cs5260-vpc
Subnet:	cs5260-public-1
Auto-assign Public IP:	Enabled
IAM Role:	LabInstanceProfile
Storage:	8GB of general-purpose SSD
Security Group:	ssh-sg
SSH Key Pair:	vockey

Instance 2

Name:	Host B
AMI:	aws-elasticbeanstalk-amzn-2018.03.20.x86_64-java8-hvm-202011090147
Instance Type:	t2.micro
Network:	cs5260-vpc
Subnet:	cs5260-public-2
Auto-assign Public IP:	Enabled

IAM Role:	LabInstanceProfile
Storage:	8GB of general-purpose SSD
Security Group:	ssh-sg
SSH Key Pair:	vockey

Step 5 – Download the SSH key and test connectivity

Find the PEM or PPK file on the Learner Lab launch download it. Move it to a save spot on your local system, rename it to something that makes sense (like cs5260.pem), and then use it to make sure you can connect to both Host A and Host B.

Step 6 – Create S3 buckets

Create three S3 buckets hosted in the us-east-1 region with the following names, where <unique-word> is some unique word of your choice.

Bucket 1	usu-cs5260-<unique word>-dist
Bucket 2	usu-cs5260-<unique word>-requests
Bucket 3	usu-cs5260-<unique word>-web

Setup Buckets 1 and 2 so they block public access and Bucket 3 so it allows public access. Enable versioning for Bucket 3. Also, setup Bucket 3 as a static website, with the following bucket policy, replacing <bucket-3> with your Bucket 3's name.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": "arn:aws:s3:::<bucket-3>/*"
        }
    ]
}
```

Edit the access control list for Bucket 3, to grant “Everyone (public access)” the “List” and “Read” privileges. Again, this is not usually wise, but it is okay for this assignment. Then, Add the following JSON for the *Cross-origin Resource Sharing* (CORS).

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

Download the index.html file, modify it to contain your Bucket 3 S3 URL, and upload it to your Bucket 3. Also, download the Sample Object and upload it to your Bucket 3 with a key prefix of “widgets/”. Test the static website to make sure you can see the sample object.

Step 7 – Setup a DynamoDB Table

Create a DynamoDB table called “widgets”, with

- a primary index with a partition key of “id” (a string) and no sort key;
- “On-demand” read/write capacity; and
- a global secondary index with partition key of “owner” (a string) and sort key of “last_modified_on” (a string) and that contains the “label” and “description” attributes.