

## Lab 07 – Setting Up a Cloud Server

**Name:** Olivia Flores

**Course:** IS-3033-ON2

**Date:** 4/12/2024

### INTRODUCTION

In this lab, I will be taking you along with me to learn how to deploy a virtual machine in a cloud environment, specifically the Amazon Elastic Compute Cloud (EC2) through the Amazon Web Services (AWS) cloud provider. The objectives I will be covering include setting up an account first, securing my root account, configuring an IAM group and user, deploying an instance (virtual server) as a user, and lastly, SSH'ing into the instance (then terminate). The Amazon EC2 provides on-demand, scalable computing capacity in the AWS Cloud. This is very useful as it allows us to launch as many or few virtual servers as needed, configure security and networking, and manage storage [1]. This lab will also help transition into our next and final lab, which entails using a cloud instance to deploy a honeypot. So, let's begin getting familiar with the cloud environment.

### BREAKPOINT 1

#### Step 1: Set Up Your Cloud Account

In step 1 of this lab, I will be setting up my cloud account using the resources provided, but before I do so, I will also be signing up for an Amazon Educate account using my UTSA email. This will allow me to receive free training on the cloud as well as take advantage of other benefits it provides such as the connection to employment and being able to build my network through completing various hands-on labs to eventually be a part of the talent community. This Amazon Educate account will also provide me with access to the free tier we will be needing to complete this lab.

To begin, I will first create my AWS educate account using my UTSA email and filling out the required information. The Setup Environment guide provided by AWS, demonstrates the following steps below in more detail:

1. I went to the AWS Educate page and clicked on "Register Now."
2. Next, I input information as requested such as my name and school email address and clicked "Create Account" and received the email to verify the account immediately after.

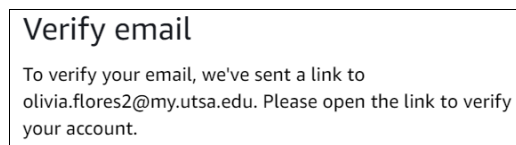


Figure 1: AWS Educate: "Verify email" message.

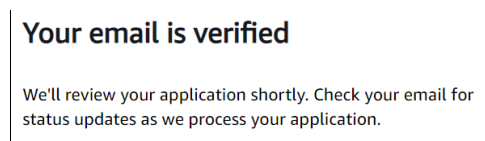


Figure 2: AWS Educate: "Your email is verified" message.

3. Next, I received a link in another email to set up a password for my Amazon Educate account.

Now, I moved on to sign up for free tier on the link provided in our lab instructions and I also followed the [Setup Environment Guide](#) provided by AWS.

1. First, I clicked on “Register Now,” and was directed to a page that requested the following information, for this AWS root user, I used my school email address again and clicked “Verify email address.”

## Sign up for AWS

### Root user email address

Used for account recovery and some administrative functions

olivia.flores2@my.utsa.edu

### AWS account name

Choose a name for your account. You can change this name in your account settings after you sign up.

Olivia Flores

**Verify email address**

Figure 3: Signing up for my AWS account.

2. Once verified through email with a code, I was able to set the Root user password and confirmed the root user password and selected “Continue.”
3. Next, I input my contact information and I plan to use AWS for school, so I selected the “Business” option, then clicked “Continue.”
4. Although I am using free tier, I was still required to put payment information on file since this will work as a free trial until expired.
5. Next, I confirmed my identity through text message (SMS) with a code provided, completed the security check, and clicked “Send SMS.”
6. After confirming my identity, I now want to sign up for the free support plan provided by AWS, as shown below and clicked “Complete signup.”

## Select a support plan

Choose a support plan for your business or personal account. [Compare plans and pricing examples](#). You can change your plan anytime in the AWS Management Console.

☒ **Basic support - Free**

- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor




Figure 4: Free Basic Support plan selected.

Now I have created both my AWS Educate account and my AWS account. I ran into the issue where I could not find the free tier link through my AWS Educate account, so I navigated to the link provided by the Setup Guide available on Module 1, “Create an AWS Account.” This then had me enter an email and I was aware of carefully considering which email to input, I continued with my school email. In addition to the school email account, I also created one for my personal email account, but I did not include that in this lab report as I am aware of payment arrangements and other account details.

## BREAKPOINT 2

### Step 2: Secure Your Root Account

Following along with the Module 2 instructions on the Setup Environment Guide, I want to learn best practices for securing my AWS account. To accomplish that, I will be logging into the Root user account, enabling additional security with Multi-Factor Authentication (MFA) and later I will dive into AWS Identity and Access Management (IAM) for additional users. This is important due to the level of permissions granted to the root user.

## Root user sign in ⓘ

**Email:** olivia.flores2@my.utsa.edu

**Password**

[Forgot password?](#)

.....

**Sign in**

Figure 5: Signing in to AWS as Root user.

Before entering my password as displayed in Figure 5 above, I was prompted to complete a small security check such as matching characters to the image generated by AWS. Once those tasks are complete, I was able to move forward with enabling MFA and creating an administrative user in the IAM Identity Center.

When MFA is enabled, in addition to providing the email address and password for the root user, I will be required to provide credentials from another authenticator, making it much more difficult for someone else to use my root user credentials without my permission [2].

So now that we understand why it is important to set up MFA, let's add more security to the root user sign-in. To do this, I will use the AWS Identity and Access Management (IAM) service and you can follow along with me in the steps provided below:

1. As I am still signed into my root user AWS account, I want to click on my account name on the right side of the navigation bar and choose "Security Credentials."
2. Once in the MFA section, I clicked "Assign MFA" as shown in the figure below.

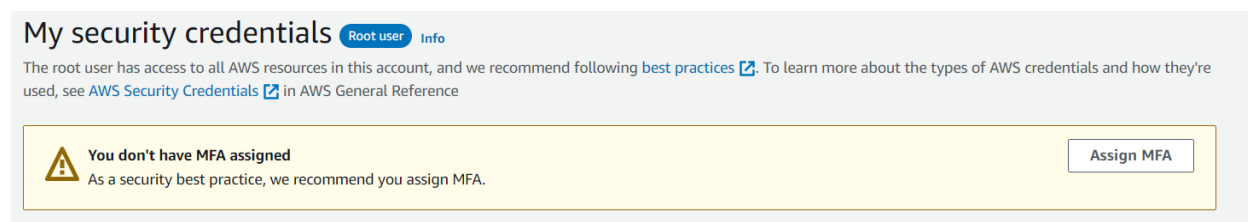


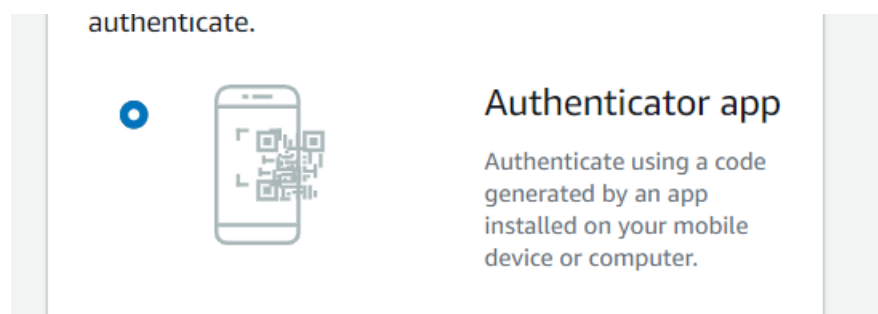
Figure 6: Assign MFA under Security Credentials section.

3. Next, the Register MFA device page opens up, I input a meaningful Device name and chose the Authenticator app option, see my example below then select "Next":

### Device name

Enter a meaningful name to identify this device.

Maximum 128 characters. Use alphanumeric and '+ = , . @ - \_ ' characters.



Now, the Set up device page opens. There are several different authenticator apps that are supported for both Android and iOS devices, I will be using DUO Mobile as I already have been using this MFA for my school account login and works for my iOS device. Next, follow along with me in the steps below:

1. I selected the “Show QR code” link to then be able to scan a unique QR code for my account through the authenticator app on my mobile device.
2. I scanned the QR code with my DUO mobile authenticator app, but if this did not work, alternatively, I could have entered a code into the authenticator app to link my authenticator device to the account.
3. After DUO Mobile authenticator established the link to my account, it generated secret codes that are only good for a limited number of seconds. In MFA code 1, I typed the code in the app, and then waited for the MFA code 2 and typed in that code into the app as well. Before the second code expired, I made sure to select “Assign MFA.”

After those three steps were complete, I was returned to the Security credentials page. A notification should be displayed at the top that states the “MFA device is assigned.” This mean that my root user credentials are now more secure than before.

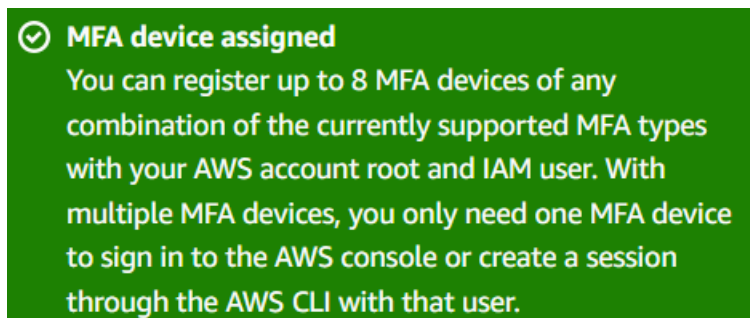


Figure 7: MFA device is assigned notification displayed.

If you want to make sure this is working, you could sign out of the console and sign back in to see that you will have to provide the credentials from your MFA device as well as your email address and password.

As another best practice, to make sure I am not overextending my monthly free compute time (hours), I want to quickly set up free alerts in the Billing and Cost Management console. There is a [link](#) to this console in the Setup Environment guide provided by AWS as well [2].

When I click on the link, I am on the main page and see a wide variety of tools that can be used. Here, I will simply be changing my alert preferences, so I scrolled down to the “Preferences and Settings” section and clicked on “Billing Preferences” and under this tab I can see a section called “Alert Preferences.” I checked both alerts as displayed in the figure below and clicked “Update” to make the changes.

**Alert preferences** [Info](#)

☒ **Receive AWS Free Tier alerts**

If this option is selected, your AWS Free Tier usage alerts will be sent to the email address that you used to create your account. To send these alerts to a different email address, specify it below.

Email address to receive Free Tier usage alerts - *optional*

☒ **Receive CloudWatch billing alerts**

Once enabled, this preference cannot be disabled.

**Update** **Cancel**

**CloudWatch billing alerts**

You can activate CloudWatch billing alerts to receive email notifications when your charges reach a specified threshold. You can monitor your AWS usage charges and recurring fees automatically to simplify tracking and managing of your AWS spends. **This preference can't be deactivated once activated.** To manage your CloudWatch billing alerts, see your CloudWatch dashboard, or view your AWS Budgets.

Figure 8: Setting up free alerts in Billing and Cost Management console.

Now my root account is more secure by following these best practices, setting up MFA and alert preferences. I did not encounter any challenges in this step of the lab.

## BREAKPOINT 3

### Step 3: Configure and Deploy an IAM Group and User Account.


In step 3 of this lab, it is important to consider a security best practice to not use our root account for everyday tasks and as we are using our local virtual machines, it is best not to operate our server as root. This leads me to my next point I want to go over- configuring a group and a user in that group, and then creating a policy to allow the deployment of EC2 instances. Under the IAM Identity Center, I am going to create a group [1].

We are using IAM Identity Center because it provides users like me with unique credentials for every session, also known as temporary credentials. Providing users these credentials results in enhanced security for their AWS account, because they are generated each time the user signs in. Once I have an administrative user, I will be able to sign in with that user to create additional Identity Center users and assign them to groups with permissions to perform specific job functions [2].

To begin this process, I signed into the AWS Management Console as the account owner, entered the password and completed the MFA as prompted. Next, I navigated to the search bar and entered "IAM Identity Center." Once I was in the service overview page, I was able to review the information to learn about the features of this service, then under Enable IAM Identity Center, I chose Enable as displayed in the figure below.

## Enable IAM Identity Center

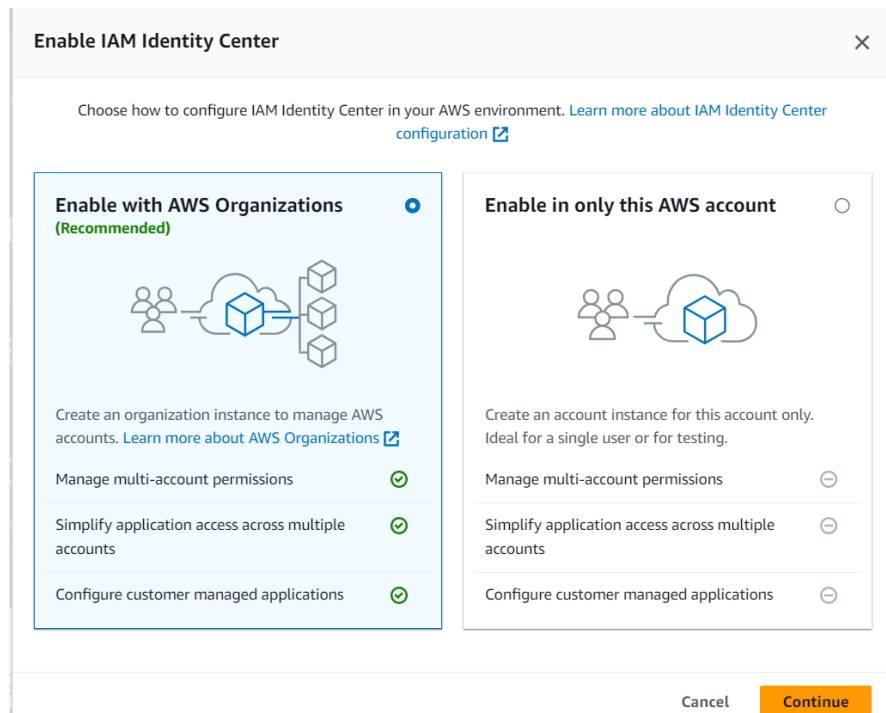
IAM Identity Center makes it easy to connect an existing directory or use the built-in Identity Center directory to manage user access to AWS accounts and cloud applications.

AWS recommends reviewing the [IAM Identity Center prerequisites](#) 


Enable

Figure 9: Enabling IAM Identity Center.


Since I enabled this service, I also enabled AWS Organizations because as I mentioned earlier, we are going to be creating a group. AWS Organizations lets you organize multiple AWS accounts so that you can have separate AWS accounts for different use cases. AWS Organizations is a feature of the AWS account offered at no additional charge [2]. You can see from the figure below this is the *recommended* way to configure IAM Identity Center. Next, I clicked “Continue” and I received confirmation that I successfully created the organization instance.





**Enable IAM Identity Center** ×


Choose how to configure IAM Identity Center in your AWS environment. [Learn more about IAM Identity Center configuration](#) 


**Enable with AWS Organizations** ☒   
(Recommended)




Create an organization instance to manage AWS accounts. [Learn more about AWS Organizations](#) 

Manage multi-account permissions 


Simplify application access across multiple accounts 


Configure customer managed applications 


**Enable in only this AWS account** ☐



Create an account instance for this account only. Ideal for a single user or for testing.

Manage multi-account permissions 

Simplify application access across multiple accounts 

Configure customer managed applications 

Cancel **Continue**

Figure 10: Creating an AWS Organization instance.

✔ You have successfully created the organization instance of IAM Identity Center 66840386546c79c6.

Figure 11: Successful creation of organization instance.

My root user account is now the management account for this AWS Organization mentioned above.

Next, I will configure my identity source which is where my users and groups are managed. After this is configured, I will be able to look up users or groups to grant them single sign-on access to AWS accounts, cloud applications, or both.

When IAM Identity Center is enabled for the first time, it is automatically configured with an IAM Identity Center directory as the default identity source. As displayed in the figures below, I confirmed the identity source and then enabled the attributes for access control.

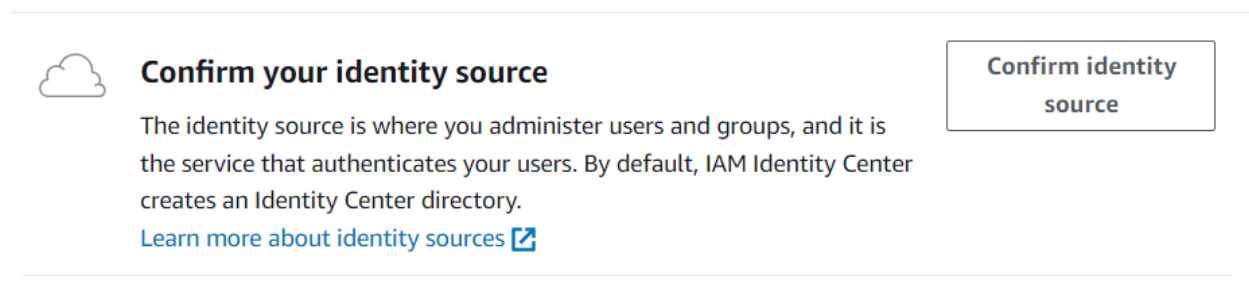


Figure 12: Confirming identity source.

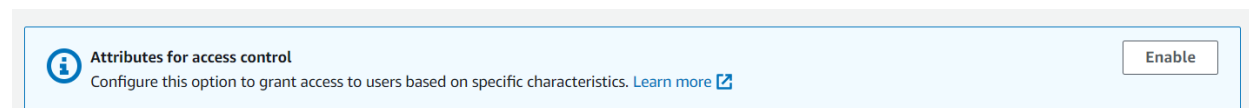


Figure 13: Granting access to users as specified.

Now for the fun of creating a user in IAM Identity Center, I completed this with the following steps:

1. I navigated to the IAM Identity console and chose "Users." Then, selected "Add user."
2. Next, I input the requested information, such as the Username, choice for how the user will receive their password, their email address, and the full name, then selected "Next." I did not complete the optional information following the figure shown below.



### Username

This username will be required for this user to sign in to the AWS access portal. The username can't be changed later.

Maximum length of 128 characters. Can only contain alphanumeric characters or any of the following: +=,.,@- \_

### Password

Choose how you want this user to receive their password. [Learn more](#) 

- ☒ Send an email to this user with password setup instructions.
- ☐ Generate a one-time password that you can share with this user.

### Email address

### Confirm email address

### First name

### Last name

### Display name

This is typically the full name of the workforce user (first and last name), is searchable, and appears in the users list.

Figure 14: Adding a user in the IAM Identity Console.

A nice tip to note is that during testing, it is possible to use email subaddressing to create valid email addresses for multiple fictitious users. If your email provider supports it, you can create a new email address by appending the plus sign (+) and then numbers or characters to your current email address, such as someone@example.com, someone+1@example.com, and someone+test@example.com. All of those email addresses would result in an email being received at the same email address [2].

3. Now I want to add my user to a group, so I selected "Create group."
4. A new browser tab opened to display the Create group page, and under group details, I entered Admins, then clicked "Create group." You can see in the figure below the group has been created shortly after.

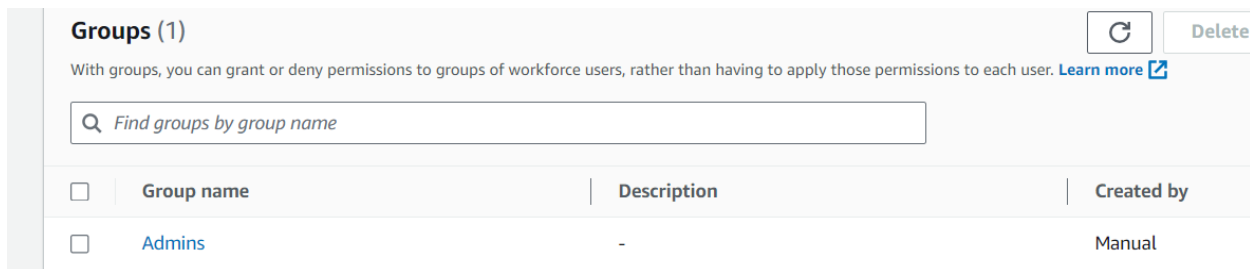


Figure 15: Group created called "Admins."

5. I returned to the Add user browser tab and refreshed the Add users page and can see the new Admins group is now displayed.
6. I selected the check box next to the Admins group, and then clicked "Next."
7. On the Review and add user page I confirmed the following:
  - Primary information appears as I intended.
  - Groups shows the user added to the group you created.

Since everything looked correct, I clicked "Add User." I have now successfully added a user in my AWS Organization. If I want to add more users and groups, I will just repeat the steps shown above.

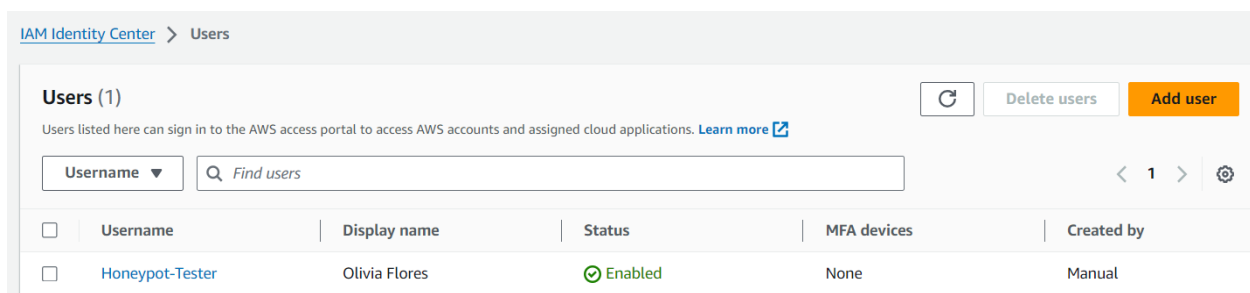


Figure 16: Successfully created a user of a group.

8. I wanted to confirm that I received the email invitation with the AWS access portal URL for the user displayed in the figures above. Then set up MFA for this user as well shortly after.

Accept invitation

This invitation will expire in 7 days.

### Accessing the AWS access portal

After you've accepted the invitation, you can sign in to the AWS access portal by using the information below.

Your AWS access portal URL:

[REDACTED]

Your Username:

Honeypot-Tester

Figure 17: Received the user portal link.

✔ Your authenticator app has been successfully registered. You can now use it when prompted for additional verification at sign in.

Honeypot-Tester's MFA 1 [Rename](#)

Figure 18: MFA successfully set up for Honeypot-Tester.

So now we have successfully added Honeypot-Tester to the group *Admins*, but if I were to access this portal displayed above, I would see a blank dashboard because I have not created any permissions to allow services and/or applications to my user yet. So, let's give my new user access to my AWS account and then allow the ability to create and deploy Amazon EC2 instances. Since we put the user into a group, we will assign the group to an account and then we will add a permission set that defines what the members of the group can access.

I will still be using the root user credentials for this procedure and following the steps shown below:

1. In IAM Identity Center console left hand navigation, I clicked on the Dashboard.
2. Under *Manage permissions for multiple AWS accounts*, I clicked "Manage permissions."
3. In the left-hand navigation, I clicked on "**AWS accounts**."
4. Under the *Organizational structure*, I check marked the account (Olivia Flores) I created initially under the Root account and then clicked "Assign users or groups" on the upper right-hand side.
5. The *Assign users and groups* workflow is displayed next and on the left-hand side, there will be 3 steps visible needed to be accomplished as demonstrated below.

For Step 1: **Select users and groups**, I selected the *Admins* group that I created previously, then clicked "Next."

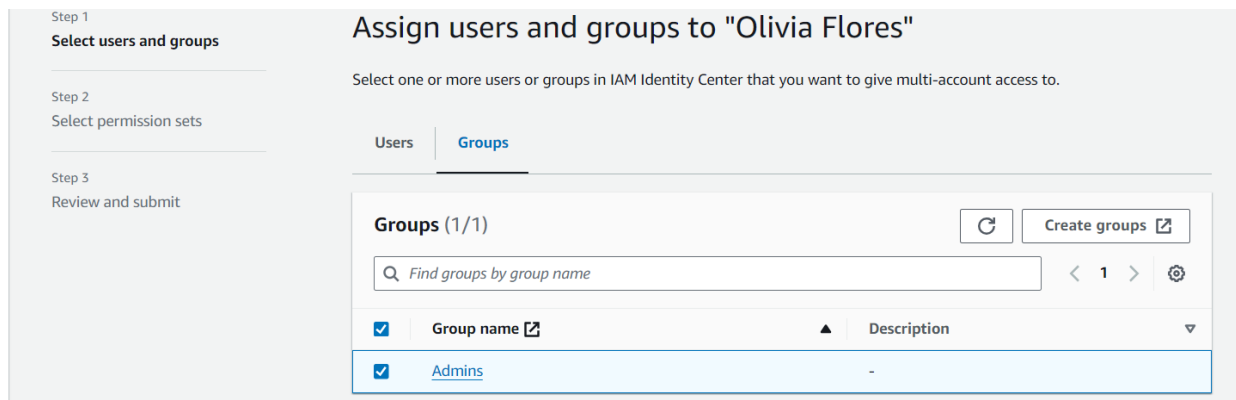


Figure 19: Assigning users and groups to Olivia Flores.

For Step 2: **Select permission sets**, I selected *Create permission set* on the upper right-hand side.

This opened a new browser tab and walked me through the three sub-steps (like the ones shown above for users and groups) involved in creating the permission set.

For **sub step 1- Select permission set type**: I selected “*Custom Permission Set*.” For a custom permission set, I will then have to select an AWS managed or customer-managed policies that I want to include, I clicked on the AWS managed policy drop down arrow and searched for “AmazonEC2FullAccess,” then clicked Next.

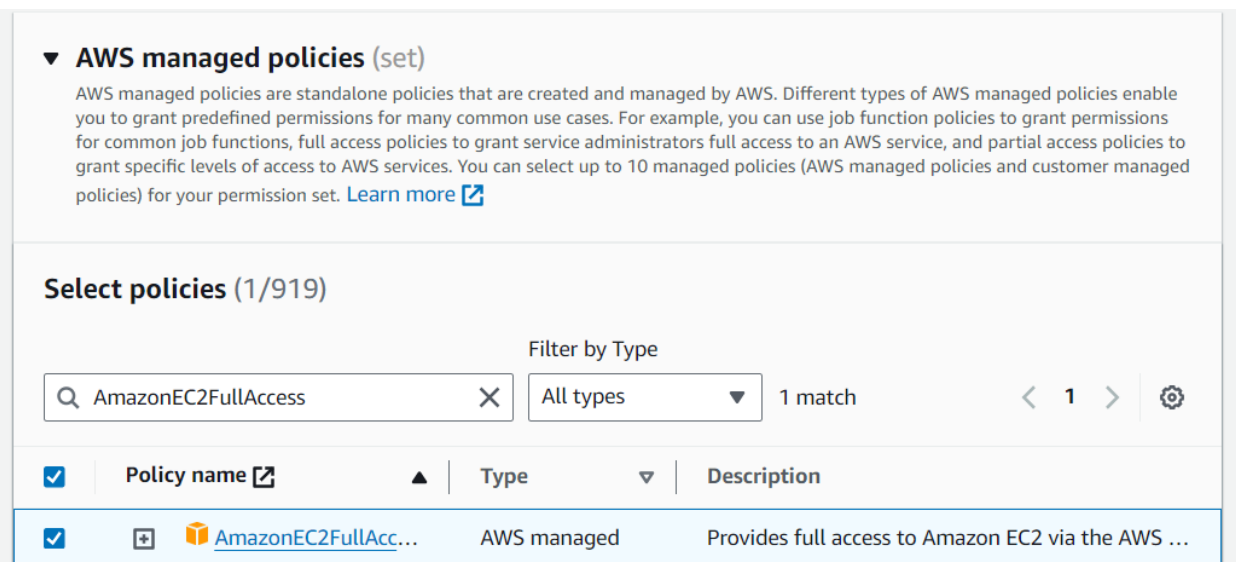


Figure 20: Adding a custom permission set.

In the setup pane, I also chose a maximum time of four hours in the Session duration to allow a honeypot to run daily for a reasonable amount of time.

## Permission set details

### Permission set name

The name that you specify for this permission set appears in the AWS access portal as an available role. After users in IAM Identity Center sign in to the AWS access portal and select an AWS account, they can choose the role.

EC2-Instance

Permission set names are limited to 32 characters or less. Names may only contain alphanumeric characters and the following special characters: + = , . @ - \_

### Description - optional

Add a short explanation for this permission set.

This EC2 instance is a honeypot.

Permission set descriptions are limited to 700 characters or less. Descriptions should match the regular expression: `[\u0009\u000A\u000D\u0020-\u007E\u00A1-\u00FF]*`

### Session duration

The length of time a user can be logged on before the console logs them out of their session. [Learn more](#)

4 hours ▼

### Relay state - optional

The value used in the federation process for redirecting users within the account. [Learn more](#)

Enter relay state

Relay states support up to 320 characters. Relay states may only contain alphanumeric characters.

Figure 21: Specifying permission set details.

Now, I will need to assign the permission set to my group as well (Olivia Flores). When I assign a permission set to a user or group, IAM Identity Center creates corresponding IAM roles in the target AWS account(s) and attaches the specified policies to those roles [1].

Next, I reviewed the parameters set and then created the permission set by clicking “Submit.” This allows the authorized users to assume the roles and access the resources in the target account(s).

## Review and submit assignments to "Olivia Flores"

### Step 1: Select users and groups

[Edit](#)

Users and groups (1)		< 1 >	
Username / group name <a href="#">🔗</a>	▲	Type	▼
<a href="#">Admins</a>		Group	

### Step 2: Select permission sets

[Edit](#)

Permission sets (1)			
Permission set	▲	Description ▼	ARN ▼
<a href="#">EC2-Instance</a>		This EC2 instance is a honeypot.	arn:aws:sso::;permission Set/ssoins-66840386546c79c6/ps-578dcf139e dad7e0

[Cancel](#)[Previous](#)[Submit](#)

Figure 22: Confirming assignment to "Olivia Flores" by clicking "Submit."

Now let's walk through how to deploy a virtual machine (EC2 server t2.micro instance)! I was able to find the EC2 dashboard under **Services** in the IAM Identity Center console.

Once in the EC2 dashboard, I clicked on the "Launch instance" button to open the instances pane. I named my instance "Lab 7 server" and selected the **Ubuntu Server 22.4 LTS image** as directed in our lab instructions.

Name and tags [Info](#)

Name

Lab 7 server

Add additional tags

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUS

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible ▼

ami-0b8b44ec9a8f90422 (64-bit (x86)) / ami-0000456e99b2b6a9d (64-bit (Arm))

Virtualization: hvm    ENA enabled: true    Root device type: ebs

Figure 23: Setting up the instance parameters.

In this step of the lab, I was able to get familiar with configuring a group and a user in that group, then being able to create a policy to allow the deployment of EC2 instances. I am quite satisfied with the resources available on AWS throughout the lab thus far and have not encountered any difficulties yet. In the next step of this lab, I will be finishing the demonstration of the EC2 instance deployment.

## BREAKPOINT 4

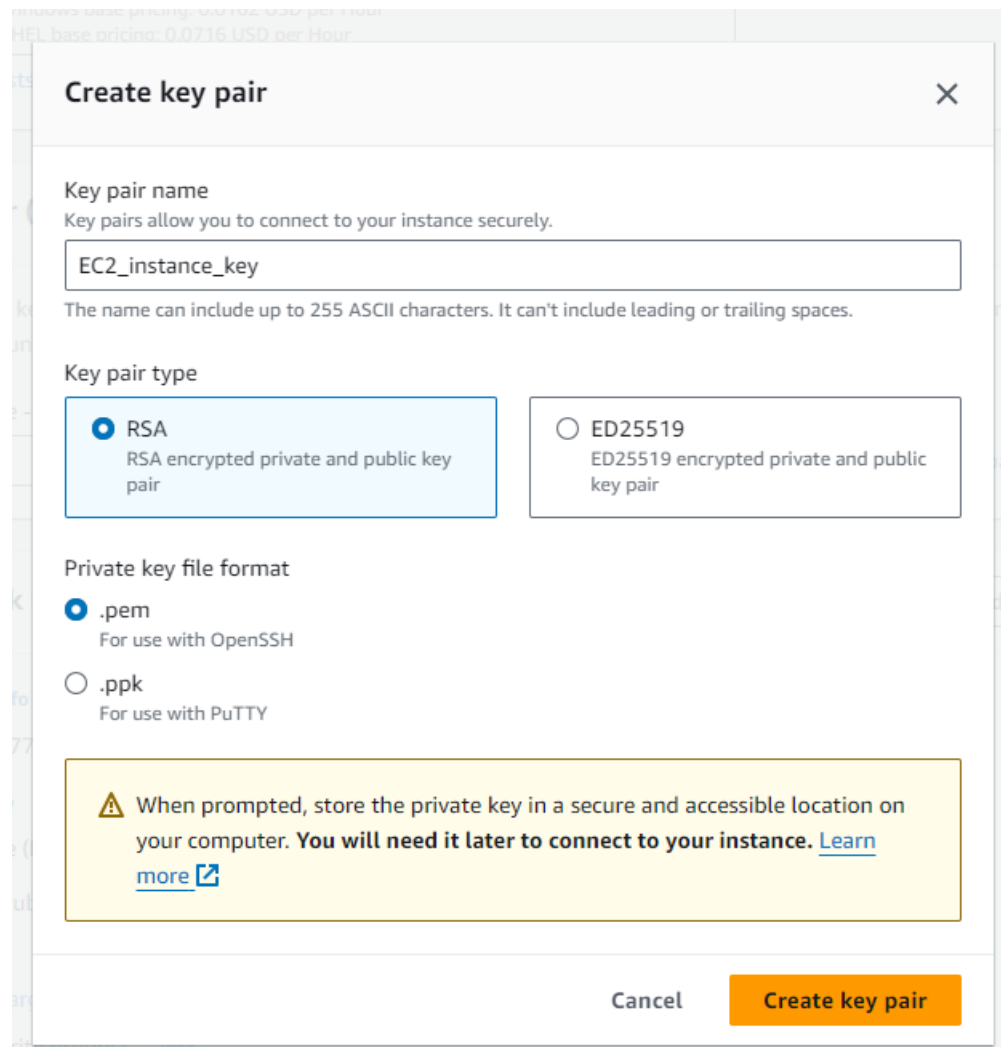
### Step 4: Set Up a Key Pair

In this step of the lab, I will be demonstrating how to set up a key pair as this is very important. I will be using **RSA** as my key generator (rather than ed25519- not supported for Windows instances). I set up a key pair so that the login into my cloud virtual machine is secure.

15

To do this I clicked “Create a new key pair” under Key pair (login). Then I named my key **EC2\_instance\_key**, choosing the .pem format and saved/downloaded it to a folder in my desktop by clicking “Create key pair.” I will need this information to connect via SSH.

The private key file is automatically downloaded by my browser. The base file name is the name specified as the name of my key pair, and the file name extension is determined by the file format I chose.



The screenshot shows the 'Create key pair' dialog box. At the top, it says 'Create key pair' with a close button (X). Below this, there's a section for 'Key pair name' with a text input field containing 'EC2\_instance\_key'. A note below the input says 'Key pairs allow you to connect to your instance securely.' and 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.' Below this is the 'Key pair type' section with two options: 'RSA' (selected with a radio button) and 'ED25519'. The RSA option is described as 'RSA encrypted private and public key pair'. The ED25519 option is described as 'ED25519 encrypted private and public key pair'. Below this is the 'Private key file format' section with two options: '.pem' (selected with a radio button) and '.ppk'. The .pem option is described as 'For use with OpenSSH' and the .ppk option as 'For use with PuTTY'. At the bottom of the dialog, there's a yellow warning box with a triangle icon and text: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)'. At the very bottom, there are two buttons: 'Cancel' and 'Create key pair'.

Figure 24: Creating a key pair.

Under **Instance type**, from the **Instance type** list, I was able to choose which hardware configuration I wanted to use for my instance. I chose the t2.micro instance type, which is also selected by default. The **t2.micro** instance type is eligible for the Free Tier. In Regions where t2.micro is unavailable, you can use a t3.micro instance under the Free Tier.

Since we are setting up a honeypot, I will be allowing SSH using TCP from any address. Typically, you would establish a security group, but in this lab we are not due to the honeypot.



We'll create a new security group called 'launch-wizard-1' with the following rules:

- ☒ Allow SSH traffic from Anywhere  
0.0.0.0/0  
Helps you connect to your instance
- ☐ Allow HTTPS traffic from the internet  
To set up an endpoint, for example when creating a web server
- ☐ Allow HTTP traffic from the internet  
To set up an endpoint, for example when creating a web server

**⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.**

Figure 25: Allow SSH from anywhere.

Next, I want to change the storage amount to 25 GB, which is also allowed under the free tier.

**▼ Configure storage** [Info](#) [Advanced](#)

1x  GiB  Root volume (Not encrypted)

**ℹ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage**

Figure 26: Configuring the storage.

Now for the fun, I launched my instance and checked the settings by navigating to the instances tab and check marking the box for the Lab 7 server instance I created. I will need to note my public IPv4 address to SSH into my server and you will see this in the figure below.

Instance: i-0794795a9d0470a92 (Lab 7 server)			
Details	Status and alarms <a href="#">New</a>	Monitoring	Security
<b>▼ Instance summary</b> <a href="#">Info</a>			
Instance ID i-0794795a9d0470a92 (Lab 7 server)	Public IPv4 address 34.207.123.104 <a href="#">open address</a>	Private IPv4 addresses 172.31.90.206	
IPv6 address -	Instance state <span>Running</span>	Public IPv4 DNS ec2-34-207-123-104.compute-1.amazonaws.com <a href="#">open address</a>	
Hostname type IP name: ip-172-31-90-206.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-90-206.ec2.internal	Elastic IP addresses -	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	AWS Compute Optimizer finding <a href="#">Opt-in to AWS Compute Optimizer for recommendations.   Learn more</a>	
Auto-assigned IP address 34.207.123.104 [Public IP]	VPC ID vpc-0c281f2252b98ba05 <a href="#">open address</a>	Auto Scaling Group name -	
IAM Role -	Subnet ID subnet-0981bd5e5c82d39fa <a href="#">open address</a>		

Figure 27: Noting instance public IPv4 address.

It is important to remember that this instance should be stopped or terminated when it is not being actively used. Stopping it will halt all compute costs and will retain files and storage so that we can restart later with the same data. Terminating it will completely halt the machine, just as if we were launching a new machine the next time.

## BREAKPOINT 5

### Step 5: SSH into Your Cloud Server

Now I can test access to my server (EC2 instance) from my host machine (acting as a client). Instructions on how I accessed this instance from my Windows machine were found here:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connect-linux-inst-from-windows.html> [3].

Other options are in the same section, but in this lab I will go over how I accomplished those instructions as well.

### Verifying PowerShell prerequisites

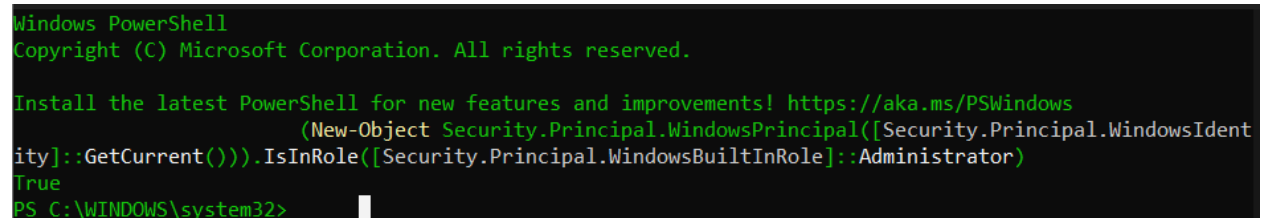
To install OpenSSH on my Windows OS using PowerShell, I wanted to make sure I was running PowerShell version 5.1 or later, and that my account is a member of the built-in Administrators group. First, I ran the command `$PSVersionTable.PSVersion` from PowerShell to check my PowerShell version. My PowerShell version shows a Major version of 5 and a Minor version of 1, so I was able to proceed with that.

To check whether I was a member of the built-in Administrators group, I ran the following PowerShell command:

#### (New-Object

`Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent()))`.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

Before the command below was ran, I made sure to open Powershell as Administrator by right clicking and "Running as Administrator." Since I am a member of the built-in Administrators group, the output below shown is **True**.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(New-Object Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
True
PS C:\WINDOWS\system32>
```

Figure 28: Checking whether I am a member of the built-in Administrators group on my local machine.

Next, to install OpenSSH for Windows using PowerShell, I ran the following PowerShell command:

**Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0**

The figure below demonstrates the expected output as shown in the instructions from AWS:

```
PS C:\WINDOWS\system32> Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0

Path      :
Online    : True
RestartNeeded : False
```

Figure 29: Installing OpenSSH for Windows in PowerShell.

After I installed OpenSSH, I use the following procedure to connect to my Ubuntu instance from Windows using OpenSSH.

### Connecting my instance using OpenSSH

1. In PowerShell or the Command Prompt, I used the **ssh** command to connect to the instance. This is done by specifying the path and file name of the public key (.pem), the username for my instance, and the public DNS name for my instance [4].

Below, my .pem file is stored under Desktop, so I simply navigated to that directory and ran the command (using my public IPv4 address).

```
ssh -i EC2_instance_key.pem ubuntu@34.207.123.104
```

The default username for my EC2 instance is determined by the AMI that was specified when I launched the instance [4].

The default username for an **Ubuntu AMI** is **ubuntu** as shown in the figure below.

I received the following response:

```
PS C:\Users\david\Desktop> ssh -i /EC2_instance_key.pem ubuntu@ec2-34-207-123-104.compute-1.amazonaws.com
Warning: Identity file /EC2_instance_key.pem not accessible: No such file or directory.
The authenticity of host 'ec2-34-207-123-104.compute-1.amazonaws.com (34.207.123.104)' can't be established.
ED25519 key fingerprint is SHA256:aOPQrvYft6UC4f9lt1me89Q7dZpDdzd+c45qkn2HRSA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Figure 30: Initializing the SSH into the instance.

Next, I entered “yes” for the fingerprint and received the following warning which I was not worried about as this should happen:

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-207-123-104.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

Figure 31: Response to establishing the fingerprint.

Initially, I did cause an error because if you can see from figure 30, I was inputting the wrong path to the instance key by simply adding a forward slash to the command. Once I fixed this, I was able to fix the error and get the output as shown in the figure below.

```
PS C:\Users\david\Desktop> ssh -i EC2_instance_key.pem ubuntu@34.207.123.104
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Apr 17 00:00:32 UTC 2024

System load:  0.0               Processes:            97
Usage of /:   6.4% of 24.05GB   Users logged in:     0
Memory usage: 20%              IPv4 address for eth0: 172.31.90.206
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-90-206:~$
```

Figure 32: Successful SSH into the Ubuntu server.

After the SSH into my Ubuntu server was successful, I ran the standard commands to make sure it was working as intended, such as `ls`, **whoami**, **date**, **echo**, **touch**, and lastly, **exit**. Here you will also notice that my private IP address is displayed in the figure below, known as **172.31.90.206**.

```


ubuntu@ip-172-31-90-206:~$ ls
ubuntu@ip-172-31-90-206:~$ whoami
ubuntu
ubuntu@ip-172-31-90-206:~$ date
Wed Apr 17 00:03:51 UTC 2024
ubuntu@ip-172-31-90-206:~$ echo "It worked!"
It worked!
ubuntu@ip-172-31-90-206:~$ touch "This is not a permanent machine."
ubuntu@ip-172-31-90-206:~$ ls
'This is not a permanent machine.'
ubuntu@ip-172-31-90-206:~$ exit
logout
Connection to 34.207.123.104 closed.
PS C:\Users\david\Desktop>

```

In the next lab, we will resume this learning but for now I want to terminate my instance. To do this I clicked on the Actions drop down arrow on the upper right hand side, and clicked on Manage instance state, then clicked Terminate and lastly, “Change state” to officially terminate.

### Instance state settings

☒ Start  
Available when the instance is stopped
   
☐ Stop
   
☐ Hibernate  
This instance did not have Stop - Hibernate enabled at launch
   
☐ Reboot
   
☒ Terminate


**Note that when your instances are terminated:**
  
On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.

Cancel
Change state

Figure 33: Terminating instance.

## CONCLUSION

In this lab, I successfully learned how to deploy the EC2 virtual machine in a cloud environment through the Amazon Web Services (AWS) cloud provider. I was satisfied with the resources provided, including the Free tier that AWS offers as it gave me a lot of insight on new tools I can use in the future. I was able to smoothly walk through the instructions provided and ultimately use best practices to SSH into the instance we created. This lab is also going to help transition into our next and final lab, which entails using a cloud instance to deploy a honeypot.

## REFERENCES

- [1] Mitra, Department of Information Systems and Cyber Security. [Online] "Lab-07 Setting Up a Cloud Server" by Rita Mitra (2024). UTSA, 2024 [Accessed: April 12, 2024.]
- [2] "How to Setup Your Development Environment for AWS | Module 2." *Amazon Web Services, Inc.*, <https://aws.amazon.com/getting-started/guides/setup-environment/module-two/>. [Accessed: April 12, 2024.]
- [3] *Connect to Your Linux Instance from Windows* - Amazon Elastic Compute Cloud. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connect-linux-inst-from-windows.html>. [Accessed: April 16, 2024.]
- [4] *Connect to Your Linux Instance from Windows with OpenSSH* - Amazon Elastic Compute Cloud. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/openssh.html#openssh-prereqs>. [Accessed: April 16, 2024.]

## COLLABORATION

In this lab, I made complete use of the lab instructions provided to us as well as the instructions provided through AWS- the Setup Environment Guide and the instructions to SSH into the instance from a Windows Operating System. This helped in clarifying which commands to use when using Windows PowerShell and confirming some outputs I received were correct.