# LTRSEC-2000

# ISE Deployments in the Cloud - Automate ISE Deployments in AWS and Integrate Them with Entra ID

Jesse Dubois, TAC Security Technical Leader

Patrick Lloyd, Senior Solutions Architect, CX Security Services

## Learning Objectives or Table of Contents

Upon completion of this lab, you will be able to:

- Provision an Ubuntu Linux machine with Terraform and Ansible to automate configurations.
- Automatically provision AWS VPC's, Subnets, Routes, and Transit Gateway Attachment via Terraform.
- Provision ISE using a CloudFormation template.
- Provision ISE programmatically into the AWS cloud with your own credentials.
- Configure network device groups, network access devices, users and roles via Ansible configuration scripts.
- Integrate a ROPC external database for authentication to Microsoft Entra ID.

## Scenario

In this lab activity, you will learn how to utilize an Ubuntu Linux machine to deploy ISE into an AWS environment. Two separate orchestration models are used, Terraform, which is more suited for deploying workloads and maintaining their state, and Ansible, which is better suited to configure and modify workloads via API. Both Terraform and Ansible scripts, found on github.com for later consumption, will be used to provision the lab, with Terraform used to deploy VPC, subnet, routing table, and internet gateway into an AWS tenant already created. Ansible will then deploy ISE configurations, including users, groups, and network access devices. You will then provision the Entra ID connector, which requires manual intervention to join to Entra ID.

## Document Conventions

As part of this document the following highlighting will be used

Note: A time saving note which will indicate aspects of the lab to look out for, or could otherwise cause need for rework.

Warning: Common mistake areas students have made in the past.

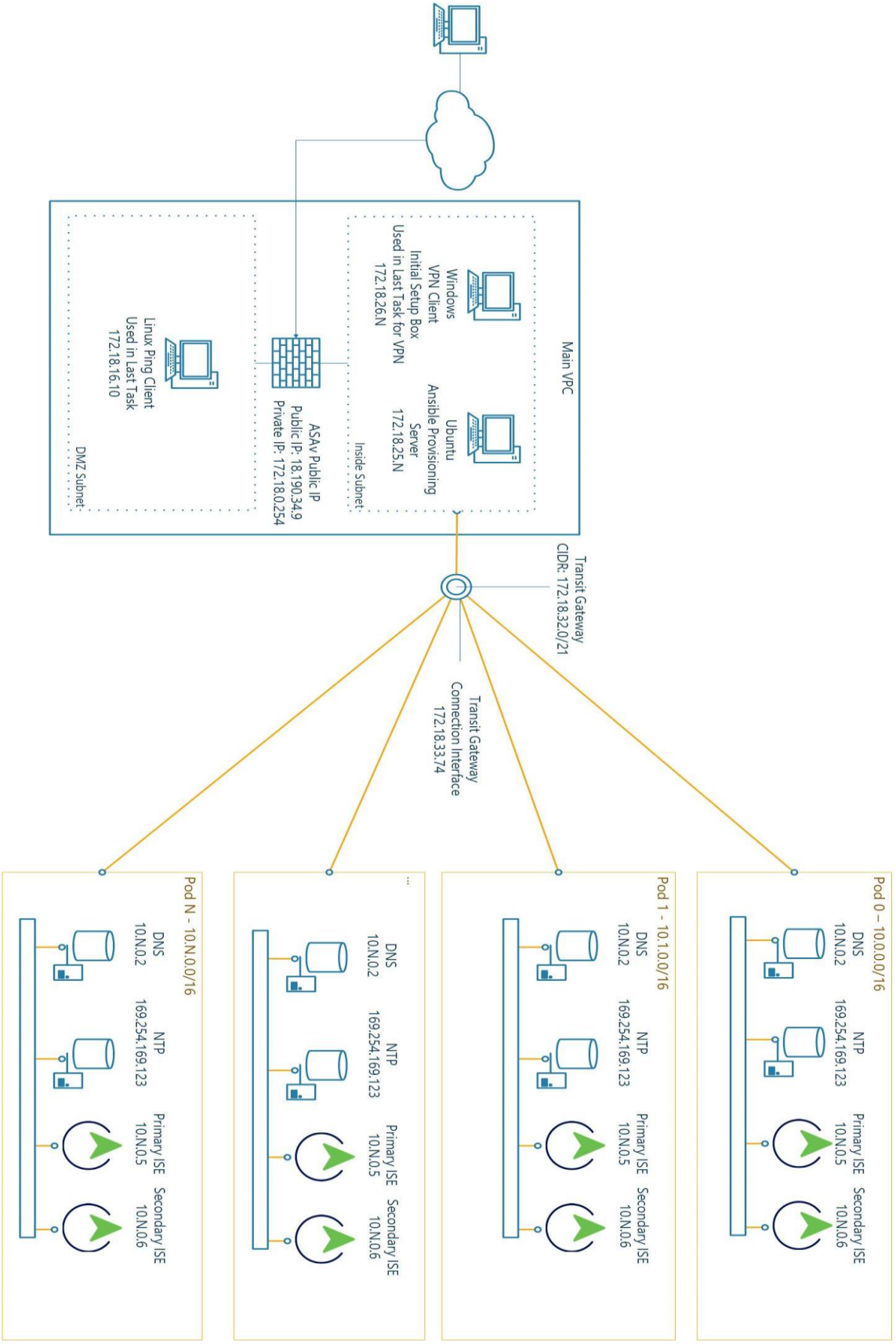
```
A configuration block.
```

Throughout the lab, there are various places where passwords must be changed. Your proctors should provide you an initial password as well as an updated password before starting.

## Recommended Equipment

1. A laptop with VMWare workstation installed.
2. A virtual image you are familiar with.
3. Cisco Secure Client on the virtual image.  If you do not have Cisco Secure Client, reach out to your lab proctor.
4. A cell phone with Google Authenticator Installed or your MFA app of choice.
5. Internet Connectivity.

# Network Diagram



## Task 1: Initial Connectivity

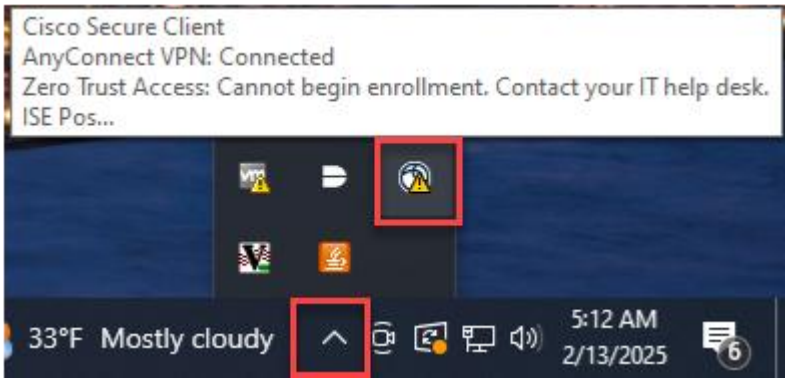
Each pod is accessible via VPN with a provisioning machine on the required subnet.  From this provisioning machine, you'll execute tasks allowing you to deploy AWS and ISE components and configurations. Access your pod based on the pod number assigned to you by the instructors:

Table 1-1

| Pod Number | Ubuntu Host | AWS Username | RAVPN Username |
|---|---|---|---|
| 1 | 172.18.25.1 | pod1-awsadmin | pod1-ravpn |
| 2 | 172.18.25.2 | pod2-awsadmin | pod2-ravpn |
| 3 | 172.18.25.3 | pod3-awsadmin | pod3-ravpn |
| 4 | 172.18.25.4 | pod4-awsadmin | pod4-ravpn |
| 5 | 172.18.25.5 | pod5-awsadmin | pod5-ravpn |
| 6 | 172.18.25.6 | pod6-awsadmin | pod6-ravpn |
| 7 | 172.18.25.7 | pod7-awsadmin | pod7-ravpn |
| 8 | 172.18.25.8 | pod8-awsadmin | pod8-ravpn |
| 9 | 172.18.25.9 | pod9-awsadmin | pod9-ravpn |
| 10 | 172.18.25.10 | pod10-awsadmin | pod10-ravpn |
| 11 | 172.18.25.11 | pod11-awsadmin | pod11-ravpn |
| 12 | 172.18.25.12 | pod12-awsadmin | pod12-ravpn |
| 13 | 172.18.25.13 | pod13-awsadmin | pod13-ravpn |
| 14 | 172.18.25.14 | pod14-awsadmin | pod14-ravpn |
| 15 | 172.18.25.15 | pod15-awsadmin | pod15-ravpn |
| 16 | 172.18.25.16 | pod16-awsadmin | pod16-ravpn |
| 17 | 172.18.25.17 | pod17-awsadmin | pod17-ravpn |
| 18 | 172.18.25.18 | pod18-awsadmin | pod18-ravpn |
| 19 | 172.18.25.19 | pod19-awsadmin | pod19-ravpn |
| 20 | 172.18.25.20 | pod20-awsadmin | pod20-ravpn |
| 21 | 172.18.25.21 | pod21-awsadmin | pod21-ravpn |
| 22 | 172.18.25.22 | pod22-awsadmin | pod22-ravpn |
| 23 | 172.18.25.23 | pod23-awsadmin | pod23-ravpn |
| 24 | 172.18.25.24 | pod24-awsadmin | pod24-ravpn |
| 25 | 172.18.25.25 | pod25-awsadmin | pod25-ravpn |
| 26 | 172.18.25.26 | pod26-awsadmin | pod26-ravpn |
| 27 | 172.18.25.27 | pod27-awsadmin | pod27-ravpn |
| 28 | 172.18.25.28 | pod28-awsadmin | pod28-ravpn |
| 29 | 172.18.25.29 | pod29-awsadmin | pod29-ravpn |
| 30 | 172.18.25.30 | pod30-awsadmin | pod30-ravpn |

## Step 1: Disconnect Cisco Secure Client

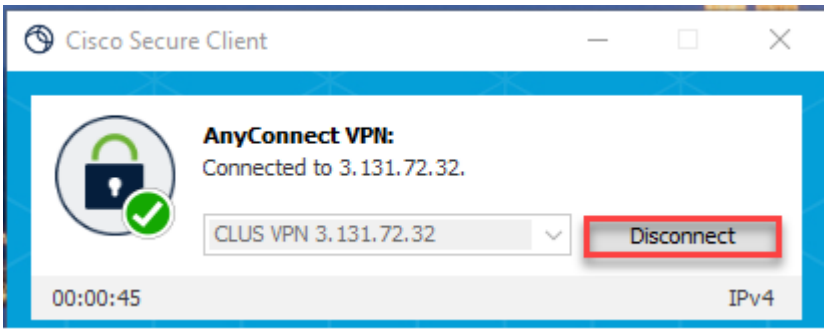
1. Previous labs in the same room may have left Cisco Secure Client connected to a VPN head end different from the one used for this lab. First, navigate to the status bar on the bottom right of the screen to find the Cisco Secure Client icon:




2. Double click on the Secure Client icon to open the GUI.

   a. If you are unable to find the Secure Client icon in the status bar, navigate to **Start > Cisco > Cisco Secure Client** and open it there.

3. Click the disconnect button to disconnect from the current VPN termination point.




## Step 2: Connect to the Lab VPN

1. Using Cisco Secure Client VPN, connect to the ASAv to gain access to the inside network. Connect to IP **3.131.72.32**. Choose the group "RA_VPN" and login with the username (**podX-ravpn**) assigned to your pod in table 1-1 above.

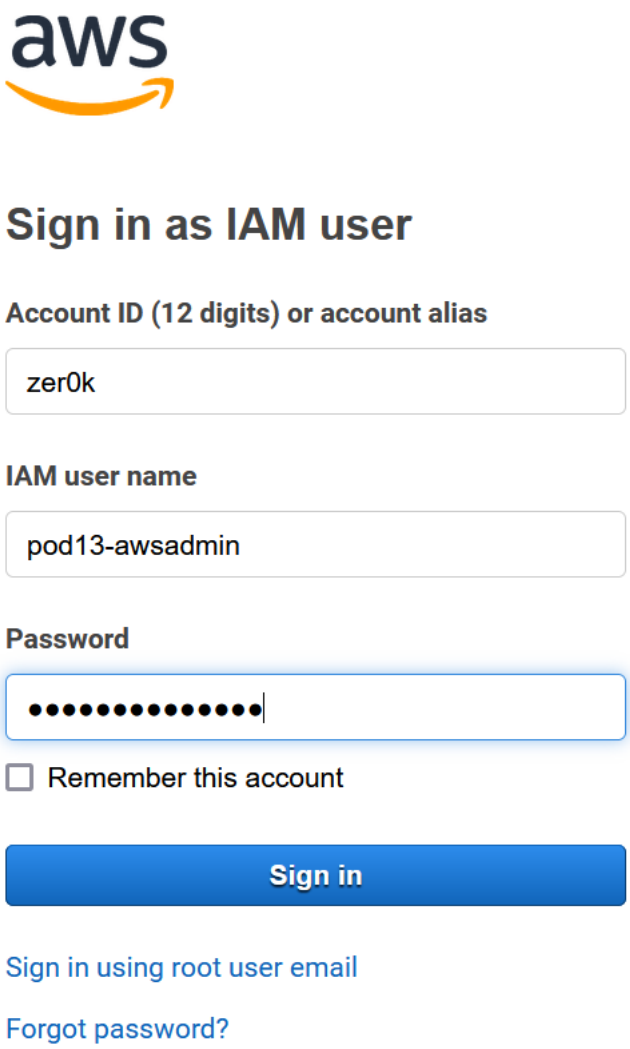
*At this time, do not log into your "PODXX_VPN" tunnel group.*

The password for the account is provided by your proctor. Ignore any certificate warnings presented on connection. There is no public IP access to any of the machines in the pods. When trying to connect to any of the lab machines, ensure VPN is active.

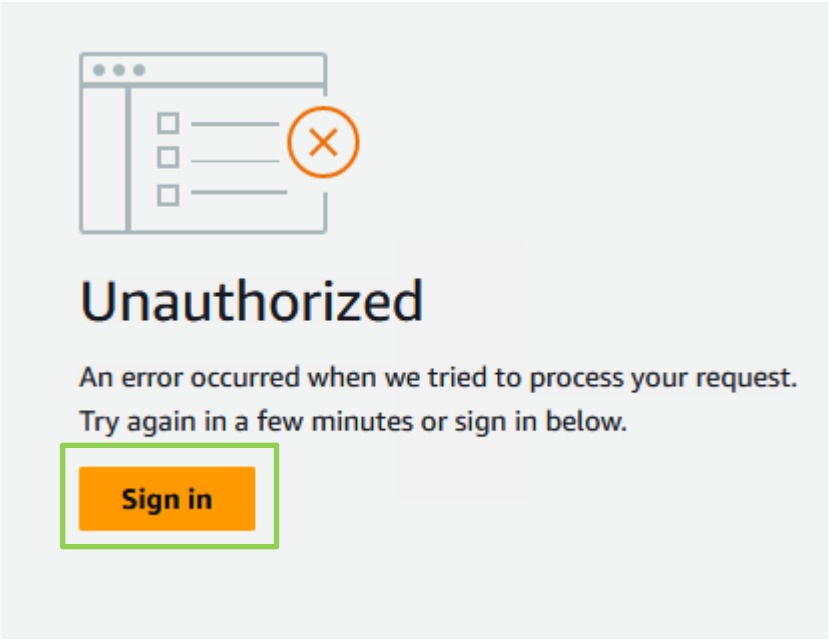
## Step 2: Connect to AWS

1. Open a web browser on the lab laptop and navigate to

https://zer0k.signin.aws.amazon.com/console. Use Account ID zer0k if not already populated. Login with the username assigned to your pod in table 1-1 above (**podX-awsadmin**). The password for the account is **provided by your proctor.**

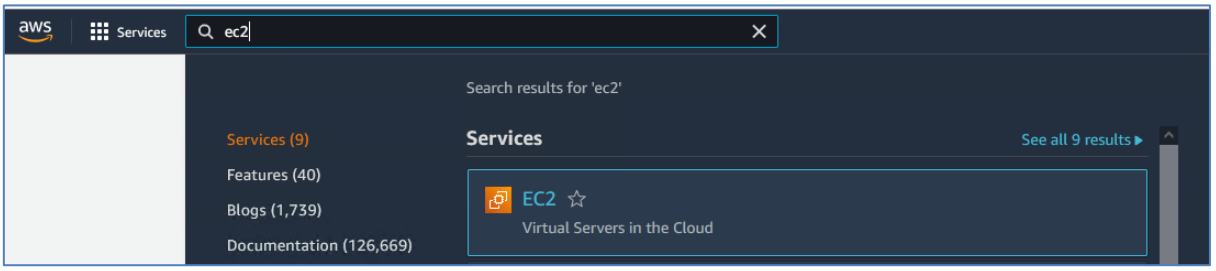



If you are presented with an "Unauthorized" screen, click the Login button and you'll be redirected to the AWS home page.



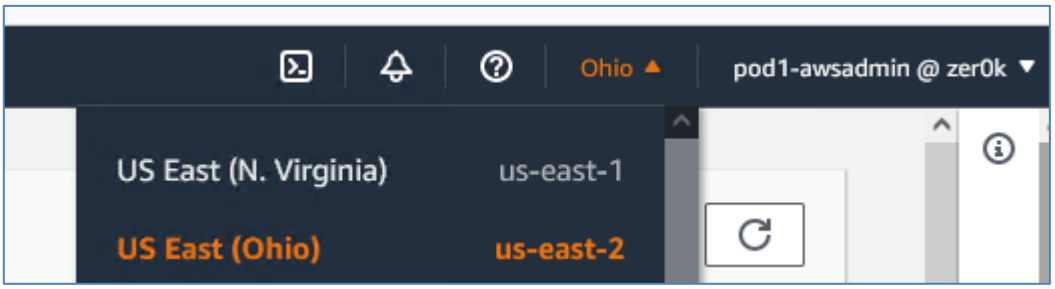

## Step 3: EC2

1. Navigate to the EC2 area of AWS to access available virtual machines and virtual machine settings. In the search box at the top of the screen, type “EC2” and use the EC2 service link. Please do not access any virtual resources not associated with your pod number. Consult a lab proctor for assistance if you are unsure if a resource is one assigned to you or is one that you created.



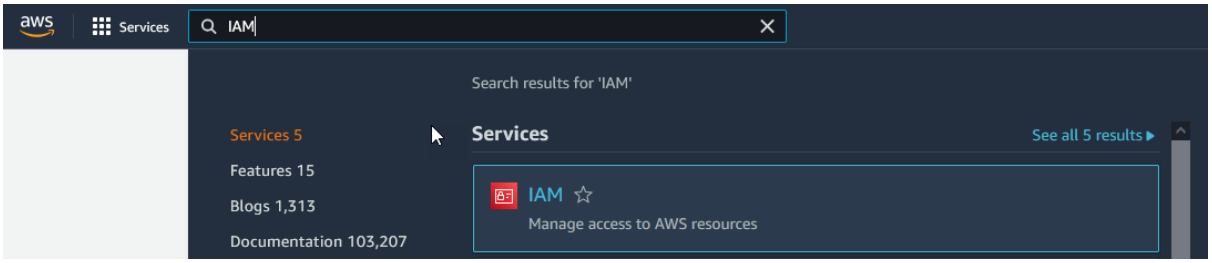
## Step 4: Change AWS Regions

1. The Ansible lab is built within the us-east-2 Ohio region, which is required for connectivity to work between machines. In the EC2 dashboard, ensure that the region, located on the top right of the page, is us-east-2, or “Ohio”.



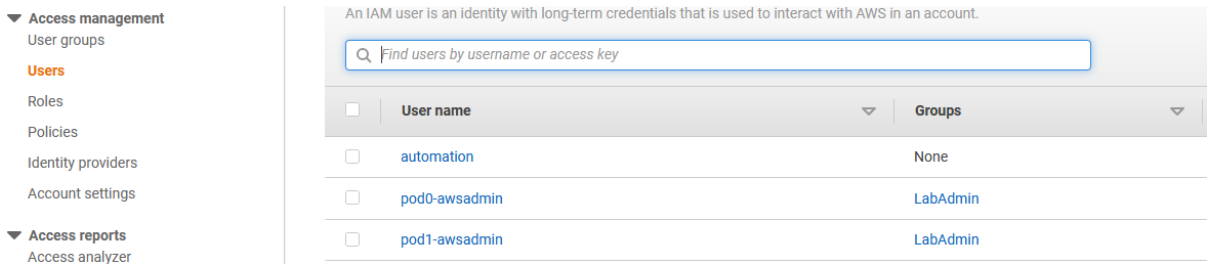

## Step 5: Generate your AWS Access and Secret Key

1. From the top search bar in AWS, type IAM to navigate to Identity and Access Management. Click IAM.



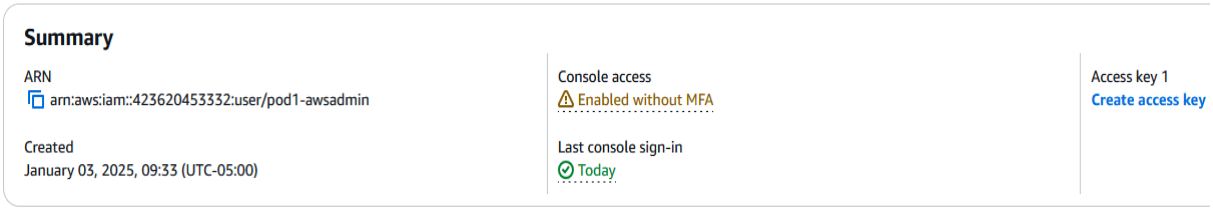

2. From the left bar, click “Users” and select your pod username.

**Note:** You may need to navigate to the second page of credentials or use the search box on this page depending on display settings

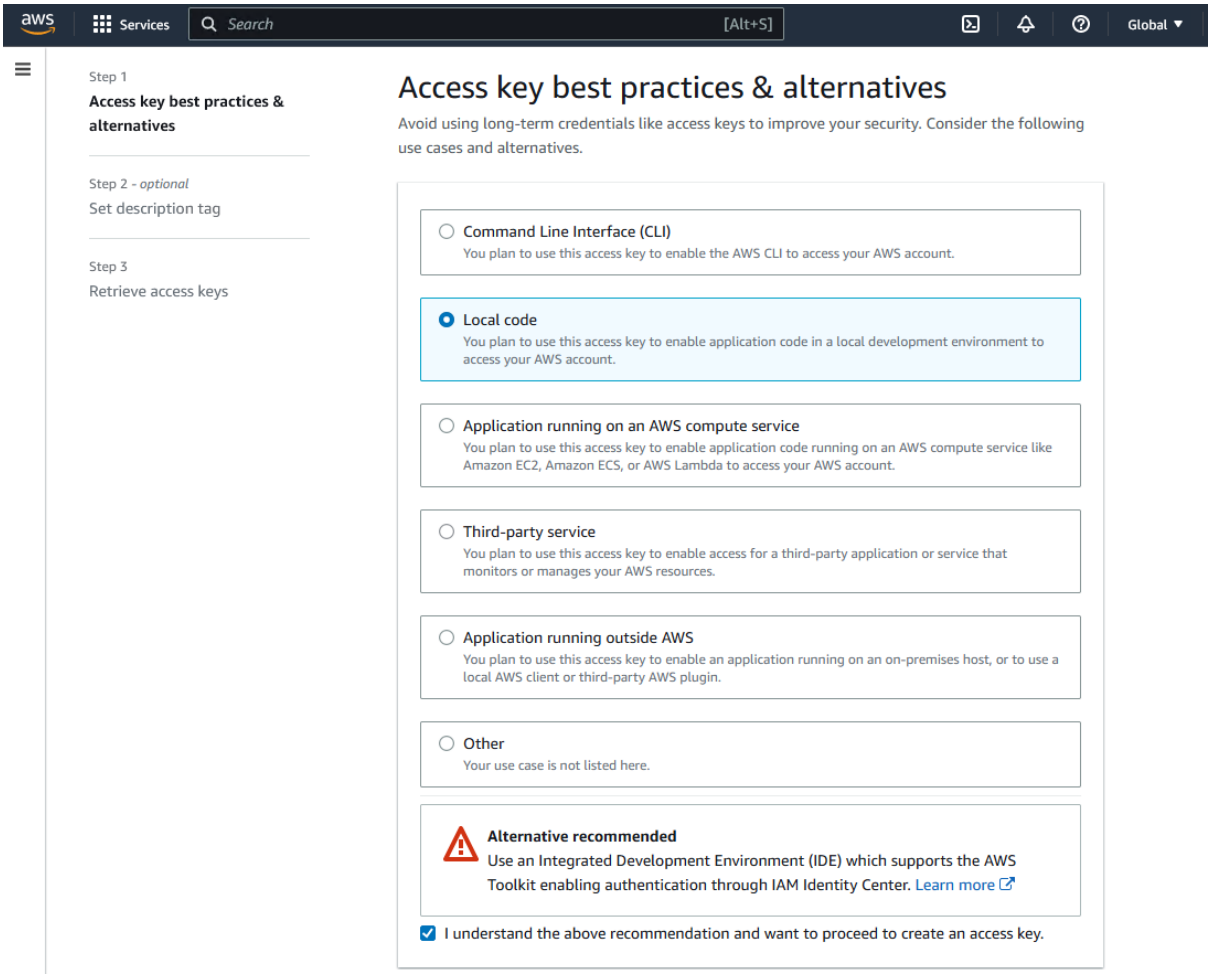



***Warning: Common mistake area!***

3. Under the Summary subsection and under Access Key 1, select “Create Access Key”.




You will use this access key for your environmental variables used in future steps so that Terraform is able to access your AWS account. When presented with the “Access key best practices & alternatives” select “Local Code” for your use case. Check the box at the bottom of the page for “I understand the above recommendation and want to process to create an access key”.



4. Click Next.

5. For the tag description enter “Pod X local code access key” where pod X is your pod number.

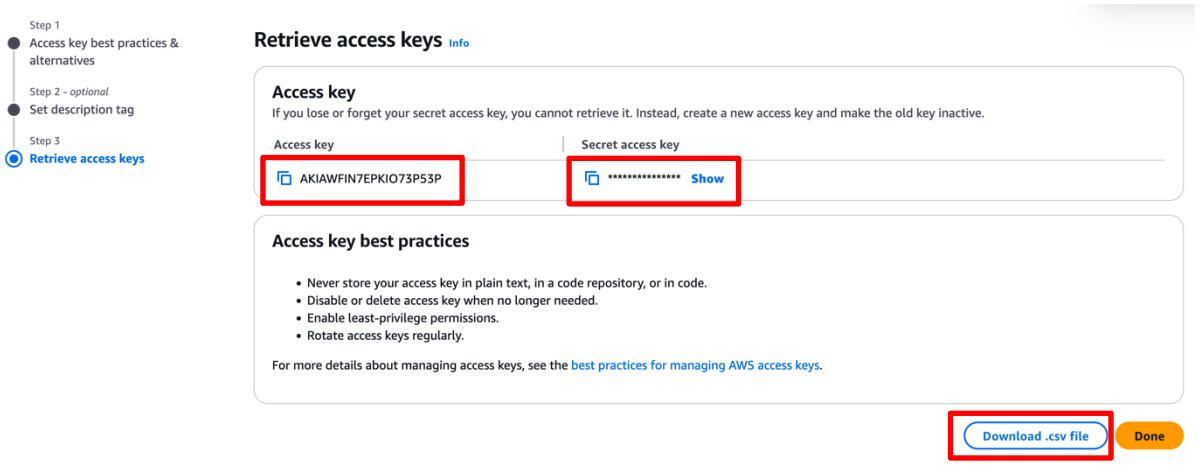
6. Click “Create Access key”

7. Click “Show” to show this secret key.

8. Copy the Access Key and Secret Key to a notepad.

> **Ensure you copy the secret key, as it will not be available again.**
>
> **Label your access key and secret access key accordingly, ensuring you paste the right string into the proper variables later in the lab.**

In addition, you can download a CSV file with the access and security keys contained within it.

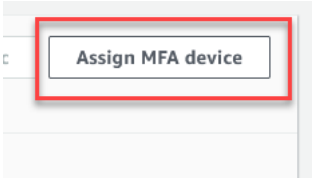



## Task 2: Set Up Two Factor Authentication

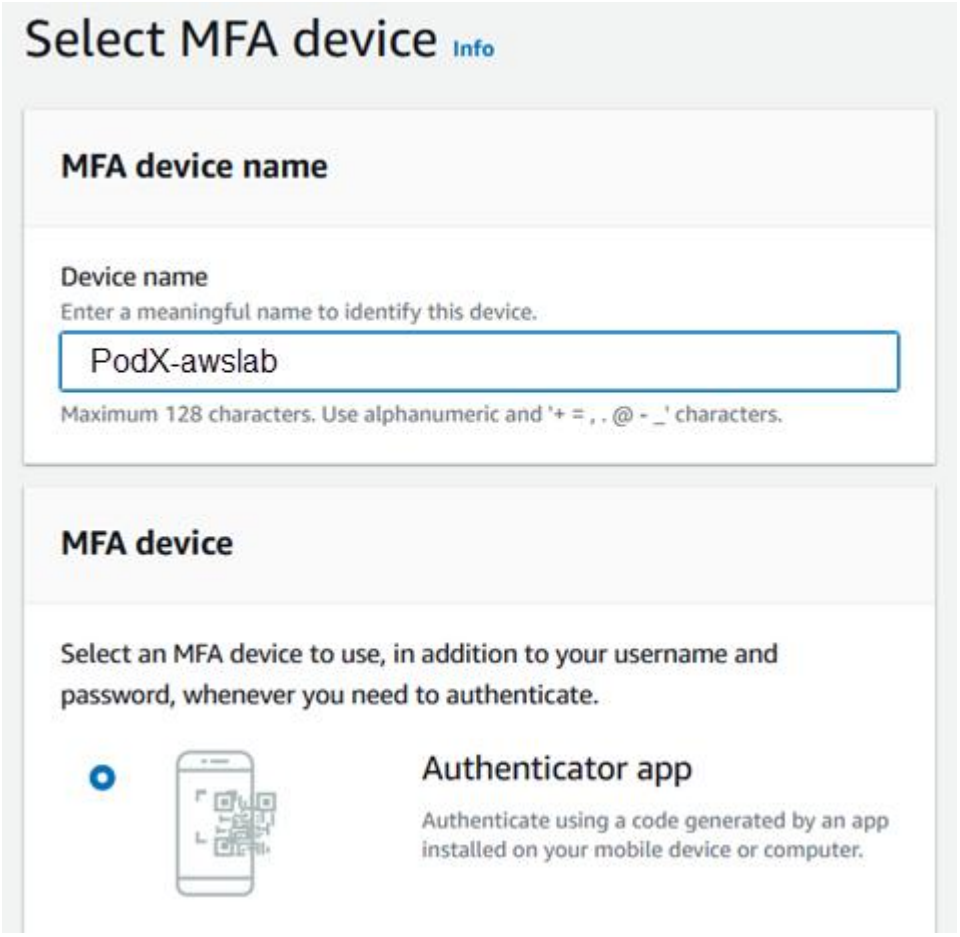
Each pod used within this lab has administrative access to the test deployment. Therefore, it is best practice that two factor authentication be set up for the administrative user. For this task you’ll need an authenticator app such as Google Authenticator

1. In the IAM -> Users -> <your username> -> Security Credentials area, navigate to **Multi-factor authentication (MFA)** section.

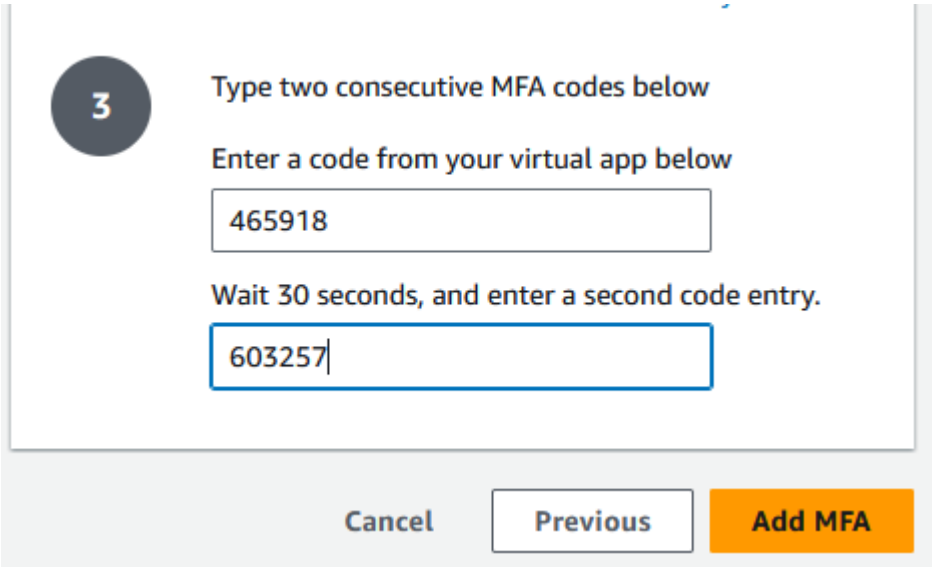
2. Click “Assign MFA Device”.




3. Enter a name for the device, we recommend **podX-awslab** and select the preferred method of “Authenticator App”.




4. Using Google Authenticator (or your app of choice), scan the QR code and confirm the next two MFA codes.




5. Logout of the console and log in with the user to which you assigned the MFA device. Enter the next two factor authentication code when prompted.

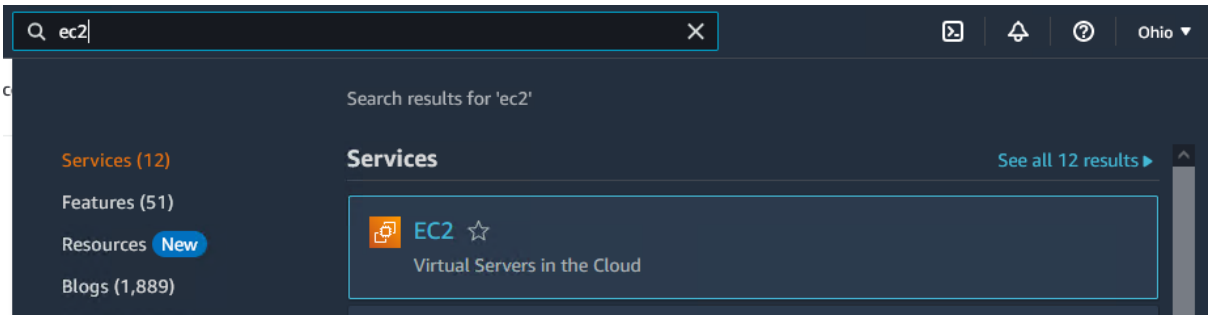
## Task 3: Deploy Ubuntu Provisioning Machine / Building Blocks

There are many ways to create an instance in AWS without creating each object ahead of time, but it is helpful to understand each object when CloudFormation templates and scripts are used later in the lab. It is also helpful to know where each of these objects resides when troubleshooting connectivity problems.

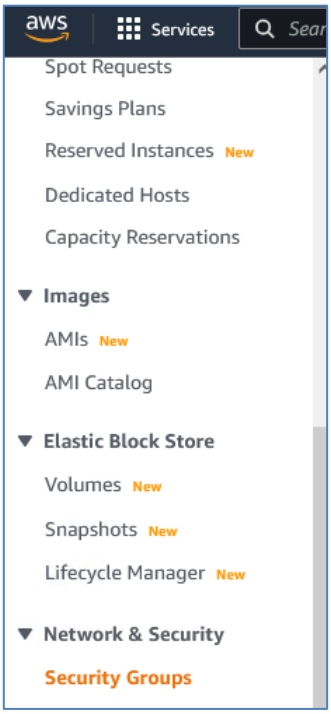
### Step 1: Create Security Group

Security Groups are AWS ACLs that control network traffic, by default all outbound network traffic from a security group is permitted as well as the return traffic from that outbound connection (stateful). By default all traffic is denied inbound to an AWS EC2 instance.

1. Navigate back to EC2 using the search bar at the top of the page.

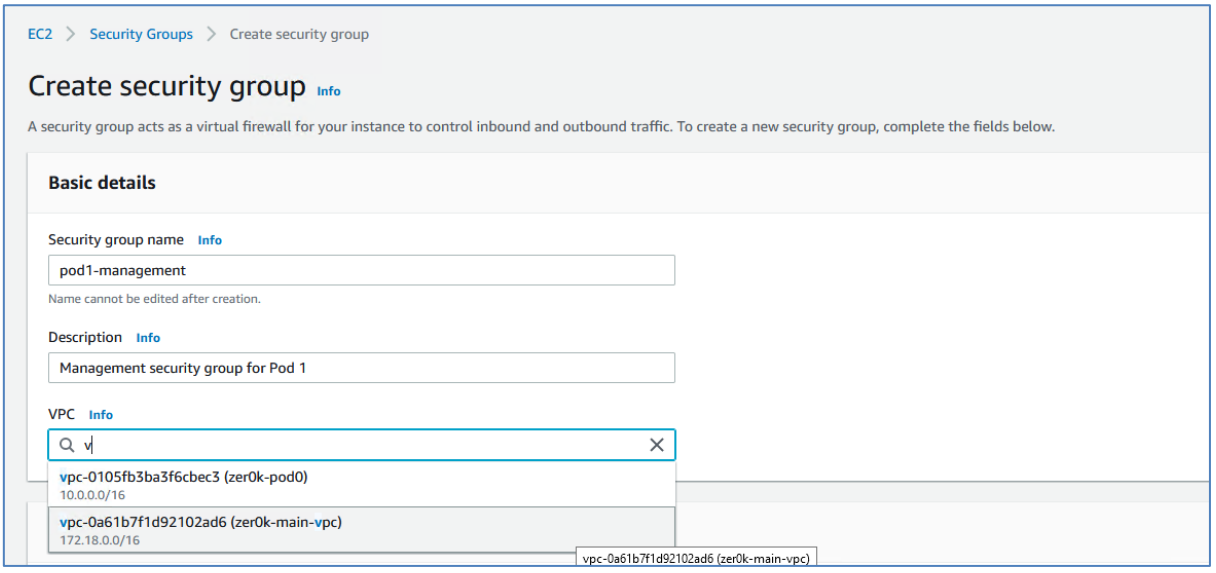



2. In the left navigation panel within the EC2 dashboard, Browse to Network & Security -> Security Groups.

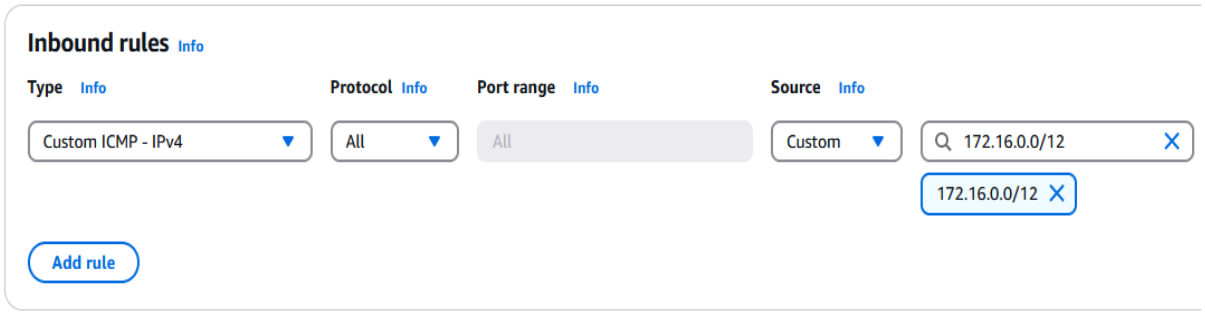


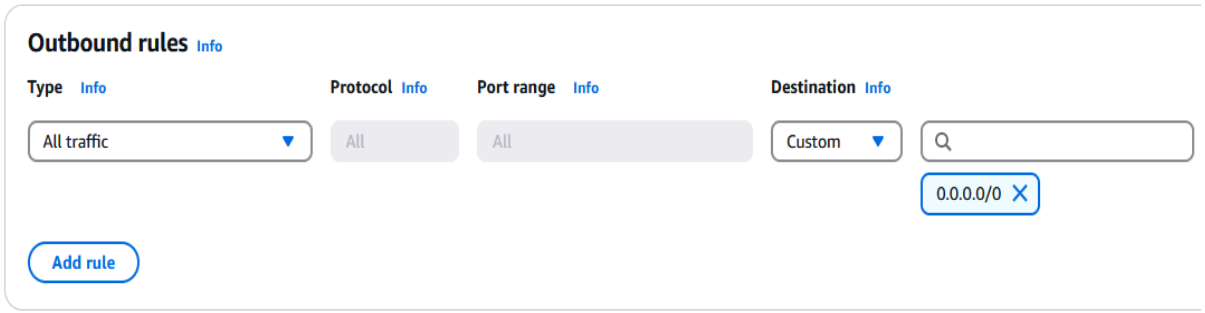

3. Select “Create Security Group” from the upper right-hand side of the page.
4. Name the security group **podX-management-sg** where X is your assigned pod number, add a required description, we recommend “**Management security group for pod X**” where X is your pod number. See the screenshot in step 5 for verification.
5. Make sure the VPC is set to vpc-0a61b7f1d92102ad6 (zer0k-main-vpc). Security groups for all pods will be associated with the main VPC.

Note: You should be able to delete the current VPC name (if populated) and begin typing “zer0k” to show available VPC’s.




6. Add an inbound rule allowing ICMPv4 traffic from 172.16.0.0/12.



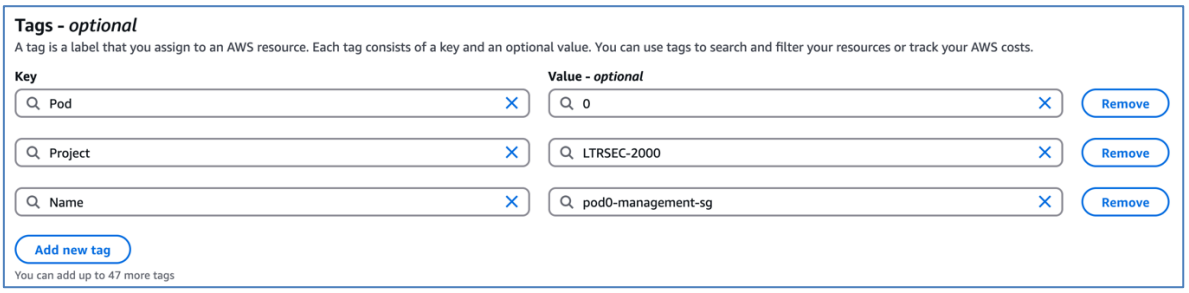




**Note: Ensure you added this rule under the inbound rules header.**

**Note: Cisco monitors all activity from users within the AWS cloud to ensure that limits are placed on reachability. Please ensure you add 172.16.0.0/12 and NOT 0.0.0.0/0 inbound. If this mistake is made, the rule will be removed and proctors will receive an automated notification that a reachability rules has been triggered.**

7. Do not add any other rules at this time.
8. Under Tags, add two new tags
   - Key of “Pod” and a value of your pod number, typically in the range of 1 to 60.
   - Key of “Project” and a value of “LTRSEC-2000”

Tip – AWS does not name anything by default, always add a “Name” tag to easily find the object later.




9. Create the security group.

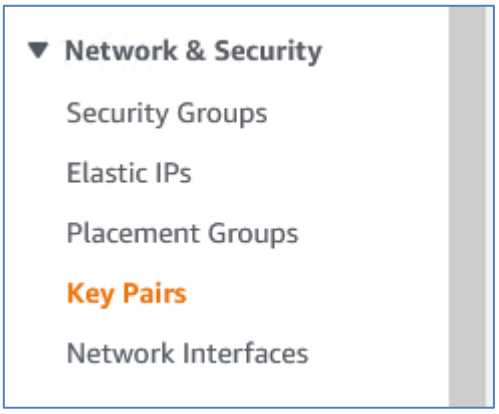
## Step 2: Create a Key Pair

Key pairs are used to connect to instances after creation and can be used

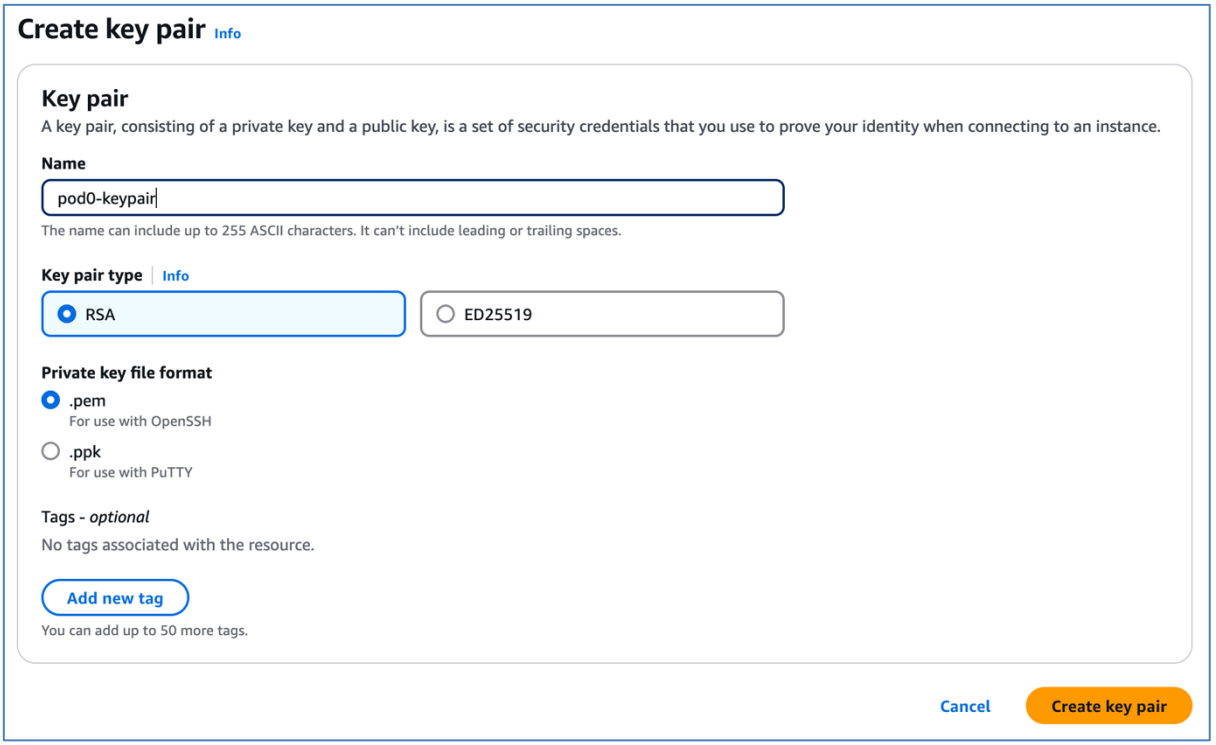
for other tasks in AWS such as decrypting Windows passwords.

**Note: For the following step, the private key created is NEVER available for download again, if it is lost a new key pair must be created.**

1. In the EC2 service, navigate in the left bar to Network & Security -> Key Pairs.




2. Select “Create Key Pair”.
3. Name your keypair podX-keypair where X is your pod number. Leave the default value of RSA and ensure .pem (For use with OpenSSH) is used. In this case the AWS UI uses the Name field as the Tag value, so no tags are required.




4. Select “Create key pair” and ensure the .pem file is saved to your lab laptop. Downloading the file as .pem will allow it to be used from a Linux or Mac host and can be converted later for use in Putty.  This should happen automatically.

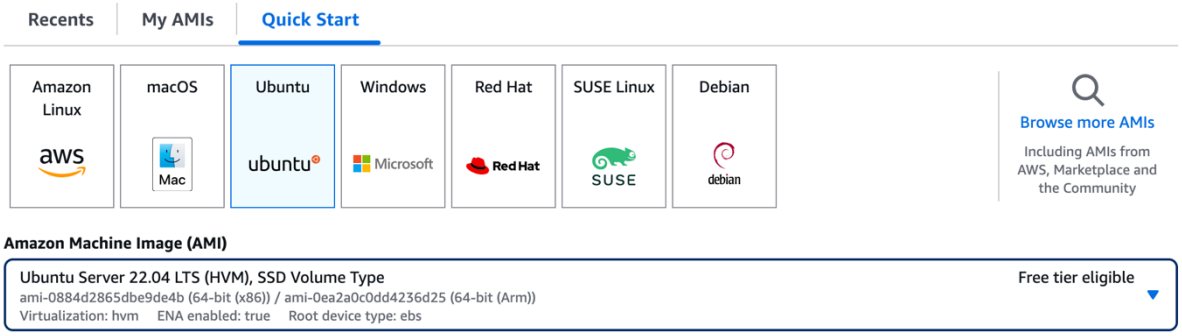
**This private key is never available for download again, if it is lost a new**

**key pair must be created.**

## Step 3: Deploy Ubuntu Provisioning Machine

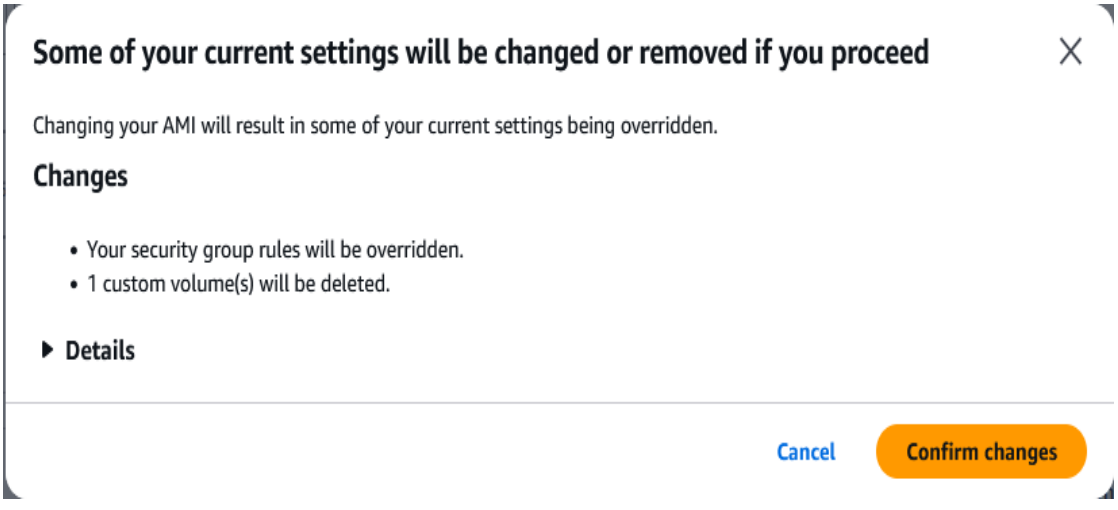
All of the building blocks are in place from a networking standpoint to create an Ubuntu Linux instance.

1. From the Left Navigation bar, navigate to Instances -> Instances.
2. Select “Launch instances” from the upper right corner.
3. Under Name enter “podX-Ubuntu” where X is the number of your assigned pod.
4. Under Application and OS Images, choose Ubuntu under quick start and **Ubuntu Server 22.04** as the AMI. Leave the rest default. There should be a Free Tier Eligible Ubuntu instance chosen.



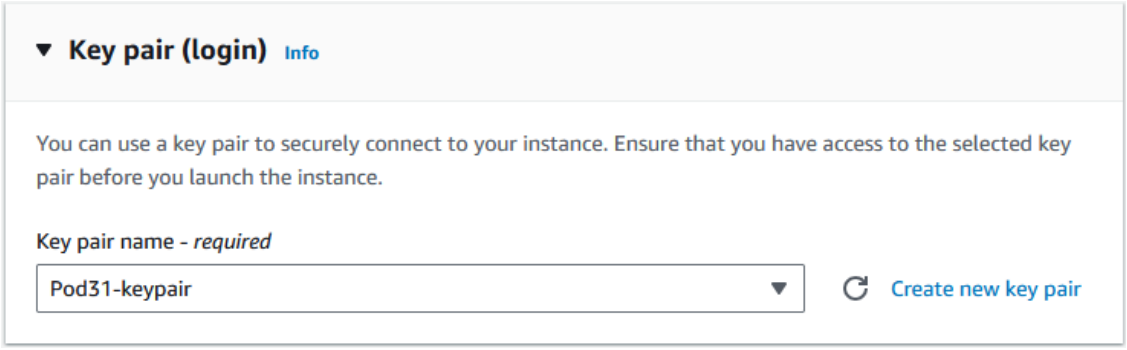

**Note**: Some of the settings within this lab are reliant upon Ubuntu 22.04. If a mistake is made to select 24.04 here, you’ll need to redeploy from this step onward.

5. If you click another image by mistake in this process, you may be presented with a dialog box warning that some of the current settings will be changed or removed. Confirm these changes.

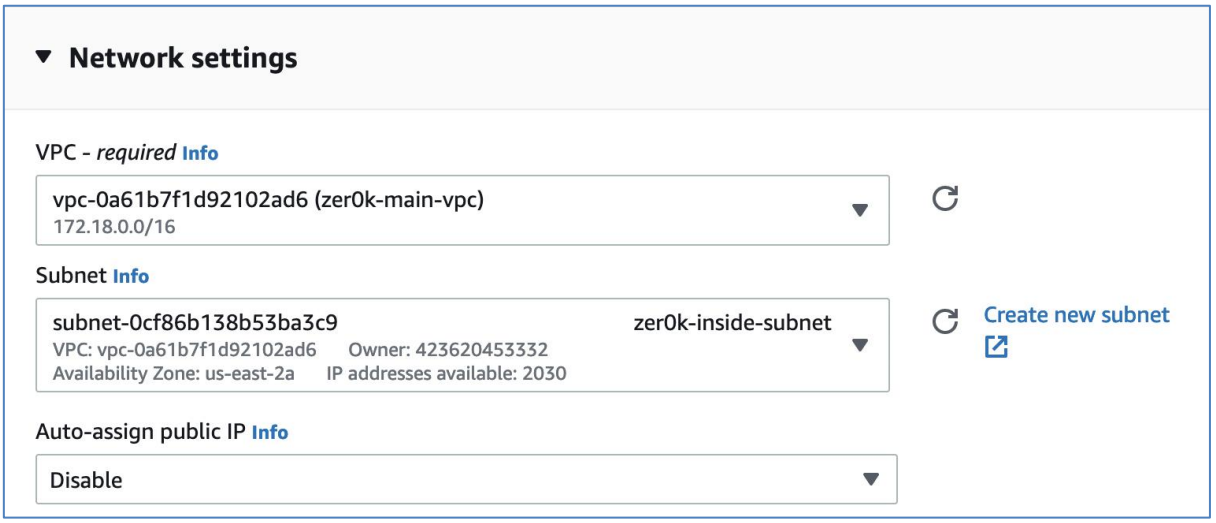



6. Under Instance Type leave it as default x86 t2.micro.
7. Under Key pair, choose the key pair you created under Step 2 of this

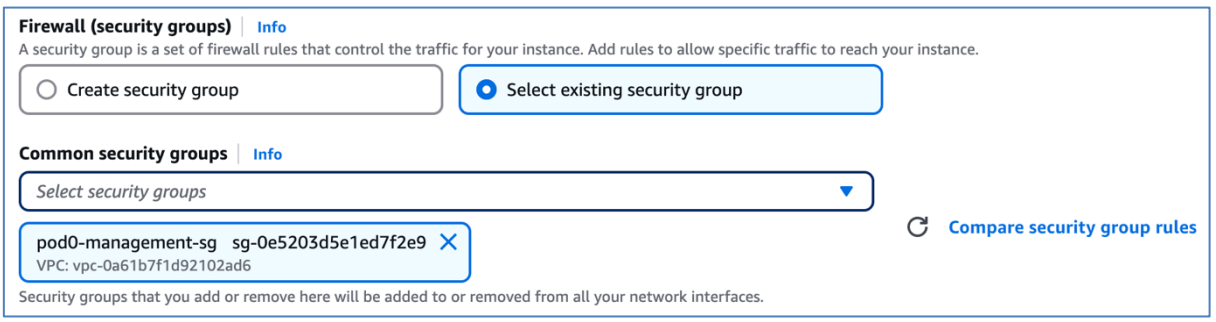
task. This is the key pair you have downloaded to your local machine. It should have been named “podX-keypair” where X is your assigned pod number.




8. Under Network Settings click “Edit”.
9. Set the VPC to vpc-0a61b7f1d92102ad6 (zer0k-main-vpc).
10. Set the Subnet to subnet-0cf86b138b53ba3c9 (zer0k-inside-subnet).
11. Leave Auto Assign Public IP disabled as there is no direct internet access from this subnet.

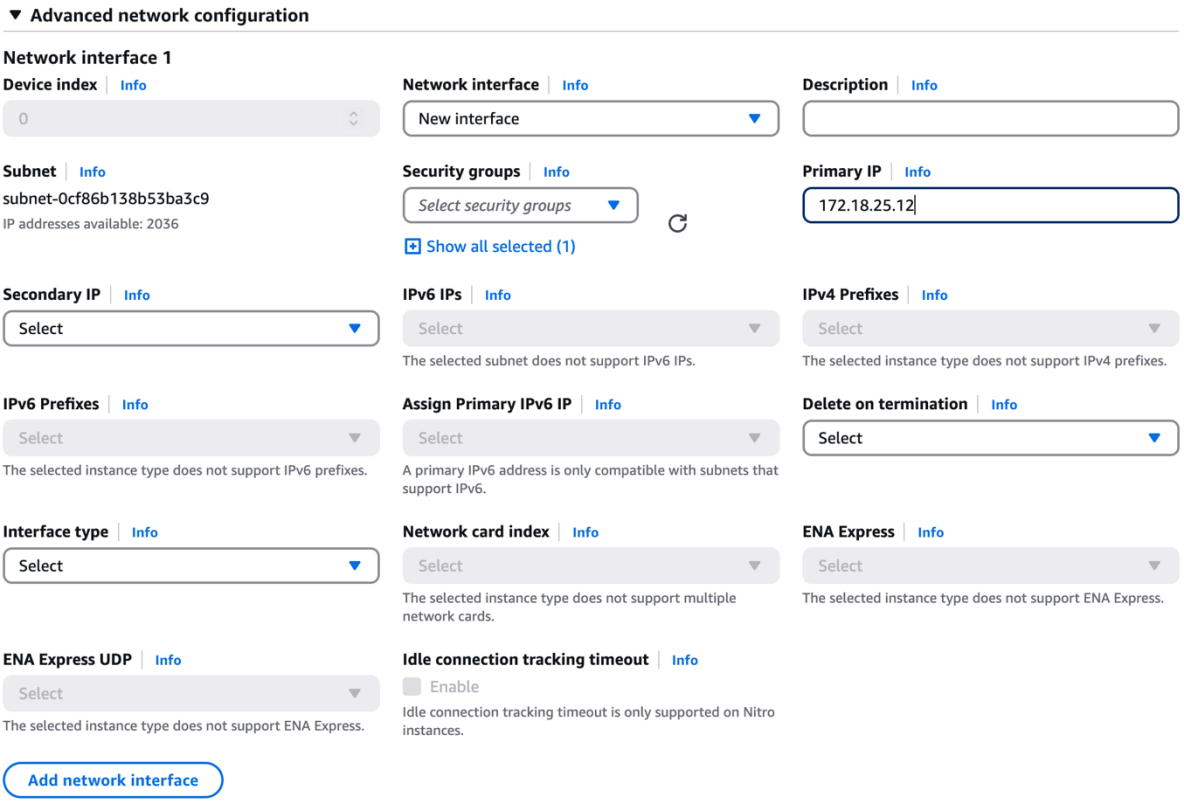



12. Choose Select existing security group and select the security group you created in Step 1 above. It should be podX-management-sg where X is your pod number.




13. Expand “Advanced network configuration”.

14. Under Primary IP, configure 172.18.25.X where X is your pod number.




15. Leave the options under Configure storage and Advanced details as defaults.
16. Select “Launch instance” on the bottom right of the page.
17. If the creation of the instance is successful, click on “View all instances” to go back to the instance list. Otherwise, notify a proctor.
18. In the EC2 service, browse to Instances -> Instances.
19. Verify that the status check for your podX-Ubuntu machine is showing 2/2 checks passed. If it isn’t, refresh the page until the status is 2/2 checks passed or you see a red status. It will have to be troubleshot if red, move on once it is 2/2 checks passed.

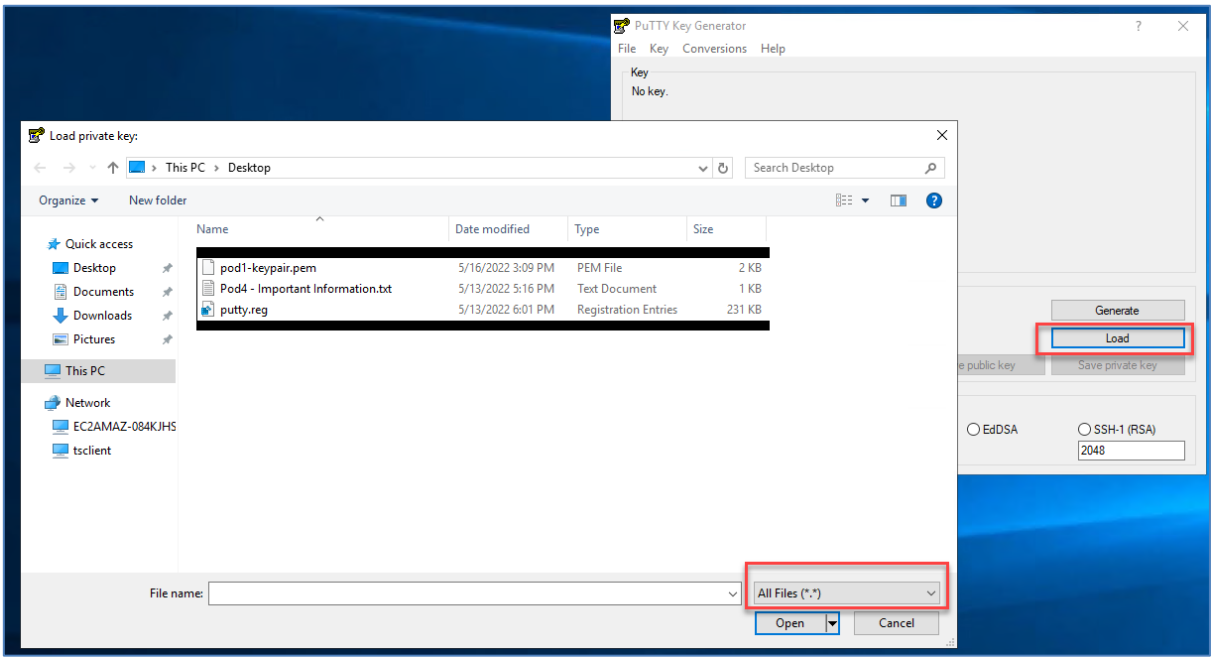

## Step 4: Convert PEM to PPK format for dual use

The PEM file downloaded for use as a keypair within AWS will both serve as the keypair within AWS, as well as the connectivity file used for Putty. To use this file within Putty, it must be in PPK format. Locate the PEM file on the lab laptop, it is recommended you move it to the desktop for easy retrieval. By default, Firefox will put it in the Downloads folder.

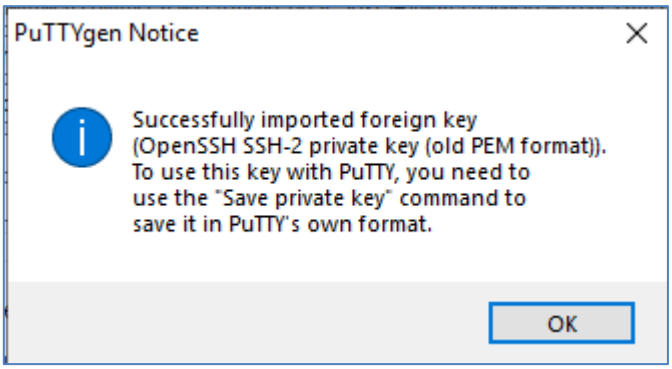
*Putty on Windows:*

PuTTY is installed on the lab laptop and is recommended to be used, however it can be downloaded if doing this on an alternative machine. If you don’t already have PutTTY and PuTTYgen, download them from here: https://www.chiark.greenend.org.uk/~sgtatham/putty/

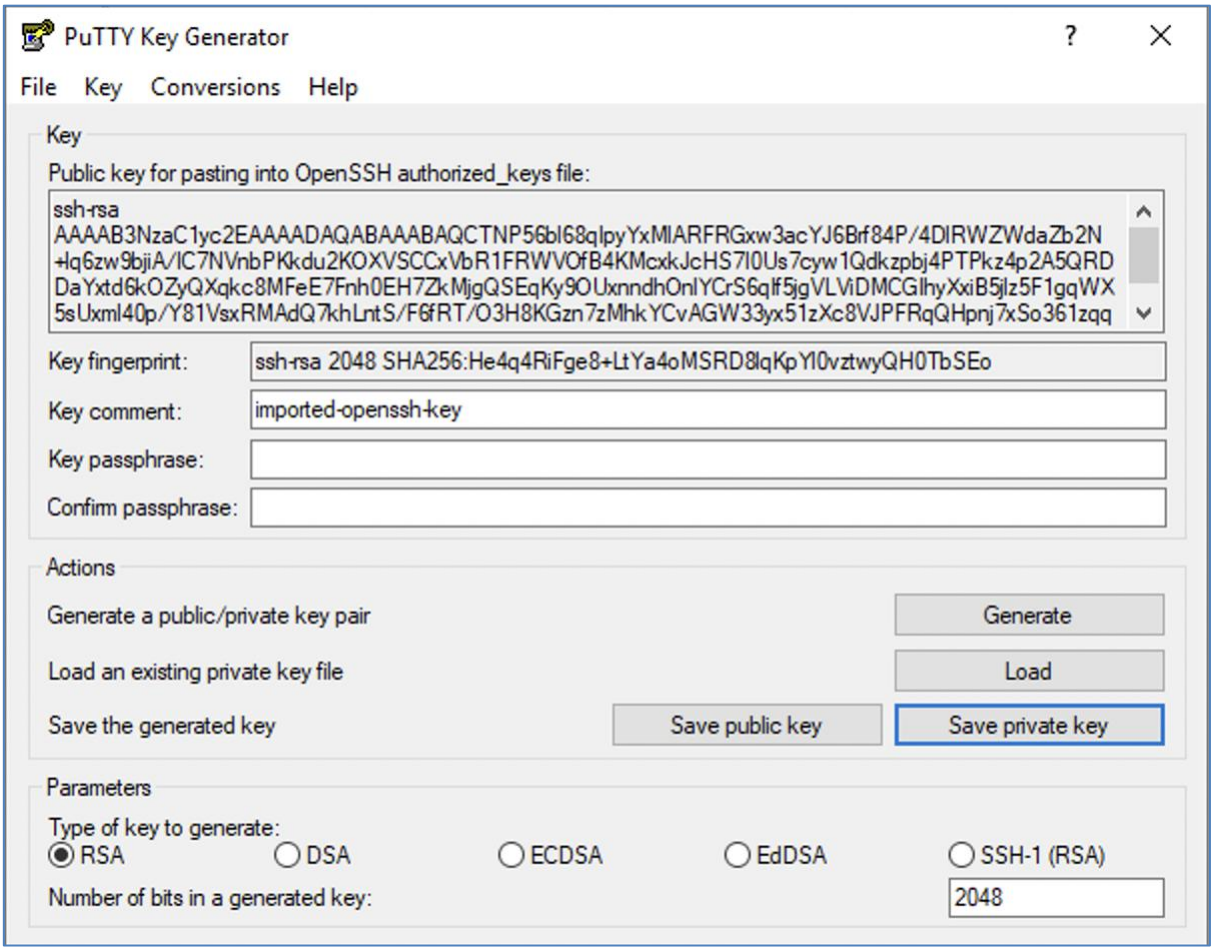
1. In the start menu of the lab laptop, navigate to PuTTY -> PuTTYgen and open PuTTYgen.
2. Click “Load” and change the file type to “All Files (*.*)” to select the PEM keypair you created in the last task, it should have been named podX-keypair.pem where X is your assigned pod number.




3. The selection of the keypair should result in a successful imported foreign key dialog.




4. Leave RSA chosen as the default key type under parameters. Save the **private key** without passphrase to the same location as the PEM file. Name this keypair “podX-keypair.ppk” where X is your assigned pod number.



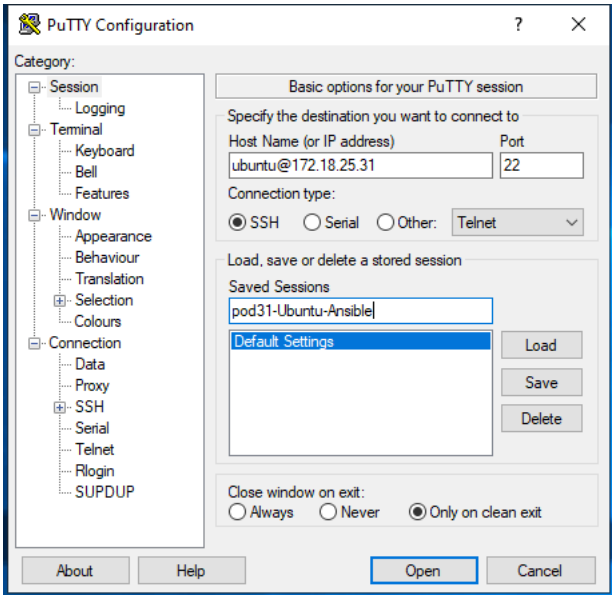

5. Close PuTTYgen.
6. This converted key will be used later in the next step.

## Step 5: Configure Putty for SSH on Local Machine

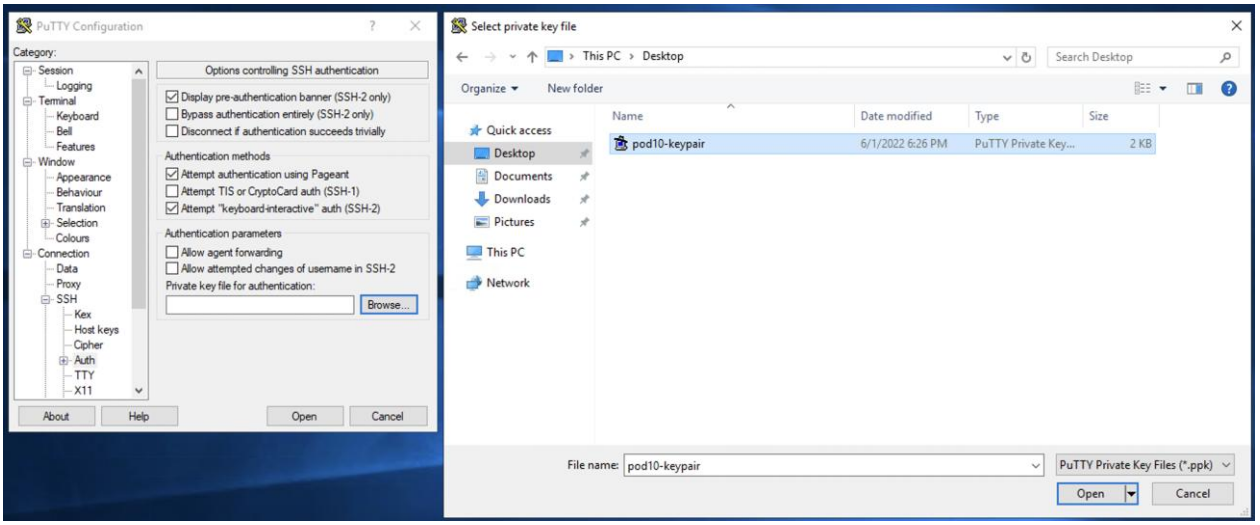
In this section the SSH client is just being setup,

| Note: SSH will not work until Step 6 is completed. |
|---|

1. On the lab laptop, navigate back to the desktop and open Putty from the shortcut.
2. The default screen is Session, set the Hostname to "ubuntu@172.18.25.X" and Saved Sessions as "podX-Ubuntu" where X is your assigned pod number. Click the save button.




3. Navigate to Connection -> SSH -> Auth -> Credentials. In the "Private key file for authentication" box at the bottom of the screen, select the private key created from the PEM file in step 3. It should have been named podX-keypair.ppk located on the desktop from the previous preparation step.




4. Optional: Navigate to Session -> Logging, select "All Session Output", browse, and place the log file in the desktop. This is strictly for future troubleshooting if desired.
5. Do not change any additional settings.
6. Go back to Session and save again.
7. Attempt to SSH to your podX-Ubuntu machine from your Windows host using the previously set up profile. **It timed out!**

*Optional: If using Linux/Mac Command Line*

The permissions of the key file must be changed to 400 for the command line ssh client to use the key:

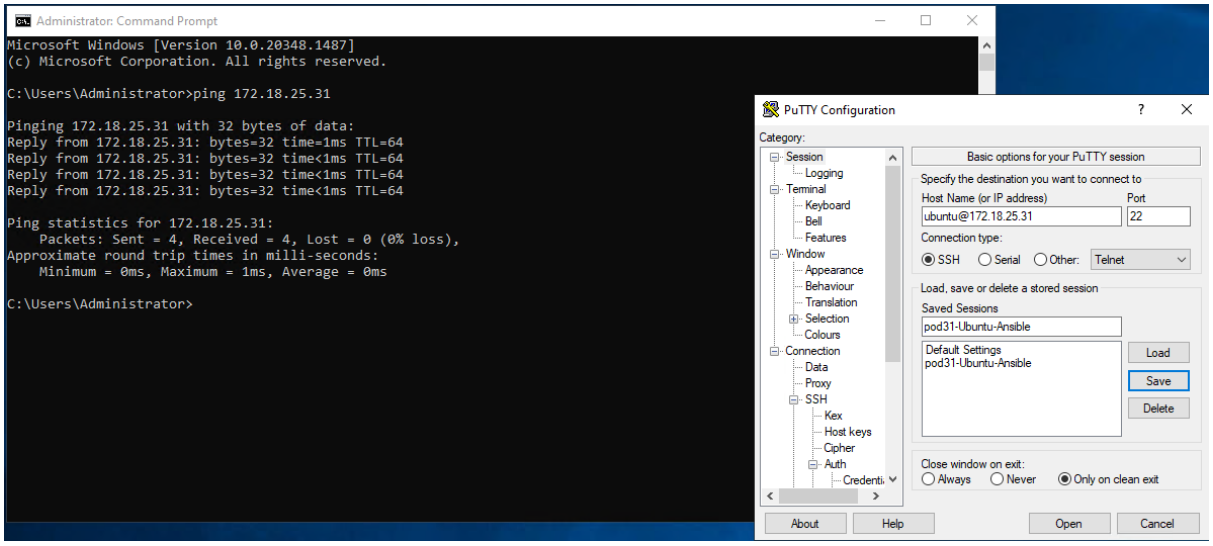
```
> chmod 400 <keyfile>
```

To use the private key file use ssh -i

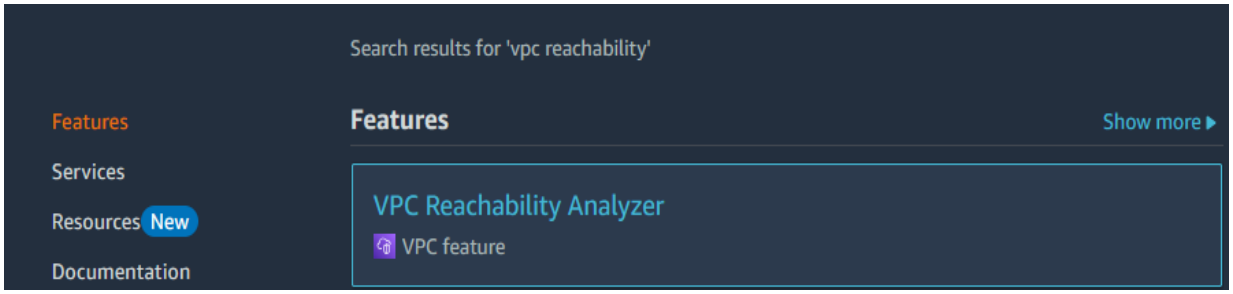
```
> ssh -i <keyfile> <username>@<host>
```

## Step 6: Verify Connectivity

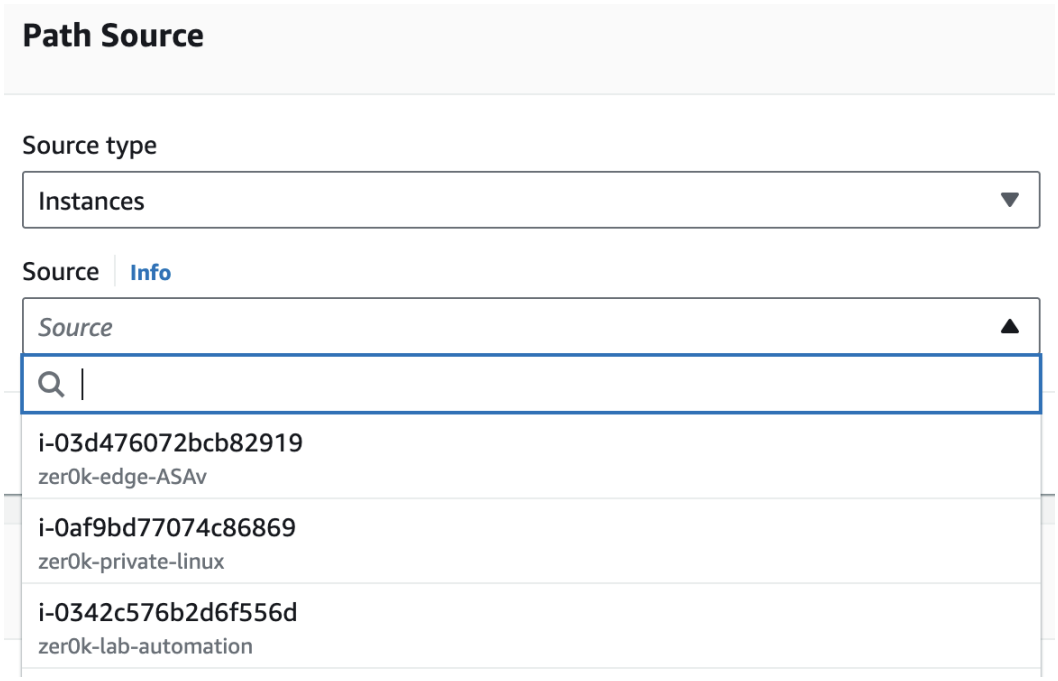
1. From your lab laptop, ping your podX-Ubuntu server, remember the IP is 172.18.25.X. This should be successful. If it isn't, recheck your security group. The following steps will also help to troubleshoot.



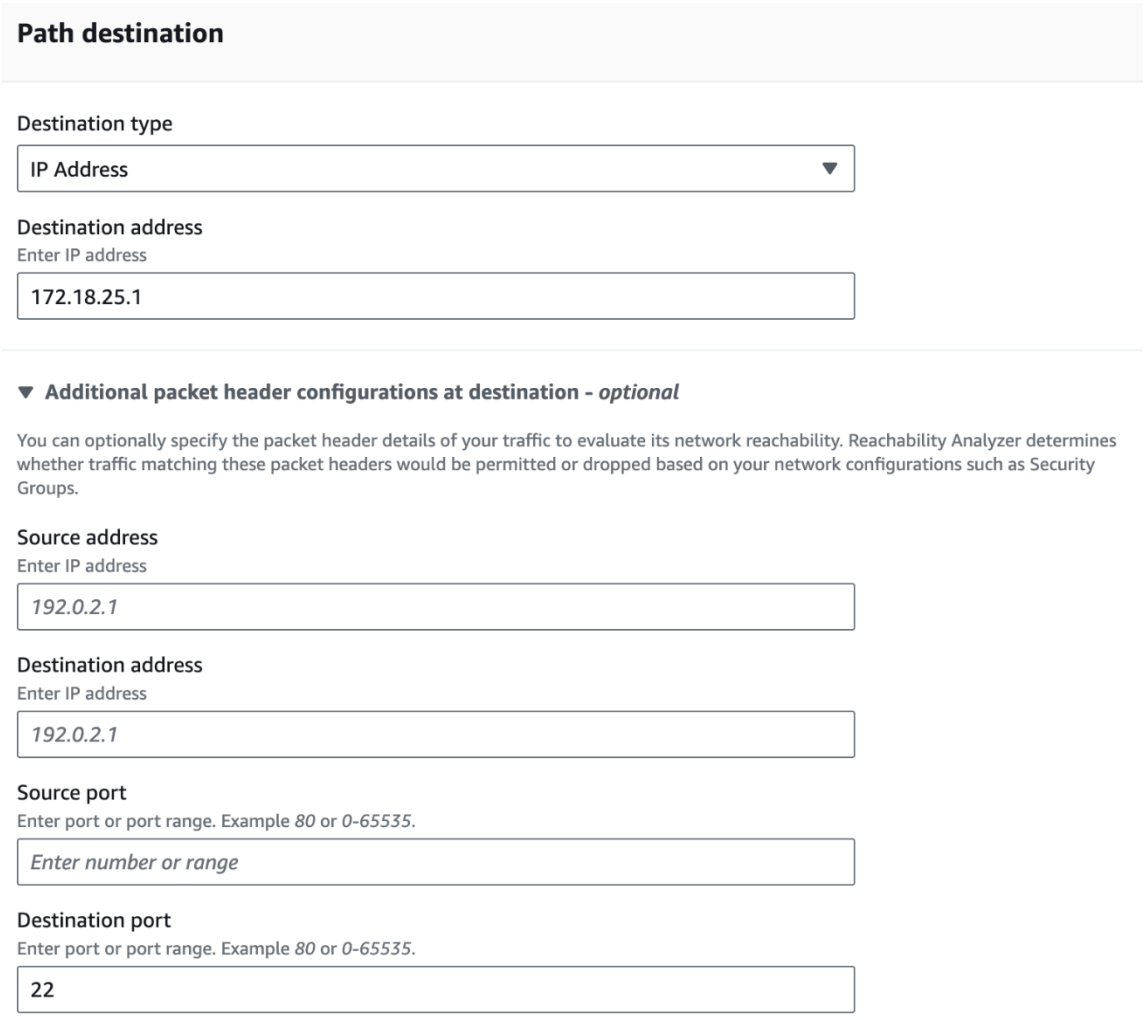

2. Back in the AWS console, use the search bar to search for "VPC Reachability Analyzer"



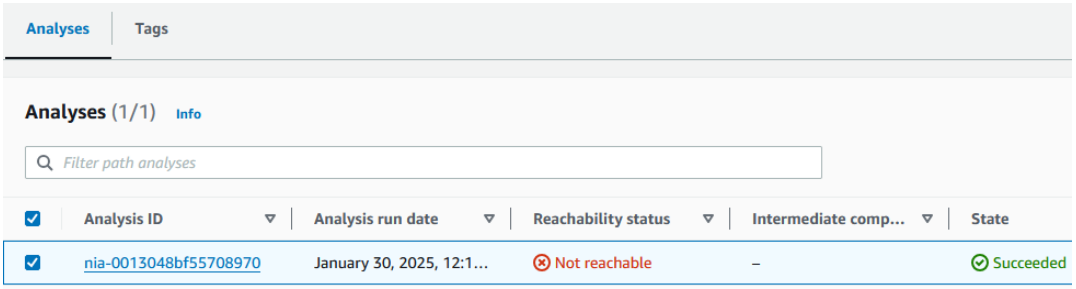

3. Click Create and analyze path.
4. Name it "SSHtoPodX-Ubuntu" where X is the pod number you were assigned.
5. Set the source type to "Instances" and locate the instance named "zer0k-lab-automation"



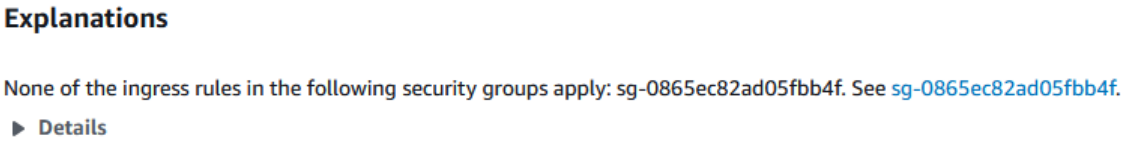

6. Set the destination type to “IP Address” and enter the IP of your Ubuntu machine which should be 172.18.25.X where X is your pod number.
7. Expand “Additional packet header configurations at source – *optional.*"
8. Set the destination port to “22” for SSH and the protocol as TCP



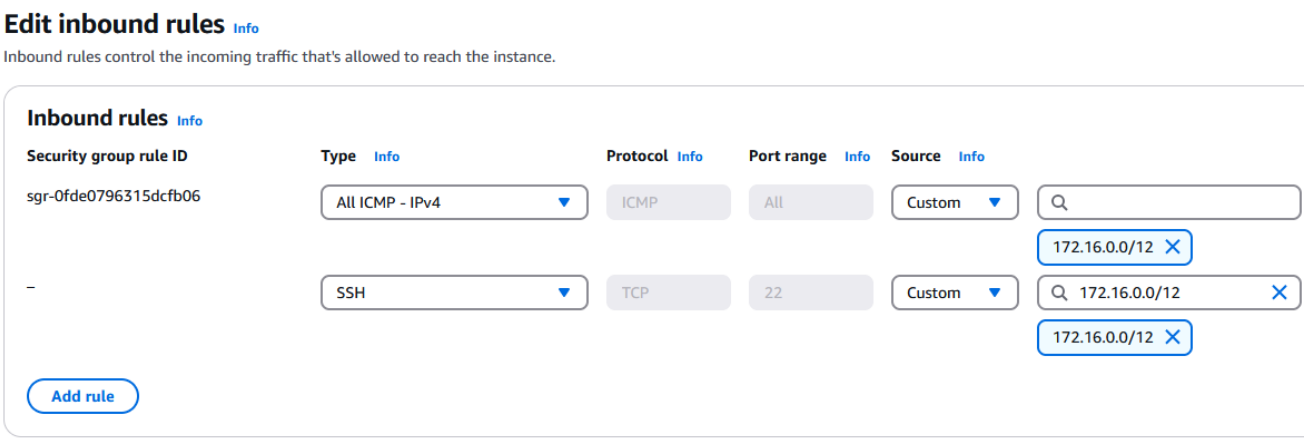

9. Select Create and analyze path in the bottom right.
10. You will see the Analyses as pending, refresh the page until you see success. The reachability status should be Not Reachable.




11. Check under Explanations to see why AWS thinks your instance isn't reachable on port 22. It should be because of a missing security group rule.

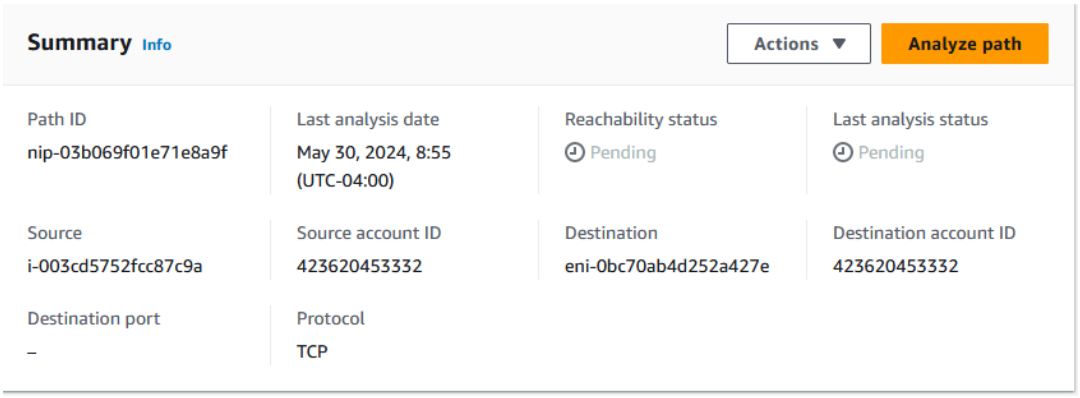


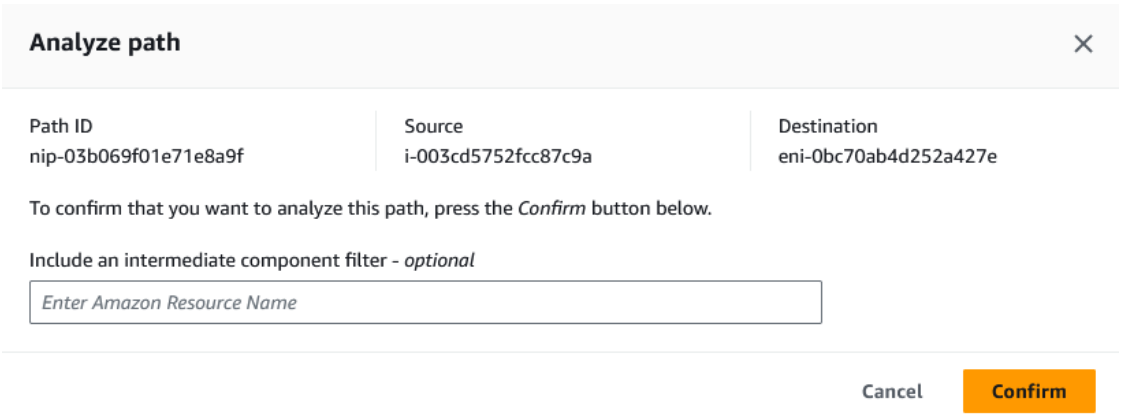

12. Click on the link to the security group. Click "Edit inbound rules" under the "Inbound rules" tab.
13. Add another inbound entry for SSH from 172.16.0.0/12, all hosts are behind the ASA firewall already so allowing from the private subnet is fine for this lab.



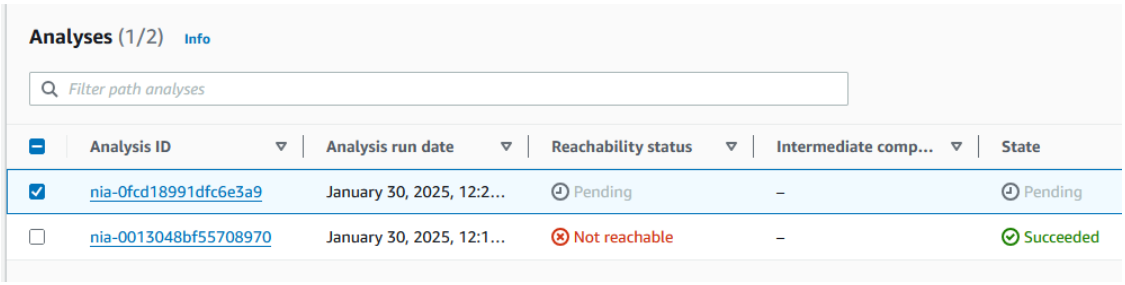

14. Save the rule.
15. Go back to the VPC Reachability Analyzer using the search bar.

16. Check the box next to your previously saved analysis. Click “Analyze path”. Click confirm.

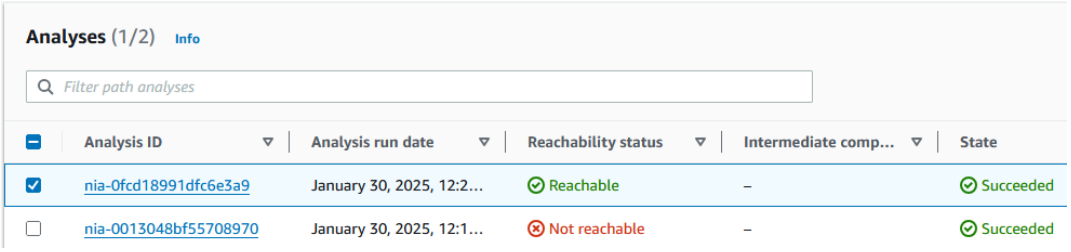



17. You may need to use the refresh button in the pane to see a new entry in the analyses table.

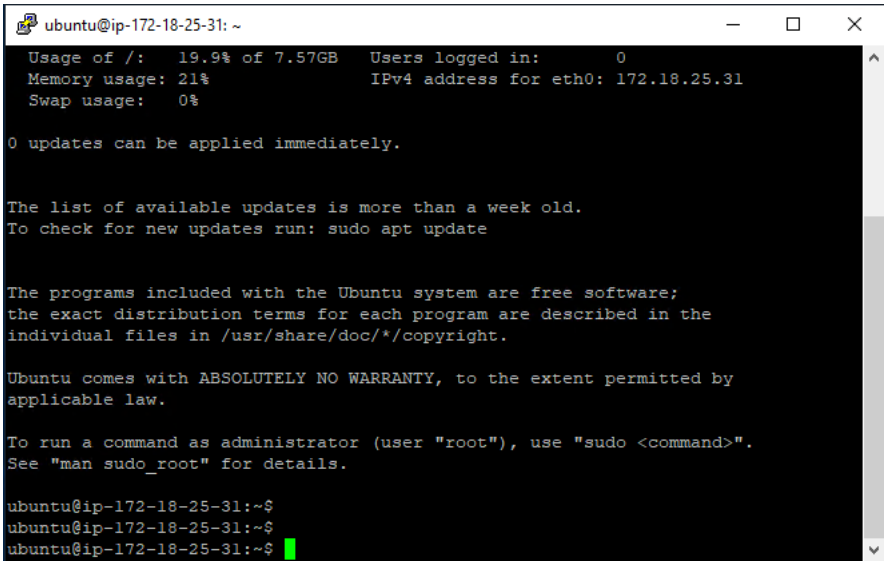



18. There will be a new entry pending under Analysis, click refresh until it succeeds. The Reachability status should now be reachable. If it is not, view the Explanations.




19. From your lab laptop, attempt to SSH to your provisioning machine,

accepting any key acceptance messages.  It should succeed this time and you should be at an ubuntu command prompt.



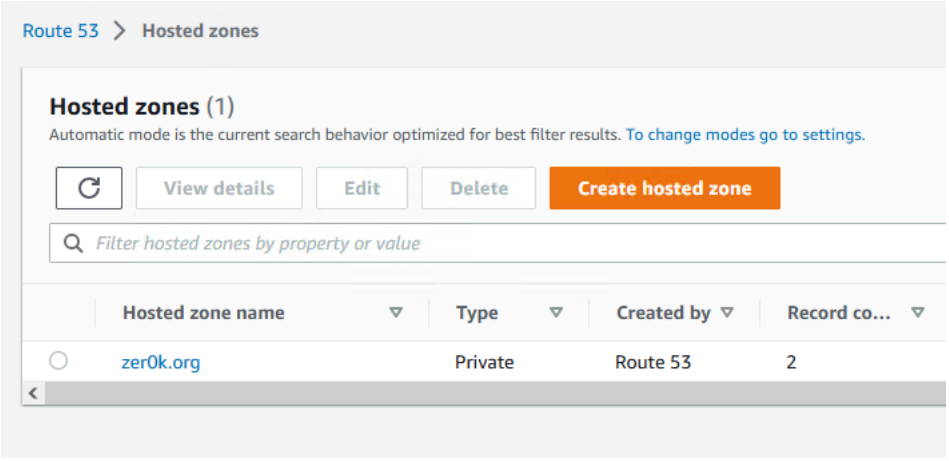

## Task 4: Configure DNS in Route 53

ISE requires DNS for many operations including initial database priming and any internode operations.

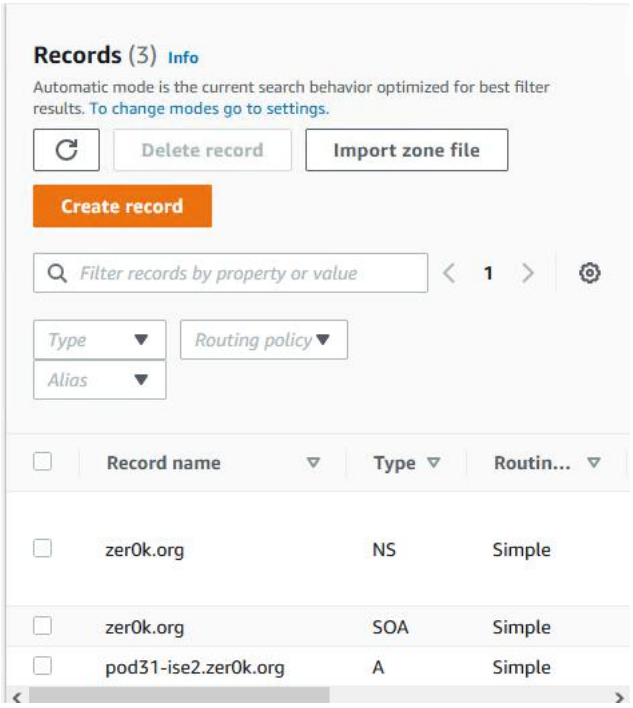
Table 4-1

| Pod | Primary Hostname | Primary IP Address | Secondary Hostname | Secondary IP Address |
|---|---|---|---|---|
| Pod 1 | pod1-ise1 | 10.1.0.5 | pod1-ise2 | 10.1.0.6 |
| Pod 2 | pod2-ise1 | 10.2.0.5 | pod2-ise2 | 10.2.0.6 |
| Pod 3 | pod3-ise1 | 10.3.0.5 | pod3-ise2 | 10.3.0.6 |
| Pod 4 | pod4-ise1 | 10.4.0.5 | pod4-ise2 | 10.4.0.6 |
| Pod 5 | pod5-ise1 | 10.5.0.5 | pod5-ise2 | 10.5.0.6 |
| Pod 6 | pod6-ise1 | 10.6.0.5 | pod6-ise2 | 10.6.0.6 |
| Pod 7 | pod7-ise1 | 10.7.0.5 | pod7-ise2 | 10.7.0.6 |
| Pod 8 | pod8-ise1 | 10.8.0.5 | pod8-ise2 | 10.8.0.6 |
| Pod 9 | pod9-ise1 | 10.9.0.5 | pod9-ise2 | 10.9.0.6 |
| Pod 10 | pod10-ise1 | 10.10.0.5 | pod10-ise2 | 10.10.0.6 |
| Pod 11 | pod11-ise1 | 10.11.0.5 | pod11-ise2 | 10.11.0.6 |
| Pod 12 | pod12-ise1 | 10.12.0.5 | pod12-ise2 | 10.12.0.6 |
| Pod 13 | pod13-ise1 | 10.13.0.5 | pod13-ise2 | 10.13.0.6 |
| Pod 14 | pod14-ise1 | 10.14.0.5 | pod14-ise2 | 10.14.0.6 |
| Pod 15 | pod15-ise1 | 10.15.0.5 | pod15-ise2 | 10.15.0.6 |
| Pod 16 | pod16-ise1 | 10.16.0.5 | pod16-ise2 | 10.16.0.6 |
| Pod 17 | pod17-ise1 | 10.17.0.5 | pod17-ise2 | 10.17.0.6 |
| Pod 18 | pod18-ise1 | 10.18.0.5 | pod18-ise2 | 10.18.0.6 |
| Pod 19 | pod19-ise1 | 10.19.0.5 | pod19-ise2 | 10.19.0.6 |
| Pod 20 | pod20-ise1 | 10.20.0.5 | pod20-ise2 | 10.20.0.6 |
| Pod 21 | pod21-ise1 | 10.21.0.5 | pod21-ise2 | 10.21.0.6 |
| Pod 22 | pod22-ise1 | 10.22.0.5 | pod22-ise2 | 10.22.0.6 |
| Pod 23 | pod23-ise1 | 10.23.0.5 | pod23-ise2 | 10.23.0.6 |
| Pod 24 | pod24-ise1 | 10.24.0.5 | pod24-ise2 | 10.24.0.6 |
| Pod 25 | pod25-ise1 | 10.25.0.5 | pod25-ise2 | 10.25.0.6 |
| Pod 26 | pod26-ise1 | 10.26.0.5 | pod26-ise2 | 10.26.0.6 |
| Pod 27 | pod27-ise1 | 10.27.0.5 | pod27-ise2 | 10.27.0.6 |
| Pod 28 | pod28-ise1 | 10.28.0.5 | pod28-ise2 | 10.28.0.6 |
| Pod 29 | pod29-ise1 | 10.29.0.5 | pod29-ise2 | 10.29.0.6 |
| Pod 30 | pod30-ise1 | 10.30.0.5 | pod30-ise2 | 10.30.0.6 |
| | | | | |

1. Search for “Route 53” in the AWS Console.
2. Under DNS Management, choose “Hosted zones”
3. Zer0k has already been registered so click on zer0k.org.



4. As part of the ISE provisioning, the Primary ISE node in your pod (10.x.0.5) will be provisioned into Route53. To help understand the task done via script, you will register the second node manually.
5. Click “Create record” and enter the following information (replace X with your pod number):
   a. Record name – podX-ise2
   b. Value – 10.X.0.6
6. Leave the rest of the values as default then click “Create records”.



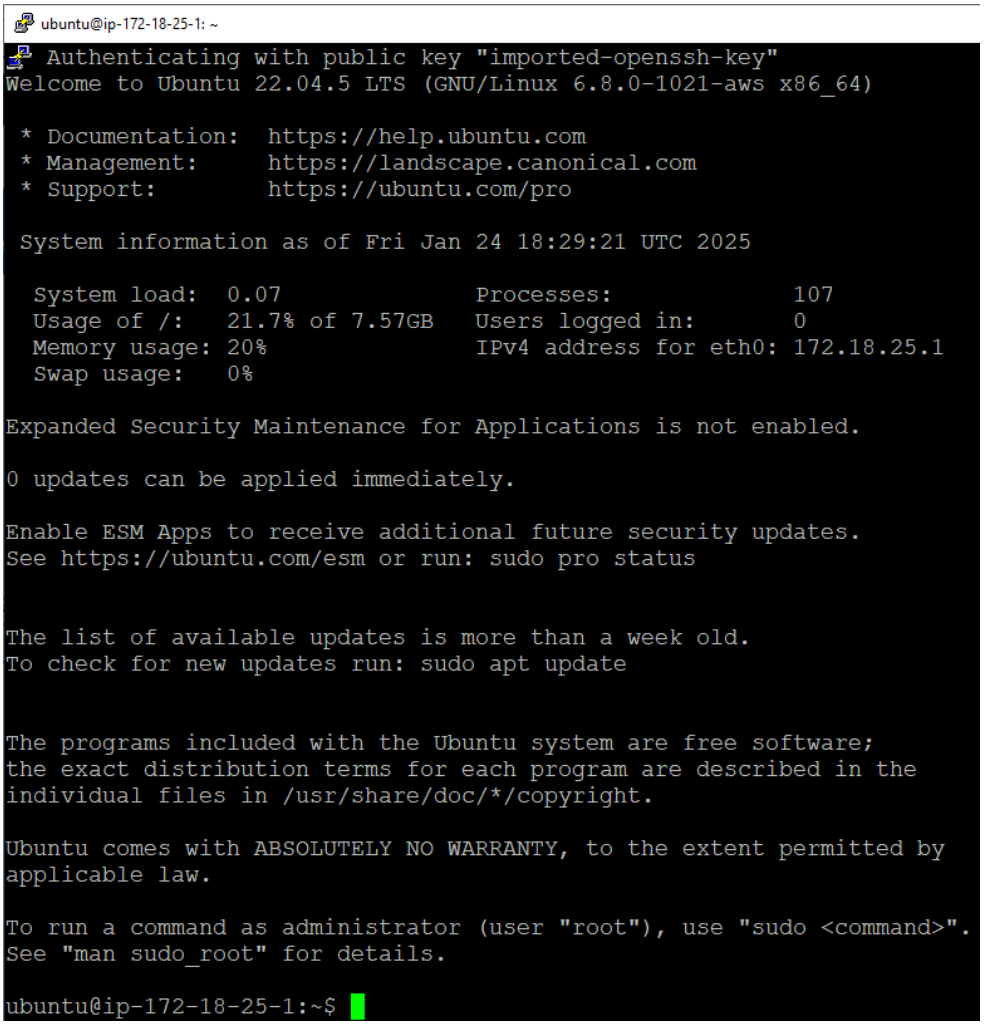

## Task 5: VPC Automation

Terraform will be used to provision ISE into AWS, and the required components which allows ISE to connect to the internet. It will maintain the state of the AWS resources and ISE instance, as well as allow for an easy tear down post-lab or in a real world scenario, should you use these instructions after Cisco Live!.

**Note**: In the following sections, blocks of text are used to install and configure the Ubuntu machine. We have found that on Mac and Linux devices, spacing sometimes suffers due to the PDF format of this guide. It is recommended that you copy commands from the guide into a simple text editor, such as notepad, if you are not typing them out, to remove any formatting issues.

### Step 1: Install Terraform

4. In the preceding step, you should have established an SSH session to the Ubuntu provisioning machine for your pod. The Ubuntu machine should be sitting at the default prompt after login.




2. Ubuntu uses apt for its package manager, first apt needs to be updated to get the most current package and version list from the default package repository:

```
sudo apt-get update
```

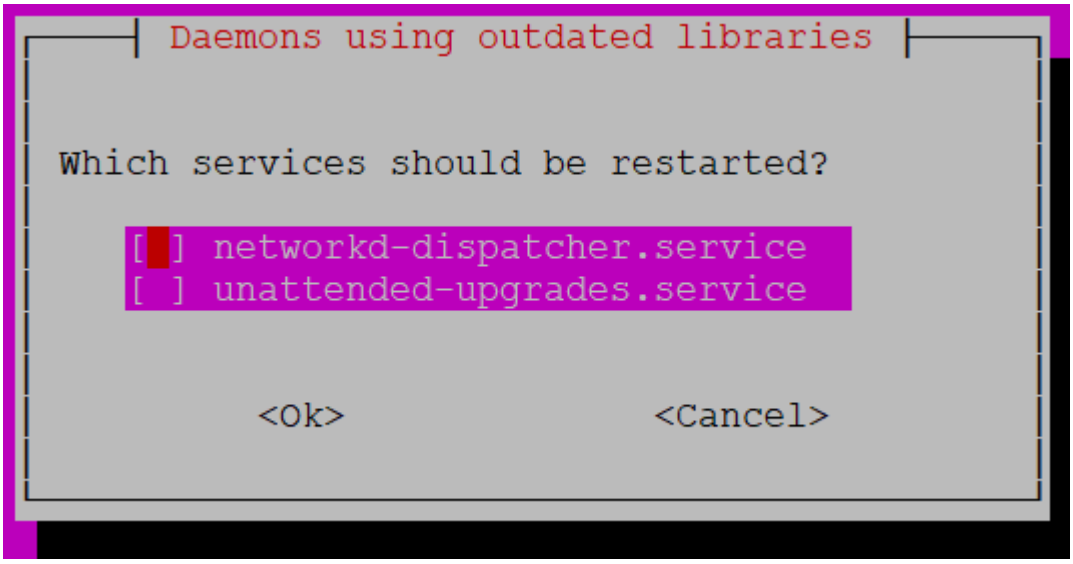
Note: If this step fails due to connectivity, check your security groups to ensure you didn't modify the outbound security group rule.

3. Upgrade the packages on the system to the most current versions:

```
sudo apt-get upgrade
```

Answer "Y" when shown how much space the upgrades will take.

4. You will be prompted to restart services that are using some of the packages that were updated. Leave the defaults selected and select (tab, enter) OK.




5. This lab will use a number of open-source libraries for Terraform. Install the GNU-PG Common Software Properties library by running:

```
sudo apt-get install -y gnupg software-properties-common
```

The package may already be up to date, at which point you would see the following screenshot. Otherwise, updates will be performed.

```
ubuntu@ip-172-18-25-1:~$ sudo apt-get install -y gnupg software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
software-properties-common is already the newest version (0.99.22.9).
software-properties-common set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
```

**Warning:** The following commands install Terraform from a custom repository, which is added from the hashicorp website. The website's GPG fingerprint is verified as part of these steps. If a typo is entered in the following commands, you'll need to delete and rerun the steps without the typo. See Appendix 2 for the procedure to resolve this error.

6. With Terraform being an open-source package, it is important to verify that the terraform package we're installing has not been tampered with. Download the Terraform package fingerprint with the following command (all one command):

```
wget -O- https://apt.releases.hashicorp.com/gpg | \
gpg --dearmor | \
sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null
```

```
--2025-01-24 18:39:53--  https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 108.156.184.65, 108.156.184.19, 108.156.184.5, ...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|108.156.184.65|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-                           100%[=====================================================>]   3.89K  --.-KB/s    in 0s

2025-01-24 18:39:53 (1.75 GB/s) - written to stdout [3980/3980]
```

7. Verify the fingerprint with the following command:

```
gpg --no-default-keyring \
--keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
--fingerprint
```

The following output should be presented indicating the fingerprint is valid, the signature in **bold** should match the screenshot until 2028:

```
gpg: directory '/home/ubuntu/.gnupg' created
gpg: /home/ubuntu/.gnupg/trustdb.gpg: trustdb created
/usr/share/keyrings/hashicorp-archive-keyring.gpg
------------------------------------------------
pub   rsa4096 2023-01-10 [SC] [expires: 2028-01-09]
      798A EC65 4E5C 1542 8C8E  42EE AA16 FCBC A621 E701
uid           [ unknown] HashiCorp Security (HashiCorp Package
Signing) <security+packaging@hashicorp.com>
sub   rsa4096 2023-01-10 [S] [expires: 2028-01-09]
```

8. Terraform is a custom repository, which can be thought of as a custom "app store". We'll need to add the repository to the set which Ubuntu uses to ensure Terraform can be downloaded. Run the following command:

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/hashicorp.list
```

9. We'll once again need to update our repositories to ensure Ubuntu pulls down the index of available packages, specifically so we get the latest version of Terraform. Run the following command:

```
sudo apt-get update
```

10. Install Terraform from the repository:

```
sudo apt-get install terraform
```

11. If asked which services should be restarted, repeat the same process as in the preceding steps and hit ***<Tab>, Enter*** to keep the defaults.
12. Once complete, run **terraform -help** to verify installation succeeded and Terraform is now an accessible package on your system.

```
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.
```

```
Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
<snip>
```

Congratulations, you now have your first orchestration package installed!

13. Terraform provides for tab completion as part of the package, which assists with ease of operation. Run the following commands to enable tab completion:

```
touch ~/.bashrc && terraform -install-autocomplete
```

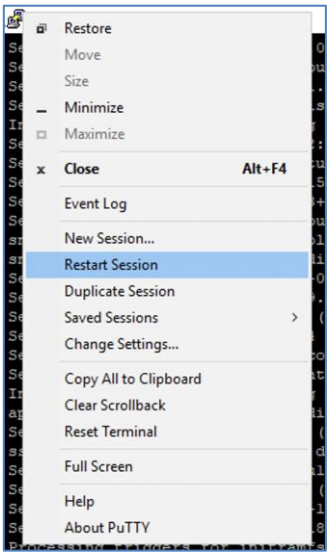
14. There are some services that were unable to be restarted individually, including the tab completion for terraform. Restart the system to get those services using the latest package versions:

```
sudo shutdown -r now
```

This will prevent further restart messages in the future.

Note: Now is the perfect time to stretch and get some water. It'll take up to 2 minutes for the virtual machine to start. In the meantime, you'll receive a "Connection Refused" error.

15. Restart the putty session by right clicking the putty icon on the window and selecting Restart Session. If it times out, try again until it connects, it could take a couple of minutes. If prompted, use username **ubuntu** once again.




## Step 2: Clone the GIT Repository for Script Execution

Clone the scripts to be used for execution from GIT into the production provisioning machine. These will be used to execute all tasks for the remainder of the lab. There isn't enough time in the lab for you to write the scripts but as you go through and execute them, take some time to read over the scripts first.

1. Verify you are in the home directory by executing ***pwd***, meaning "present working directory". You should be in the /home/ubuntu directory. If you are not, change directory with:

```
cd /home/ubuntu
```

2. Download (Clone) the GIT repository to the local machine. Execute the command:

```
git clone https://github.com/minimavus/LTRSEC-2000
```

The output should look similar to the following:

```
Cloning into 'LTRSEC-2000'...
...
Receiving objects: 100% (101/101), 165.20 KiB | 2.01 MiB/s, done.
Resolving deltas: 100% (32/32), done.
```

3. There should now be an 'LTRSEC-2000' directory in your home directory at this point, run ***ls -al*** to verify.

```
ubuntu@ip-172-18-25-2:~$ ls -al
total 44
drwxr-x--- 6 ubuntu ubuntu 4096 Jan 30 21:35 .
drwxr-xr-x 3 root   root   4096 Jan 30 21:05 ..
-rw------- 1 ubuntu ubuntu  655 Jan 30 21:34 .bash_history
-rw-r--r-- 1 ubuntu ubuntu  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3813 Jan 30 21:34 .bashrc
drwx------ 2 ubuntu ubuntu 4096 Jan 30 21:30 .cache
drwx------ 2 ubuntu ubuntu 4096 Jan 30 21:33 .gnupg
-rw-r--r-- 1 ubuntu ubuntu  807 Jan  6  2022 .profile
drwx------ 2 ubuntu ubuntu 4096 Jan 30 21:05 .ssh
-rw-r--r-- 1 ubuntu ubuntu    0 Jan 30 21:31 .sudo_as_admin_successful
-rw-rw-r-- 1 ubuntu ubuntu  181 Jan 30 21:32 .wget-hsts
drwxrwxr-x 5 ubuntu ubuntu 4096 Jan 30 21:35 LTRSEC-2000
```

4. Change directories into the LTRSEC-2000 directory and verify there is both an **Ansible** and **Terraform** directory:

```
cd LTRSEC-2000
```

```
ubuntu@ip-172-18-25-2:~/LTRSEC-2000$ ls -al
total 44
drwxrwxr-x 5 ubuntu ubuntu  4096 Jan 30 21:35 .
drwxr-x--- 6 ubuntu ubuntu  4096 Jan 30 21:35 ..
drwxrwxr-x 8 ubuntu ubuntu  4096 Jan 30 21:35 .git
-rw-rw-r-- 1 ubuntu ubuntu   110 Jan 30 21:35 .gitignore
drwxrwxr-x 6 ubuntu ubuntu  4096 Jan 30 21:35 Ansible
-rw-rw-r-- 1 ubuntu ubuntu 18456 Jan 30 21:35 README.md
drwxrwxr-x 2 ubuntu ubuntu  4096 Jan 30 21:35 Terraform
```

## Step 3: Configure the Terraform Environmental Variables to be Used

As part of the deployment process, multiple variables are used to ensure configurations are populated and access keys are available to Terraform and Ansible. This step configures variables for Terraform. Additional steps are found later in the lab to do the same for Ansible.

1. Change directory into your Terraform directory to modify your pod number and key pair. Run the following commands:

```
cd Terraform
ls -al
```

5. Verify the “variables.tf” file exists.

```
ubuntu@ip-172-18-25-1:~/LTRSEC-2000/Terraform$ ls -al
total 24
drwxrwxr-x 2 ubuntu ubuntu 4096 Jan 24 19:01 .
drwxrwxr-x 5 ubuntu ubuntu 4096 Jan 24 19:01 ..
-rw-rw-r-- 1 ubuntu ubuntu 4995 Jan 24 19:01 main.tf
-rw-rw-r-- 1 ubuntu ubuntu  263 Jan 24 19:01 userdata.tftpl
-rw-rw-r-- 1 ubuntu ubuntu 2155 Jan 24 19:01 variables.tf
```

6. Access the variables.tf file by running:

```
nano variables.tf
```

7. Change the pod number in the variables file from the default of "<TBD>" to your pod number. Replace the <TBD> and keep the quotation marks.

```
variable "pod_number" {
      description = "Assigned Pod Number"
      type = string
      default = "X"
}
```

8. Change the keypair that will be used to access ISE to "podX-keypair" where X is your pod number.

```
variable "pod_keypair" {
    description = "Key Pair Created By the Student"
    type = string
    default = "podX-keypair
}
```

9. Save the file by using the key combination Ctrl+X, answer "Y" to save the configuration, and press ***Enter*** to confirm we'll be saving to the current file.
10. To prevent malicious actors from accessing the lab with what we publish to git, there is one file that needs to be created by the user with sensitive information such as the access key and secret key. While in the Terraform working directory, run the command:

```
nano terraform.tfvars
```

This command will create a new file call terraform.tfvars.

11. Paste the following into terraform.tfvars, populating your access key, secret key, and password accordingly:

```
#AWS keys (not shared with github)
aws_access_key = "<YOUR ACCESS KEY>"
aws_secret_key = "<YOUR SECRET KEY>"
ise_password = "<INITIAL ISE PASSWORD ENDING WITH !>"
```

Maintain all quotation marks and ensure they are straight quotation marks, not curly. Replace the access key and secret key with the values you saved from the Ubuntu machine creation. Enter the password provided by the proctors.

12. Save the file by using the key combination Ctrl+X, answer "Y" to save the configuration, and press Enter to confirm we'll be saving to the current file.

## Step 4: Deploy the AWS components, and Identity Services Engine

1. Explore what the Terraform script will do by using the ***more main.tf*** command. Some areas to focus on within the script are the following:
    a. Provider “aws”
    b. #Create Pod VPC
    c. #Create Private Subnet
    d. #Create Transit Gateway Attachment
    e. #Create Private Route Table
    f. #Attach the Route Table to the Subnet
    g. #Create Security Group for ISE
    h. #Get AMI from Market Place for which ever region is being used
    i. #Create a DNS Entry for the Node
    j. #Create the ISE Instance

2. We’ll now run the terraform script, but need to first initialize terraform to pull in the files and tasks it’ll need to perform. Run the command:

```
terraform init
```

to initialize terraform.

```
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.84.0...
- Installed hashicorp/aws v5.84.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

3. We’ll now want to verify our “plan”, or what Terraform will do before finally executing on it. Run the command:

```
terraform plan
```

to see what changes will be made. Verify these plans in accordance with step 2:
    a. # aws_vpc.pod-vpc will be created
    b. # aws_subnet.pod-private-subnet will be created
    c. # aws_security_group_rule.allow_tls_ipv4 will be created
    d. # aws_security_group_rule.allow_ssh_ipv4 will be created
    e. # aws_security_group.ise-security-group will be created
    f. # aws_route_table_association.a will be created
    g. # aws_route_table.pod-private-rt will be created

h. # aws_route53_record.www will be created
i. # aws_instance.ise-instance will be created
j. # aws_ec2_transit_gateway_vpc_attachment.zer0k-transit-gateway will be created

The plan should end with the following verbiage:

```
Plan: 14 to add, 0 to change, 0 to destroy.
```

Note: If you forgot to add in the access key, secret key, or ISE password, you will be prompted at this point to add them in. This dialog will look like the following:

```
ubuntu@ip-172-18-25-1:~/LTRSEC-2000/Terraform$ terraform plan
var.aws_access_key
  Enter a value: AKIAWFIN7EPKC5YH4OLP

var.aws_secret_key
  Enter a value: 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

var.ise_password
  Enter a value: CLUS2023Party!
```

Note: You'll find that many of the configuration attributes associated with ISE and AWS are human readable within the output that Terraform returns. We encourage you to read through and understand what is being applied and compare it to the network diagram at the beginning of the lab. ISE will take up to 30 minutes to install and initialize, during which is a great time to develop your understanding.

4. You may now apply the terraform configurations with the apply command, this will create the resources into AWS. Run the following command:

```
terraform apply
```

Answer ***yes*** when prompted whether you want to perform these actions.

5. When successfully applied, the script should end with:

```
Apply complete! Resources: 14 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-053350f3ff9514710"
instance_private_ip = "10.2.0.5"
ise_password = "<Provided by your proctor>"
pod-key-name = "Creating Instance with Key Name: podX-keypair"
pod_number = "X"
```

Copy these values to your notepad for use in the next task.

## Step 5: Explore the AWS Environment to Ensure the Network Environment Was Provisioned

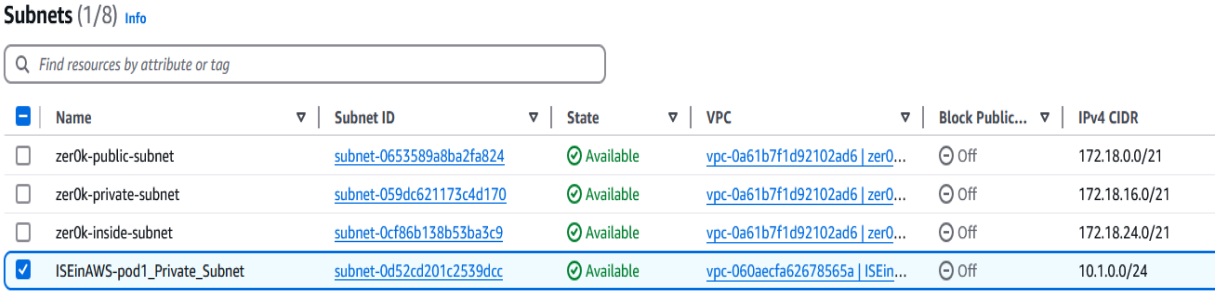
With the scripts being successfully run it is time to verify that they created and configured the environment as expected.

1. Verify the VPC was deployed. Navigate to the search bar in AWS, type

VPC. Click VPC in the results and navigate to “Your VPC’s”. The VPC should be named ISEinAWS-podX where X is your assigned pod number.

| | | | | |
|---|---|---|---|---|
| ☐ | zer0k-pod0 | vpc-0105fb3ba3f6cbec3 | Available | 10.0.0.0/16 |
| ☐ | ISEinAWS-pod9 | vpc-0dbba522e2e80f740 | Available | 10.9.0.0/16 |
| ☑ | ISEinAWS-pod10 | vpc-06c27a7f1c1c05e39 | Available | 10.10.0.0/16 |
| ☐ | zer0k-main-vpc | vpc-0a61b7f1d92102ad6 | Available | 172.18.0.0/16 |

2. Navigate to Subnets in the left menu. Verify the Private subnet is deployed in accordance with your pod. It should be named ISEinAWS_podX_Private_Subnet where X is your assigned pod number.

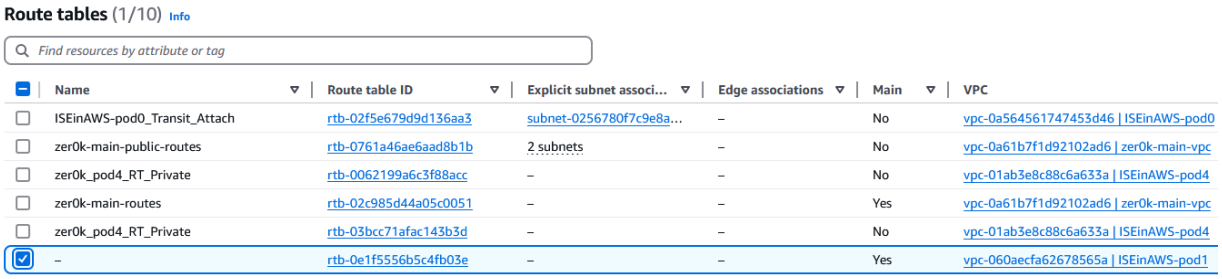


**Subnets** (1/8) Info

Find resources by attribute or tag

| | Name | Subnet ID | State | VPC | Block Public... | IPv4 CIDR |
|---|---|---|---|---|---|---|
| ☐ | zer0k-public-subnet | subnet-0653589a8ba2fa824 | Available | vpc-0a61b7f1d92102ad6 \| zer0... | Off | 172.18.0.0/21 |
| ☐ | zer0k-private-subnet | subnet-059dc621173c4d170 | Available | vpc-0a61b7f1d92102ad6 \| zer0... | Off | 172.18.16.0/21 |
| ☐ | zer0k-inside-subnet | subnet-0cf86b138b53ba3c9 | Available | vpc-0a61b7f1d92102ad6 \| zer0... | Off | 172.18.24.0/21 |
| ☑ | ISEinAWS-pod1_Private_Subnet | subnet-0d52cd201c2539dcc | Available | vpc-060aecfa62678565a \| ISEin... | Off | 10.1.0.0/24 |

3. Verify the subnet allocated to your pod is in the form of 10.X.0.0/24 where X is your pod number.
4. Navigate to Route Tables in the left menu. Verify a routing table was provisioned for your pod. Under the column “VPC” a route table should be named ISEinAWS-podX where X is your assigned pod number.



**Route tables** (1/10) Info

Find resources by attribute or tag

| | Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC |
|---|---|---|---|---|---|---|
| ☐ | ISEinAWS-pod0_Transit_Attach | rtb-02f5e679d9d136aa3 | subnet-0256780f7c9e8a... | – | No | vpc-0a564561747453d46 \| ISEinAWS-pod0 |
| ☐ | zer0k-main-public-routes | rtb-0761a46ae6aad8b1b | 2 subnets | – | No | vpc-0a61b7f1d92102ad6 \| zer0k-main-vpc |
| ☐ | zer0k_pod4_RT_Private | rtb-0062199a6c3f88acc | – | – | No | vpc-01ab3e8c88c6a633a \| ISEinAWS-pod4 |
| ☐ | zer0k-main-routes | rtb-02c985d44a05c0051 | – | – | Yes | vpc-0a61b7f1d92102ad6 \| zer0k-main-vpc |
| ☐ | zer0k_pod4_RT_Private | rtb-03bcc71afac143b3d | – | – | No | vpc-01ab3e8c88c6a633a \| ISEinAWS-pod4 |
| ☑ | – | rtb-0e1f5556b5c4fb03e | – | – | Yes | vpc-060aecfa62678565a \| ISEinAWS-pod1 |

5. Verify a Transit Gateway Attachment exists for your pod. Under the Column “Name” the transit attachment should be named “ISEinAWS-podX_Transit_Attach” where X is your pod number.

## Step 6: Install Ansible and Modify Variables for Your Pod

While ISE initializes now is a good time to get started on the next task, while Terraform is great at deploying resources, Ansible is better at configuration, let’s get Ansible installed and prepared.

1. Change your directory back to the home directory with *cd /home/ubuntu*.
2. This lab will use a python virtual environment. Venv is not installed by default so install it:

```
sudo apt-get install python3.10-venv
```

A virtual python environment allows for different projects on the same machine to use different versions of shared python libraries. Answer ***Y*** when prompted to install.

3. Create a virtual environment for this lab, this will create a new folder in the /home/ubuntu directory:

```
python3 -m venv ISEonAWS
```

4. Activate the newly created ISEonAWS virtual environment by running the activation script:

```
source ISEonAWS/bin/activate
```

This script sets up environment variables on the system to use the activated virtual environment. You should see the linux prompt prepended with (ISEonAWS) at this time.

Note: If you log out of Ubuntu, you'll need to repeat the command ***source ISEonAWS/bin/activate*** to get back into your virtual environment.

5. Install the python packages required to run ansible for ISE:

```
python3 -m pip install ansible boto boto3 botocore ciscoisesdk jmespath
```

This step will take a few minutes.

## Step 7: Modify your ansible Variables

1. Your current working directory, or directory that you are within should be /home/ubuntu/. Verify this by running the command ***pwd*** meaning "present working directory".

```
ubuntu@ip-172-18-25-1:~/LTRSEC-2000/Terraform$ pwd
/home/ubuntu/
```

2. Change directories into Ansible using command

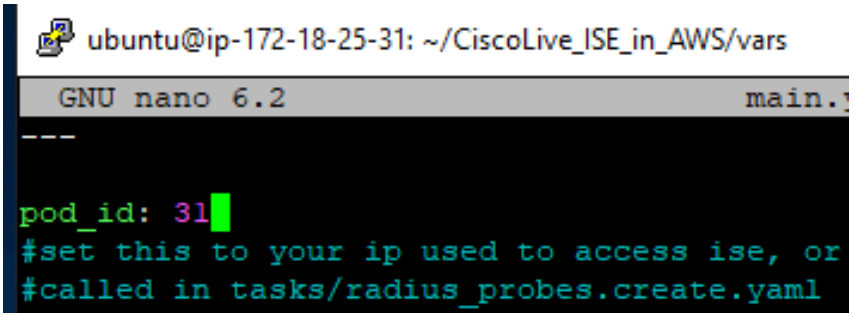
```
cd LTRSEC-2000/Ansible
```

3. Ansible's variables exist in the "vars" directory. Edit the vars/main.yaml file to update your pod number, which will be utilized when we configure ISE in later steps. Refer to the output from the terraform apply command in your notepad if you forget any of these values. Execute the commands:

```
cd vars
nano main.yaml
```

4. Edit the line in main.yaml ***pod_id:*** to match your pod number.

> **Note:** Update the pod_id: to only have the pod number in it, as seen in the following screenshot. This is a variable used throughout the lab, and only takes in a numerical value.




5. Scroll down in main.yaml to ensure the “ise_username” variable is set to ***iseadmin***. If it is not, edit it. Do not change any other variables at this time.
6. Set the ise_password to the *updated* ISE password, which ends with a $. This password is provided by your proctor.

> **Note:** Because of the order of operations within the lab, Terraform will deploy ISE with an initial password ending in ! This is expected. ISE will require a password reset before Ansible is used to configure users, groups, and network access devices. This password will end with $. Both passwords are printed on your information card for your pod.

7. Exit and save from nano utilizing the key sequence “Ctrl-X, Y, <enter>”
8. Once back at the Ubuntu command line, replace the access key and secret key with the one you created in Task 1, Step 5. These commands can also be found in the file “PodX Important Information.txt” in the lab material. It should look like the following:

```
export AWS_REGION=us-east-2
export AWS_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXX <----Your Access Key Here
export AWS_SECRET_KEY=YYYYYYYYYYYYYYYYYYYY <----Your Secret Key Here
export ise_verify=false
```

> ***Note:*** Ensure there are no spaces in any of the exported variables.

> ***Note:*** If you get logged out of Ubuntu, you’ll need to repeat the above export of variables, as they are local to your session.

9. Paste the export commands block of text into the Ubuntu CLI, line by line.

**Note:**

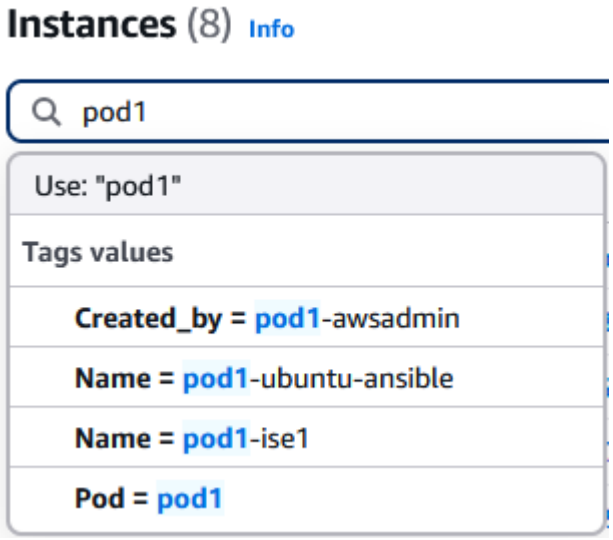
- Right click in PuTTY will paste commands, do not use ctrl+v.
- Do not paste the entire block, paste line by line to ensure no spaces are added.
- Ensure there are no spaces before or after any of the variable names.

10. Verify by running the ***env*** command:

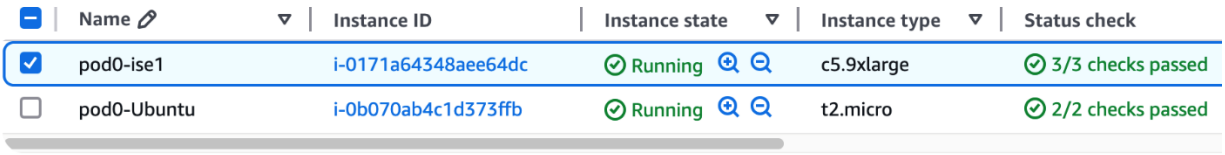
```
ubuntu@ip-172-18-25-1:~/LTRSEC-2000/Ansible/vars$ env
SHELL=/bin/bash
AWS_ACCESS_KEY=<YOUR ACCESS KEY>
AWS_SECRET_KEY=<YOUR SECRET KEY>
AWS_REGION=us-east-2
PWD=/home/ubuntu/LTRSEC-2000/Ansible/vars
```

## Step 8: Monitor the Environment and ISE Deployment

6. ISE will take up to 30 minutes to initiate. Depending on how long it took to install Ansible, you may have time before ISE is fully deployed, during which time you can monitor it from within the AWS GUI. Within your web browser, navigate back to EC2 by using the search bar on top of the AWS page.
7. Create a filter in EC2 for your pod instances, enter “podX” where x is your pod number in the search bar to filter down to just your pod instances.

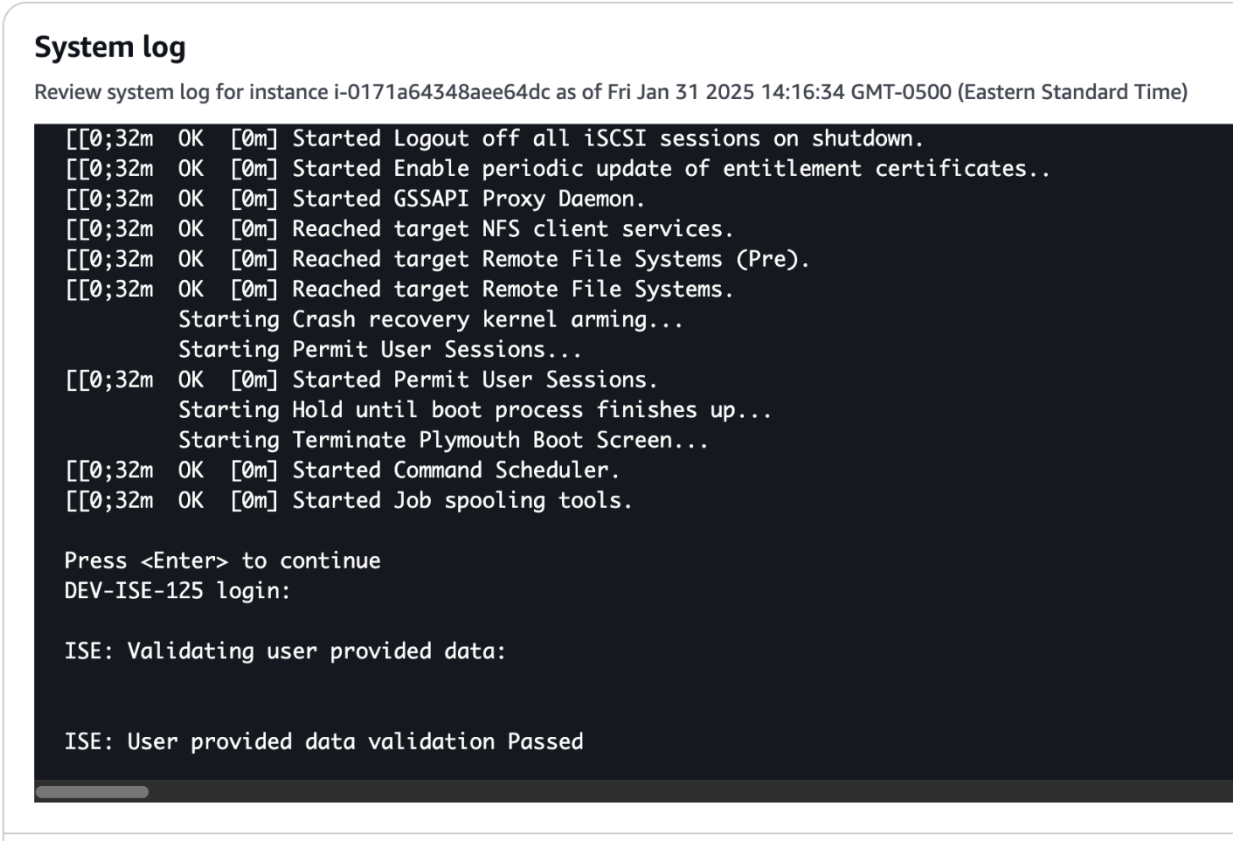



8. Check the status of podX-ise1, wait until it has completed 3/3 checks before proceeding to the next step. The node will need to cycle through initializing, ½ checks passed, and 3/3 checks passed before being pingable.



| Name | Instance ID | Instance state | Instance type | Status check |
|---|---|---|---|---|
| pod0-ise1 | i-0171a64348aee64dc | Running | c5.9xlarge | 3/3 checks passed |
| pod0-Ubuntu | i-0b070ab4c1d373ffb | Running | t2.micro | 2/2 checks passed |

9. You can also monitor the state of ISE as it comes up through the System log. Check the box next to your ISE instance then in the top right go to Actions -> Monitor and troubleshoot -> Get system log. This is helpful to ensure that the user_data provided to the instance is correct.

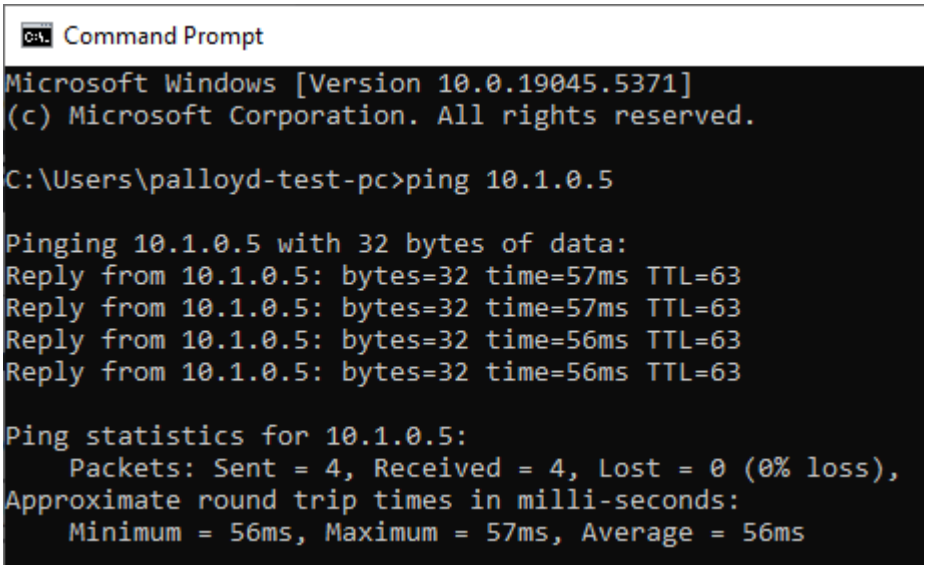



10. The serial console can also be used by following the procedure found in “Appendix 1: Using the EC2 Serial Connect to Monitor”.

## Step 9: Establish an SSH session with ISE

***Note: ISE will need to initialize and can take up to 30 minutes to fully deploy. You can verify the status of your ISE instance in EC2 -> Instances. In the Status check header in the EC2 dashboard, the status for ISE must have all checks passed, as opposed to “initializing”***


1. From the **Lab Windows PC** open a command prompt and attempt to ping ISE. ISE’s IP address should be in the form of 10.X.0.5 where X is your pod number.

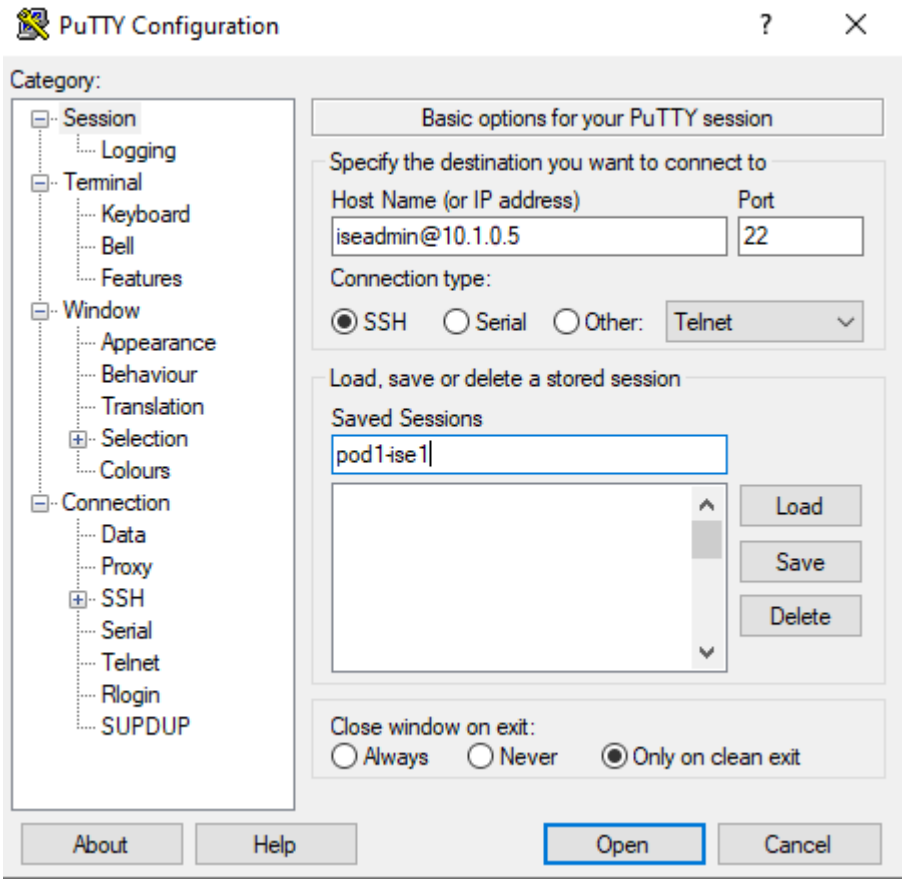



2. In the following steps, we'll establish an SSH session with ISE to verify services transition through a Not Running -> Initializing -> Running cycle. This should be done from the lab.

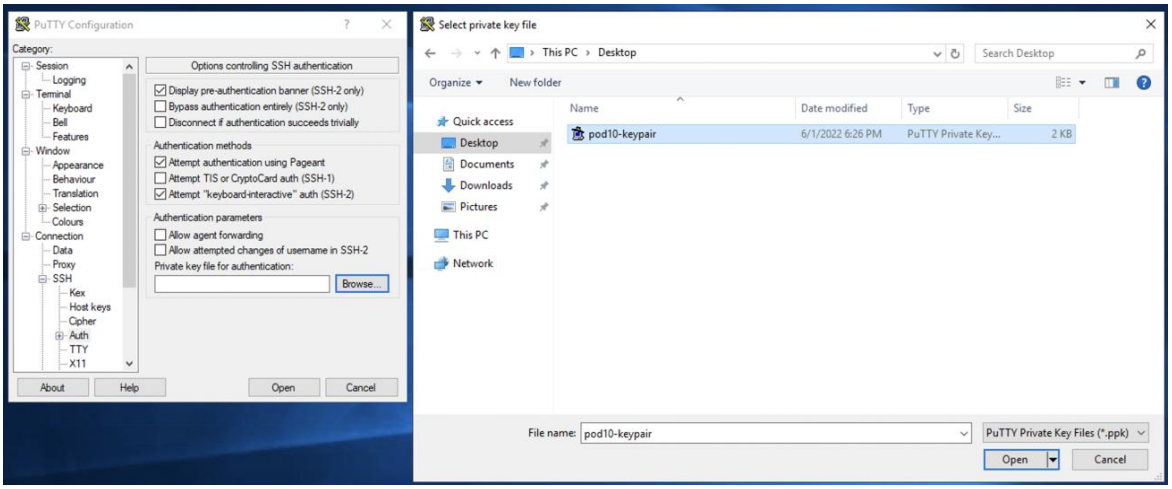
**NOTE:** SSH will fail until ISE is fully provisioned and running. Within this process the ISE server is provisioned, the database primed, the ISE node restarted, and only then is the application server and SSH process available. This can take up to ***30 minutes***. Progress can be monitored in the AWS console within the EC2 Instances area. This will manifest itself as:

"(ISEonAWS) ubuntu@ip-172-18-25-4:~ $ ssh -i ~/.ssh/ISEinAWS-pod4.pem iseadmin@10.4.0.5
admin@10.4.0.5: **Permission denied** (publickey)."

3. On the Lab Windows PC, navigate back to the desktop and open Putty from the shortcut.
4. The default screen is under the Session category. Set the Hostname to "iseadmin@10.X.0.5" and Save the Session as "podX-ise1" where X is your assigned pod number. Click the save button.




5. Navigate to Connection -> SSH -> Auth -> Credentials. In the “Private key file for authentication” box at the bottom of the screen, select the private key created from the PEM file earlier in the lab. It should have been named podX-keypair.ppk located on the desktop from the previous preparation step.

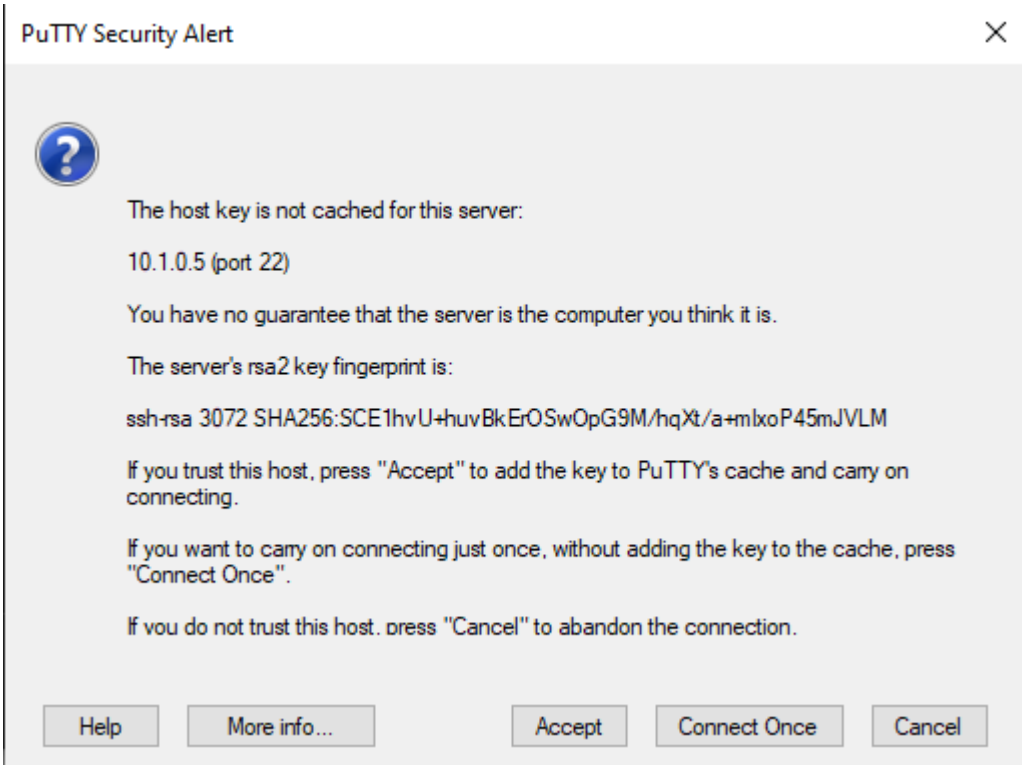



6. Optional: Navigate to Session -> Logging, select “All Session Output”, browse, and place the log file in the desktop. This is strictly for future troubleshooting if desired.
7. Do not change any additional settings.

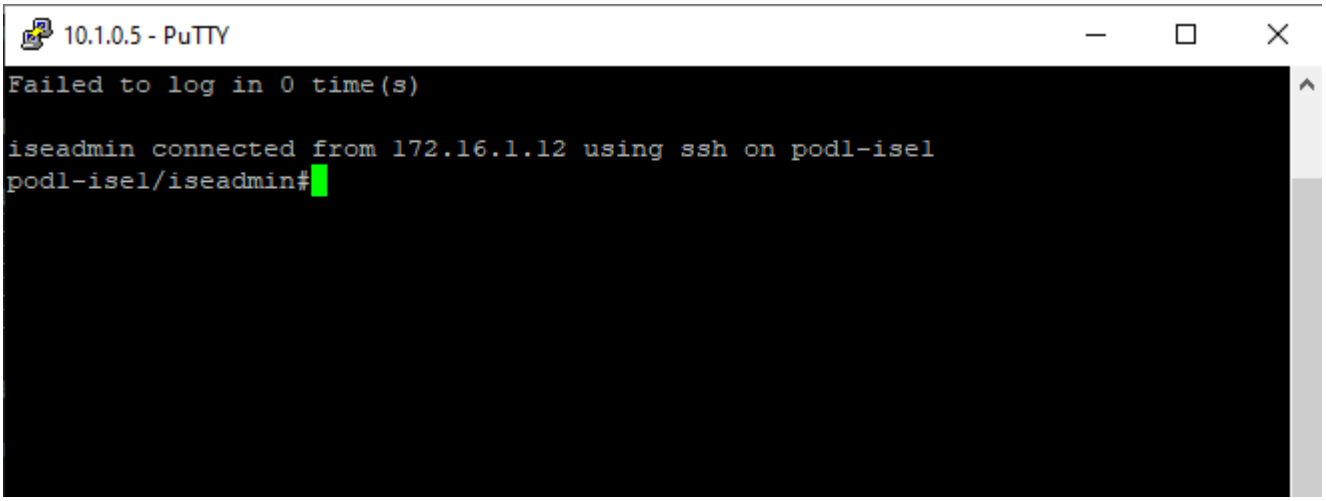
8. Go back to Session and save again.

9. Click Open.

10. If presented with a Putty Security Alert, accept the fingerprint.




11. You should now have access to the ISE console.




12. ***Optional:*** If you are unable to SSH to ISE and receive an error of **Permission denied** (publickey), ISE may still be provisioning. A procedure to connect via the Amazon EC2 Serial Console is provided in Appendix 1 to observe the provisioning process.

13. ISE will need to be in a running state to access the GUI. The Application server will need to be in a running state before this is the case.

    Run the following command:

```
show application status ise
```

```
podXX-ise/admin# show application status ise
ISE PROCESS NAME                       STATE            PROCESS ID
------------------------------------------------------------------
---------
Database Listener                      running          32895
Database Server                        running          85 PROCESSES
Application Server                     not running
...
Application Server                     initializing     49728
...
Application Server                     running
```

The application server will transition through each of the states of not running, initializing, and running as seen in the snippit above.

14. Once the Application Server shows as “running” move on to the next step.

## Step 10: Verify the running configuration of ISE to align with your Pod

Verify the running configuration, including IP address for the private facing interface, aligns with expected IP’s for your pod. Also verify DNS is provisioned to the correct server of 169.254.169.253

Execute the command:

```
show run
```

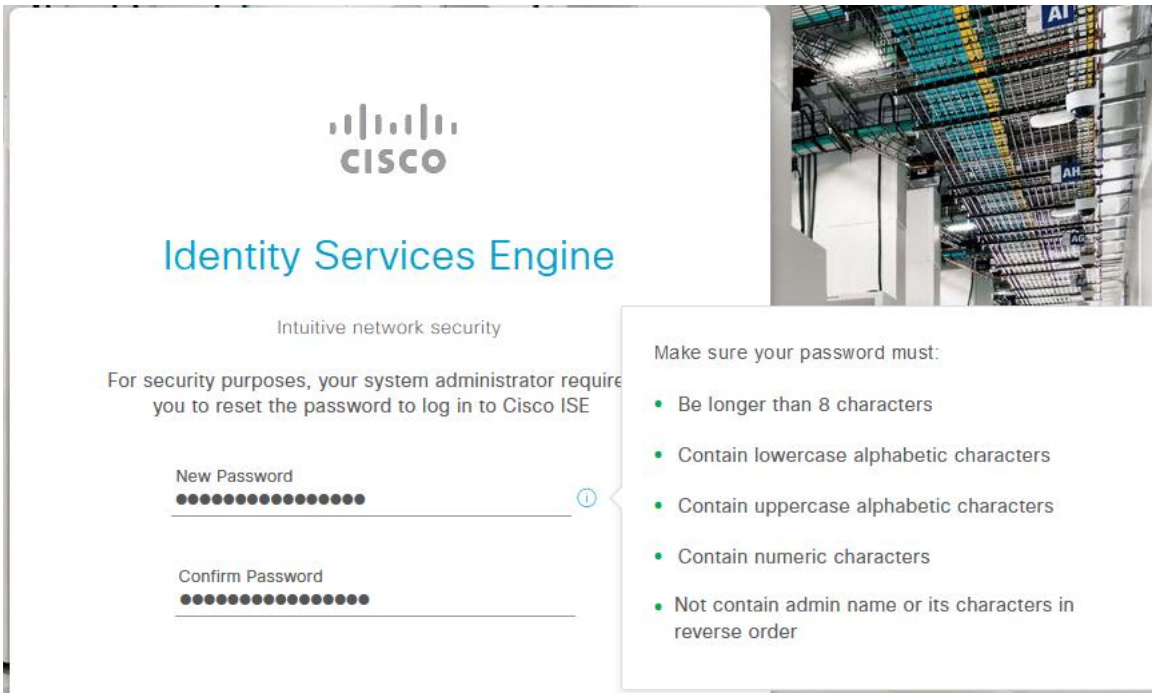
```
ise/admin# show run

Expected Output:
Generating configuration...
!
hostname ise
ip domain-name zer0k.org
interface GigabitEthernet 0
  ip address 10.X.0.5 255.255.255.0
  !
ip name-server 169.254.169.253
ip default-gateway 10.X.0.1
clock timezone UTC
!
```

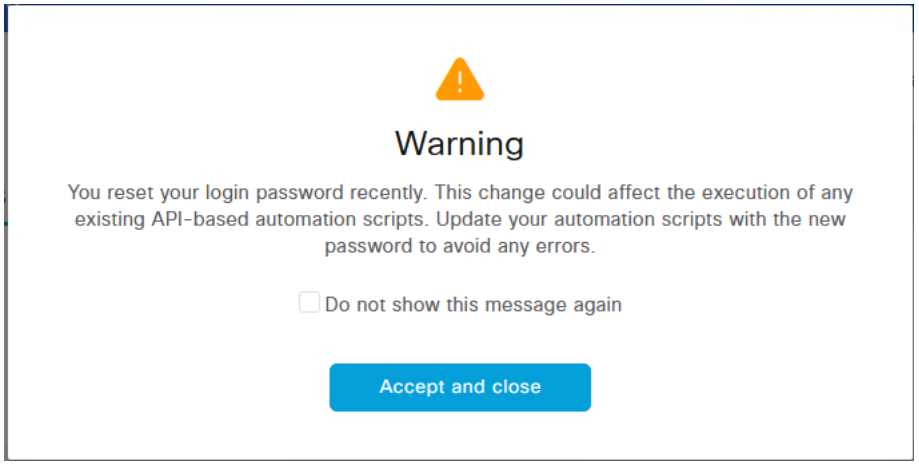
## Step 11: Reset the ISE Password

ISE forces the user to change their password on first login. Default settings of requiring complex passwords prevent the user from resetting their password back to the same password used. The password therefore needs to change. In a previous step, a note was presented indicating a different password should be used in the Ansible variables file. This password will be what you change the GUI password to as part of this step.

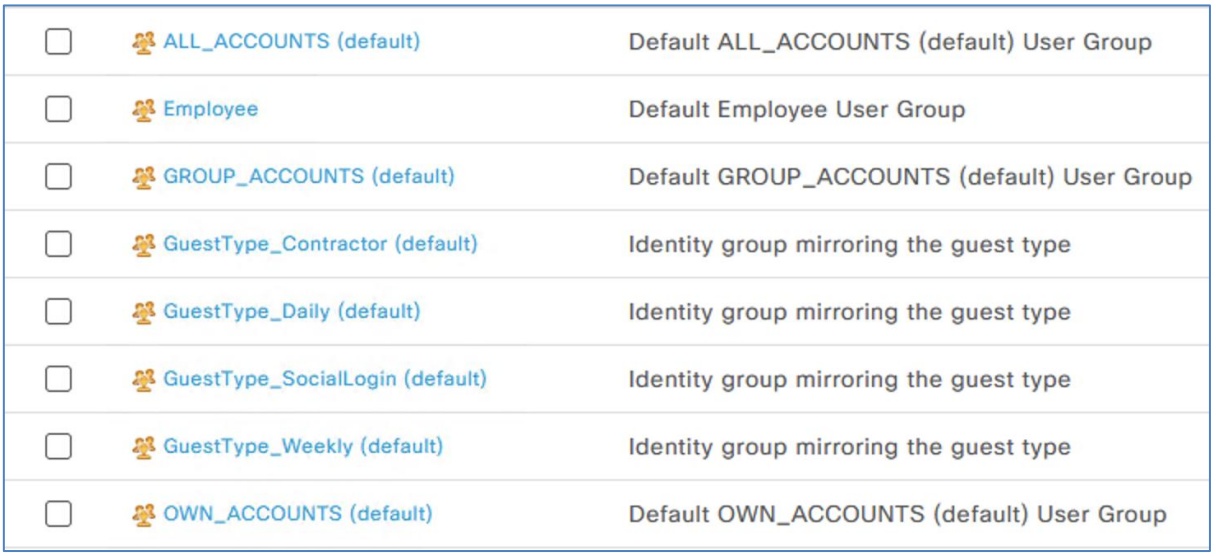
1. Launch a web browser to ISE (you will need to populate the full URL of https://10.X.0.5) from your Windows PC, accepting any certificate errors. You will be prompted to reset the ISE GUI password. Reset the password as prompted to the updated password ending in $, or the **Updated password provided by your proctor**.




2. You'll be logged out of ISE. Log back into ISE using the administrative username **iseadmin,** using the updated password.
3. If presented, accept the warning about resetting the password, noting this would be a concern for this lab.




4. Using the hamburger menu on the top left of the screen, navigate to Administration -> Identity Management -> Groups -> User Identity Groups
5. Verify only default ISE groups are populated, including OWN_ACCOUNTS, ALL_ACCOUNTS




6. Navigate to Administration -> Identity -> Identities
7. Verify there are no users present in the identity store
8. Navigate to Administration -> Network Device Groups
9. Verify only default ISE NDG's are present.


10. Navigate to Administration -> Network Devices
11. Verify no Network Devices are currently present.

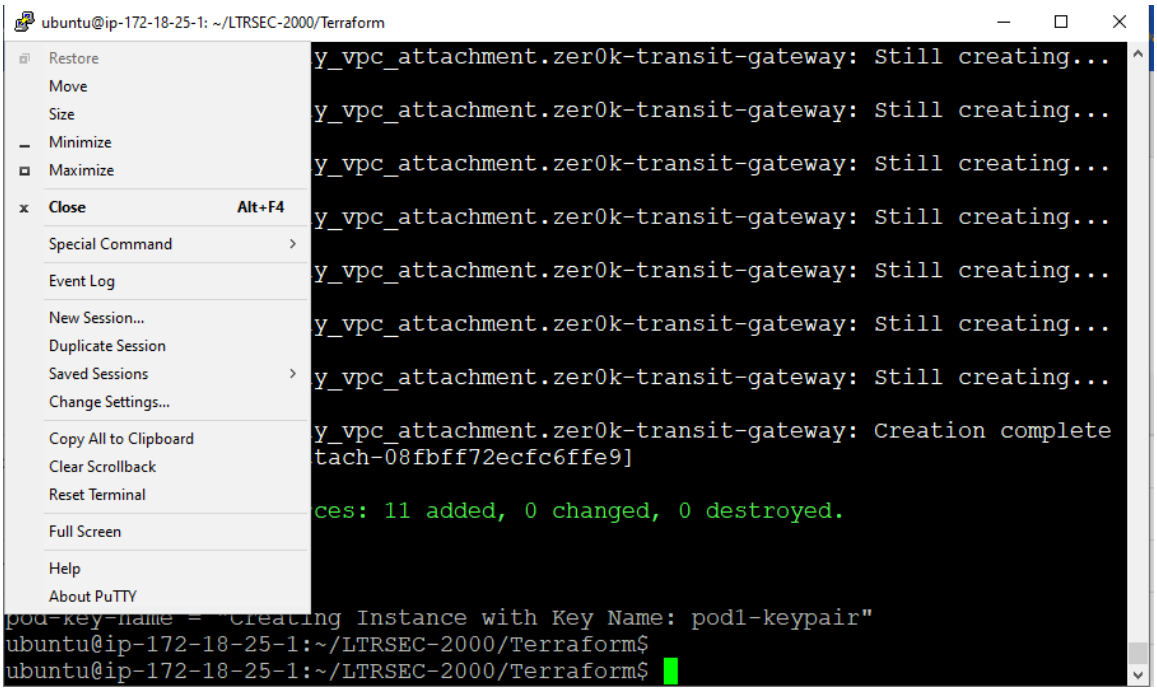
## Step 12: Configure ISE Programmatically with Ansible

**Note:** Should the following steps be performed outside of the lab or across regions, ensure that the correct IP is used for the ise.configuration.yaml script. If not, the script will perform 1000 pings before timing out.

If this happens it can be interrupted with ctrl+c.

**Warning:** If you did not complete step 5.9.6, the following will fail with an error indicating authentication is required.

1. Open the SSH session to the Ubuntu Ansible machine. You may need to reset the session using the icon in the top left of the window.




2. Navigate back to base Ansible directory with:

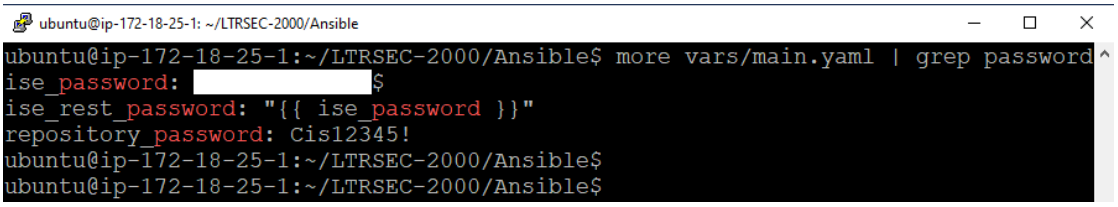
```
cd /home/ubuntu/LTRSEC-2000/Ansible
```

***3.*** Ensure you are running within the Python virtual environment for Ansible, as indicated by (ISEonAWS) preceding your machine's hostname. If not, run the command

```
source /home/ubuntu/ISEonAWS/bin/activate
```

***4.*** Verify the ISE password used for Ansible is the updated password ending in $. Run the command

```
more vars/main.yaml | grep password
```




5. Configure ISE from the Ansible virtual environment, to populate the Network Device Groups, Network Devices, User Roles, and Users. Run these commands from the Ubuntu-Ansible machine.

   Execute the command:

```
ansible-playbook ise.configuration.yaml
```

***Expected Output:***

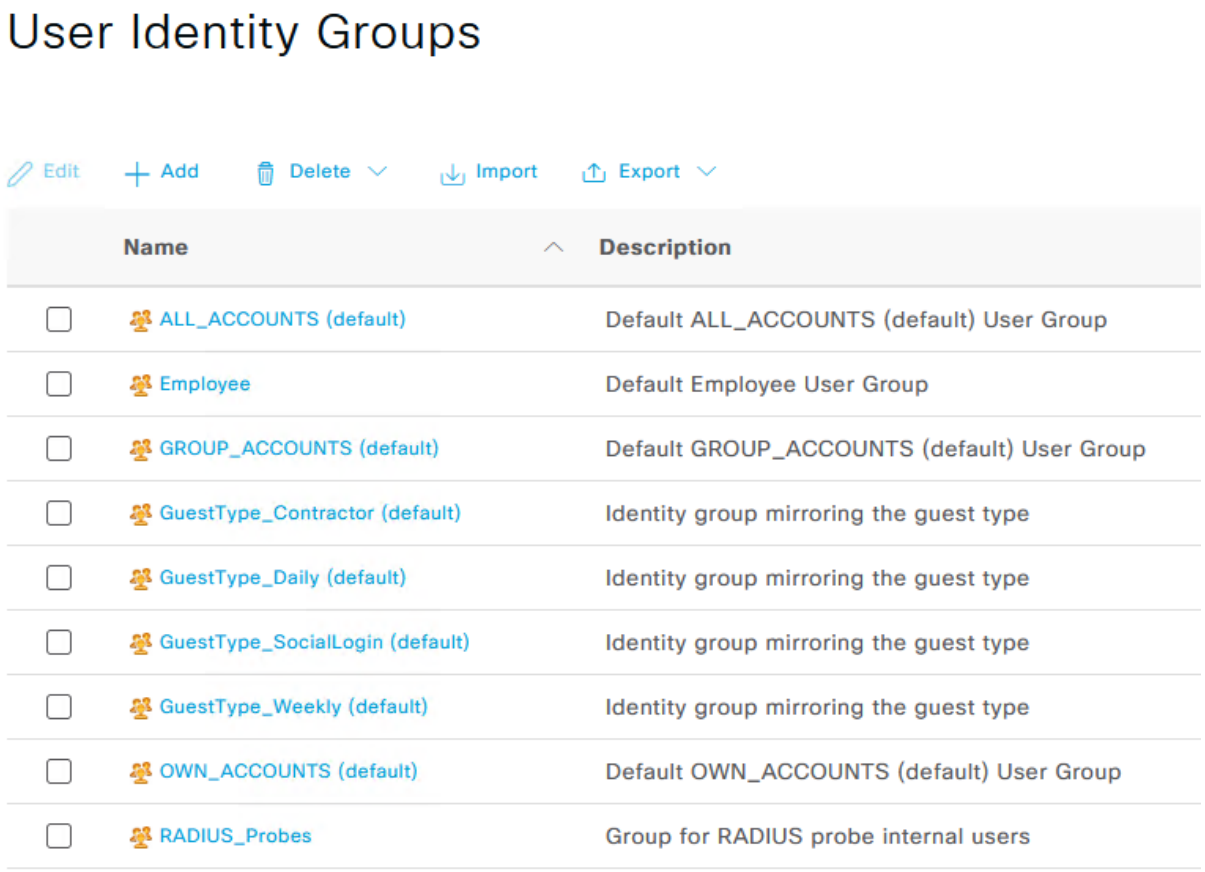
```
PLAY [ISE Configuration Playbook]
***
TASK [Query for ISE instances in project "'ISEinAWS-palloyd'"] ...
TASK [Show instances] ...
TASK [Test for ISE Application Server Initialization] ...
TASK [Ping 10.X.0.5] ...
TASK [Wait for <private_IP> App Server (GUI)] ...
TASK [Show <private_IP> Initialized] ...
TASK [Enable ISE ERS & OpenAPIs] ...
TASK [Enable ISE OpenAPIs (ISE 3.1+)] ...
TASK [Show ISE OpenAPIs Enabled Status] ...
TASK [Show ISE OpenAPIs Disabled Status] ...
TASK [Get ISE ERS APIs Status] ...
TASK [Enable ise.zer0k.org ERS APIs] ...
TASK [Show ise.zer0k.org ERS Enabled Status] ...
TASK [Show ise.zer0k.org ERS Disabled Status] ...
TASK [Create RADIUS Probes - identity_group and internal_users] ...
TASK [Create `RADIUS_Probes` identity group] ...
TASK [Create Internal Users] ...
TASK [Create Internal User Accounts] ...
TASK [Create Network Device Groups] ...
TASK [Create demo network_devices] ...
TASK [Include Endpoints] ...
TASK [Create Endpoints] ...
PLAY RECAP
localhost                  : ok=39   changed=6    unreachable=0    failed=0
skipped=4    rescued=0    ignored=0
```

## Step 13: Verify Newly Configured Users, Groups, and Network Access Devices

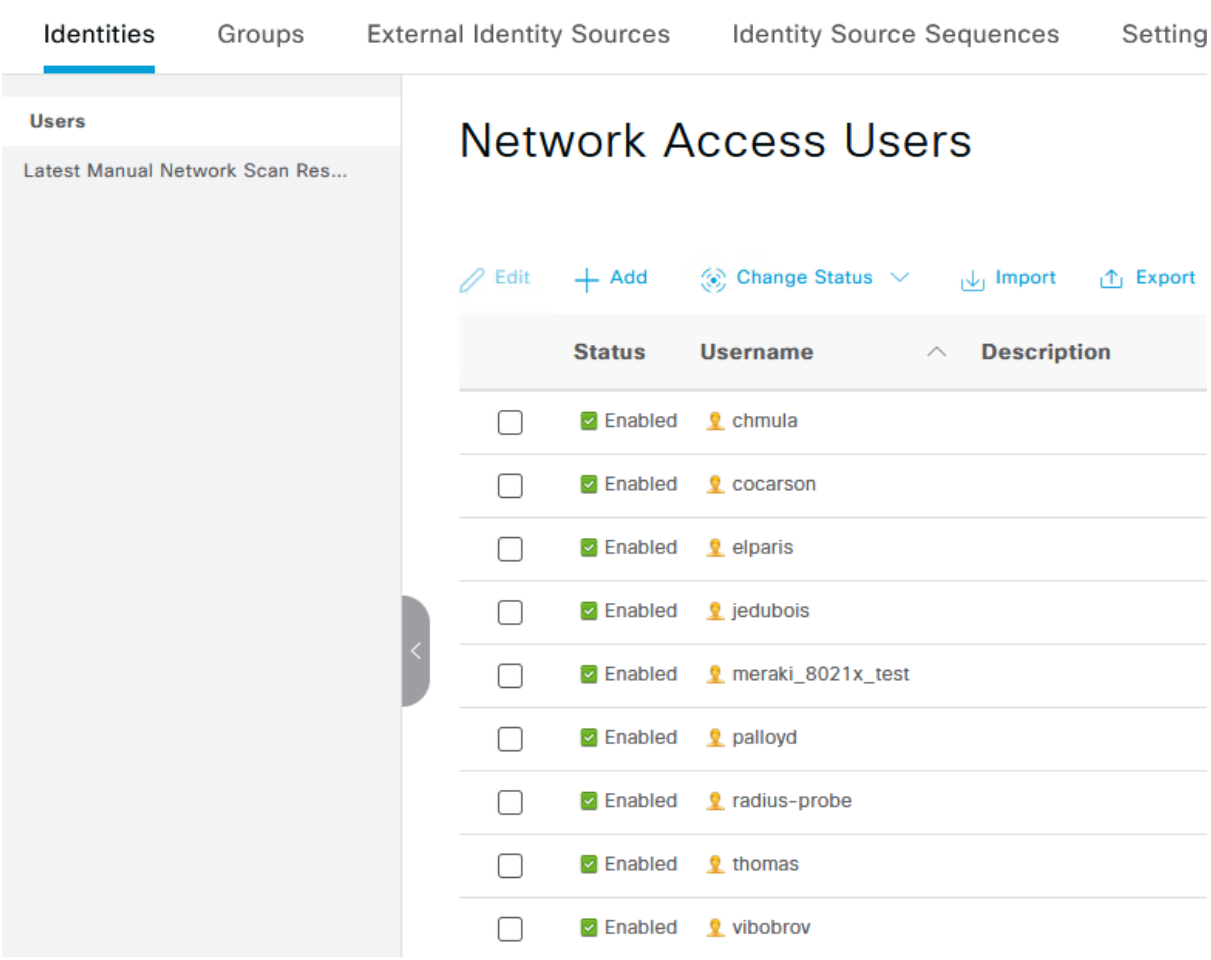
1. On your Ubuntu provisioning machine, navigate to ~/LTRSEC-

2000/Ansible/vars

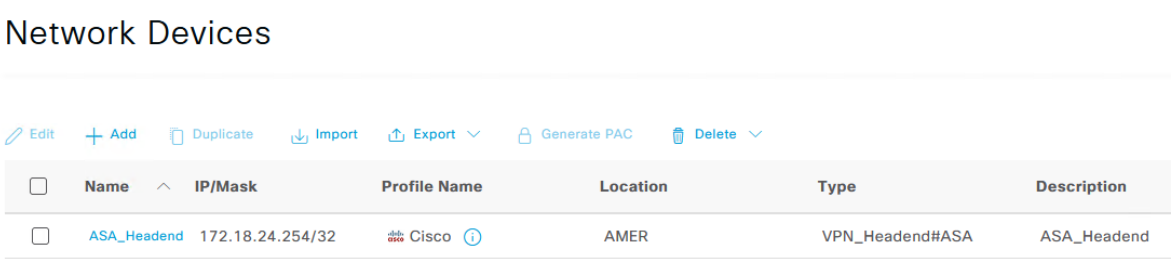
2. Execute a more on each file to explore what is expected to be configured, including users, endpoints, groups, network device groups, and network devices.
3. Using the hamburger menu on the top left of the screen, navigate to Administration -> Identity Management -> Groups -> User Identity Groups
4. Verify the RADIUS_Probes group has been created as a group to assign users to.




5. Navigate to Administration -> Identity -> Identities
6. Verify the users found in the “internal_users.thomas.yaml” are configured as expected.




7. Navigate to Administration -> Network Devices
8. Verify only one network device, the ASA Headend, was configured.




## Task 6: Entra ID integration via ROPC for Remote Access VPN

This task will have you create the basics needed for ISE integration with Entra ID for VPN authentication.

Note: Midway through the following task, you are asked to create a VPN user and set an initial password. Use the Initial password **provided by your proctor** and change this in the subsequent step to change the password to **Updated password provided by your proctor**

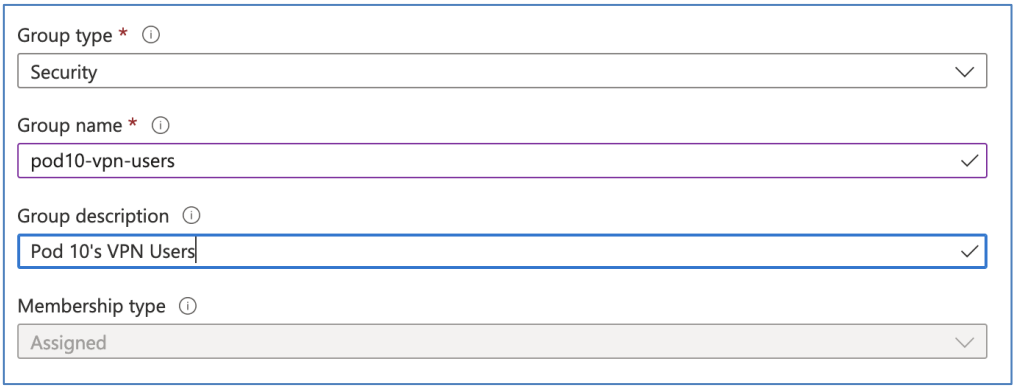
### Step 1: Log into Entra ID and Configure Users/Groups

Note: You may be asked to change your password during the first login.

1. Log into https://entra.microsoft.com/ with the pod administrator:

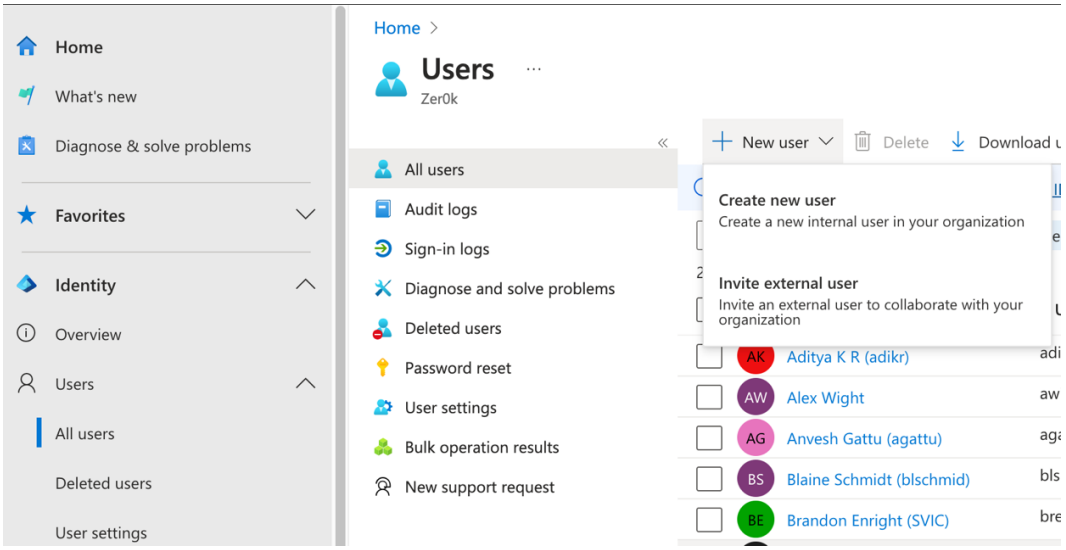
**podX-admin@zer0k.onmicrosoft.com** where X is the number of your pod and the password **is provided by your proctor**

2. On the left navigation bar browse to Identity -> Groups -> All Groups.
3. Select New Group. This will be used on ISE to determine which users are allowed to log into the VPN.
4. Leave the Group Type as Security and Name the group **podX-vpn-users** where X is your pod number. Give it a description Pod X VPN Users, where X is your pod number.

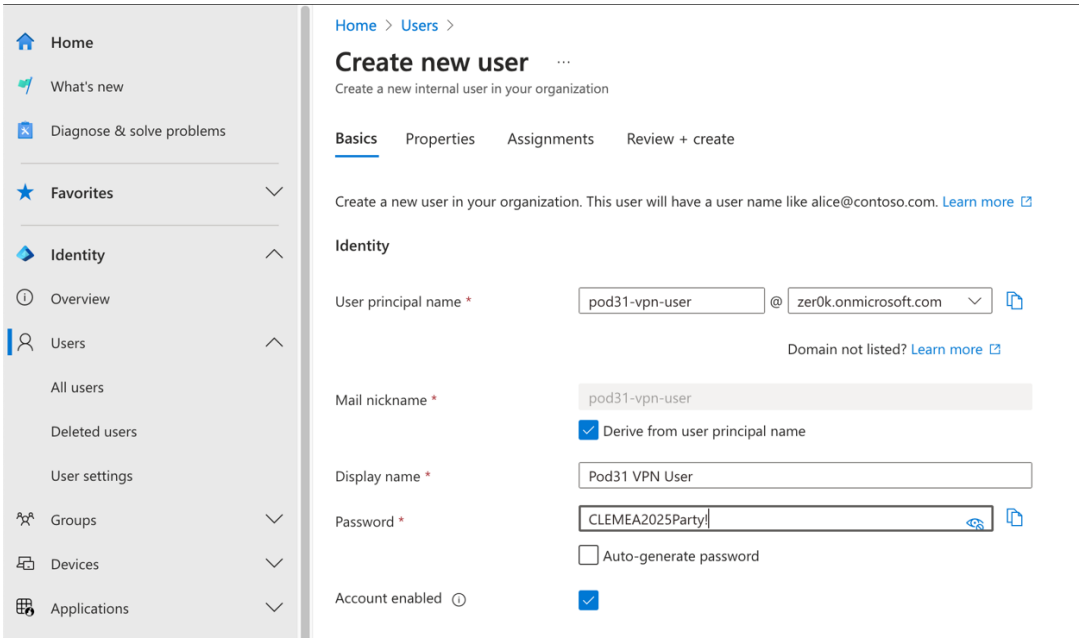



Note: It may take a minute before the group shows up in the All Groups area of Entra ID. Please be patient and hit refresh.

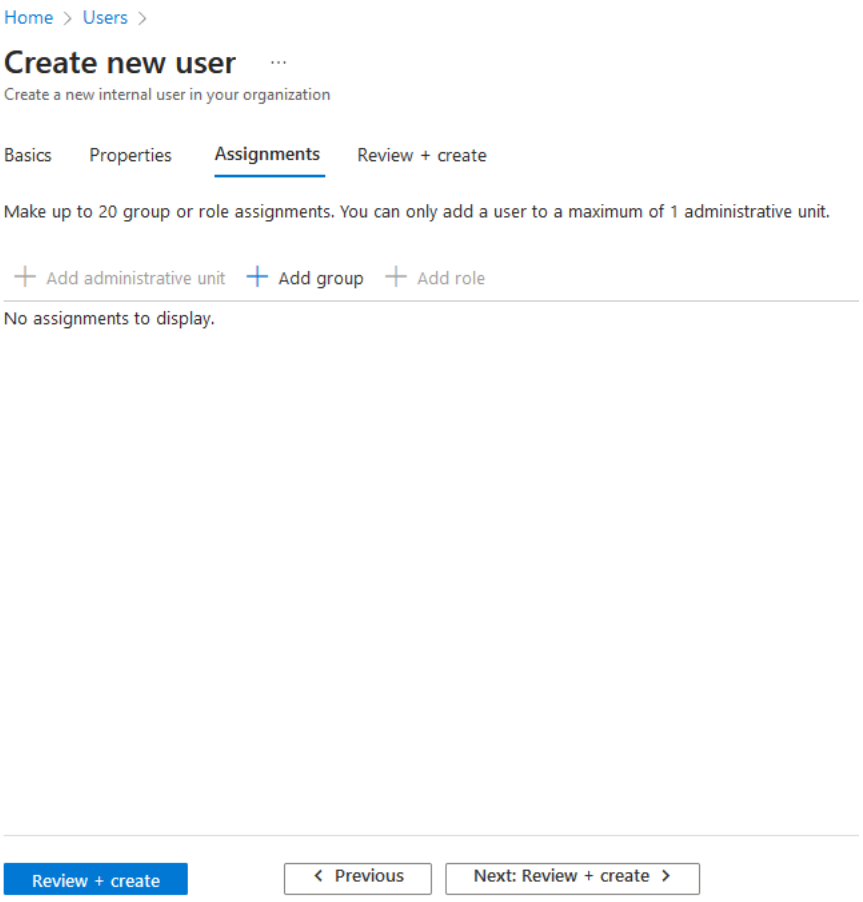
5. Using the left navigation bar, choose Identity -> Users -> All users.
6. Select New user at the top and select Create new user.
7. Name it **podX-vpn-user** where X is your pod number and give it a display name.



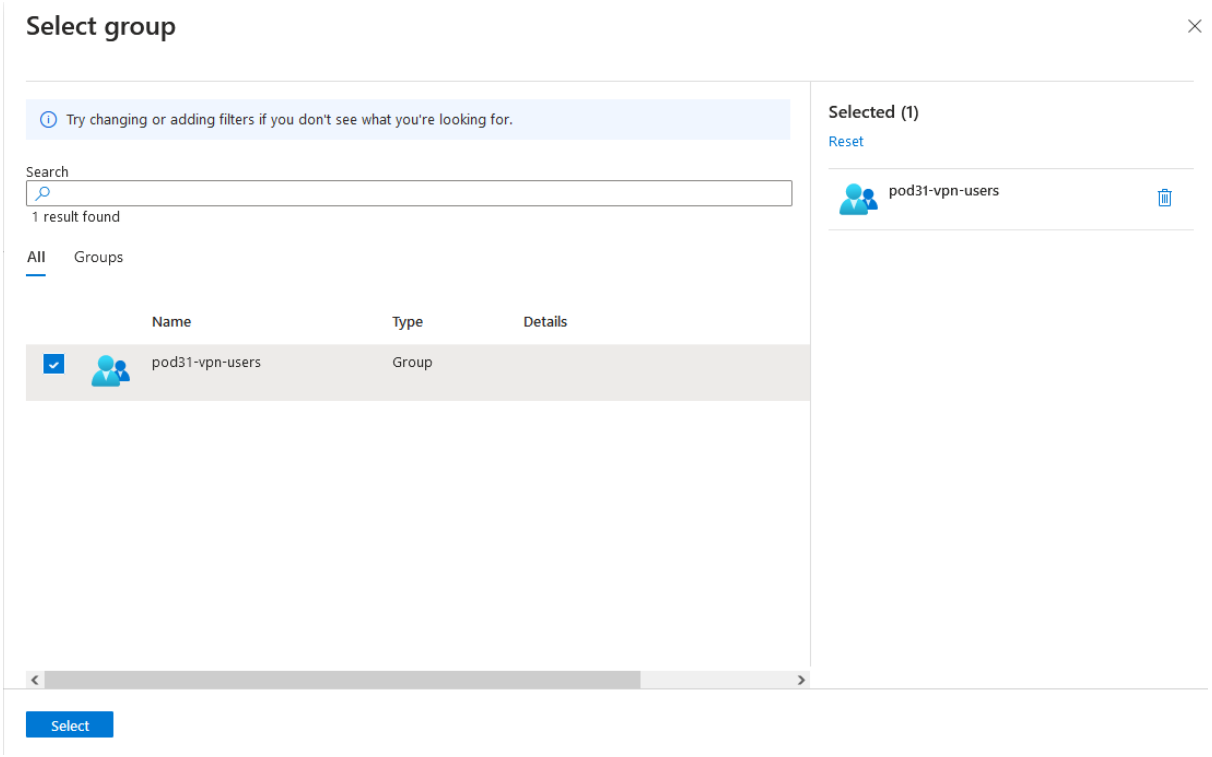

8. Complete the basic information for a user, utilizing the @zer0k.onmicrosoft.com domain, and the initial password **provided by your proctor.**



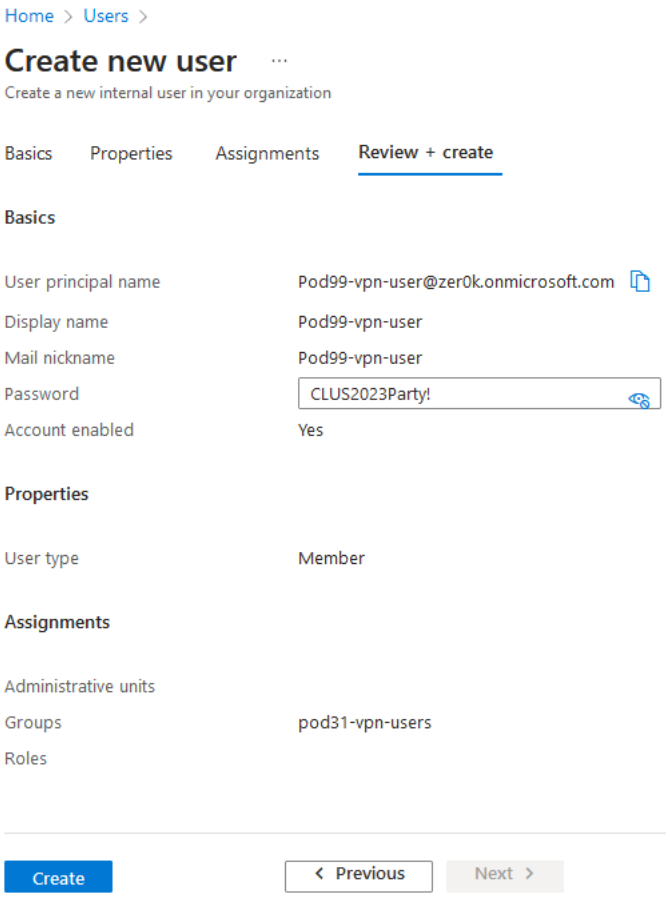

9. Click Next to advance to Properties. No information is required in this tab. Click Next to advance to Assignments.
10. Click “Add Group” at the top of the page to be presented with the “Select Group” pop out.



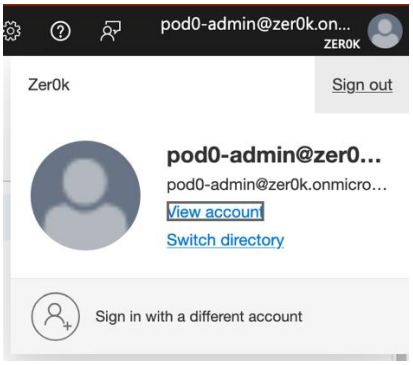

11. Under “Select Group”, check the box next to the group you created in the prior step, click “Select”.



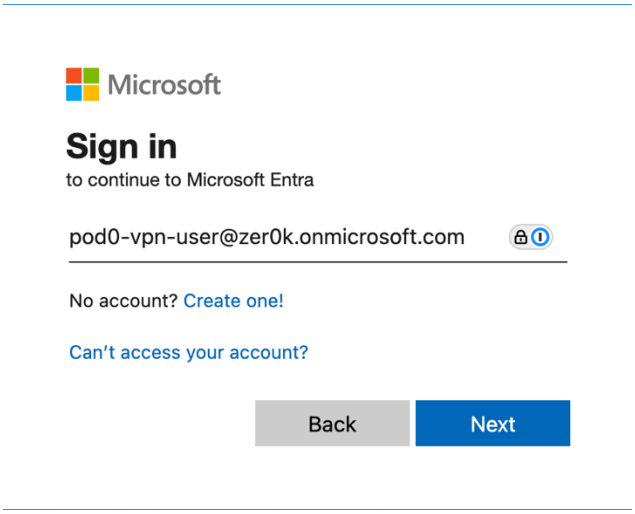

12. Click the Next button to advance to “Review + Create”. Create the user.



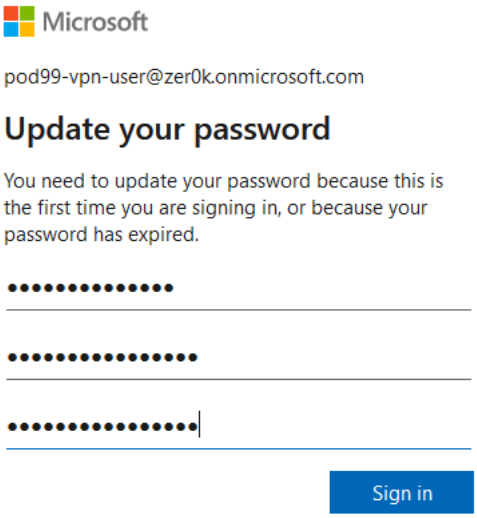

13. Entra ID requires the password to be reset before authentication can be done to that user. This cannot be done through the VPN auth so click your username at the top right of the screen and “sign out”



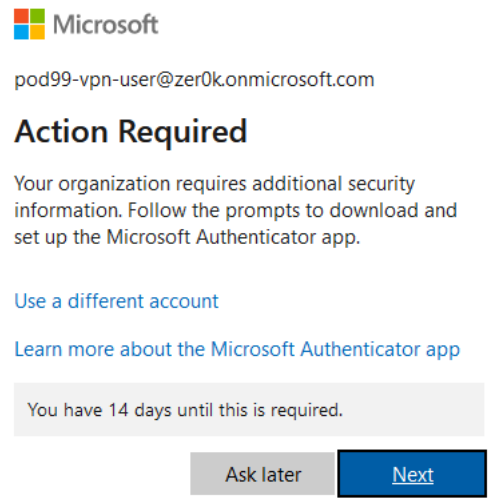

14. Navigate back to https://entra.microsoft.com, choose “User another account”, and sign in with the user you just created (the @zer0k.onmicrosoft.com suffix is required).



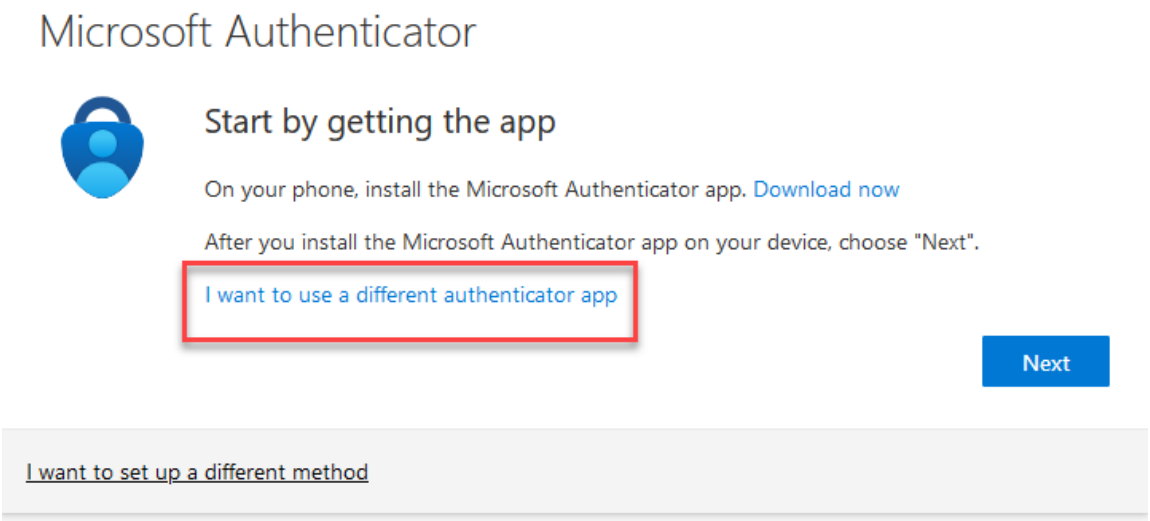

15. You will be prompted to reset your password. Use the **Updated password provided by your proctor** Password as your new password.

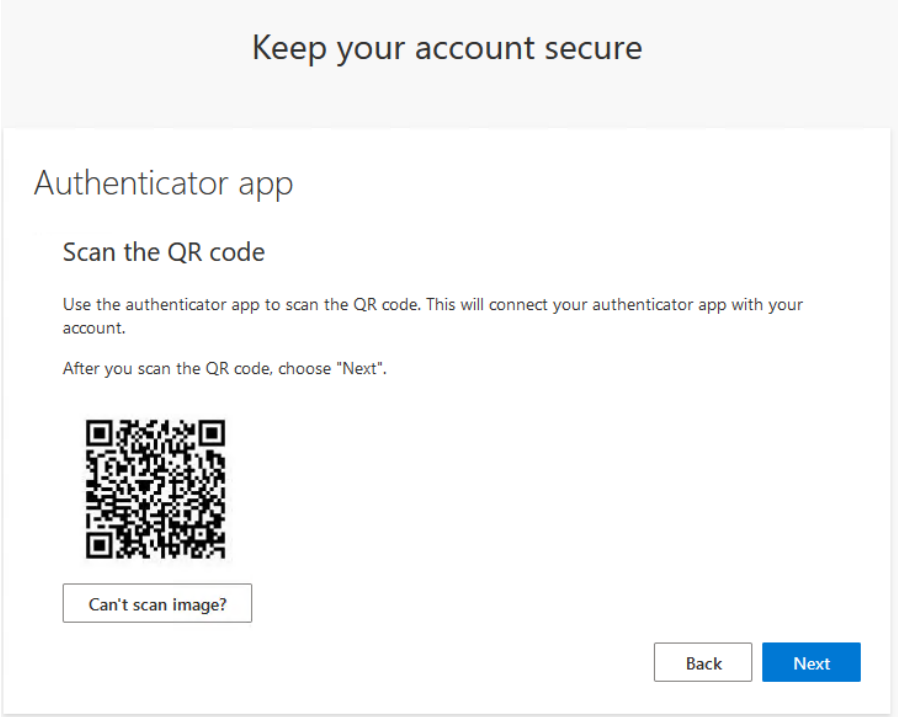



16. When asked to configure the Microsoft Authenticator app, click “Ask Later”

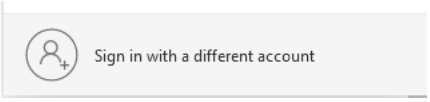



17. If Entra ID does not allow you to delay the MFA configuration, follow the prompts with the authenticator app used when logging into AWS to set up an additional account for Entra ID. Use the “I want to use a different authenticator app” if you used Google Authenticator or another app of your choice.



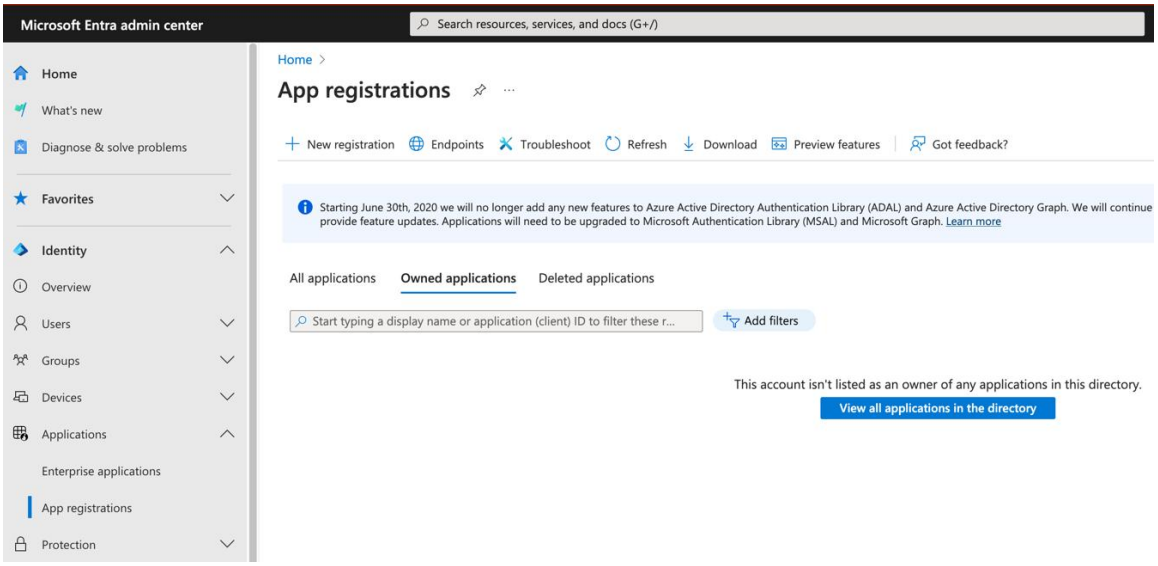



18. Once you successfully log in, switch back to the pod admin user.

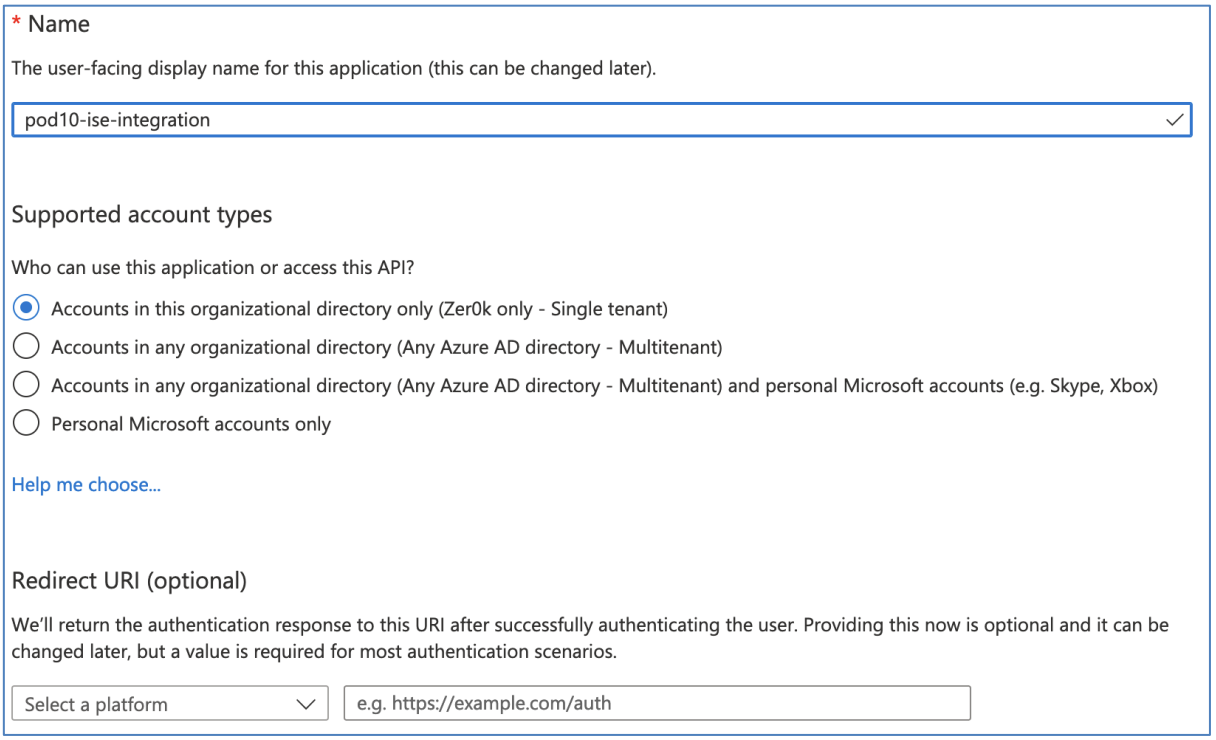



## Step 2: Configure application integration in Entra ID

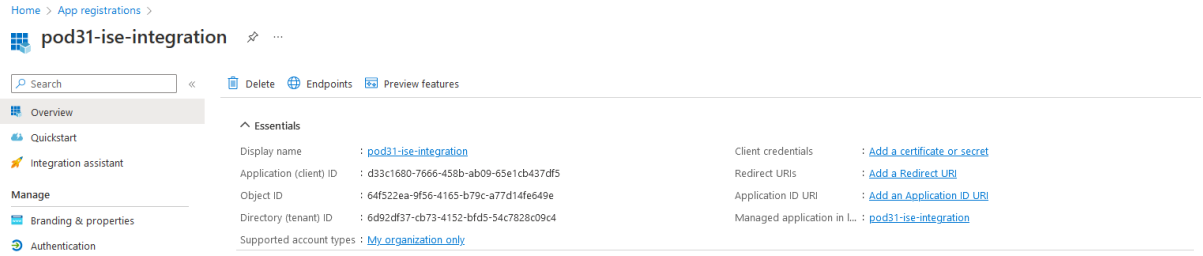
1. Using the left navigation bar brows to Identity -> Applications -> App registrations and choose “New registration”



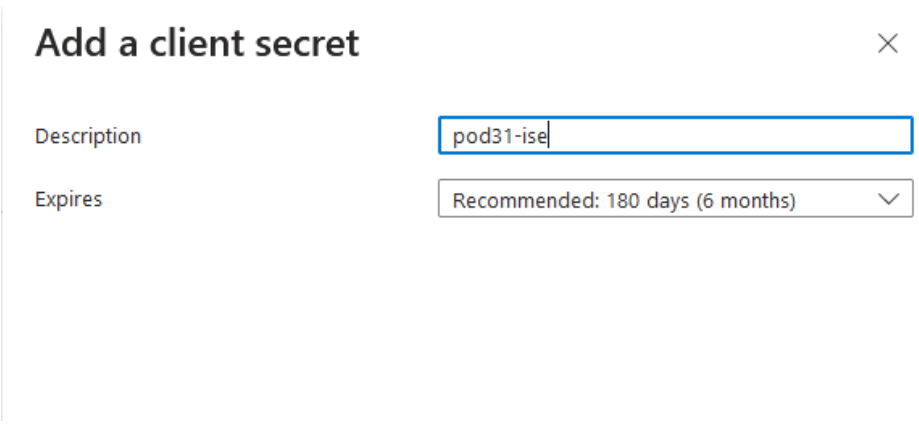

2. Name the app **podX-ise-integration** where X is your assigned pod number, and leave the Support account types as “Accounts in this organizational directory only”. This application will not need a redirect URI.




3. Click Register.
4. On the overview screen now presented, copy out the “Essentials” displayed at the top of the page. You will need the client ID and tenant ID later.



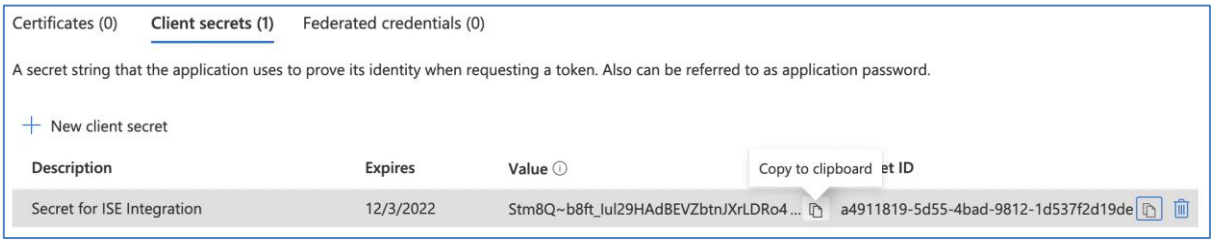

5. On the left sub-menu bar, select “Manage -> Certificates & secrets”. This is the OAuth secret key that will be used to integrate ISE with Entra ID. Click “New Client Secret” and enter a description of “podX-ISE” where X is your pod number. Click Add at the bottom of the dialog box.



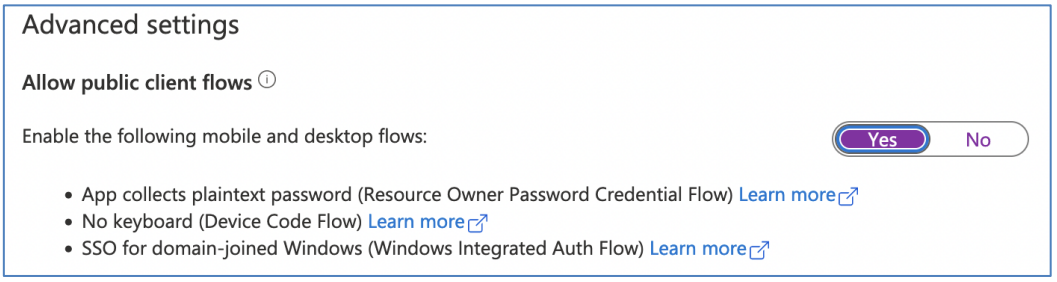
6. Copy the secret value to a notepad before leaving the page!

***Warning: Common mistake area!***

**This value cannot be viewed again after leaving the page. If you forget this step, delete the client secret and create a new one. It is *highly* recommended that you label the ID and value separately in the notepad document.**

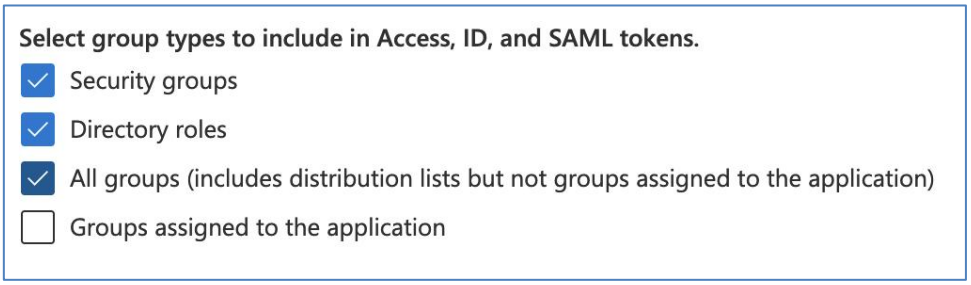



7. Configure the application for ROPC by going to the Authentication tab on the left sub-menu. ISE does not use a URI based redirect flow, therefore scroll down to Advanced Settings and set the “Allow public client flows” setting to Yes. Click Save.




8. On the left sub-menu, click Token Configuration
9. Under Token Configuration on the top menu, add a groups claim so that you can later create ISE policies that refer to groups in Entra ID. The groups claim should allow access to the following group types:
    - Security Groups
    - Directory Roles
    - All Groups

    Click Add.



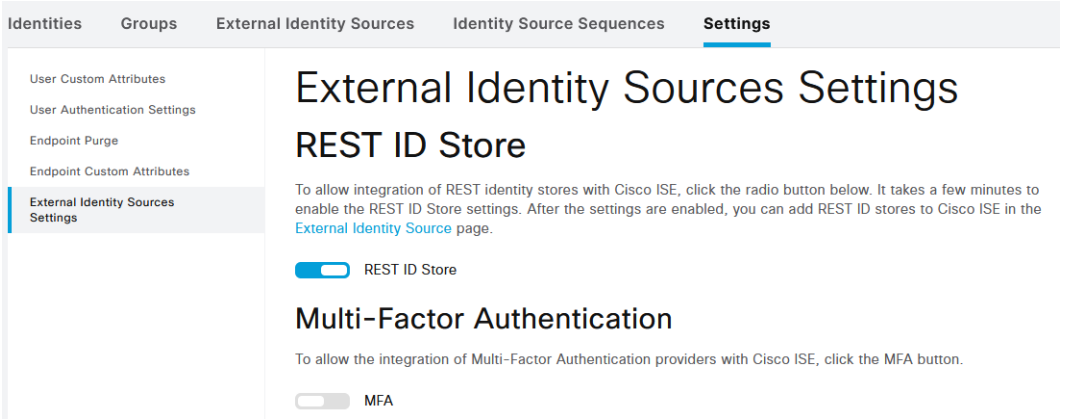

10. Add API permission for the groups graph API. In the left menu select “API permissions” then click Add a permission. In the window that pops up choose “Microsoft Graph” then “Application Permissions”.

11. Search for “group” then under the group drawer, select Group.Read.All, notice that Admin consent required is shown as Yes.
12. Select Add Permissions at the bottom of the screen.


13. Before moving on ask your lab proctor to provide consent for your new API Permission.
14. Open the overview page for the application, that information will be needed in the next steps.

## Step 3: Enable the ROPC feature in ISE

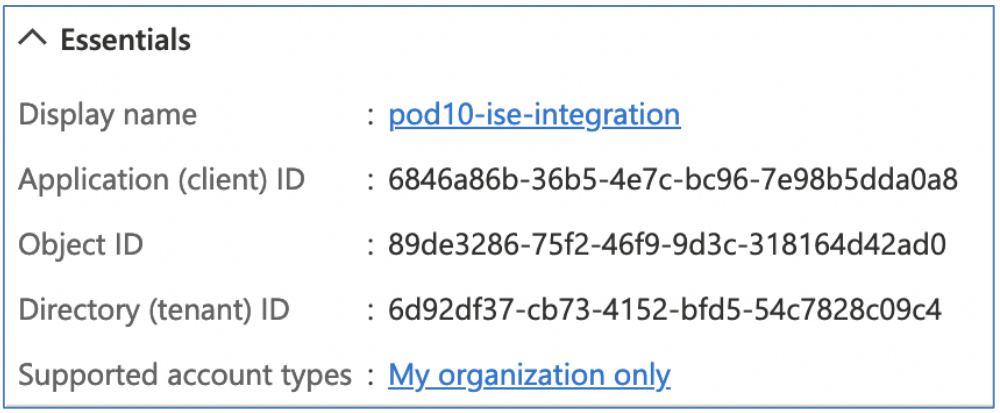
1. Go to Administration -> Identity Management -> Settings -> External Identity Sources Settings and ensure the REST ID Store is enabled. In previous versions of ISE this was disabled by default, so this is a verification step.



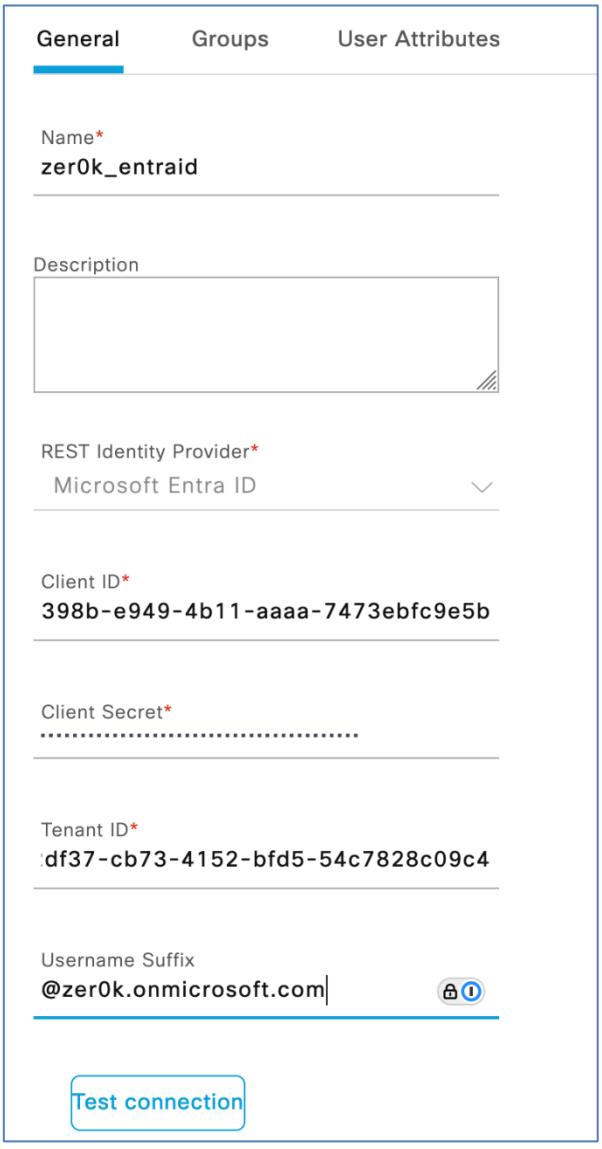

## Step 4: Register ISE with Entra ID

1. In ISE, browse to Administration -> Identity Management -> External Identity Sources from the hamburger menu icon at the top left of the ISE UI.
2. From the REST option, add a new entry.

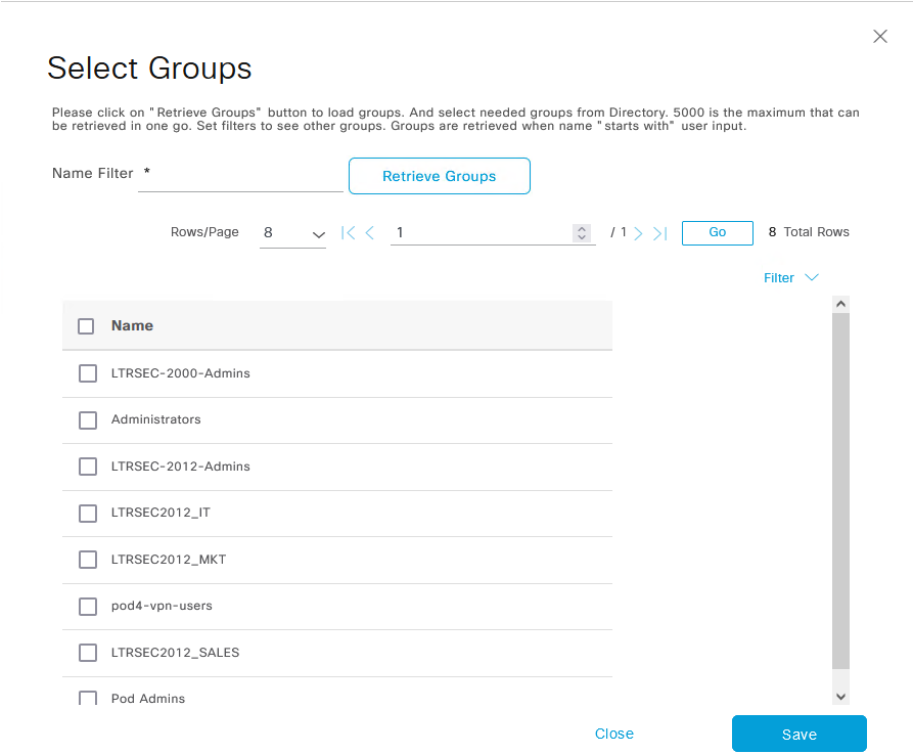
3. Name it **zer0k_entraid** and fill in the information obtained from the overview page under the application created in Step 2 from Entra ID. The client secret (Secret Value) should have been saved to a notepad. The username suffix for this lab is **@zer0k.onmicrosoft.com**.




Note: You'll need BOTH the secret **value** and the Client ID from two separate screens in this step. Both should be contained in a notepad document. In the Entra ID overview, the Client ID is the "Application (client) ID" and the tenant ID is the "Directory (tenant) ID".



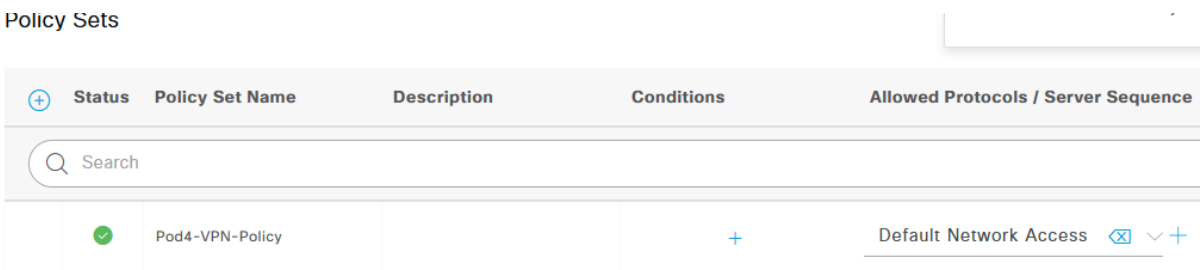
4. Test the connection, if it is successful go to the groups tab and select the **podX-vpn-users** group you created earlier.
5. In order to use groups in policy they need to be selected here in the ROPC connection. Click Add.  In the “Select Groups” screen, click “Retrieve Groups”




6. Select the group created in the previous step, with the format “**podX-vpn-users**” where X is your pod number.
7. Save the connection.

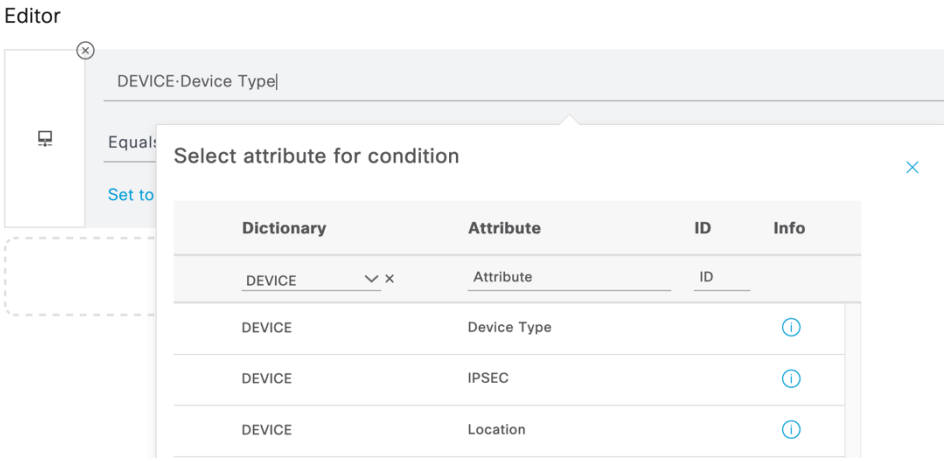
## Step 5: Configure ISE Policy for VPN Authentication

1. Go to the ISE menu -> Policy -> Policy Sets and select the “+” in the top left to add a new policy set.
2. Give the policy set a name  of “PodX-VPN-Policy” where X is your pod number, and select “Default Network Access’ under Allowed Protocols / Server Sequence.

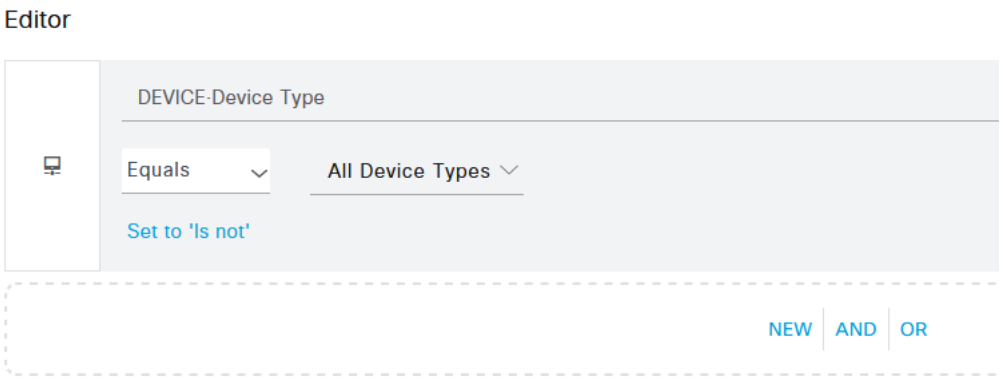



3. Click the “+” under conditions which will launch the conditions studio.
4. The network device and network device group were created by the initial configuration ansible scripts. They will be used here to match authentication from the VPN headend.

- Use the “All Dictionaries” drop down to filter for “DEVICE”.



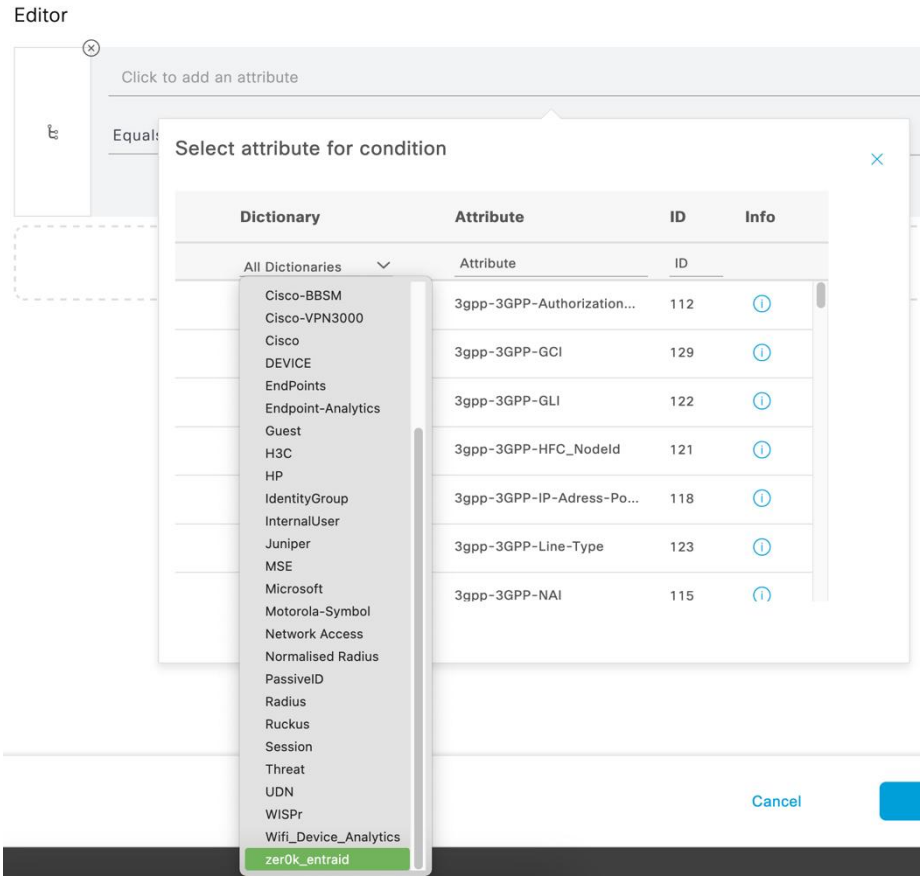

- Choose Device: Device Type and choose the “All Device Types” device type with the equals operator. The save button will allow you to save the condition for later, in this case choose “Use” at the bottom of the page to use it now.




5. The Conditions studio should close after hitting “Use”.
6. Save the policy sets.
7. Click the “View” arrow on the right side of the policy set line to enter the policy set configuration.
8. Expand “Authentication Policy” and set the default entry to what you named your ROPC connection (zer0k_entraid).

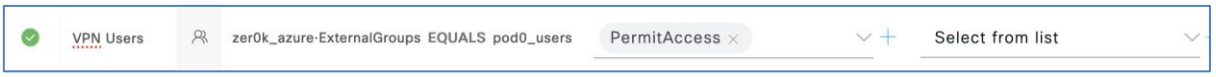

9. Expand “Authorization Policy” and click the “+” on the top of the authorization policy list to add a new entry, this entry will be used to match the group created on Entra ID and permit access to the VPN.

- Name the rule “VPN Authorization” then click the + under conditions to launch the condition studio and create a new condition.
- Click “click to add an attribute” and use the “All Dictionaries” drop down to filter for your “zer0k_entraid” external dictionary.
- Choose your ROPC connection name: External Groups and then select the group you have added your user to in Entra ID (podX-vpn-users).




- Leave the default as equals and choose use at the bottom of the screen to use the newly created condition.

10. Back on the Policy Set page set the Result to “Permit Access” then save.

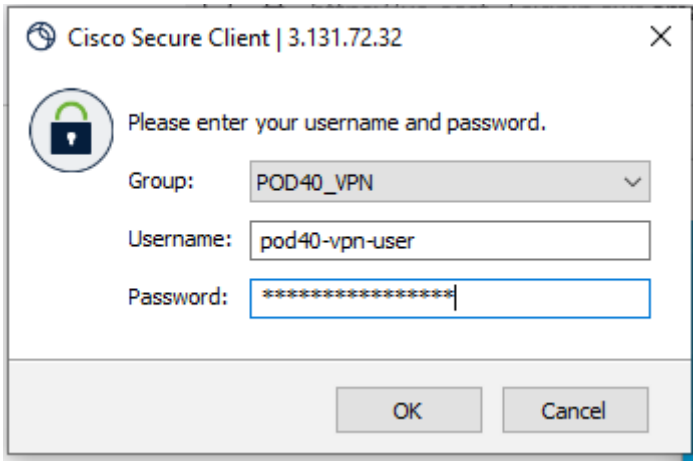



## Task 7: Test VPN Authentication

### Step 1: Test VPN Authentication

> Note: In this section of the lab you will be logging out of the lab VPN, while you are logged out any lab SSH or GUI connections will be unavailable. When reconnecting you may need to re-add environment variables to your SSH connections.

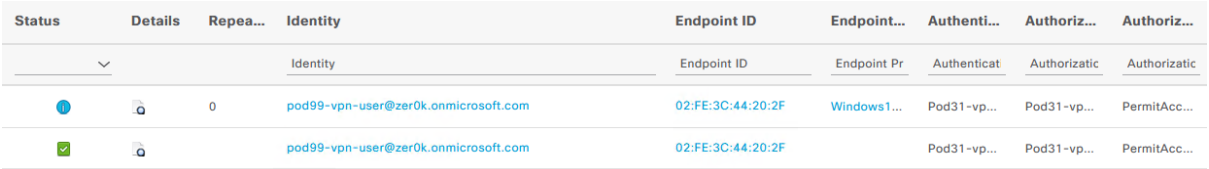
1. Log out of the current VPN connection from Secure Client on the lab laptop.
2. Reconnect to the same VPN address **3.131.72.32** but this time use your **PODX_VPN** group and **podX-vpn-user** entraid user. Use the password ending in $



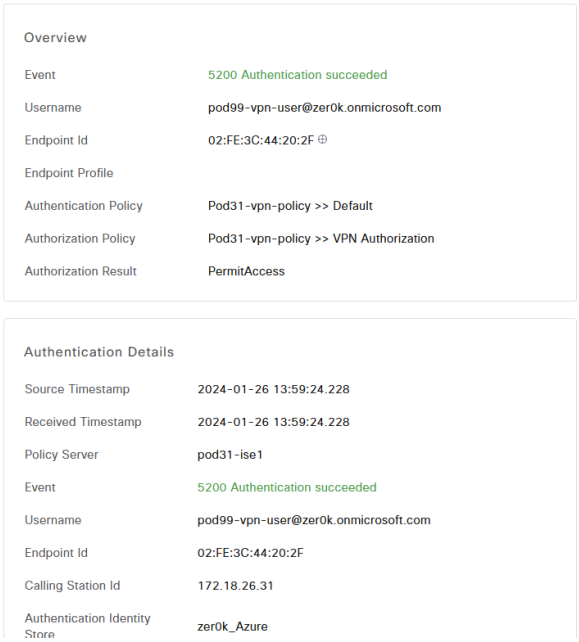

Note: There are two common errors encountered here, which lead to a radius dropped error, or an error the process. The first occurs if a misspelling occurred in step 6.4.3. Notice that the user suffix should be zer0klab.onmicrosoft.com. The fourth character is the **number zero** not an **o**.

The second common error here is if you attempt to add a suffix to the username. The username should be podX-vpn-user, it **should not** include the suffix @zer0klab.onmicrosoft.com if this was successfully populated in 6.4.3.

3. After successfully connecting you should be able to ping the super secret linux server at 172.18.16.10.
4. On ISE, browse to the ISE Menu -> Operations -> Radius -> Live Logs. Here you can see the authentication attempts and successful authentications to the VPN.




5. Click the details icon on one of the successful authentications where you can see the attributes available on the authentication and the steps ISE took to authenticate the user.



Overview

| | |
|---|---|
| Event | 5200 Authentication succeeded |
| Username | pod99-vpn-user@zer0k.onmicrosoft.com |
| Endpoint Id | 02:FE:3C:44:20:2F |
| Endpoint Profile | |
| Authentication Policy | Pod31-vpn-policy >> Default |
| Authorization Policy | Pod31-vpn-policy >> VPN Authorization |
| Authorization Result | PermitAccess |

Authentication Details

| | |
|---|---|
| Source Timestamp | 2024-01-26 13:59:24.228 |
| Received Timestamp | 2024-01-26 13:59:24.228 |
| Policy Server | pod31-ise1 |
| Event | 5200 Authentication succeeded |
| Username | pod99-vpn-user@zer0k.onmicrosoft.com |
| Endpoint Id | 02:FE:3C:44:20:2F |
| Calling Station Id | 172.18.26.31 |
| Authentication Identity Store | zer0k_Azure |

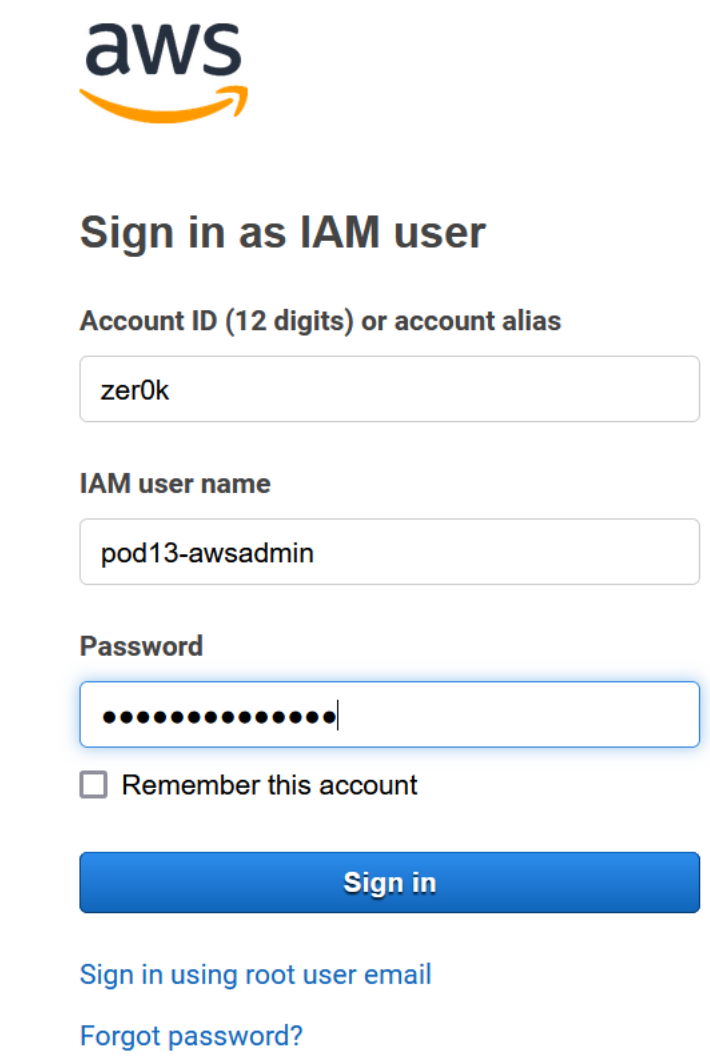
Steps

| Step ID | Description | Latency (ms) |
|---|---|---|
| 11001 | Received RADIUS Access-Request | |
| 11017 | RADIUS created a new session | 0 |
| 15049 | Evaluating Policy Group | 0 |
| 15008 | Evaluating Service Selection Policy | 0 |
| 15048 | Queried PIP - DEVICE.Device Type | 4 |
| 15041 | Evaluating Identity Policy | 7 |
| 15013 | Selected Identity Source - zer0k_Azure | 2 |
| 25103 | Perform plain text password authentication in external REST ID store server - zer0k_Azure | 0 |
| 25100 | Connecting to external REST ID store server - zer0k_Azure | 36 |
| 25101 | Successfully connected to external REST ID store server - zer0k_Azure | 240 |
| 25104 | Plain text password authentication in external REST ID store server succeeded - zer0k_Azure | 0 |
| 25107 | REST ID store server respond with groups - zer0k_Azure | 0 |
| 25110 | User groups inserted to session cache - zer0k_Azure | 1 |
| 22037 | Authentication Passed | 0 |
| 24715 | ISE has not confirmed locally previous successful machine authentication for user in Active Directory | 0 |
| 15036 | Evaluating Authorization Policy | 0 |
| 24209 | Looking up Endpoint in Internal Endpoints IDStore - pod99-vpn-user@zer0k.onmicrosoft.com | 1 |
| 24217 | The host is not found in the internal endpoints identity store | 16 |
| 15016 | Selected Authorization Profile - PermitAccess | 9 |

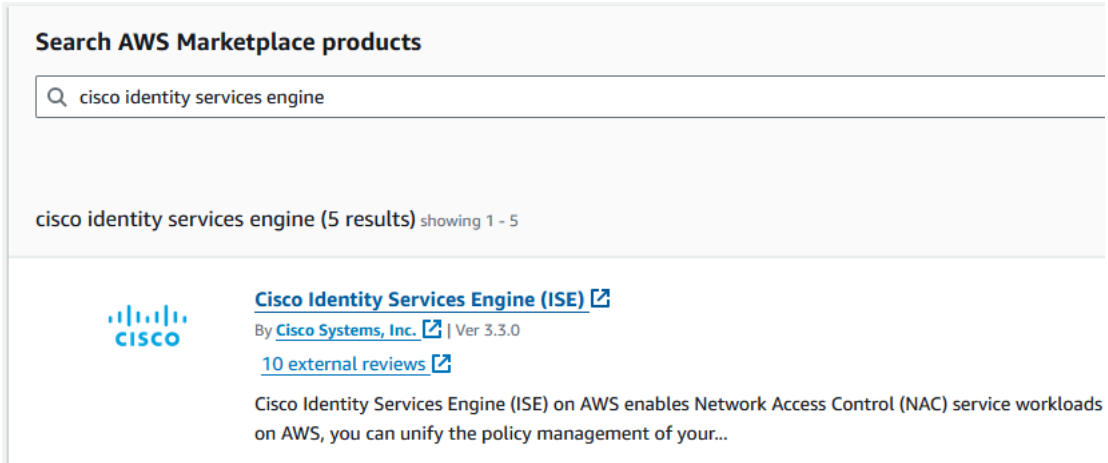
# Task 8: (Bonus) Deploy Secondary ISE through AWS Marketplace from CloudFormation Template

## Step 1: Subscribe to ISE in AWS Marketplace

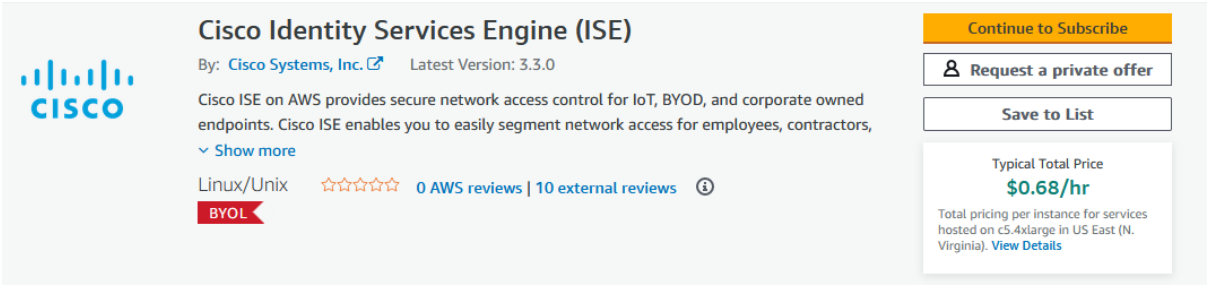
1. Navigate to https://zer0k.signin.aws.amazon.com/console Use Account ID zer0k if not already populated. Login with the username assigned to your pod in table 1-1 above (**podX-awsadmin**). The password for the account **is provided by your proctor.**



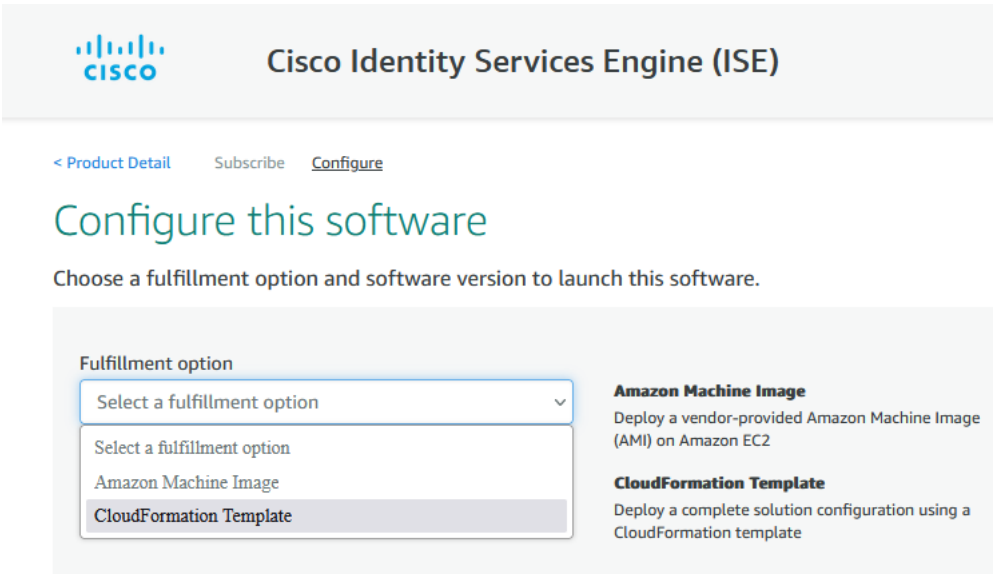

2. In the AWS console search for AWS Marketplace Subscriptions.
3. Go to Discover Products on the left menu.
4. Search for ISE and click on Cisco Identity Services Engine (ISE).



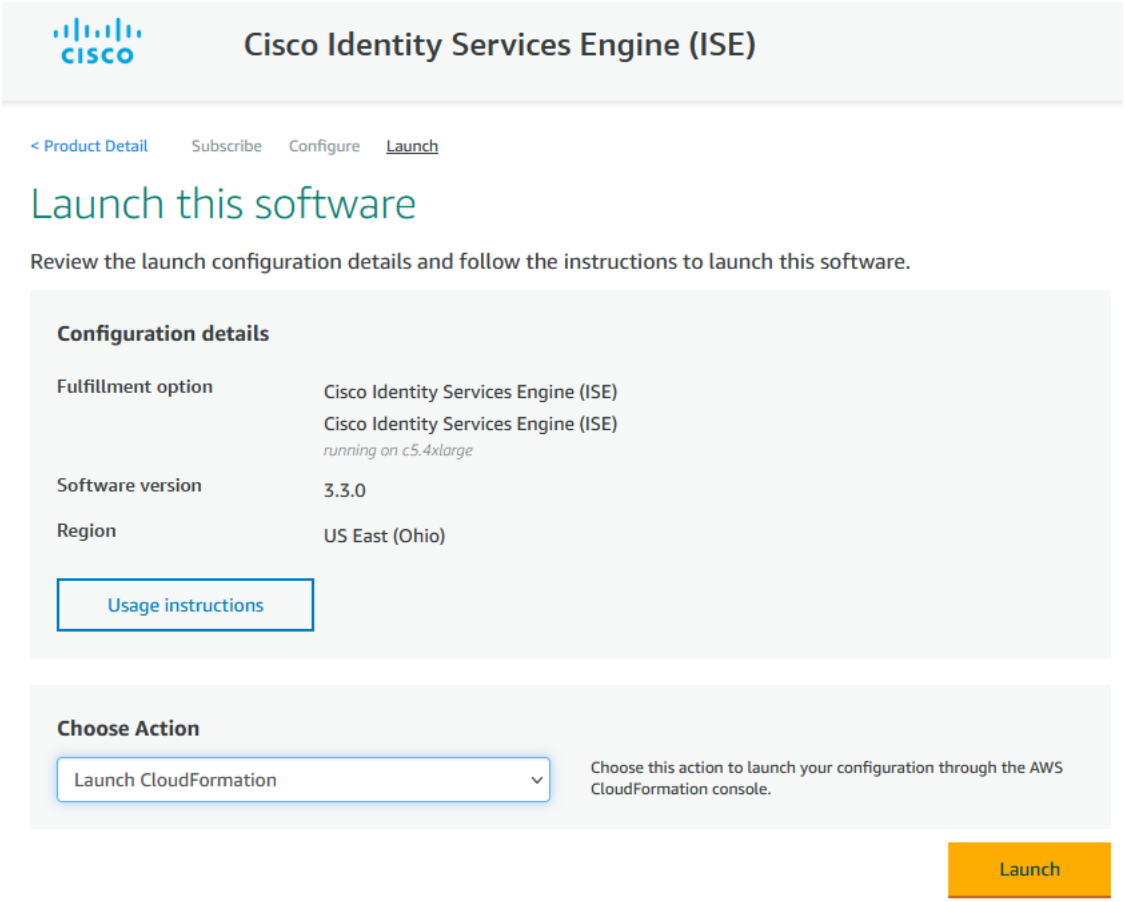

5. Click on “Continue to Subscribe” on the top right




6. Click on Continue to Configuration.
7. Under fulfilment option, select “CloudFormation Template”.



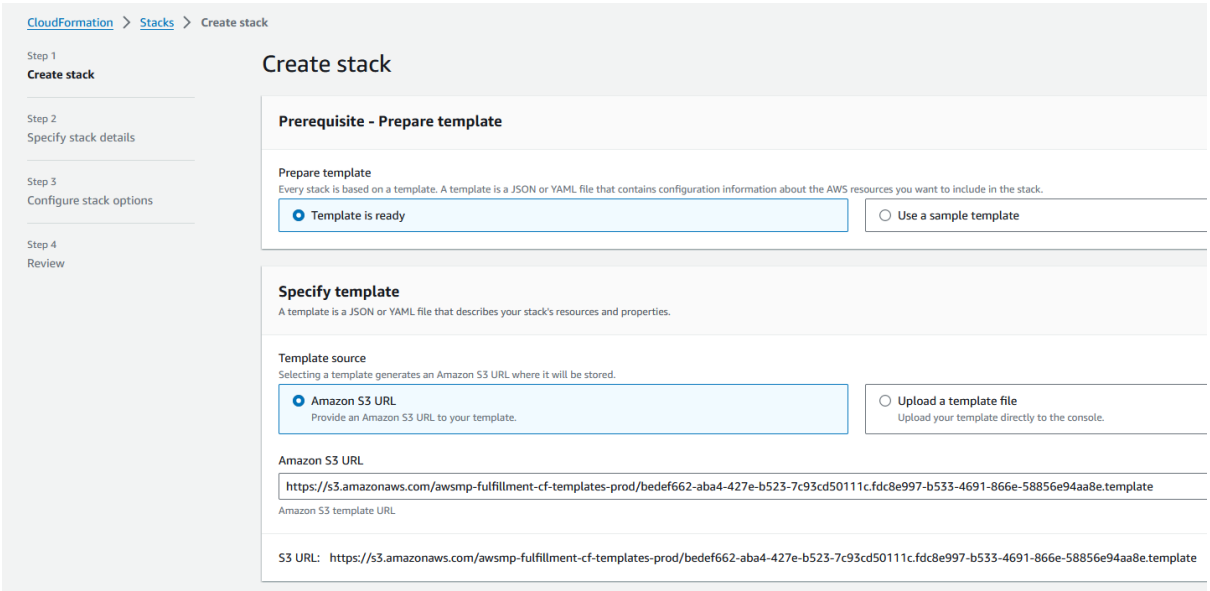

8. A new drop down will be displayed, select Cisco Identity Services Engine (ISE).
9. 2 new drop downs will be presented, select 3.3.0 (July 09, 2023) and select the Region as US East (Ohio). If you don’t use 3.3.0 you won’t be able to join the node created by ansible earlier.
10. Click Continue to Launch on the top right.
11. Under Choose Action, select “Launch CloudFormation", then select the Launch button.




## Step 2: CloudFormation Stack Creation

Marketplace will launch a new tab for CloudFormation with a link to the publicly available CloudFormation Template uploaded to S3.

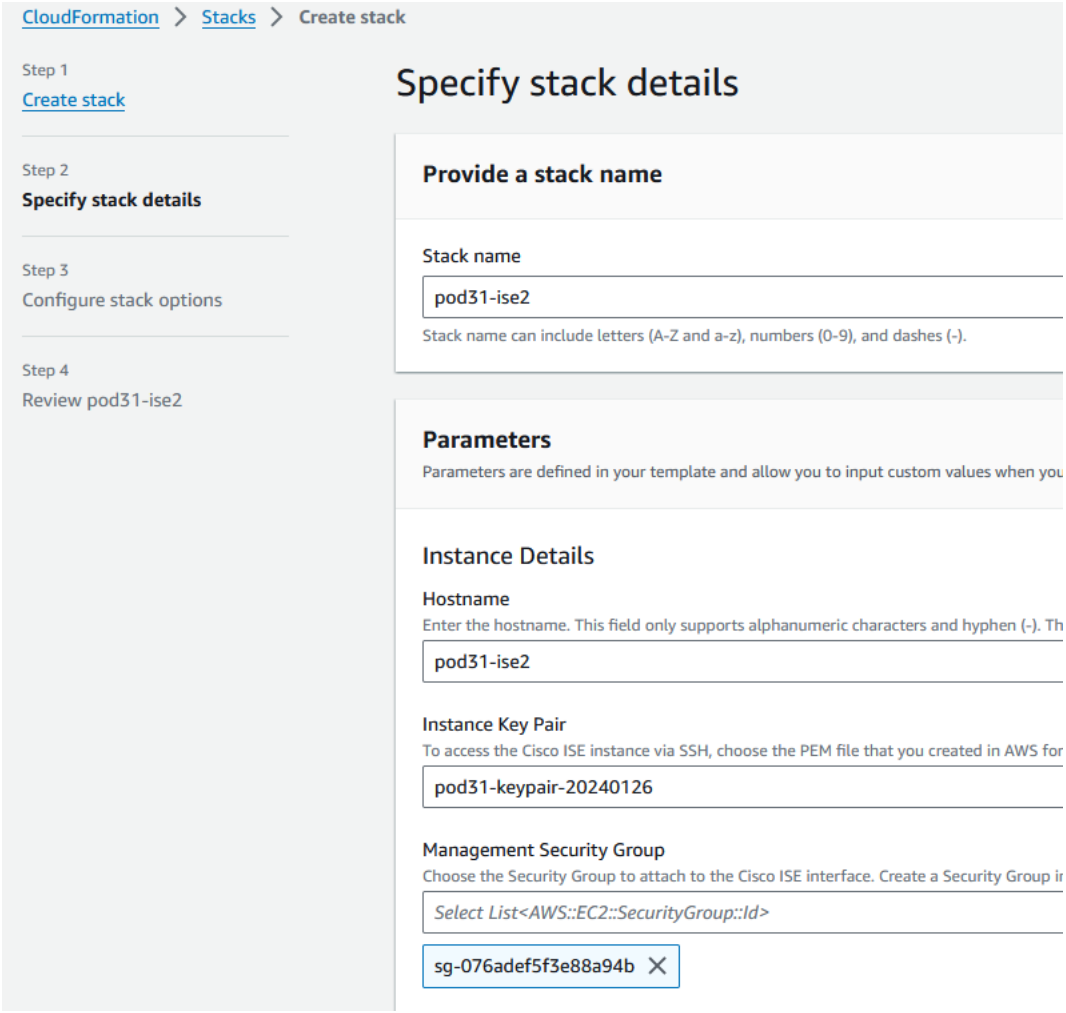
1. Copy the Amazon S3 URL and open it in a new tab/browser and save it to your local machine. This can be used later directly in CloudFormation rather than going through the subscription as another option. This lab will not use this method unless initial Cloud Formation fails.




2. With the “Template is ready” option selected, and the Amazon S3 URL still populated, click Next.

3. The template includes all the provisioning information needed to launch an ISE instance. Fill out all the fields:

a. Stack Name – Since this is a single instance set this to the hostname podX-ise2 where X is your pod number.
b. Hostname – podX-ise2 where X is your pod number.
c. Instance Key Pair – Select the key pair you created in Task 1 and have already saved to your local machine.
d. Management Security Group – ISEinAWS-podX where X is your assigned pod number. This was created in the Ansible ISE deployment steps.
e. Management Network – Select the subnet in the new VPC created in Task 3, it should be zer0k_podX_Private_Subnet where X is your assigned pod number.
f. Management Private IP – Given in the Table above as configured in Route53. It will be 10.X.0.6 where X is your assigned pod number.
g. Time Zone – Etc/UTC
h. Instance Type – C5.4xlarge
i. EBS Encryption – false
j. Volume Size – 600
k. DNS Domain – zer0k.org
l. Name Server – 10.X.0.2 where X is the number of your assigned pod.
m. NTP Server – 169.254.169.123
n. ERS – yes
o. OpenAPI – yes
p. pxGrid – no
q. pxGrid Cloud – no
r. Password – **Provided by your proctor**




4. Click Next.
5. Under Tags add a Key “Name” with value “podX-ise2” where X is the number of your pod. CloudFormation is going to add an instance, EBS volume, and network interface that will all get named via this tag.
6. Under Tags, add a tag. Enter “project” under Key and “ISEinAWS-podX” under value, where X is your pod number.
7. Leave the rest of the options default though you are welcome to browse through them to see the options AWS provides. Click Next when you are done.
8. Review the information to make sure you entered everything correctly, then click “Submit” at the bottom of the page.
9. Under events you should see “CREATE_IN_PROGRESS”, refresh the table until you see podX-ise2 with “CREATE_COMPLETE”.

| Timestamp | Logical ID | Status | Status reason |
|---|---|---|---|
| 2022-05-09 16:44:06 UTC-0400 | pod0-ise | CREATE_COMPLETE | - |
| 2022-05-09 16:44:04 UTC-0400 | IseEc2Instance | CREATE_COMPLETE | - |
| 2022-05-09 16:43:57 UTC-0400 | IseEc2Instance | CREATE_IN_PROGRESS | Resource creation Initiated |
| 2022-05-09 16:43:55 UTC-0400 | IseEc2Instance | CREATE_IN_PROGRESS | - |
| 2022-05-09 16:43:49 UTC-0400 | pod0-ise | CREATE_IN_PROGRESS | User Initiated |

## Task 9: (Bonus) Clean Up

### Step 1: Terraform Destroy

If you are finished with your lab and have no further questions, Terraform tracks the status of the ISE node and configurations that were run. However, ISE-2 was deployed outside of the stateful tracking of the deployment. If you attempt to destroy the terraform state, ise-2 continues to hold on to the security group, and prevents AWS from deleting all components. Follow the procedure below to tear down the lab.

1. If you completed task 8, you'll first need to destroy the ise-2 instance, as this uses a common subnet to ise-1. In AWS navigate to ***EC2 > Instances > check the box next to your ise2 instance > Instance State -> Terminate***. Wait for the status of the instance to be "Terminated". This may take approximately 10 minutes.
2. Navigate back to the Ubuntu machine.
3. Use the command ***pwd*** to verify you are currently under the Ansible directory:

```
(ISEonAWS) ubuntu@ip-172-18-25-2:~/LTRSEC-2000/Ansible$ pwd
/home/ubuntu/LTRSEC-2000/Ansible
```

4. Change directory back to the Terraform directory. Use the command

```
cd ..
cd Terraform
```

5. Enter command:

```
terraform destroy
```

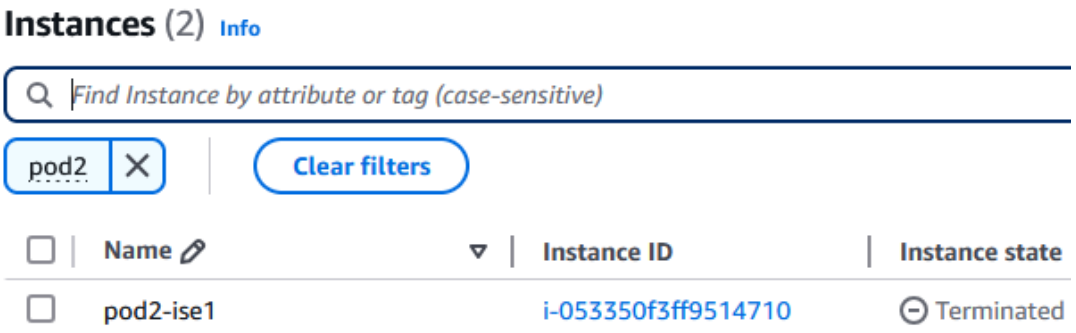
Enter "yes" when prompted:

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.
```

6. Verify the last line of the output is:

```
Destroy complete! Resources: 11 destroyed.
```

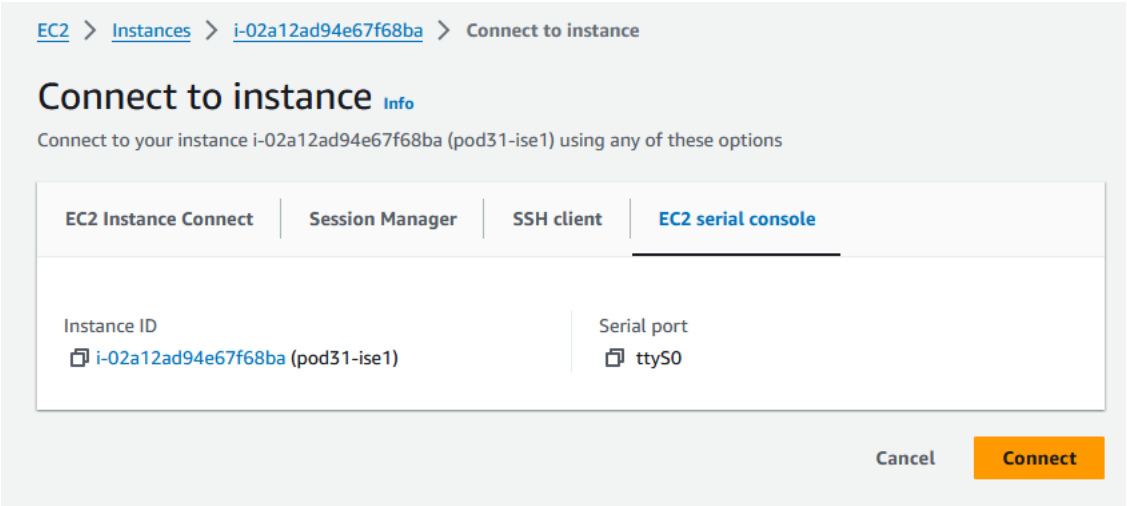
7. Within your web browser, navigate back to your AWS instance. Navigate to EC2 -> Instances.
8. Use the filter at the top of the screen to filter to your pod instances, entering "podX" where X is your pod number.
9. Verify your ISE node has been terminated.



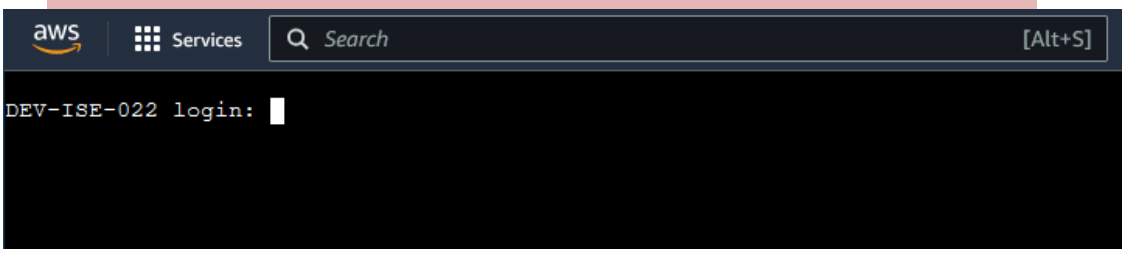

10. You Subnet, routing table, and transit gateway attachment all rely on your VPC being present. Navigate to VPC using the search bar, and "Your VPCs". Verify your VPC, named "ISEinAWS-podX" where X is your pod number, is no longer present.

## Appendix 1: Using the EC2 Serial Connect to Monitor ISE Deployment Status

1. If you would like to see the ongoing status of the ISE deployment you can connect through the virtual serial. In your AWS Console, click the instance name and click the "Connect" button on the top left.
2. Select EC2 Serial Console, and connect




Note: If your ISE host name is not in the format of podXX-ise upon initial connection, ***do not issue any commands***. The ISE node is still programmatically deploying. Wait 5 minutes before hitting enter again to get the proper login prompt.




3. ***Optional:*** Run the following command:

```
show application status ise
```

```
podXX-ise/admin# show application status ise
ISE PROCESS NAME                 STATE          PROCESS ID
------------------------------------------------------------
---------
Database Listener                running        32895
Database Server                  running        85 PROCESSES
Application Server               not running
...
Application Server               initializing           49728
...
Application Server               running
```

4. On the CLI where the ssh command was run, you will be prompted to accept the ssh fingerprint of the ISE server. Enter yes.
5. Run the following command:

```
show application status ise
```

```
podXX-ise/admin# show application status ise
ISE PROCESS NAME                 STATE          PROCESS ID
------------------------------------------------------------
--------
Database Listener                running        32895
Database Server                  running        85 PROCESSES
Application Server               not running
...
Application Server               initializing           49728
...
Application Server               running
```

6. Once the Application Server shows as “running” move on to the next step.

## Appendix 2: Package Terraform Not Found Error Remediation

In the case that a copy-paste error occurred, you may receive the error that the system is unable to locate the package “terraform”. This is typically due to a missing character in step 5.1.6 or 5.1.7, either a missing space in the repository between “jammy” and “main”, or a missing – in the GPG command between hashicorp-archive-keyring. Perform the following to recover from this error.

```
ubuntu@ip-172-18-25-40:~$
ubuntu@ip-172-18-25-40:~$ sudo apt-get install terraform
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package terraform
```

1. In 5.1.6 you download package for Terraform and subsequently add it as a valid source in the sources list. We'll need to remove the sources list and re-run 5.1.6 onward.

   Run the following command to verify the hashicorp source exists in the sources directory:

   ```
   ls -al /etc/apt/sources.list.d
   ```

   If the hashicorp.list file does not exist, contact a proctor.

2. Remove the hashicorp.list file, which most likely does not contain the correct package.

   Run the command:

   ```
   sudo rm /etc/apt/sources.list.d/hashicorp.list
   ```

3. This should remove the hashicorp repository from the system. Verify this by running:

   ```
   ls -al /etc/apt/sources.list.d
   ```

4. This procedure is built on the official Hashicorp installation instructions. More information on how to install Terraform can be found at the following page, specific to the Linux deployment:

   https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

**Thank you for your interest in the lab and joining us at Cisco Live! Please remember to do your session survey, it helps us continue to offer sessions like this!**