ISE Deployments in the Cloud - Automate ISE Deployments in AWS and Integrate Them with Azure Active Directory

LTRSEC-2000

Speakers:

Jesse Dubois

Patrick Lloyd

Table of Contents

Learning Objectives or Table of Contents

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

- Provision an Ubuntu Linux machine with Ansible to automate configurations.
- Automatically provision AWS VPC's, Subnets, Routes, and Transit Gateway Attachment via Ansible.
- 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.
- Create the ODBC object for provision of Azure Active Directory.

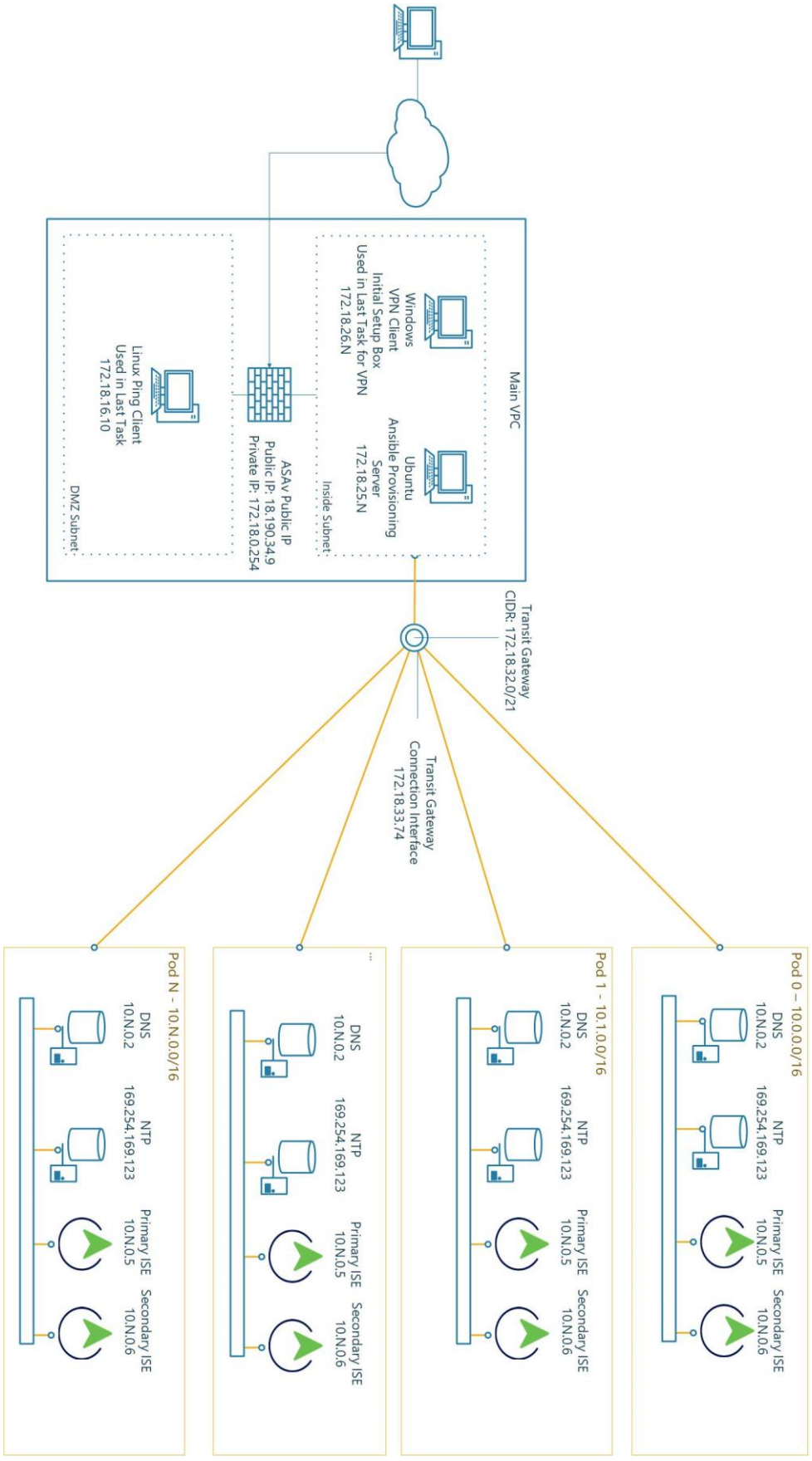
Scenario

In this lab activity, you will learn how to utilize an Ubuntu Linux machine to deploy ISE into an AWS environment, which is dynamically allocated via Ansible. The ansible script, found on github.com for later consumption, will be used to provision the VPC, subnet, routing table, and internet gateway into an AWS tenant already created. It will then deploy ISE within the VPC with a specified private IP address which will then be used to provision network device groups, network access devices, user roles, and users. You will then provision the Azure Active Directory connector, which requires manual intervention to join to Azure Active Directory.

Recommended Equipment

1. A laptop with VMWare workstation installed.
2. A virtual image you are familiar with.
3. AnyConnect on the virtual image. If you do not have AnyConnect, reach out to your lab proctor.
4. 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	Windows Provisioning Host	Windows Password	Ubuntu Ansible Host	AWS Username	RAVPN Username	RAVPN and AWS Password
1	172.18.26.1	CLEMEA2023Party!	172.18.25.1	pod1-awsadmin	pod1-ravpn	CLEMEA2023Party!
2	172.18.26.2	CLEMEA2023Party!	172.18.25.2	pod2-awsadmin	pod2-ravpn	CLEMEA2023Party!
3	172.18.26.3	CLEMEA2023Party!	172.18.25.3	pod3-awsadmin	pod3-ravpn	CLEMEA2023Party!
4	172.18.26.4	CLEMEA2023Party!	172.18.25.4	pod4-awsadmin	pod4-ravpn	CLEMEA2023Party!
5	172.18.26.5	CLEMEA2023Party!	172.18.25.5	pod5-awsadmin	pod5-ravpn	CLEMEA2023Party!
6	172.18.26.6	CLEMEA2023Party!	172.18.25.6	pod6-awsadmin	pod6-ravpn	CLEMEA2023Party!
7	172.18.26.7	CLEMEA2023Party!	172.18.25.7	pod7-awsadmin	pod7-ravpn	CLEMEA2023Party!
8	172.18.26.8	CLEMEA2023Party!	172.18.25.8	pod8-awsadmin	pod8-ravpn	CLEMEA2023Party!
9	172.18.26.9	CLEMEA2023Party!	172.18.25.9	pod9-awsadmin	pod9-ravpn	CLEMEA2023Party!
10	172.18.26.10	CLEMEA2023Party!	172.18.25.10	pod10-awsadmin	pod10-ravpn	CLEMEA2023Party!
11	172.18.26.11	CLEMEA2023Party!	172.18.25.11	pod11-awsadmin	pod11-ravpn	CLEMEA2023Party!
12	172.18.26.12	CLEMEA2023Party!	172.18.25.12	pod12-awsadmin	pod12-ravpn	CLEMEA2023Party!
13	172.18.26.13	CLEMEA2023Party!	172.18.25.13	pod13-awsadmin	pod13-ravpn	CLEMEA2023Party!
14	172.18.26.14	CLEMEA2023Party!	172.18.25.14	pod14-awsadmin	pod14-ravpn	CLEMEA2023Party!
15	172.18.26.15	CLEMEA2023Party!	172.18.25.15	pod15-awsadmin	pod15-ravpn	CLEMEA2023Party!
16	172.18.26.16	CLEMEA2023Party!	172.18.25.16	pod16-awsadmin	pod16-ravpn	CLEMEA2023Party!
17	172.18.26.17	CLEMEA2023Party!	172.18.25.17	pod17-awsadmin	pod17-ravpn	CLEMEA2023Party!
18	172.18.26.18	CLEMEA2023Party!	172.18.25.18	pod18-awsadmin	pod18-ravpn	CLEMEA2023Party!
19	172.18.26.19	CLEMEA2023Party!	172.18.25.19	pod19-awsadmin	pod19-ravpn	CLEMEA2023Party!
20	172.18.26.20	CLEMEA2023Party!	172.18.25.20	pod20-awsadmin	pod20-ravpn	CLEMEA2023Party!

21	172.18.26.21	CLEMEA2023Party!	172.18.25.21	pod21-awsadmin	pod21-ravpn	CLEMEA2023Party!
22	172.18.26.22	CLEMEA2023Party!	172.18.25.22	pod22-awsadmin	pod22-ravpn	CLEMEA2023Party!
23	172.18.26.23	CLEMEA2023Party!	172.18.25.23	pod23-awsadmin	pod23-ravpn	CLEMEA2023Party!
24	172.18.26.24	CLEMEA2023Party!	172.18.25.24	pod24-awsadmin	pod24-ravpn	CLEMEA2023Party!
25	172.18.26.25	CLEMEA2023Party!	172.18.25.25	pod25-awsadmin	pod25-ravpn	CLEMEA2023Party!
26	172.18.26.26	CLEMEA2023Party!	172.18.25.26	pod26-awsadmin	pod26-ravpn	CLEMEA2023Party!
27	172.18.26.27	CLEMEA2023Party!	172.18.25.27	pod27-awsadmin	pod27-ravpn	CLEMEA2023Party!
28	172.18.26.28	CLEMEA2023Party!	172.18.25.28	pod28-awsadmin	pod28-ravpn	CLEMEA2023Party!
29	172.18.26.29	CLEMEA2023Party!	172.18.25.29	pod29-awsadmin	pod29-ravpn	CLEMEA2023Party!
30	172.18.26.30	CLEMEA2023Party!	172.18.25.30	pod30-awsadmin	pod30-ravpn	CLEMEA2023Party!

Step 1: Connect to the Lab VPN

Using AnyConnect VPN, connect to the ASAv to gain access to the inside network. Connect to IP **18.190.34.9**. Choose the group "RA_VPN" and login with the username (**podX-ravpn**) assigned to your pod in table 1-1 above. 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.

If you do not have the AnyConnect client, browse to https://18.190.34.9/ and authenticate with the information above. The AnyConnect client will be provisioned during the initial connection process.

Step 2: Establish RDP Session to Windows Provisioning Host

Establish a remote desktop connection to the Windows provisioning host associated with your pod. User credentials for this machine are username **Administrator**, password **is provided by your proctor**. This will be the host which provisions Ubuntu and provides the SSH key to be used for connectivity. Verify two files exist on the desktop:

- A shortcut to putty for connectivity: Putty will be used to establish an SSH session to Ubuntu where ansible and other tasks will be performed.
- A Pod Important Information Document: The Important information document contains the environmental variables for Ansible. These will be used in future steps.

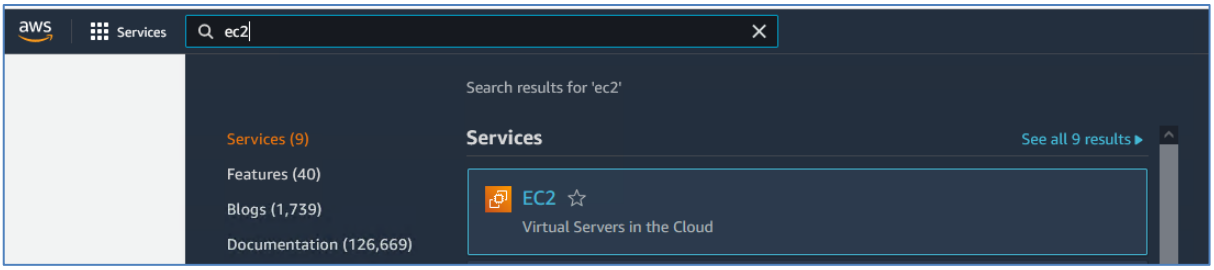
Step 3: Connect to AWS

Open the Firefox web browser on the desktop of the windows provisioning machine and utilize the link **AWS Sign In** on the bookmarks toolbar to connect to the AWS cloud under the zer0k account. If this is not shown or is unavailable, 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 **is provided by your proctor.** You may be prompted to switch to the new AWS home

experience, select “Switch to the new Console Home”. Anytime you are prompted to use the new AWS experience select to do so as the lab guide is written based on this.

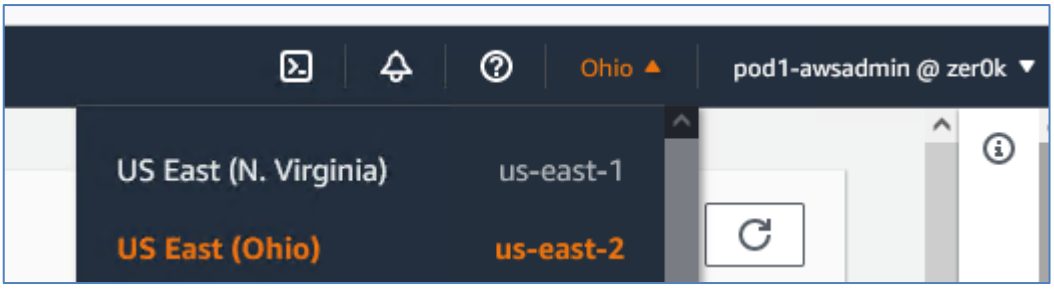
Step 4: EC2

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.



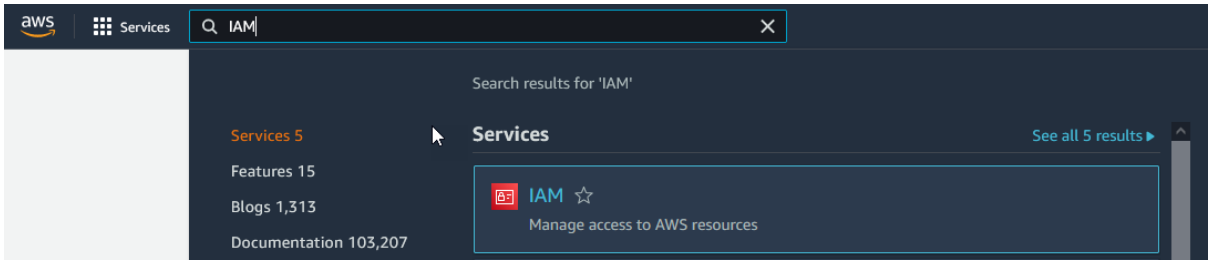
Step 5: Change AWS Regions

The Ansible lab is built within the 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 EAST-2, or “Ohio”.



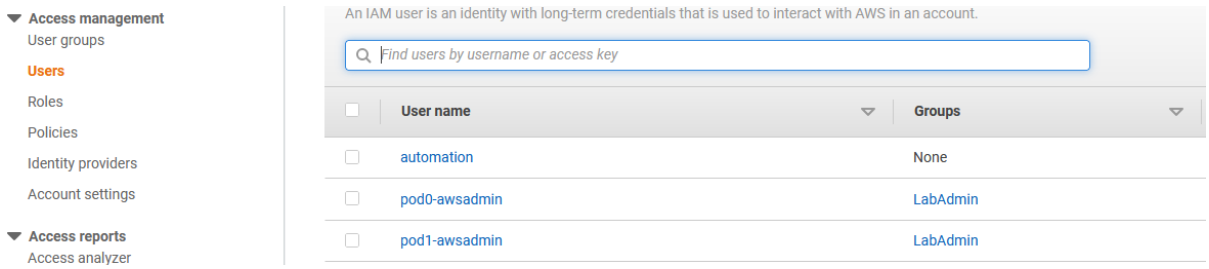
Step 6: Generate your AWS Access and Secret Key

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



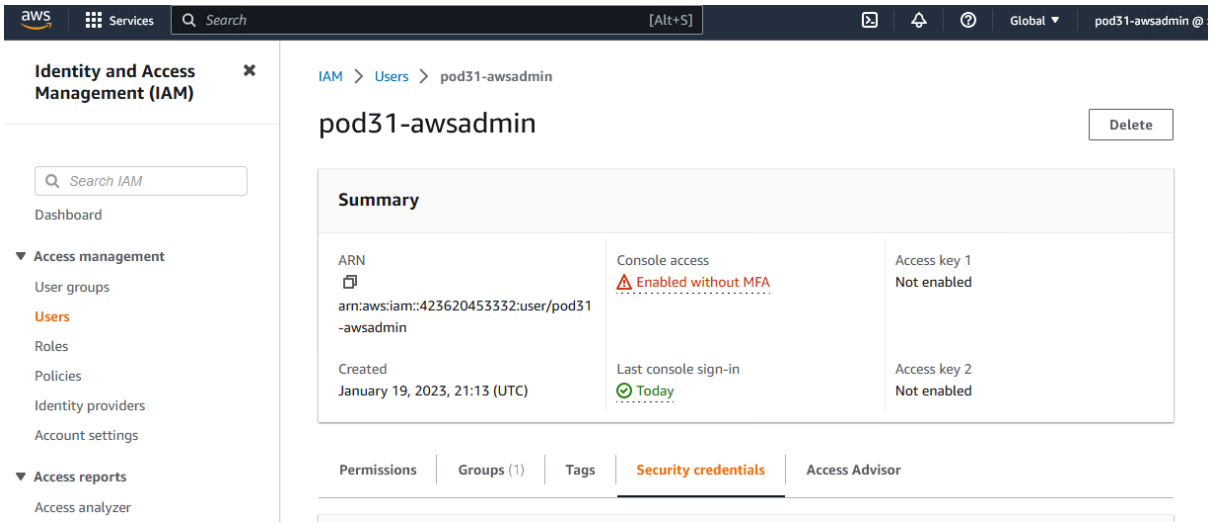
From the left bar, click “Users” and select your pod username.

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

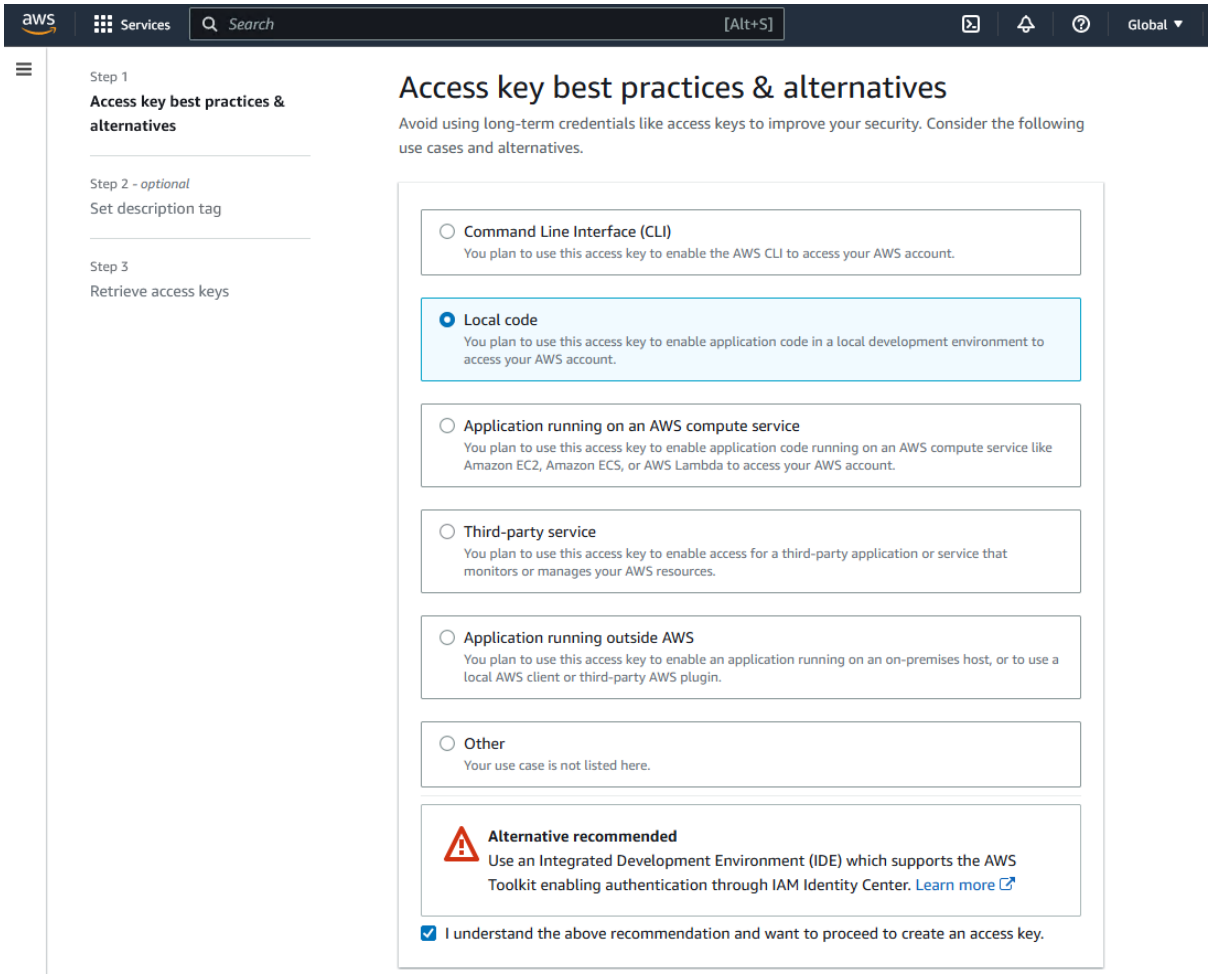


Warning: Common mistake area!

Select “Security Credentials” and under Access Keys, select “Create Access Key”.



You will use this access key for your environmental variables used in future steps so that ansible 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”.



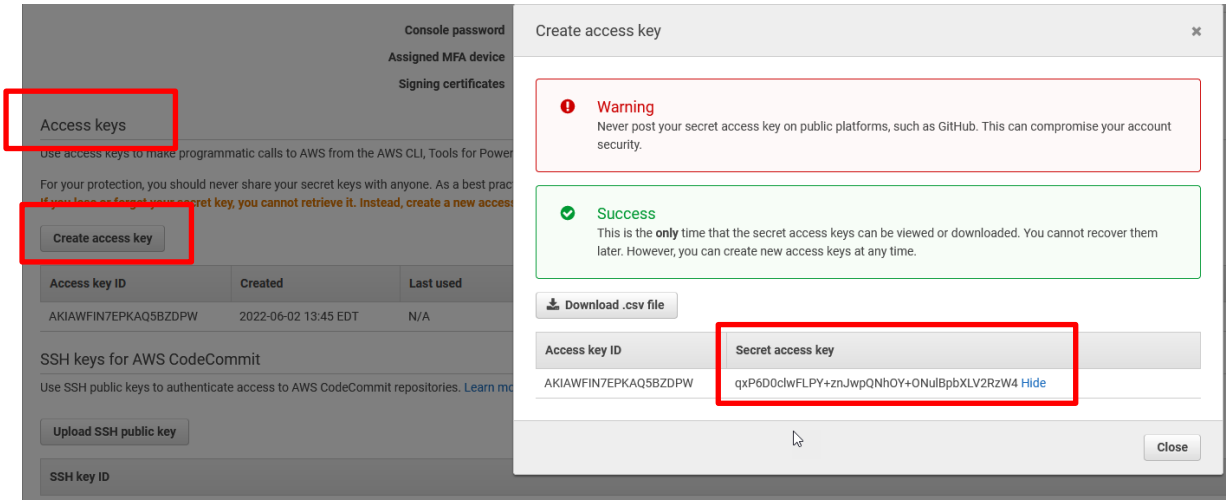
Click Next.

For the tag description enter “Pod <Your pod number> local code access key”

Click “Create Access Code”

Click “Show” to show this secret key. **Ensure you copy the secret key, as it will not be available again. We recommend placing this in your “Pod<#> - Important Information” file.**

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



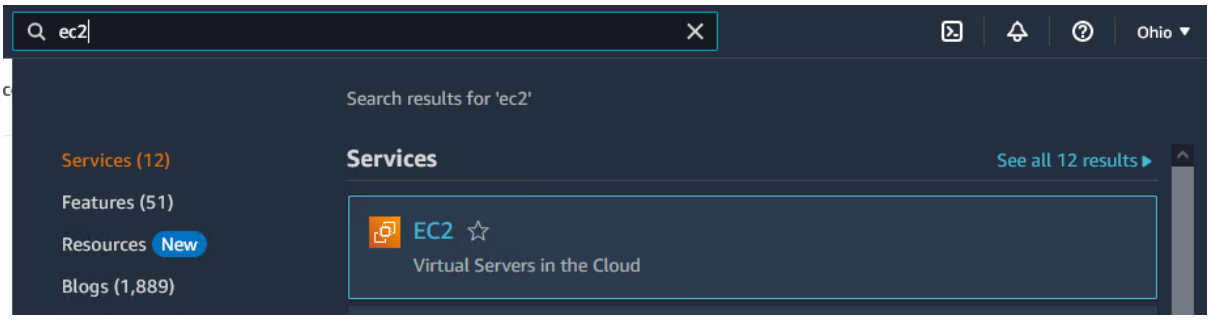
Task 2: Deploy Ubuntu Provisioning Machine / Building Blocks

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

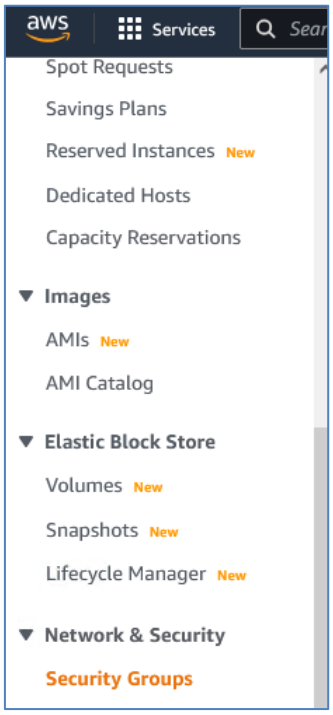
Security Groups are AWS ACLs that control network traffic, by default all outbound network traffic from a security group is permitted as well as the return traffic from that outbound connection (stateful).

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



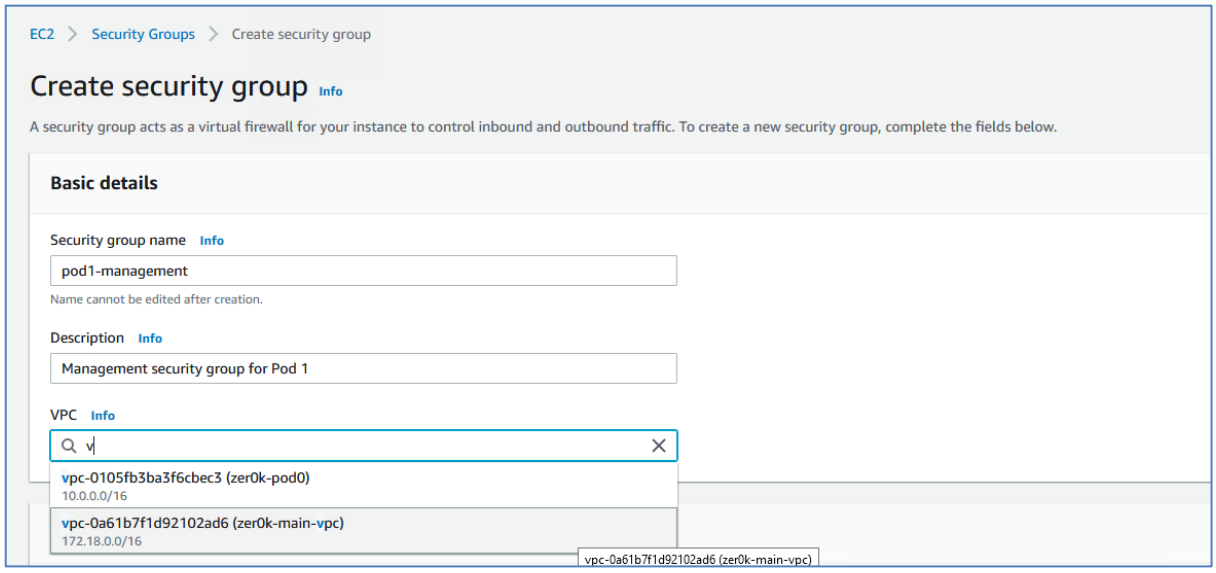
2. Ensure you are using the “New EC2 Experience indicated at the top left of the screen.

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

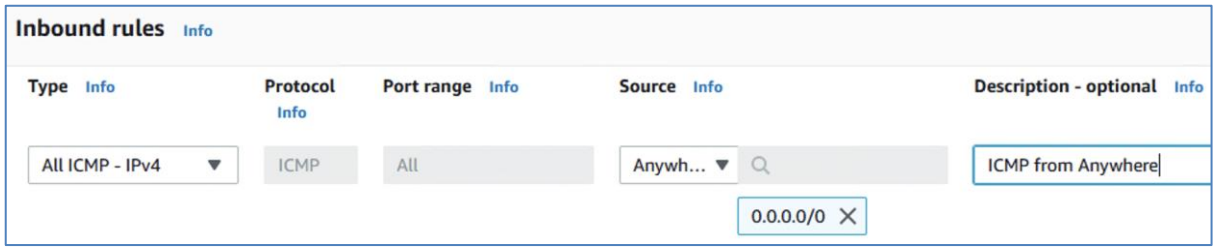


4. Select “Create Security Group” from the upper right-hand side of the page.
5. Name the security group podX-management where X is your assigned pod number, add a required description, we recommend “Management security group for pod <X>”. See the screenshot in step 6 for verification.
6. Make sure the VPC is set to vpc-0a61b7f1d92102ad6 (zer0k-main-vpc) since this is where the provisioning of additional VPCs will be done from.

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



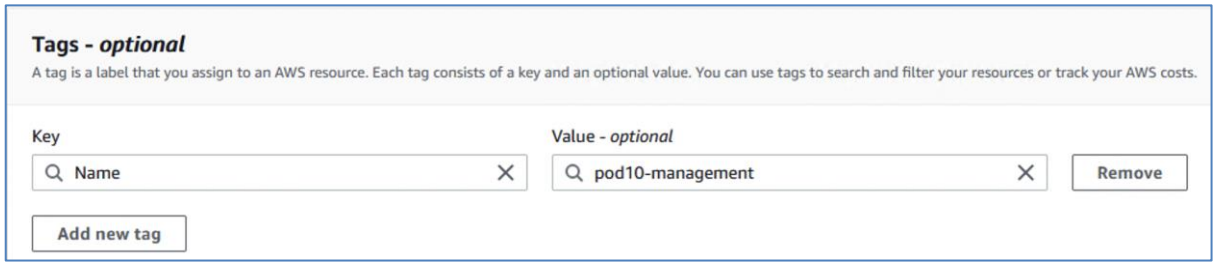
7. Add an inbound rule allowing ICMPv4 traffic from all IPs.



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

8. Do not add any other rules at this time.
9. Under Tags, add a new tag. Enter "Name" under Key and podX-management under value where X is your pod number.
10. Under Tags, add a second tag. Enter "event" under Key and "CLEMEA2023" under value.
11. Under Tags, add a third tag. Enter "Project Name" under Key and "ISEinAWS-podX" under value, where X is your pod number.

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



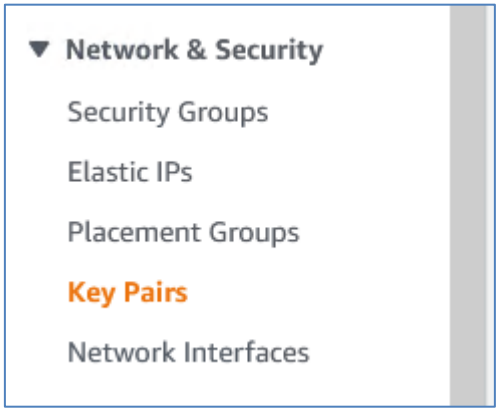
12. Create the security group.

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.

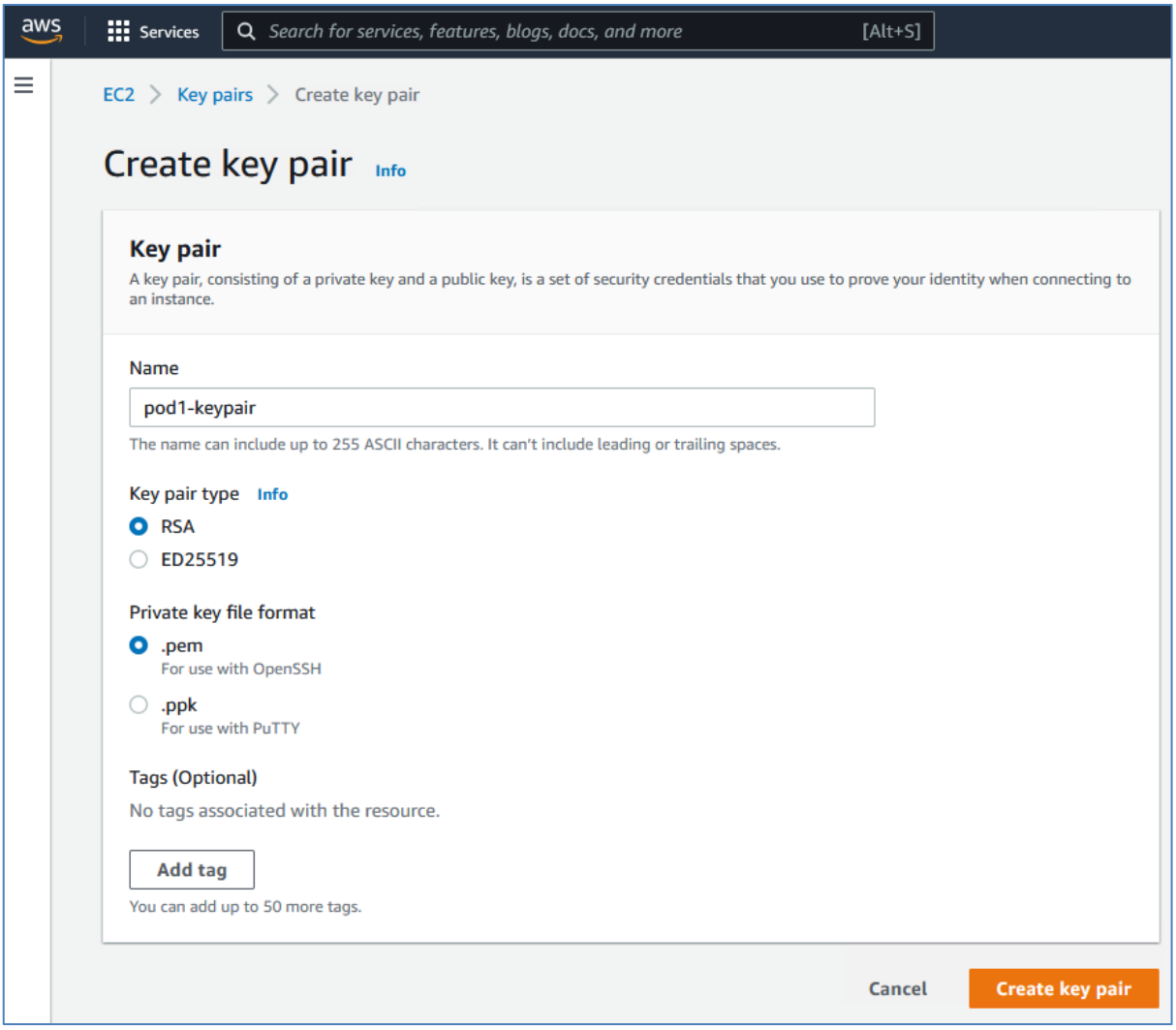
Note: For the following step, the private key created is NEVER available for download again, if it is lost a new key pair must be created.

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 is used. In this case the AWS UI populates the Name tag as an option, so no tags are required.



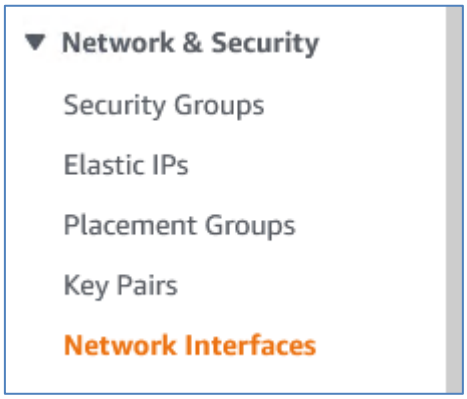
4. Select "Create key pair" and ensure the .pem file is saved to your Windows Provisioning Machine. Downloading the file as .pem will allow it to be used from a Linux or Mac host and can be converted later for use in Putty. This should happen automatically.

This private key is never available for download again, if it is lost a new key pair must be created.

Step 3: Create Network Interface

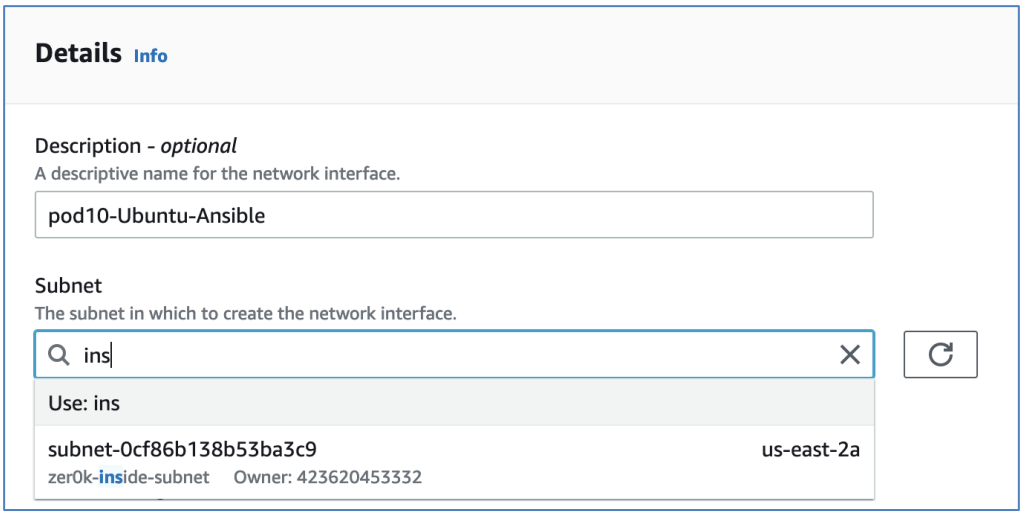
Double check that the Ohio region is chosen in the top right of the AWS console.

1. In the EC2 service, browse to Network & Security -> Network Interfaces.

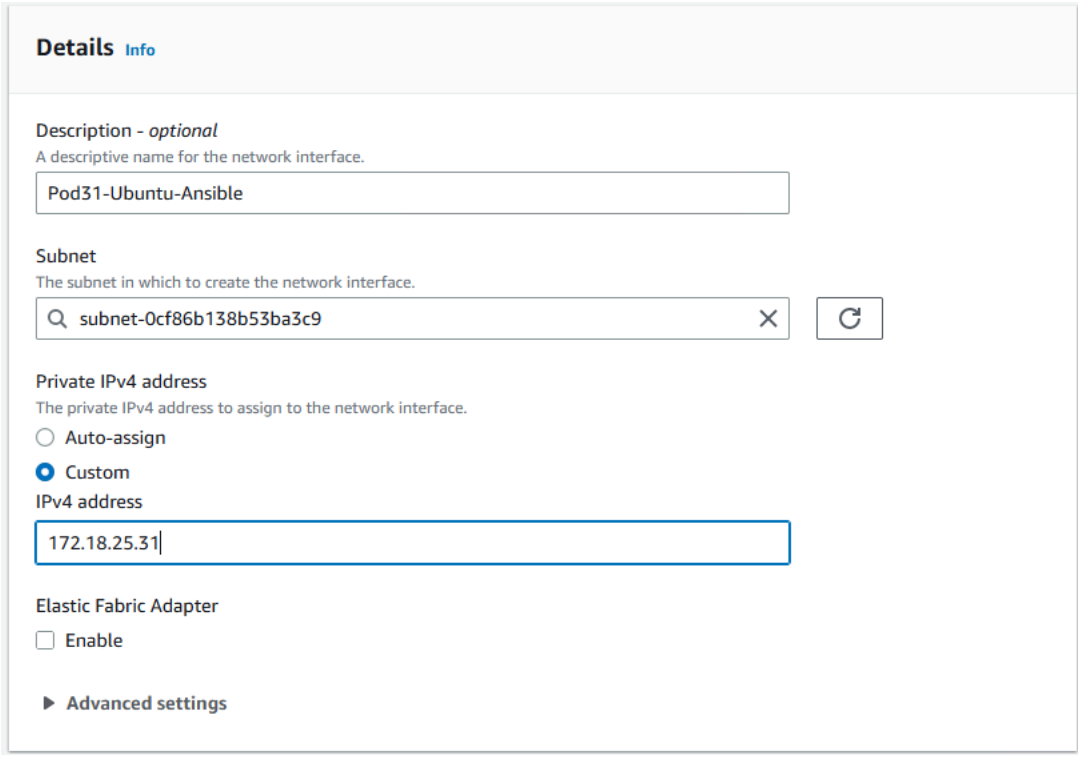


2. Select “Create network interface”.
3. Enter a description of “podX-Ubuntu-Ansible” with X being your pod number
4. Under subnet, choose subnet-0cf86b138b53ba3c9 (zer0k-inside-subnet).

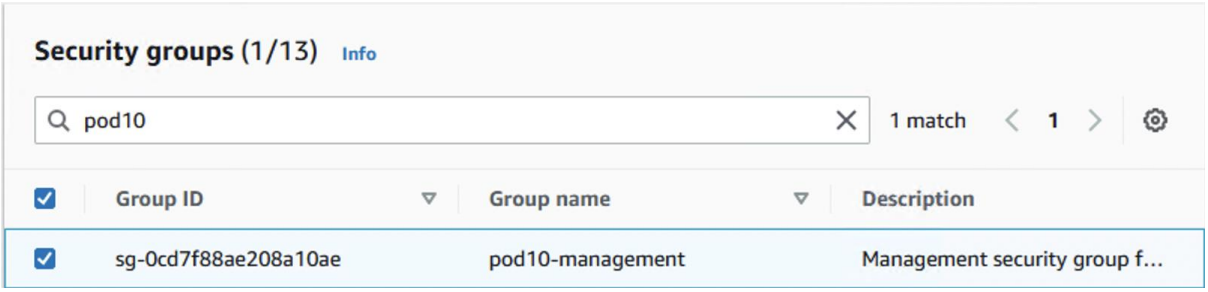
Note: You should be able to delete the current VPC name and begin typing “zer0k” to show available Subnets.



5. Under Private IPv4 address, choose custom and enter the IP address of the Ubuntu Ansible Host from table 1-1 above from your assigned pod. It will be 172.18.25.X where X is your assigned pod number.



6. Under Security Groups choose the security group created in step 1 of this task above, it should be podX-management where X is your pod number. The filter box can be used to find security groups with your pod number in the name.



Note: There are multiple pages in the security groups table. Navigate through them using the arrows next to the search bar, or search for the proper security group.

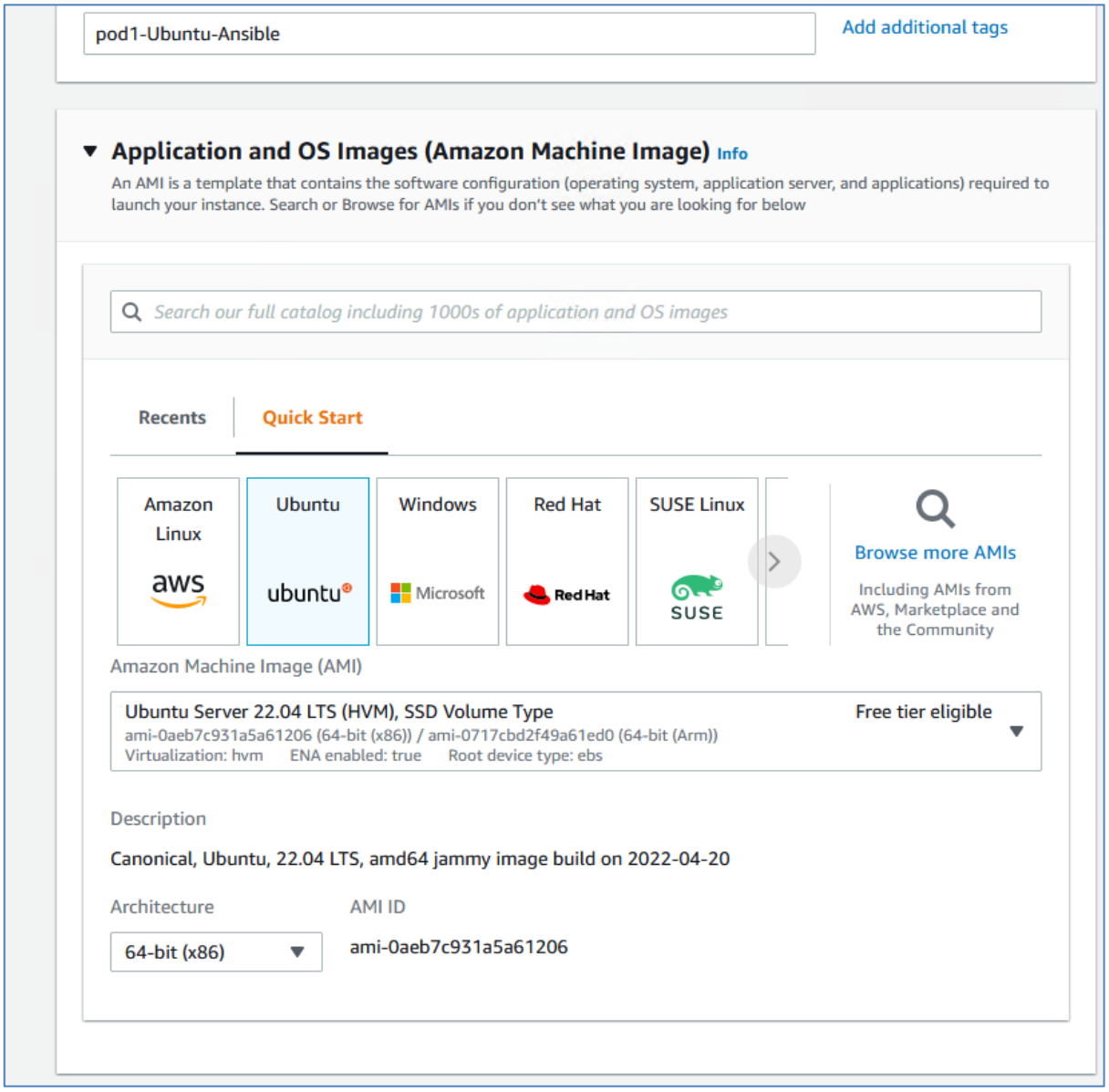
7. Under Tags, add a new tag with a key of “Name” and a value of “podX-Ubuntu-Ansible” where X is the number of your assigned pod.
8. Under Tags, add a new tag with a key of “Event” and a value of “CLEMEA2023” where X is the number of your assigned pod.
9. Under Tags, add a third tag. Enter “Project Name” under Key and “ISEinAWS-podX” under value, where X is your pod number.
10. Select “Create network interface” to finish the configuration.

Step 4: Deploy Ubuntu Provisioning Machine

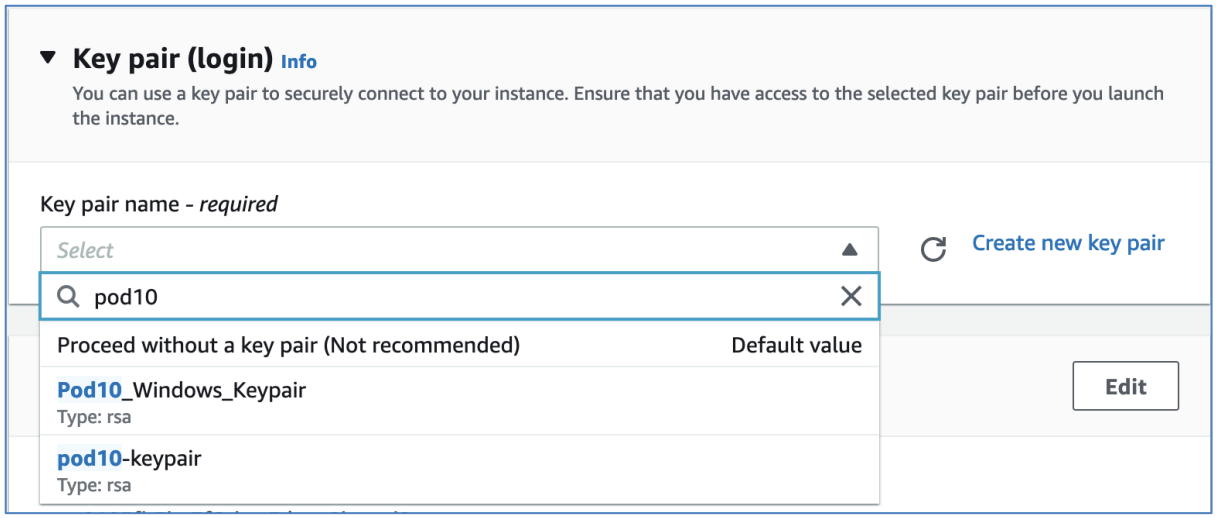
All of the building blocks are in place from a networking standpoint to create an Ubuntu linux instance. The only piece not pre-created is an EBS volume (storage) which will be created as part of the current step.

1. From the Left Navigation bar, navigate to Instances -> Instances.
2. Select “Launch instances” from the upper right corner.

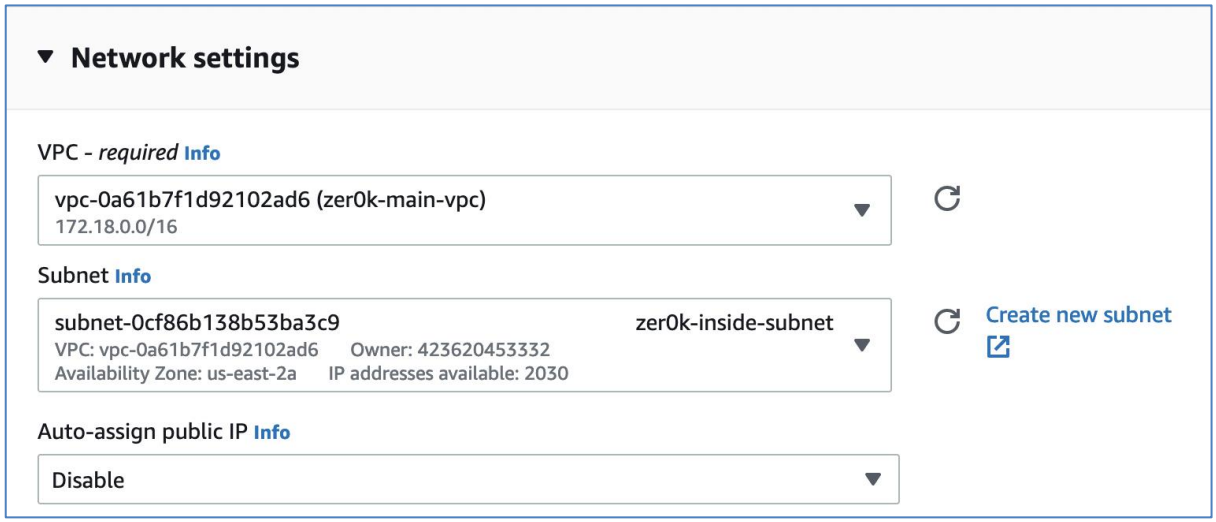
3. Under Name enter “podX-Ubuntu-Ansible” where X is the number of your assigned pod.
4. Under Application and OS Images, choose Ubuntu under quick start and leave the rest default. There should be a Free Tier Eligible Ubuntu instance chosen.



5. Under Instance Type leave it as default t2.micro.
6. 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.

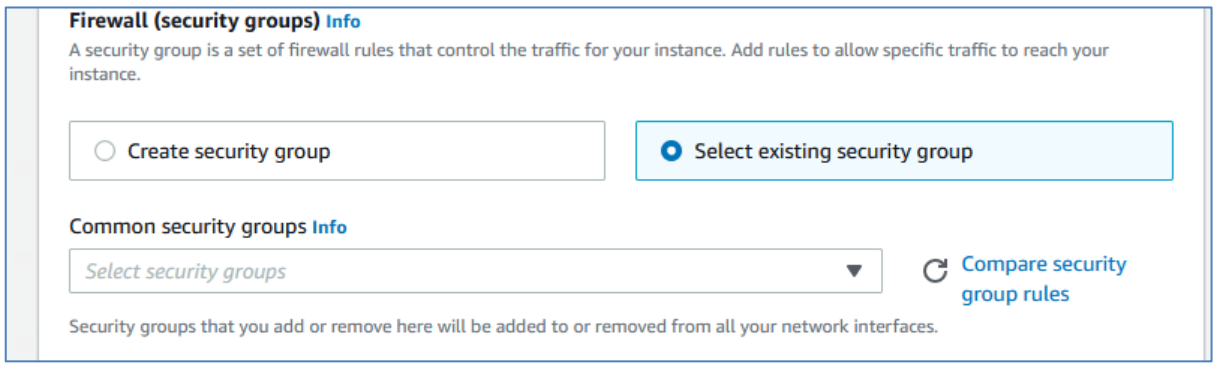


7. Under Network Settings click “Edit”.
8. Set the VPC to vpc-0a61b7f1d92102ad6 (zer0k-main-vpc).
9. Set the Subnet to subnet-0cf86b138b53ba3c9 (zer0k-inside-subnet).
10. Leave Auto Assign Public IP disabled as there is no direct internet access from this subnet.



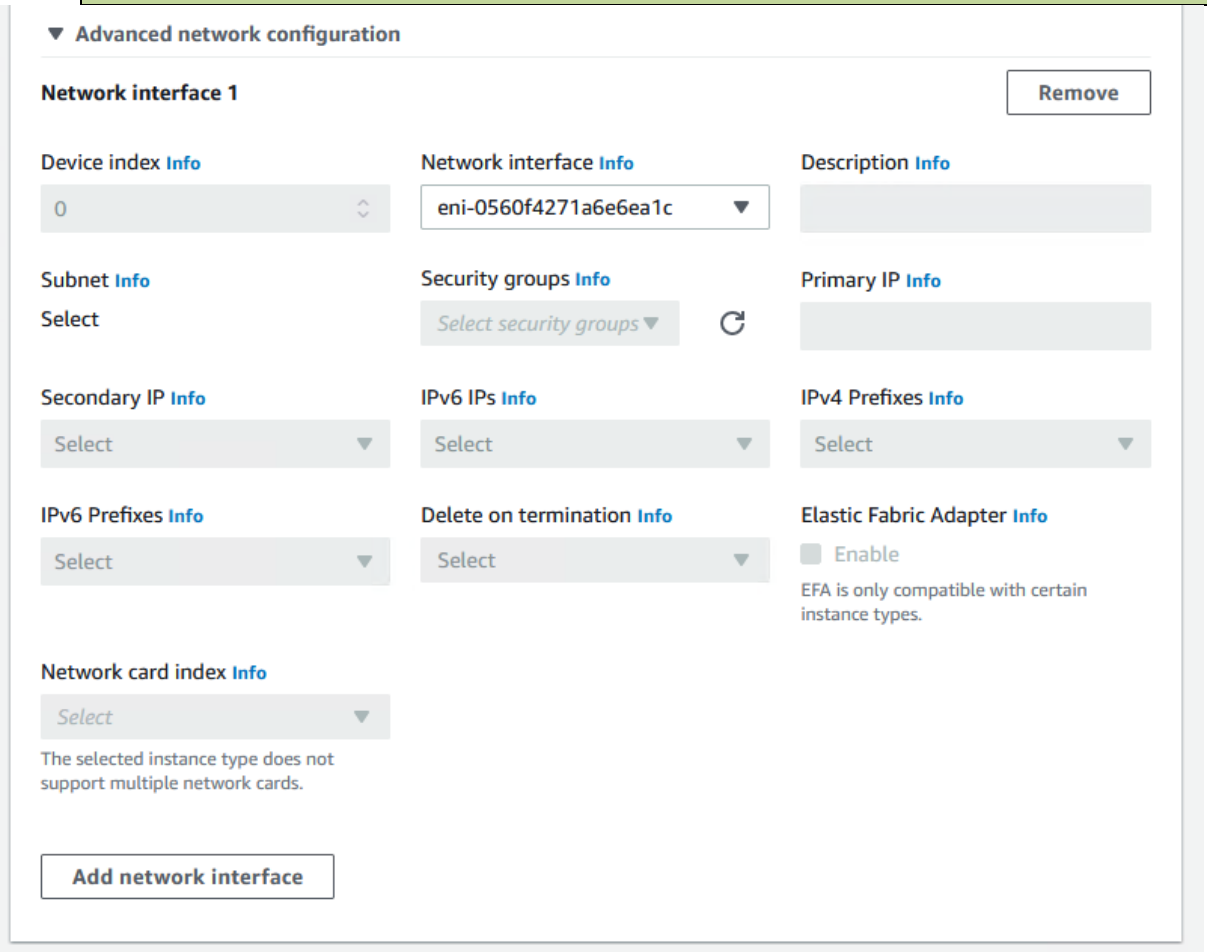
Warning: Common mistake area!

11. Choose Select existing security group but do not select a security group. This was already set under the interface in step 3! This is a quirk of AWS since normally it would just be created or selected here instead of the previously created interface.



12. Expand “Advanced network configuration”.
13. Under the Network Interface, select the network interface created in step 3 of this task. It should have been named podX-Ubuntu-Ansible, and the box should allow for searching on that name.

You can filter for the appropriate interface by typing podX- in the box and select the interface accordingly.



14. Leave the options under Configure storage and Advanced details as defaults.
15. Select “Launch instance” on the bottom right of the page.
16. If the creation of the instance is successful, click on “View all instances” to go back to the instance list. Otherwise, notify a proctor.
17. In the EC2 service, browse to Instances -> Instances.
18. Verify that the status check for your podX-Ubuntu-Ansible 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 5: 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.

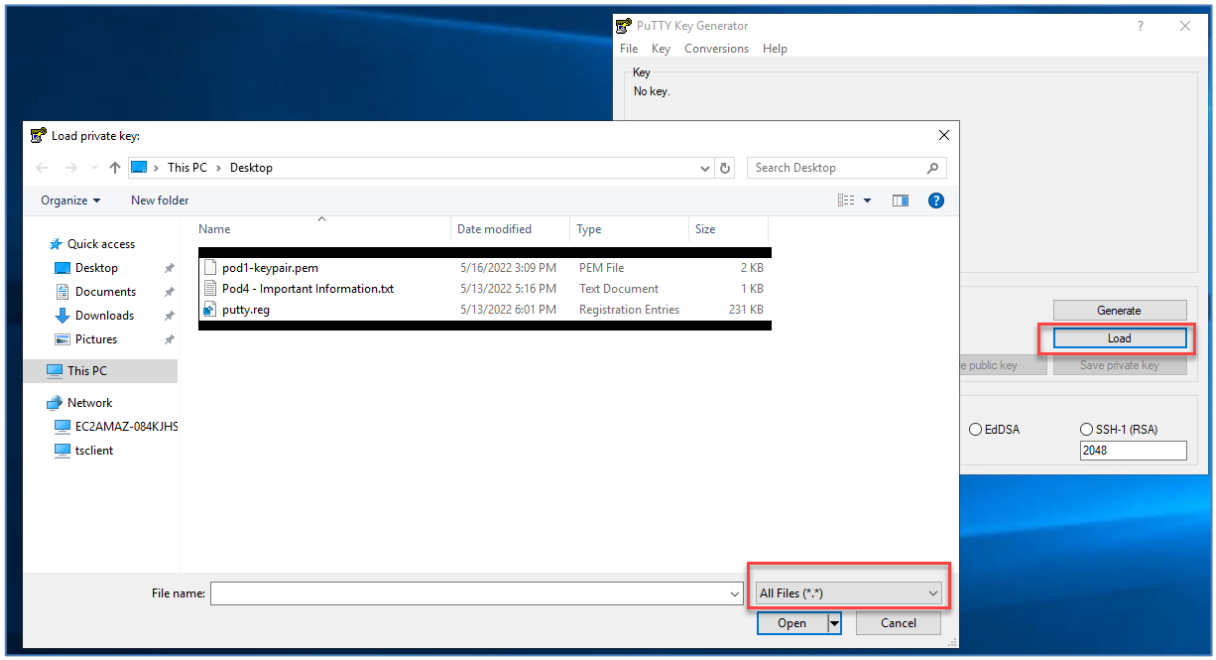
Locate the PEM file on the Windows Provisioning Machine, it is recommended

you move it to the desktop for easy retrieval. By default, Firefox will put it in the Downloads folder.

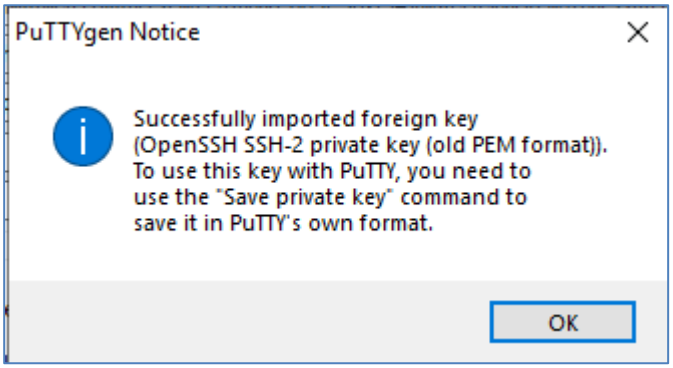
Putty on Windows:

PuTTY is installed on the Windows Provisioning Host 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/

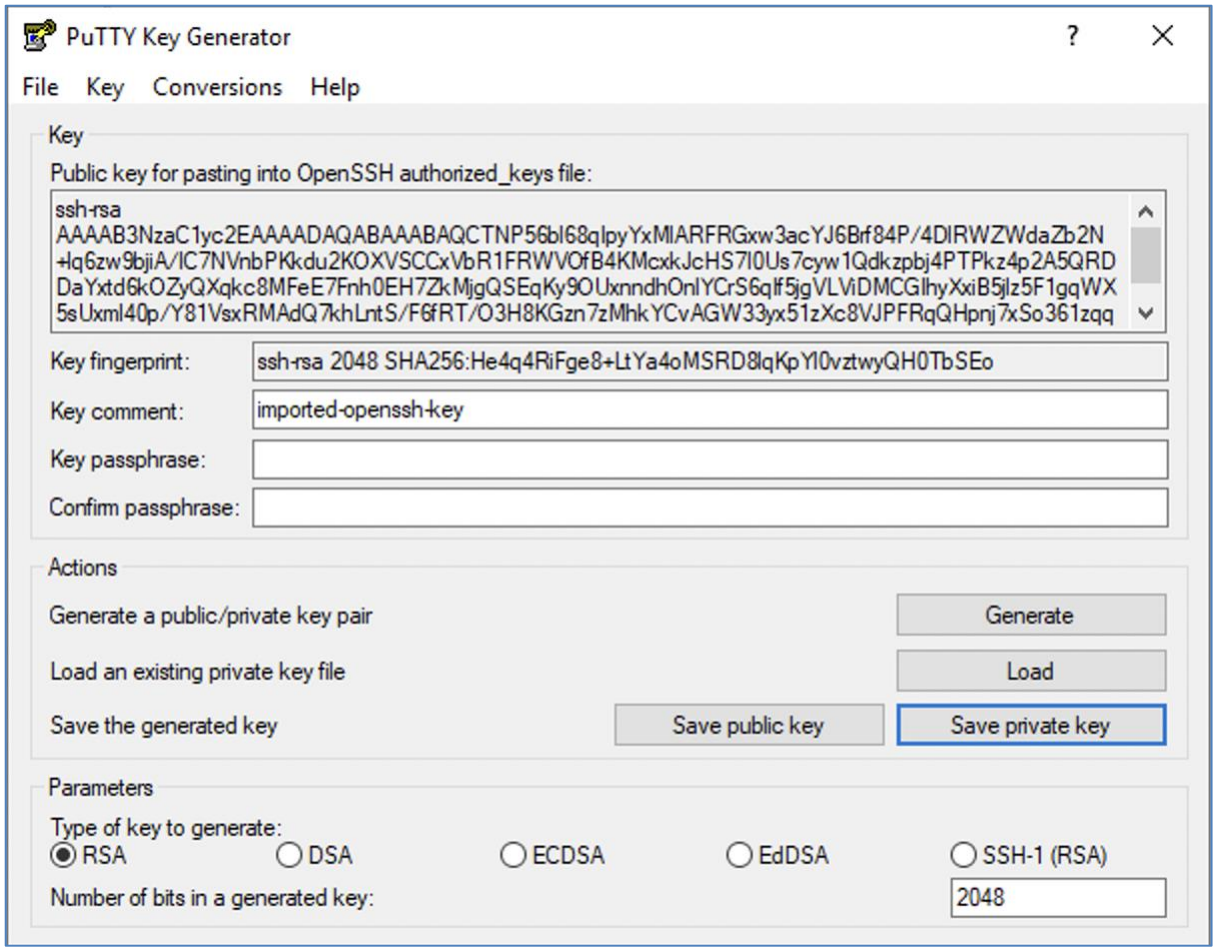
1. In the start menu of the Windows Provisioning Machine, navigate to PuTTY -> PuTTYgen and open PuTTYgen.
2. Click "Load" and change the file type to "All Files (*.*)" to select the PEM keypair you created in the last task, it should have been named podX-keypair.pem where X is your assigned pod number.



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.

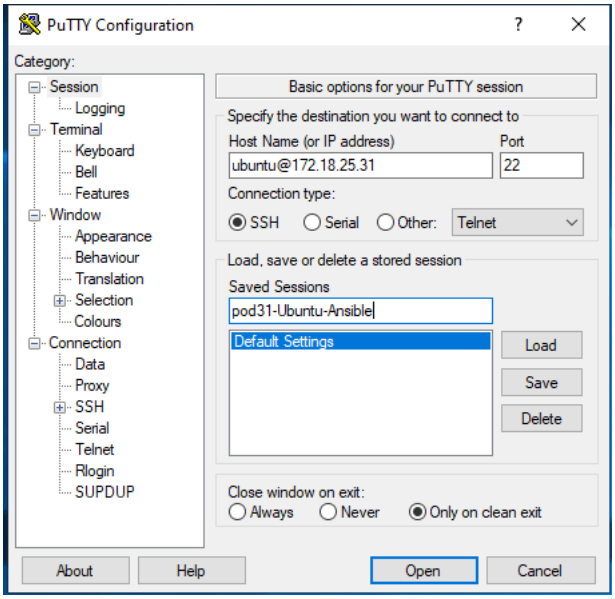


5. Close PuTTYgen.
6. This converted key will be used later in the next step.

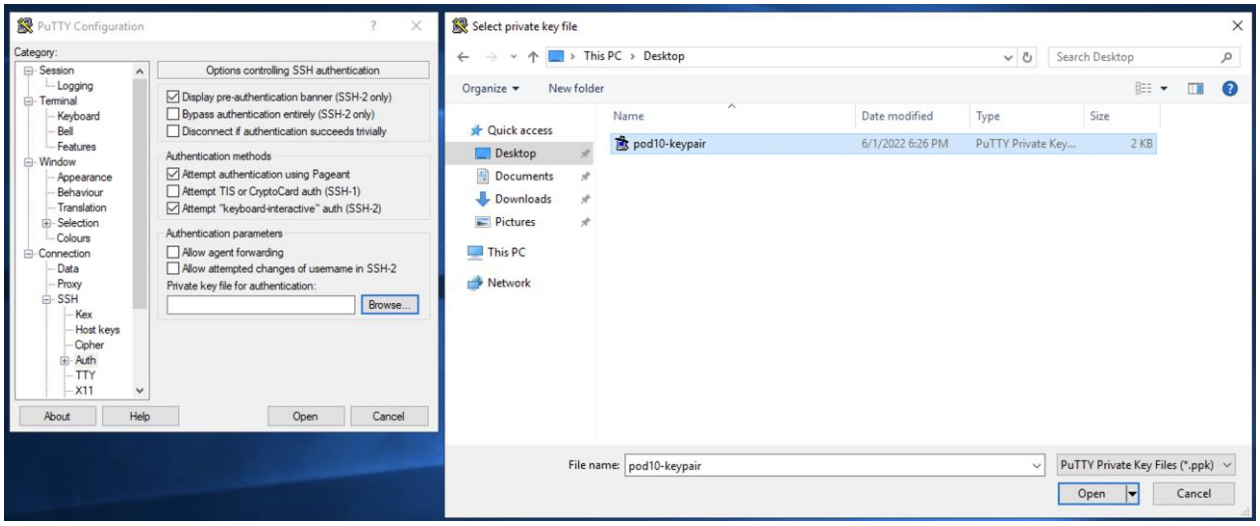
Step 6: Configure Putty for SSH on Local Machine

In this section the SSH client is just being setup, SSH will not work until Step 6 is completed.

1. On the Windows provisioning host, 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-Ansible” 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.

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:

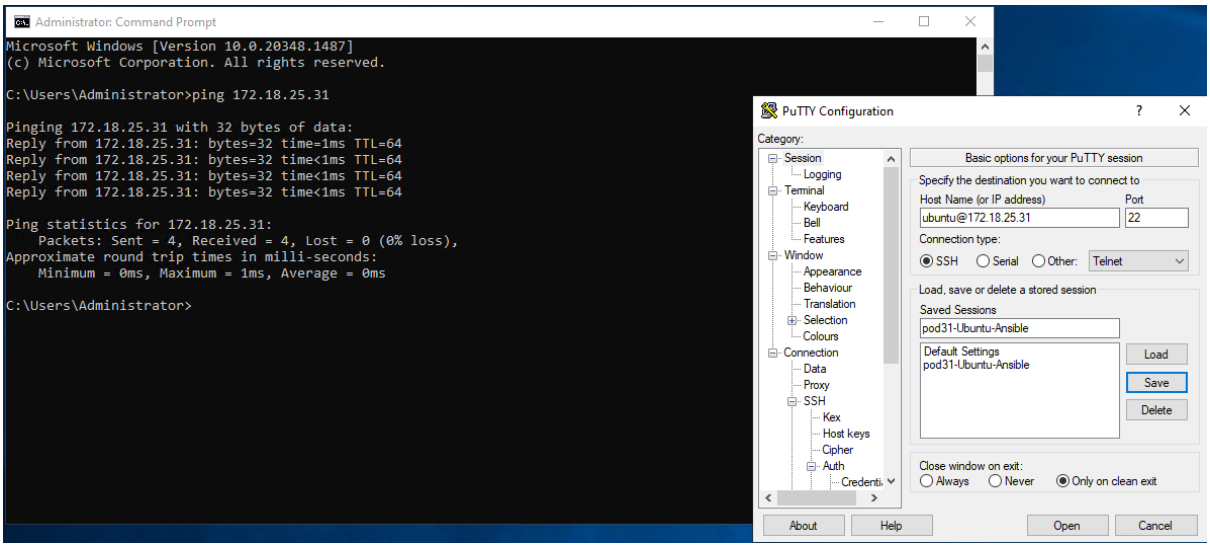
> *chmod 400 <keyfile>*

To use the private key file use ssh -i

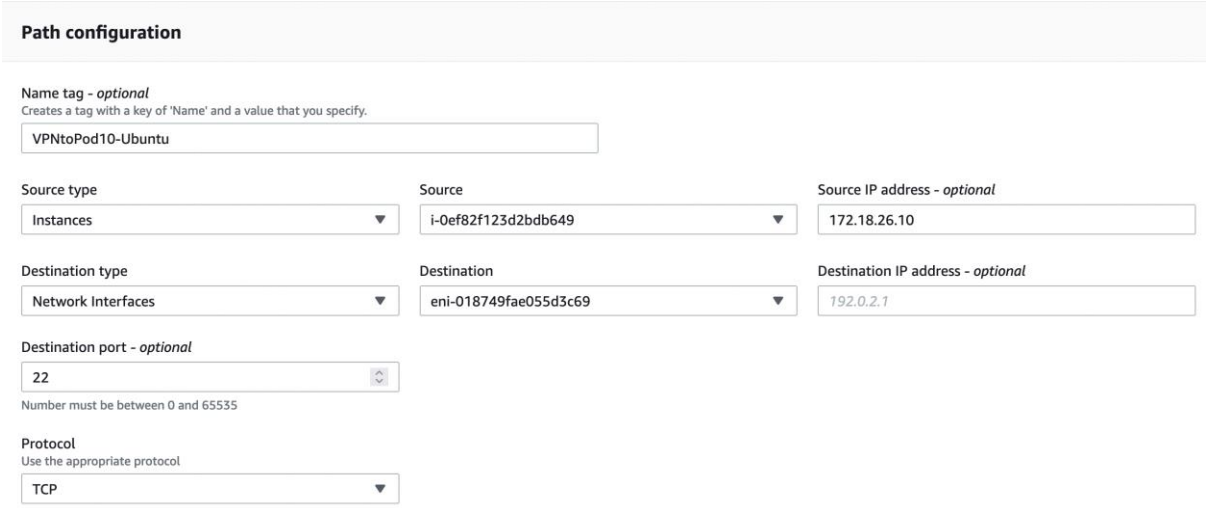
> *ssh -i <keyfile> <username>@<host>*

Step 7: Verify Connectivity

1. From your Windows provisioning machine, ping your podX-Ubuntu-Ansible 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. Attempt to SSH to your podX-Ubuntu-Ansible machine from your Windows host using the previously set up profile. It timed out!
3. On the AWS EC2 console -> Instances, select your podX-Ubuntu-Ansible machine and click on the networking tab below. There should be a message to run the Reachability Analyzer, select this. If you don't see the option, use the search feature to search for VPC Reachability Analyzer, click Create and analyze path.
4. Name it "VPNtoPodX-Ubuntu" where X is the pod number you were assigned.
5. Set the source type to "Instances" and locate the instance named "PodX_Provisioning-Server", enter the IP address as 172.18.26.X which is your Windows Provisioning host.
6. Set the destination type to "Network Interfaces" and locate the network interface named "podX-Ubuntu-Ansible" where X is your pod number. Also leave the IP blank.
7. Set the destination port to "22" for SSH and the protocol as TCP.

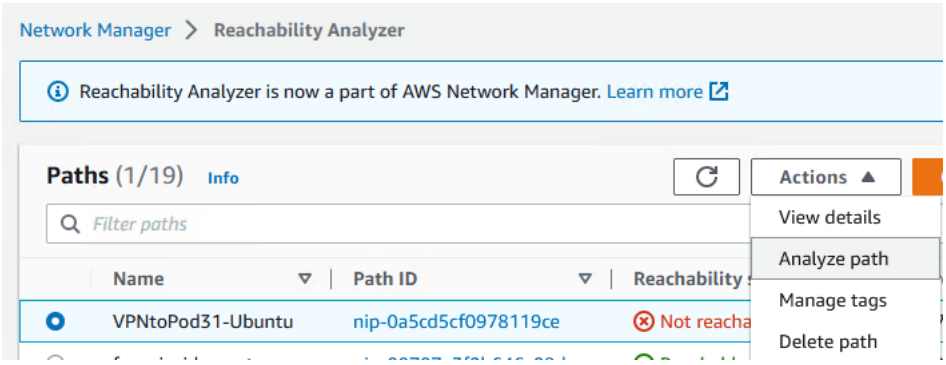


8. Select Create and analyze path in the bottom right.
9. You will see the Analyses as pending, refresh the page until you see success. The reachability status should be Not Reachable.

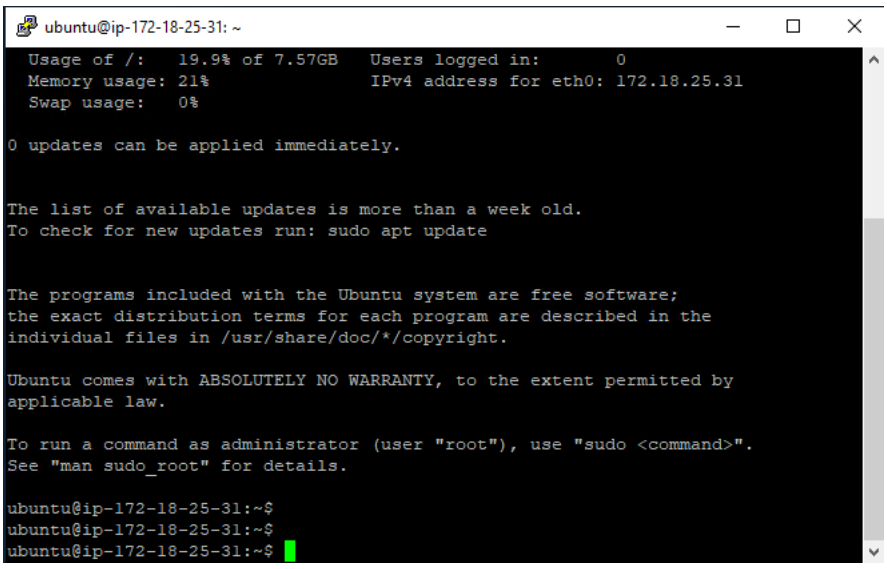
10. Check under Explanations to see why AWS thinks your instance isn't reachable on port 22. It should be because of a missing security group rule.

11. Click on the link to the security group and add another inbound entry for SSH from anywhere, all hosts are behind the ASA firewall already so allowing from anywhere is fine for this lab.

12. Go back to the VPC Reachability Analyzer using the search bar, and select the radio button for your path analysis. From the Actions menu select Analyze Path, click confirm.



13. 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.
14. From your Windows machine, attempt to SSH to your provisioning machine, it should succeed this time and you should be at an ubuntu command prompt.



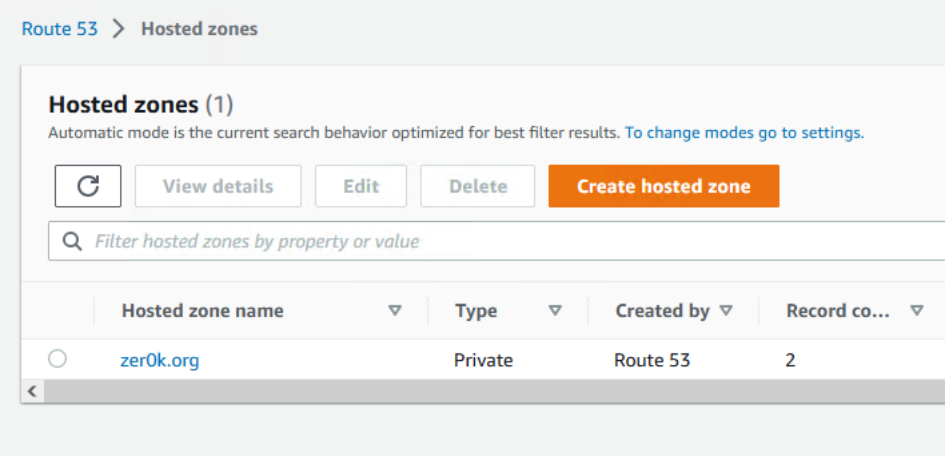
Task 3: Configure DNS in Route 53

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

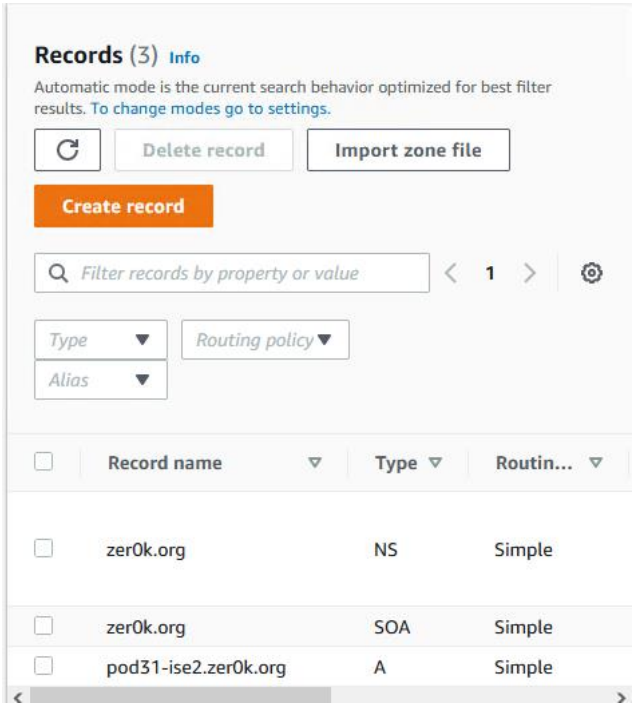
Figure 4-1

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 4: VPC Automation

Step 1: Deploy Ansible

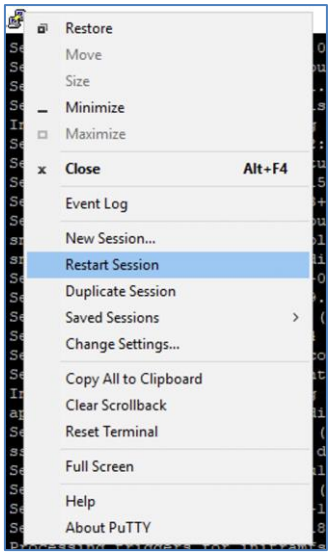
First the ansible/python environment needs to be set up on the Ubuntu server. If you are not connected to your pod's Ubuntu server, do that now. The private key currently only exists on the Windows Provisioning machine so use the previously setup putty profile to connect.

1. Open PuTTY and load the saved session created previously.

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***.
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. There are some services that cannot be restarted individually so 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.

6. 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.



7. This lab will use a python virtual environment. Venv is not installed by default so install it: ***sudo apt install python3.10-venv***. A virtual python environment allows for different projects on the same machine to use different versions of shared python libraries. Accept any prompts presented.
8. Create a venv for this lab, this will create a new folder in the /home/ubuntu directory: ***python3 -m venv ISEonAWS***.
9. 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.
10. Install the python packages required to run ansible for ISE: ***python -m pip install ansible boto boto3 botocore ciscoisesdk jmespath***. This step will take a few minutes.
11. Add the ISE collection to ansible: ***ansible-galaxy collection install cisco.ise***.

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

   git clone https://github.com/plloyd44/CiscoLive_ISE_in_AWS.git

   Cloning into ' *CiscoLive_ISE_in_AWS* '...

   …

   Receiving objects: 100% (228/228), 190.28 KiB | 17.30 MiB/s, done.

   Resolving deltas: 100% (125/125), done.

2. There should be 2 directories in your home directory at this point, run ***ls -l*** to verify.

```
(ISEonAWS) ubuntu@ip-172-18-25-10:~$ ls -l
total 8
drwxrwxr-x 7 ubuntu ubuntu 4096 Jun  1 20:33 CiscoLive_ISE_in_AWS
drwxrwxr-x 5 ubuntu ubuntu 4096 Jun  1 20:27 ISEonAWS
(ISEonAWS) ubuntu@ip-172-18-25-10:~$
```

Step 3: Configure the 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 the Python virtual environment running Ansible. There are two areas where variables are stored for the script to work, the first is locally in the vars/main.yaml file, the second is as deployed into the environment via export statements.

The following is to be done within the SSH session to Ubuntu:

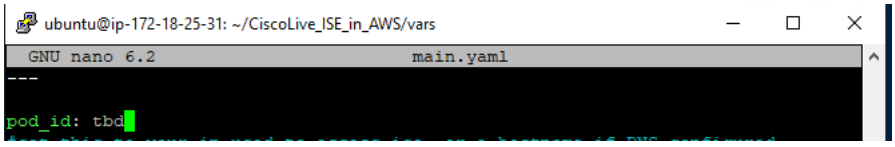
1. Edit the vars/main.yaml file to update your pod number, which will be utilized throughout the lab. Execute the commands:

   cd CiscoLive_ISE_in_AWS

   cd vars

   nano main.yaml

2. Edit the line in main.yaml ***pod_id:*** to match your pod number. Exit and save from nano utilizing the key sequence “Ctrl-X, Y, <enter>”



3. Once back at the Ubuntu command line, execute the export commands found in the file “PodX Important Information.txt” on the Windows desktop. It should look like the following:

   export ise_username=admin

   export ise_password=CLEMEA2023Party!

   export AWS_REGION=us-east-2

export AWS_ACCESS_KEY= XXXXXXXXXXXXXXXXXXXX

export AWS_SECRET_KEY= YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY

export ise_verify=false

4. Replace the access key and secret key with the one you created in Task 1, Step 6.
5. Paste this block of text into the Ubuntu CLI.

Note: Right click in PuTTY will paste commands, do not use ctrl+v

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

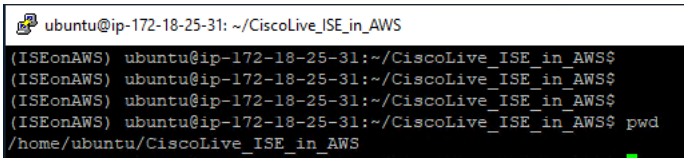
```
(ubuntu) root@ip-10-0-1-217:/home/ubuntu# env
…
ise_password=CLEMEA2023Party!
AWS_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXX
AWS_SECRET_KEY=YYYYYYYYYYYYYYYYYYYYYYYY
ise_username=admin
…
AWS_REGION=us-east-2
…
ise_verify=false
```

Step 4: Deploy the AWS Network Environment and Primary ISE Node

Configure the AWS environment to create a network into which ISE will get deployed.

1. View the playbook that will create the ssh key pair ansible will use to install and connect to ISE. This task will generate a new key and save it for use later locally.
   cd /home/ubuntu/CiscoLive_ISE_in_AWS/
   cat ssh_key_pair.yaml

2. Ensure you are in the ~/CiscoLive_ISE_in_AWS directory of your Ubuntu server, use “cd ..” to back up one directory if need be. Verify your currently working directory by using the command “pwd”.



3. Generate the SSH Key to be used from the provisioning machine to access all nodes within the AWS environment. Execute the following command:

ansible-playbook ssh_key_pair.yaml

Expected Output:

PLAY [AWS VPC with Cisco ISE and Meraki vMX]
***

TASK [Gathering Facts]
***

```
ok: [localhost]

TASK [Check for existing keypair]
***
ok: [localhost]

TASK [Create EC2 SSH Key Pair]
***
changed: [localhost]

TASK [Show key_pair]
***
   msg: key pair created

TASK [Save Private Key Locally]
***
changed: [localhost]

PLAY RECAP
***
localhost                  : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

4. Verify that there are no skipped or failed steps in the play recap. If so, determine which step needs to be followed to remediate.

> Note: If the private key is not displayed, or the task is marked as "skipped", contact a lab proctor.

5. To verify a private key was generated use the command "ls ~/.ssh". A .pem extension private key should exist in this directory, which can be viewed with the command "cat ~/.ssh/ISEinAWS-podX.pem" where X is your pod number.

6. Deploy the VPC, Subnet, Routing Table, and Internet gateway for the ISE environment. This will deploy your pod VPC as shown in the initial network diagram. Execute the command:

   ansible-playbook ise_in_aws.vpc.yaml

   Expected Output:

   PLAY [AWS VPC with Cisco ISE and Meraki vMX]
   ***

   TASK [Gathering Facts]
   ***
   ok: [localhost]

   TASK [Create VPC]
   ***
   changed: [localhost]

   TASK [Create an Internet Gateway to connect VPC to Internet]
   ***
   changed: [localhost]

   TASK [Create Public_Subnet]
   ***
   changed: [localhost]

   TASK [Create Private_Subnet]
   ***
   changed: [localhost]

   TASK [Create Public Route Table; Add Route from VPC to Internet Gateway]
   ***
   changed: [localhost]

TASK [Create Private Route Table]
***
changed: [localhost]

TASK [Show VPC(s)]
***

ok: [localhost]

TASK [Create SG-ISE Security Group]
***

changed: [localhost]

7. Deploy the ISE primary node into the ISE environment. Execute the command:

   ansible-playbook ise_in_aws.ise.yaml

   Expected Output:

   PLAY [AWS VPC with Cisco ISE and Meraki vMX]

   TASK [Create ISE 3.1 Instance in AWS]
   ***

   changed: [localhost] => (item={'name': 'ise', 'instance_type': 'c5.4xlarge', 'public_dns_name': 'ise.zer0k.org', 'private_dns_name': 'ise.zer0k.org', 'dns_alias': 'ise.zer0k.org', 'private_ip': '10.0.0.5, 'role': 'Primary', 'personas': ['Standalone']})

   TASK [Show SSH Commands for ISE Instances]
   ***

   ok: [localhost] => (item={'name': 'ise', 'instance_type': 'c5.4xlarge', 'public_dns_name': 'ise.zer0k.org', 'private_dns_name': 'ise.zer0k.org', 'dns_alias': 'ise.zer0k.org', 'private_ip': '10.0.0.5', 'role': 'Primary', 'personas': ['Standalone']}) =>
    msg:
    - ping 10.192.168.44
    - ssh -i ~/.ssh/ISEinAWS-palloyd.pem admin@10.0.0.5
    - ping ise.zer0k.org
    - ssh -i ~/.ssh/ISEinAWS-palloyd.pem admin@ise.zer0k.org

   PLAY RECAP
   ***
   localhost: ok=11 changed=8 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

Step 5: Visit 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.

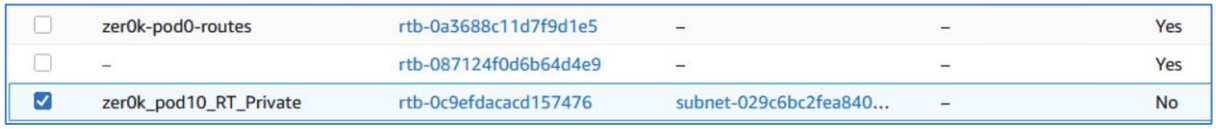
1. Verify the VPC was deployed. Navigate to the search bar in AWS, type VPC. Click VPC in the results, the VPC should be named ISEinAWS-podX where X is your assigned pod number.

☐	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 zer0k_podX_Private_Subnet where X is your assigned pod number.

☐	zer0k-public-subnet	subnet-0653589a8ba2fa824	Available	vpc-0a61b7f1d92102ad6 | zer...	172.18.0.0/21
☐	zer0k-private-subnet	subnet-059dc621173c4d170	Available	vpc-0a61b7f1d92102ad6 | zer...	172.18.16.0/21
☐	zer0k-inside-subnet	subnet-0cf86b138b53ba3c9	Available	vpc-0a61b7f1d92102ad6 | zer...	172.18.24.0/21
☑	zer0k_pod10_Private_Su...	subnet-029c6bc2fea840f4c	Available	vpc-06c27a7f1c1c05e39 | ISEi...	10.10.0.0/24

3. Navigate to Route Tables in the left menu. Verify a routing table was provisioned for your pod. It should be named zer0k_podX_Private_Subnet where X is your assigned pod number.



☐	zer0k-pod0-routes	rtb-0a3688c11d7f9d1e5	–	–	Yes
☐	–	rtb-087124f0d6b64d4e9	–	–	Yes
☑	zer0k_pod10_RT_Private	rtb-0c9efdacacd157476	subnet-029c6bc2fea840...	–	No

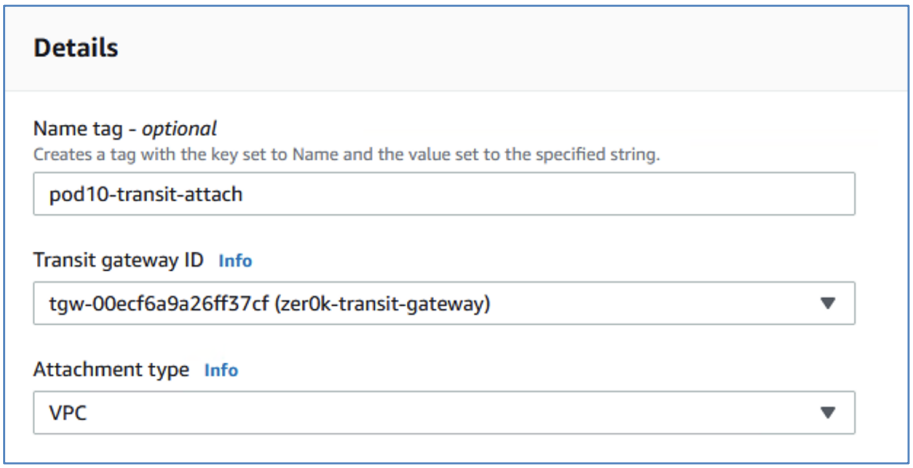
Step 6: Add Transit Gateway Attachment and Route

The lab is routing all traffic through the main VPC through the ASA. To get traffic to the main VPC from the Pod VPC a transit gateway is needed. As of the writing of this lab the ansible collection does not include the ability to attach to a transit gateway so that will be done in this step manually. The Transit gateway has already been created so all that is left is to attach to the transit gateway then update the routing in the Pod VPC to route traffic to the transit gateway.

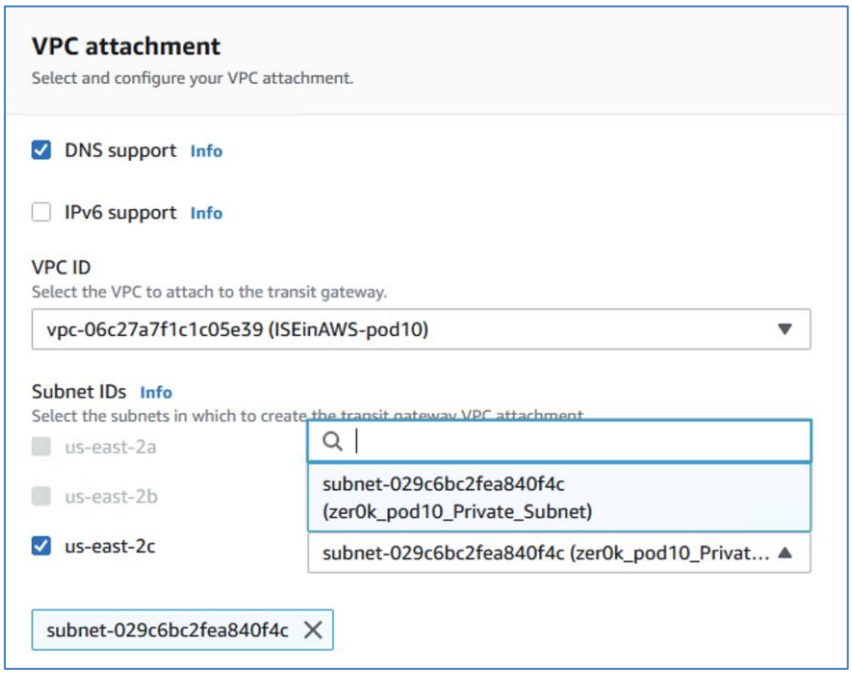
1. In the AWS console go to the VPC service -> Transit Gateways -> Transit Gateway Attachments

> Note: There are two options which are commonly mistaken here: Transit Gateway Attachments and Transit Gateways. Please choose Transit Gateway Attachments!

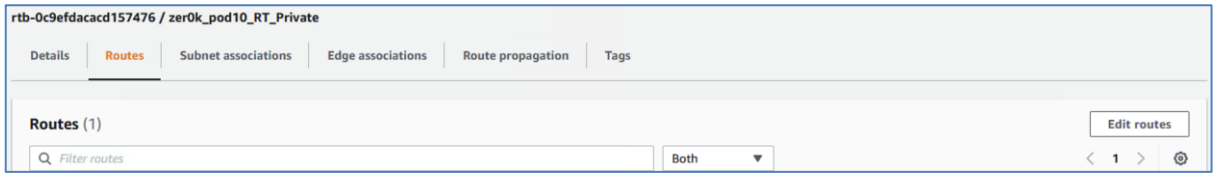
2. Click Create transit gateway attachment on the top right of the page.
3. Name it podX-transit-attach where X is your assigned pod number.
4. Choose the “zer0k-transit-gateway” under the Transit gateway ID, it should be the only entry.
5. Leave the Attachment type as VPC since you are attaching to another VPC.



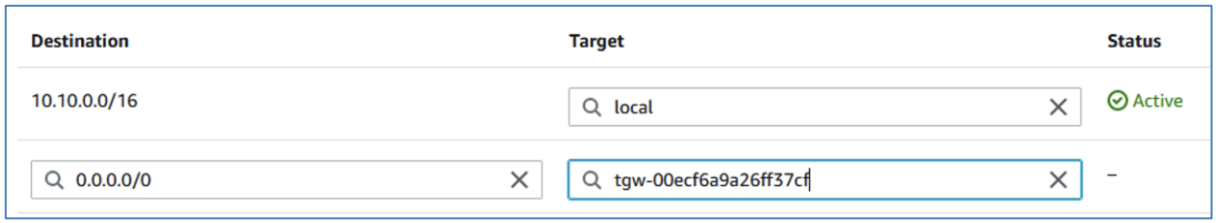
6. Under VPC attachment, leave DNS support enabled and select your VPC from the list of VPC IDs. It will be named ISEinAWS-podX where X is your assigned pod.
7. A list of subnets will be shown, select the zer0k_podX_Private_Subnet. It should be the only entry available as the VPC setup script only created a single subnet.



8. The Name tag should already be filled out from above.
9. Create an additional tag of “Project” with Value “ISEinAWS-pod31”
10. Select Create transit gateway attachment.
11. Next browse to Virtual Private Cloud -> Route Tables and find your pod VPC routing table. It is named zer0k_podX_RT_Private where X is your assigned pod number. Check the box next to the name of the Route Table.
12. The bottom half of the screen will load the Route table details, select the Routes tab and then select Edit routes.



13. Add a default route pointing to the transit gateway by selecting 0.0.0.0/0 under the Destination and by selecting Transit Gateway -> podX-transit-attach under Target where X is your assigned pod number.



14. Save the changes.

Step 7: Establish an SSH session with ISE

Note: ISE will need to initialize and can take up to 15 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. Establish an SSH session with ISE to verify services transition through a Not Running -> Initializing -> Running cycle. This should be done from the Ubuntu machine.

   NOTE: SSH will fail until ISE is fully provisioned and running. This can take up to 15 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$ ssh -i ISEinAWS-pod4.pem admin@10.4.0.5 admin@10.4.0.5: **Permission denied** (publickey)."

2. Execute the following command:

   ssh -i ~/.ssh/ISEinAWS-pod<#>.pem admin@10.X.0.5 where X is your pod number.

3. You will be prompted to accept the ssh fingerprint of the ISE server. Enter yes.

   Run the following command:

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

Step 8: 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.

Execute the command: 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

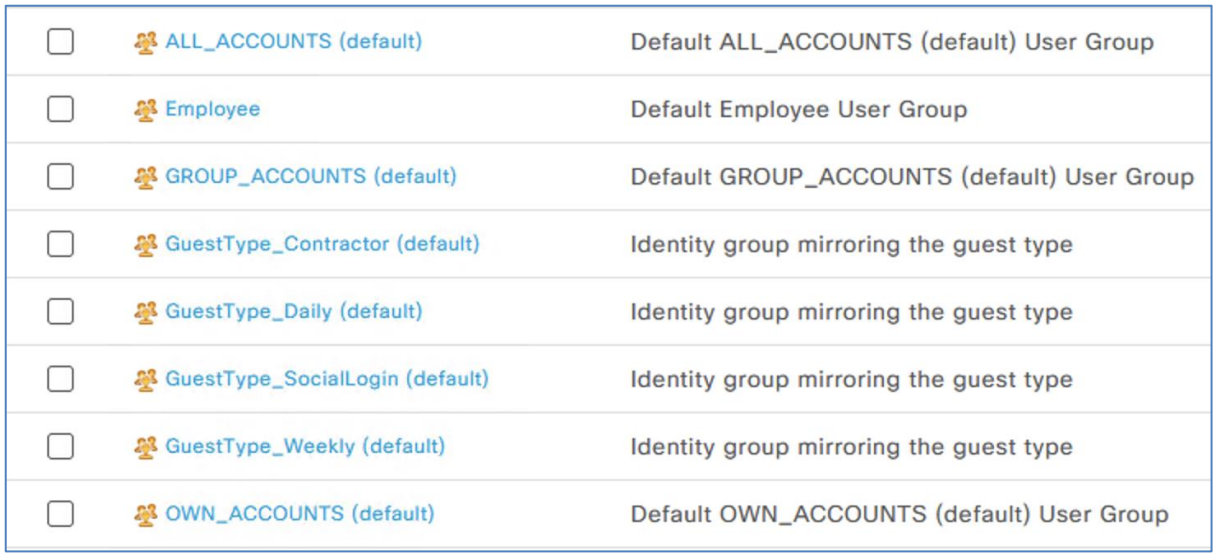
clock timezone EST5EDT

!

Step 9: Launch ISE from a Web Browser

Launch a web browser to ISE from your Windows Provisioning machine, ignoring any certificate errors. Ensure login to the GUI is successful, and that no configurations are currently present in the Users, Roles, Network Devices or Network Device Groups areas.

1. Log into ISE using the administrative username **admin** password **is provided by your proctor**
2. Using the hamburger menu on the top left of the screen, navigate to Administration -> Identity Management -> Groups -> User Identity Groups
3. Verify only default ISE groups are populated, including OWN_ACCOUNTS, ALL_ACCOUNTS



☐	ALL_ACCOUNTS (default)	Default ALL_ACCOUNTS (default) User Group
☐	Employee	Default Employee User Group
☐	GROUP_ACCOUNTS (default)	Default GROUP_ACCOUNTS (default) User Group
☐	GuestType_Contractor (default)	Identity group mirroring the guest type
☐	GuestType_Daily (default)	Identity group mirroring the guest type
☐	GuestType_SocialLogin (default)	Identity group mirroring the guest type
☐	GuestType_Weekly (default)	Identity group mirroring the guest type
☐	OWN_ACCOUNTS (default)	Default OWN_ACCOUNTS (default) User Group

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

☐	All Device Types	All Device Types
☐	All Locations	All Locations
☐ >	Is IPSEC Device	Is this a RADIUS over IPSEC Device

8. Navigate to Administration -> Network Devices
9. Verify no Network Devices are currently present.

Step 10: Configure ISE Programmatically

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.

1. Exit from the ISE SSH session using the "exit" command.
2. 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] …

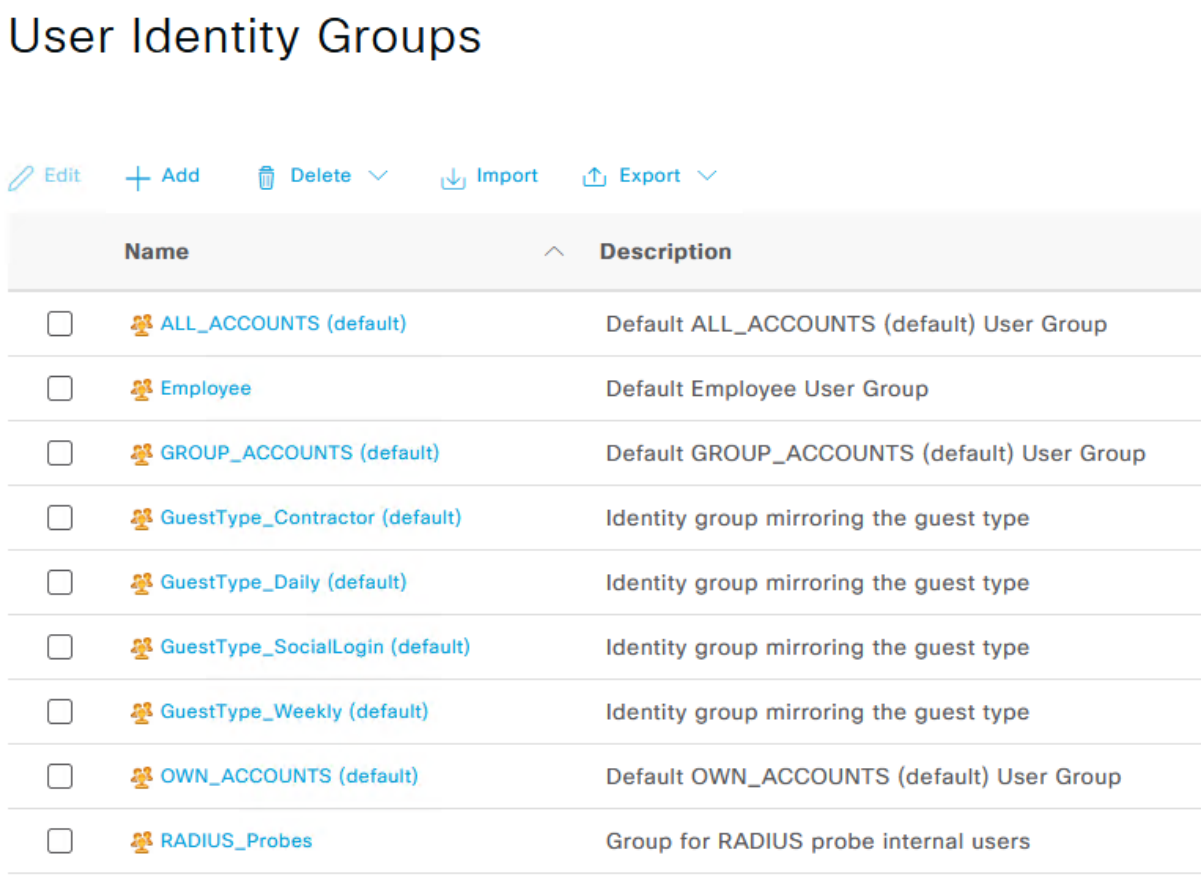
PLAY RECAP

localhost : ok=39 changed=6 unreachable=0 failed=0 skipped=4 rescued=0 ignored=0

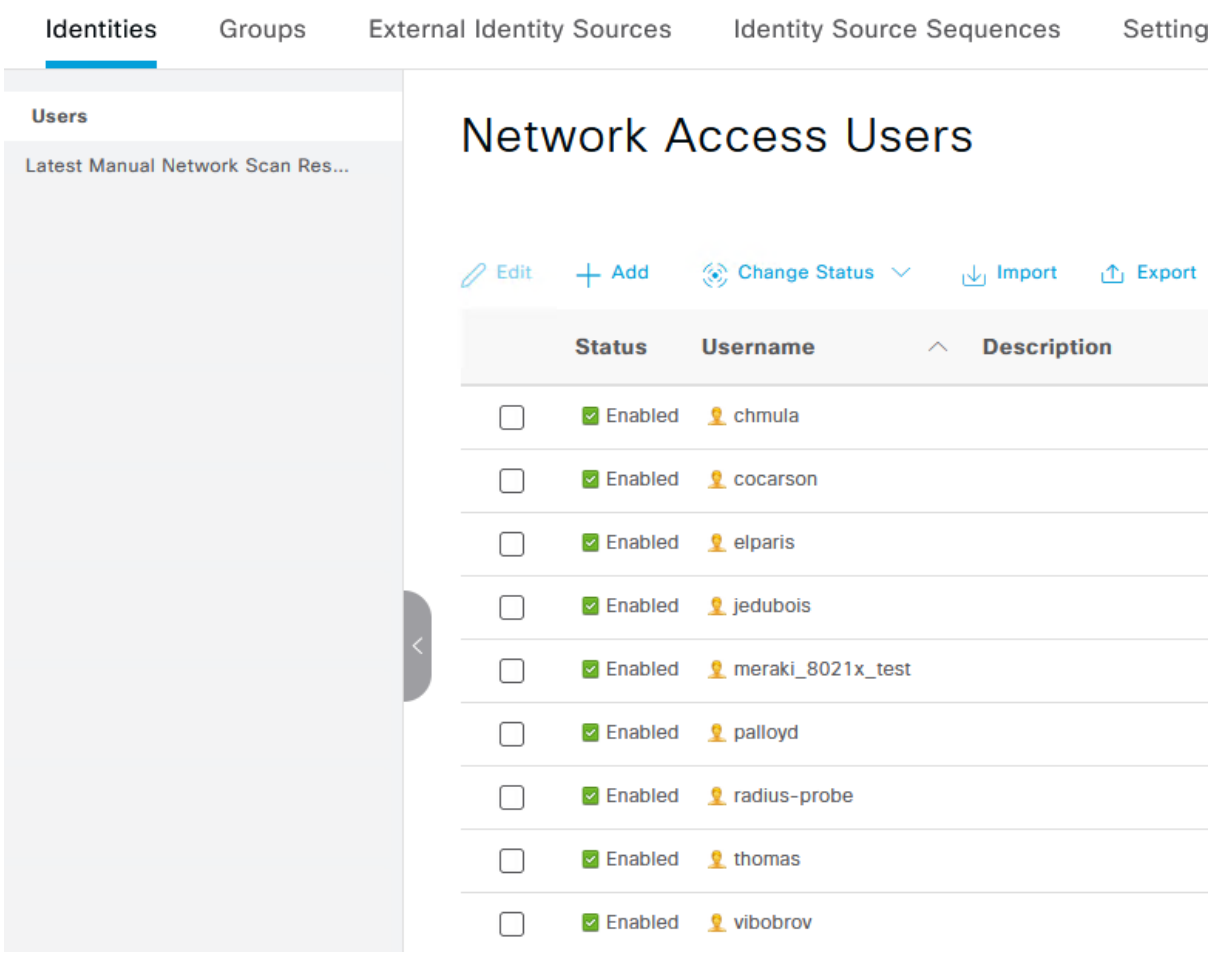
Step 11: Verify Newly Configured Users, Groups, and Network Access Devices

1. On your Ubuntu provisioning machine, navigate to ~/CiscoLive_ISE_in_AWS/vars.
2. Execute a more on each file to explore what is expected to be configured, including users, 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

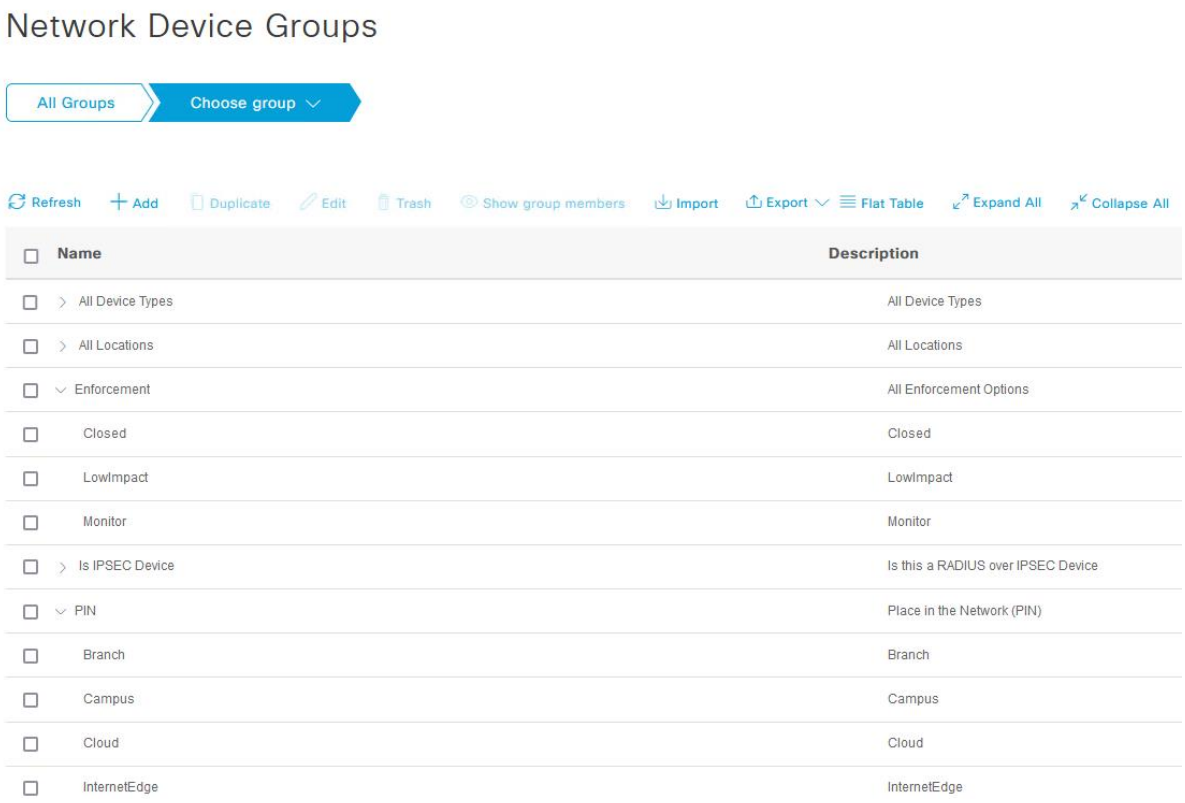
4. Verify the RADIUS_Probes group has been created as a group to assign users to.



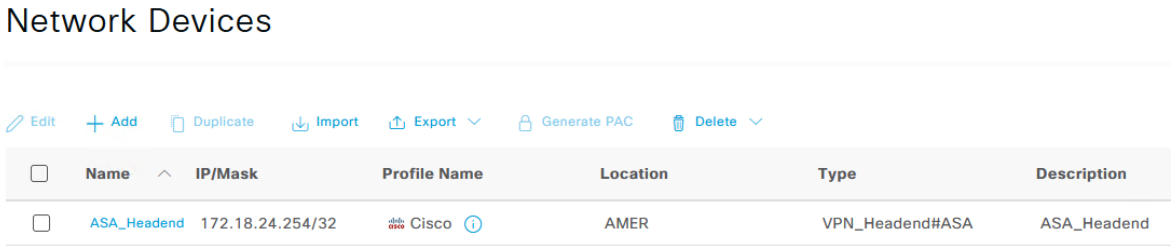
5. Navigate to Administration -> Identity -> Identities
6. Verify the users found in the “internal_users.thomas.yaml” are configured as expected.



7. Navigate to Administration -> Network Device Groups
8. Verify the network device groups found in “network_device_groups.yaml” are configured as expected.



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

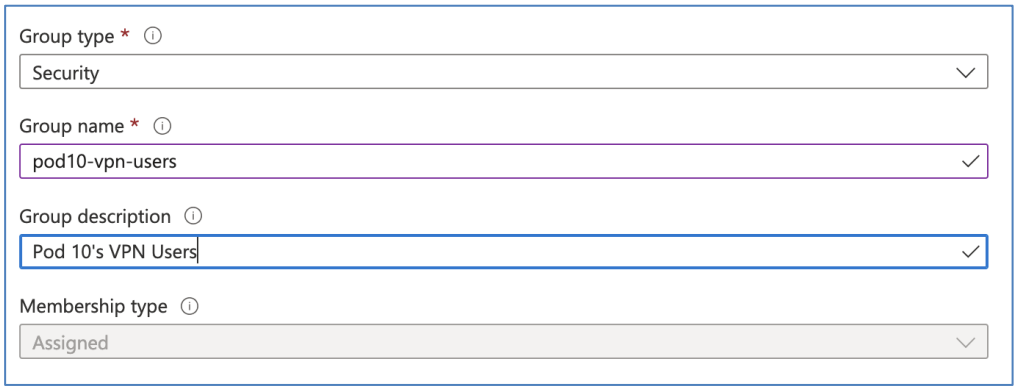


Task 5: Azure AD integration via ROPC for Remote Access VPN

This task will have you create the basics needed for ISE integration with Azure Active Directory for VPN authentication.

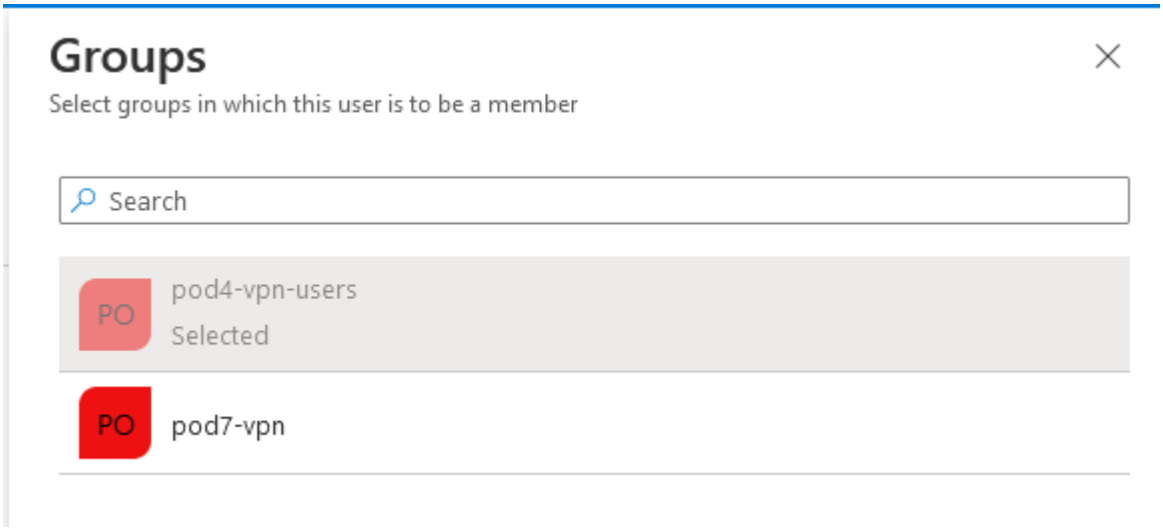
Step 1: Log into Azure and Configure Users/Groups

1. Log into https://portal.azure.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. Using the search bar, search for “groups” and add a new group of type Security. This will be used on ISE to determine which users are allowed to log into the VPN.
3. Name the group **podX-vpn-users** where X is your pod number, give it a description for users who will be allowed to log into VPN later.

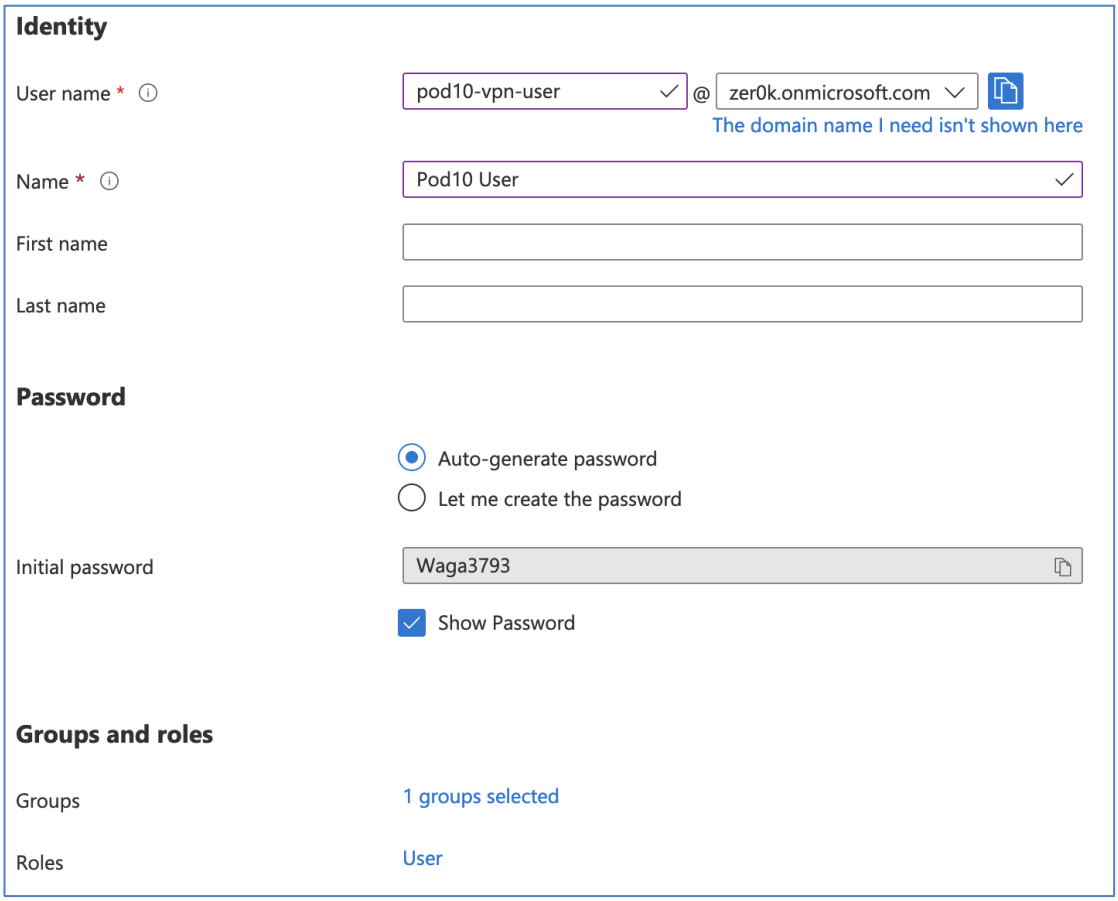


Note: It may take a minute before the group shows up in the All Groups area of Azure Active Directory. Please be patient and hit refresh.

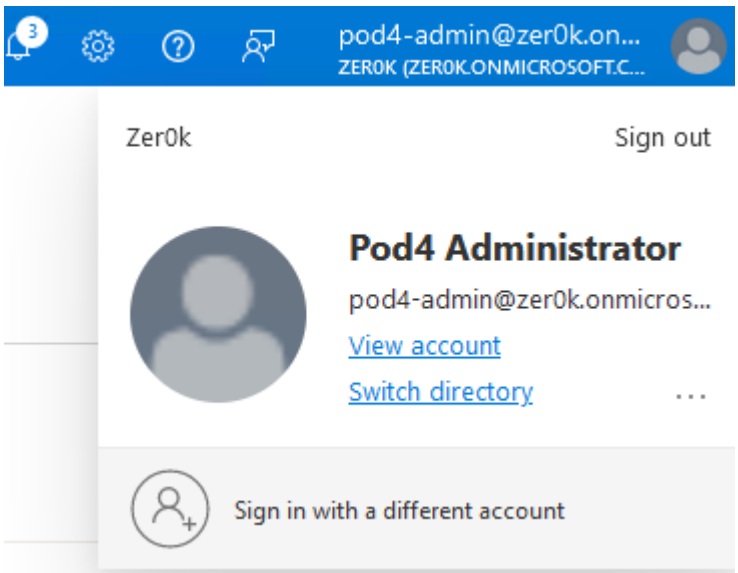
4. Using the search bar at the top of the page, search for “users” and add a new user. Name it **podX-vpn-user** where X is your pod number and give it a name.
5. Under “Groups and roles, click the “0 groups selected” link to select the group you created in the prior step, click “Select”.



6. You can use an auto-generated password or create a password, make sure you note the password before submitting the user with the “Create” button.



7. Azure 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 in with a different account"

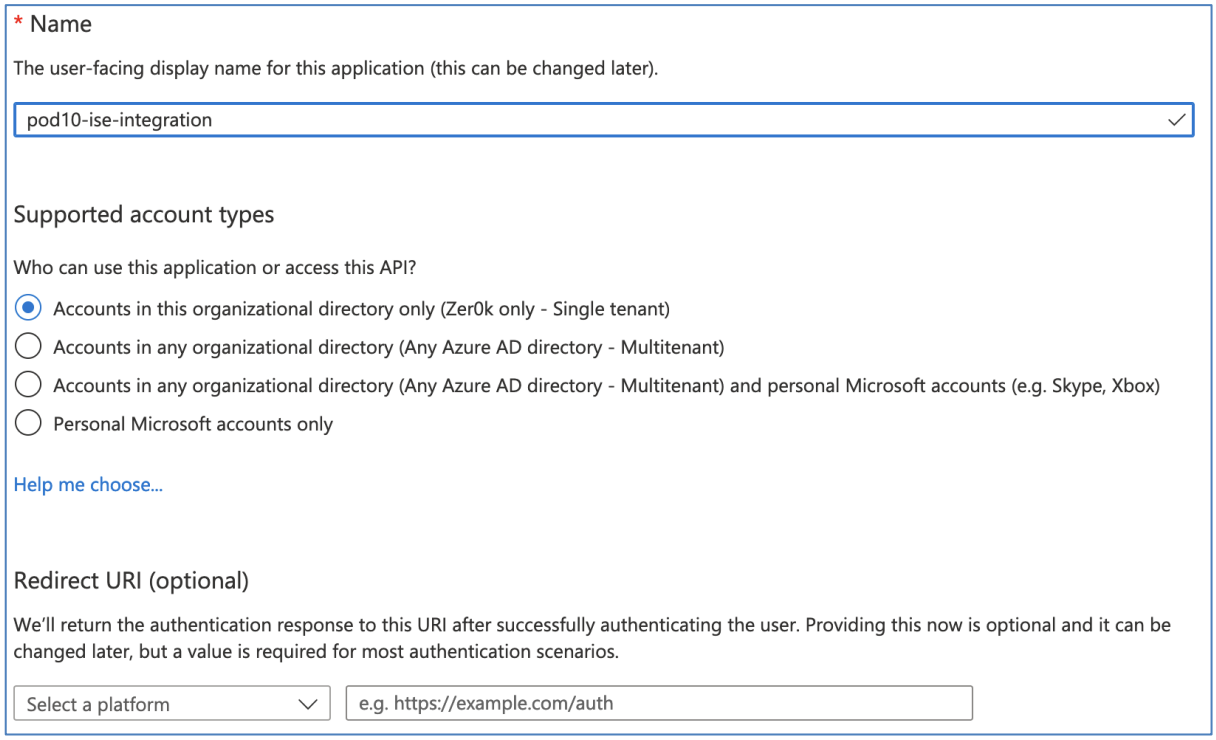


8. Sign in with the user you just created (the @zer0k.onmicrosoft.com suffix is required). You will be prompted to reset your password. Once you successfully log in, note the new password you just created and switch back to the pod admin user.

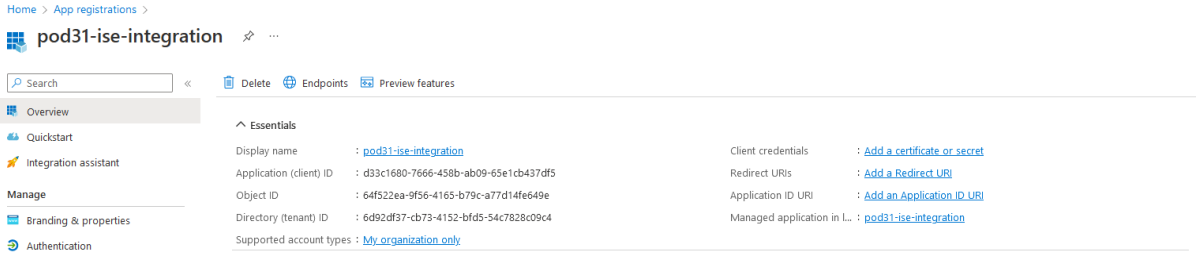
Step 2: Configure application integration in Azure

1. Search for App Registrations and choose "New registration", 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.

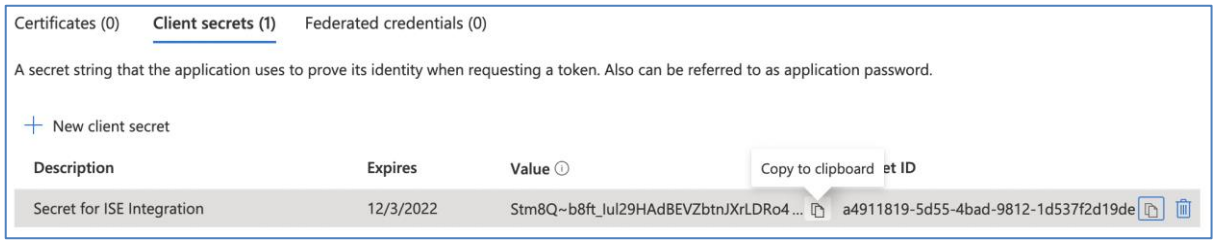


2. Click Register.
3. On the overview screen now presented, copy out the “Essentials” displayed at the top of the page. You will need the client ID and tenant ID later.

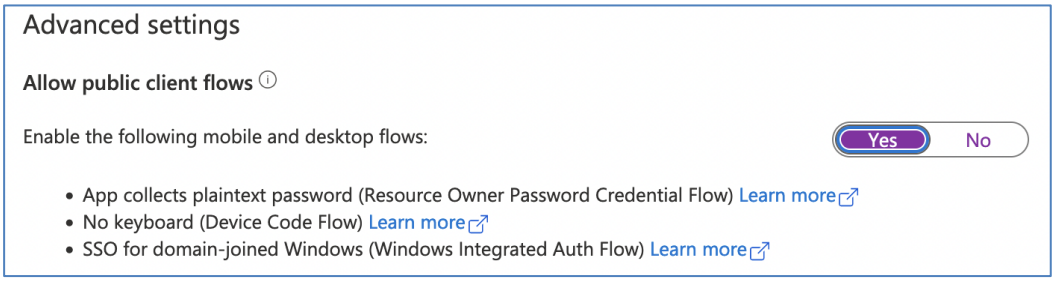


4. On the left menu bar, select “Certificates & secrets”. This is the OAuth secret key that will be used to integrate ISE with Azure. Click “New Client Secret” and enter a description of “podX-ISE” where X is your pod number. Click Add at the bottom of the dialog box.
5. Copy the secret value to a notepad before leaving the page!

Warning: This value cannot be viewed again after leaving the page. If you forget this step, delete the client secret and create a new one.

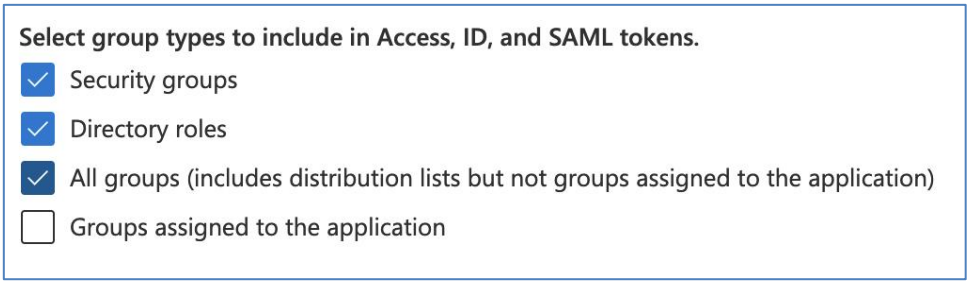


6. Configure the application for ROPC by going to the Authentication tab on the left menu. ISE does not use a URI based redirect flow so under Advanced Settings set the "Allow public client flows" setting to Yes. Click Save.



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

    Click Add.

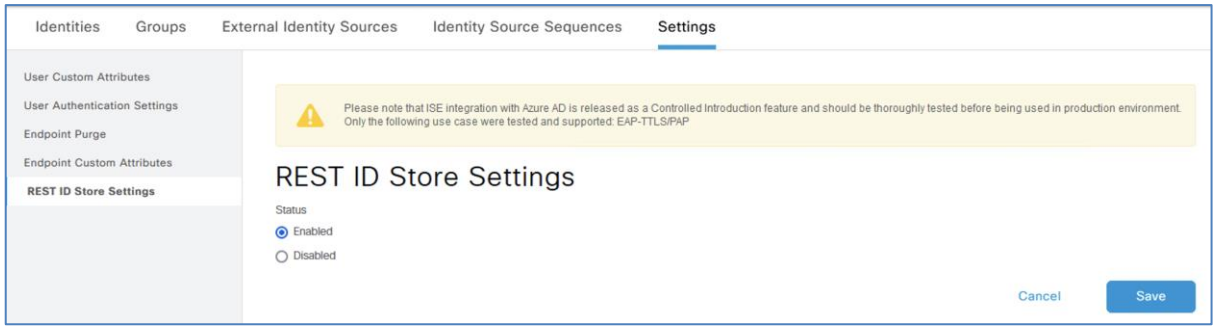


9. Add API permission for the groups graph API. In the left menu select API permissions then click Add a permission. In the window that pops up choose Microsoft Graph then Application Permissions.
10. Search for "group" then under the group drawer, select Group.Read.All, notice that Admin consent required is shown as Yes. Select Add Permissions at the bottom of the screen.

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

Step 3: Enable the ROPC feature in ISE

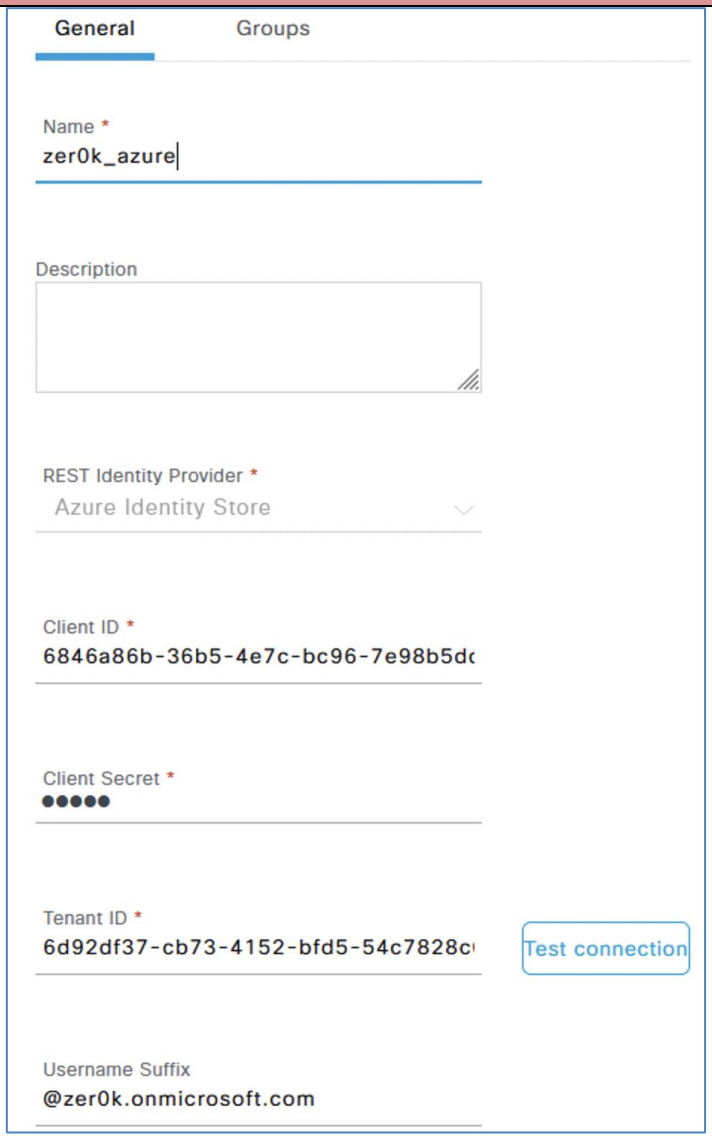
1. Go to Administration -> Identity Management -> Settings -> REST ID Store Settings and enable the feature. This is required as of ISE 3.1 since the feature is currently a beta feature.



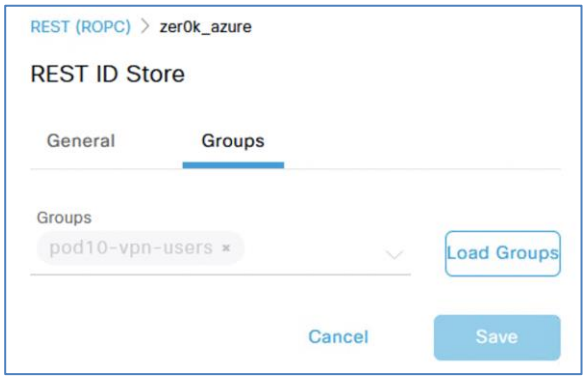
Step 4: Register ISE with Azure

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 (ROPC) option, add a new entry.
3. Name it **zer0k_azure** and fill in the information obtained from the overview page under the application created in Step 2 from Azure. The client secret (Secret Value) should have been saved to a notepad. The username suffix for this lab is **@zer0k.onmicrosoft.com**.

Note: In the Azure overview, the Client ID is the "Application (client) ID" and the tenant ID is the "Directory (tenant) ID".



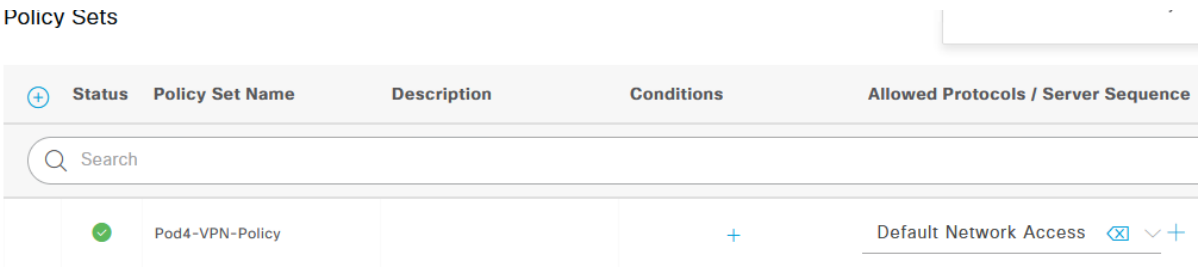
4. Test the connection, if it is successful go to the groups tab and select the **podX-vpn-users** group you created earlier.
5. In order to use groups in policy they need to be selected here in the ROPC connection. Click in the drop down to select the group you created in earlier steps. You only have to hit the Load Button once to get groups to populate into the dropdown.



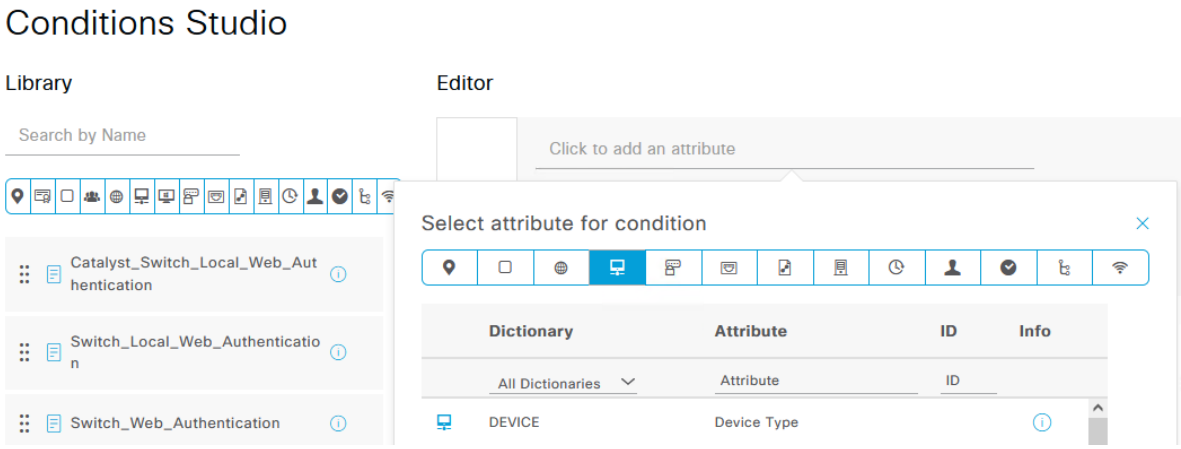
6. Save the connection.

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.



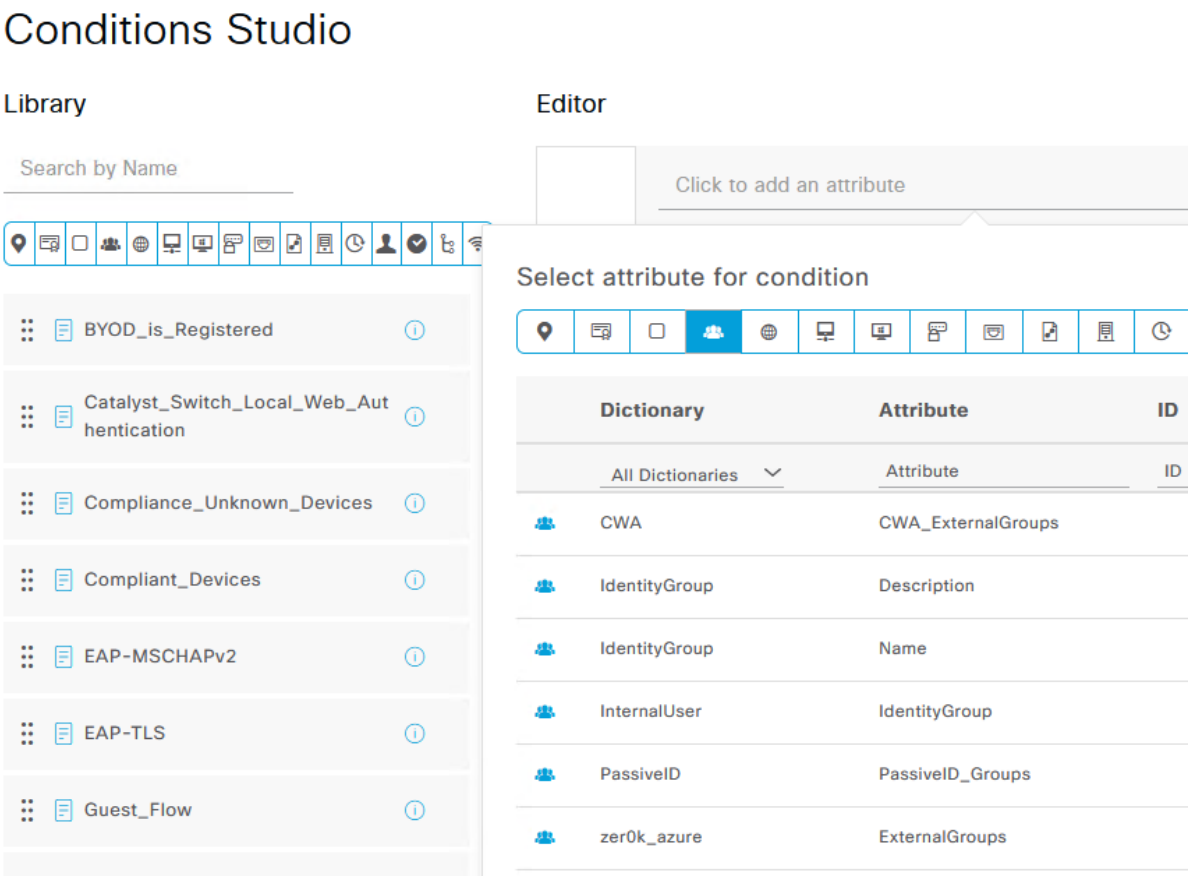
3. Click the "+" under conditions which will launch the conditions studio.
4. The network device and network device group were created by the initial configuration ansible scripts. They will be used here to match authentication from the VPN headend.
    - "click to add an attribute" and use the "network device" filter icon: 



- Choose Device: Device Type and choose the "All Device Types#VPN Headends#ASA" 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.

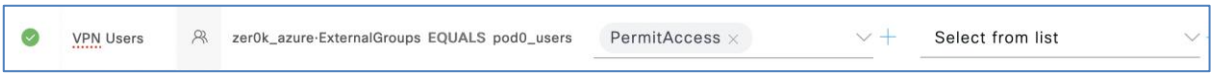
5. Save the policy sets.
6. Click the “View” arrow on the right side of the policy set line to enter the policy set configuration.
7. Expand “Authentication Policy” and set the default entry to what you named your ROPC connection (zer0k_azure).

8. 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 Azure 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 select the groups icon to filter the attributes:
    - Choose your ROPC connection name: External Groups and then select the group you have added your user to in Azure.



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

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



Task 6: Test VPN Authentication

Step 1: Test VPN Authentication

1. On your Windows Provisioning Machine, browse to https://172.18.24.254, accept any certificate errors that are presented.
    - Choose your pod number from the drop down (If authentication fails the dropdown is reset every time).
    - Log in using the Azure user previously created. The username should be **podX-vpn-user** where X is your assigned pod number.

If you did not populate the user suffix in previous steps, you may need to add @zer0k.onmicrosoft.com

2. You will be prompted to install AnyConnect, complete the installation. When the install completes, launch AnyConnect from the Windows start menu and connect to 172.18.24.254. You will be prompted for the pod number and the same username/password.
3. You will get a Certificate warning - Click change settings -> De-Select Block connections to untrusted servers, Connect again -> Connect anyway.
4. After successfully connecting you should be able to ping the super secret linux server at 172.18.16.10.
5. On ISE, browse to the ISE Menu -> Operations -> Radius -> Live Logs. Here you can see the authentication attempts and successful authentications to the VPN.
6. 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.

Task 7: (Bonus) Deploy Secondary ISE through AWS Marketplace from CloudFormation Template

Step 1: Subscribe to ISE in AWS Marketplace

1. In the AWS console search for AWS Marketplace Subscriptions.
2. Got to Discover Products on the left menu.
3. Search for ISE and click on Cisco Identity Services Engine (ISE).
4. Click on “Continue to Subscribe” on the top right after reading through some of the options on the subscription if you like.
5. Click on Continue to Configuration.
6. Under fulfilment option, select “CloudFormation Template”.
7. A new drop down will be displayed, select Cisco Identity Services Engine (ISE).
8. 2 new drop downs will be presented, select 3.1 Patch1 and select the Region as US East (Ohio). If you don’t use 3.1 patch 1 you won’t be able to join the node created by ansible earlier.
9. Click Continue to Launch on the top right.

10. Under Choose Action, select “Launch CloudFormation", then select the Launch button.

Step 2: CloudFormation Stack Creation

Market place has launched into 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 Name” 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.

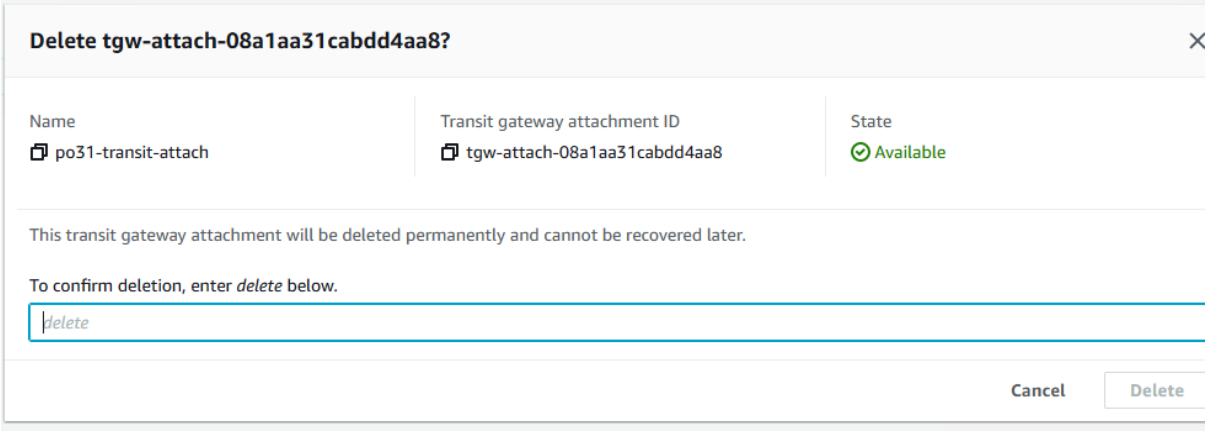
9. Under events you should see “CREATE_IN_PROGRESS”, refresh the table until you see podX-ise2 with “CREATE_COMPLETE”.

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

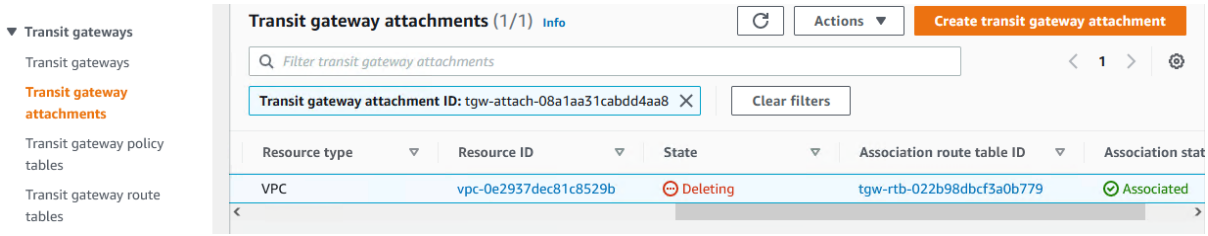
Task 8: (Bonus) Clean Up

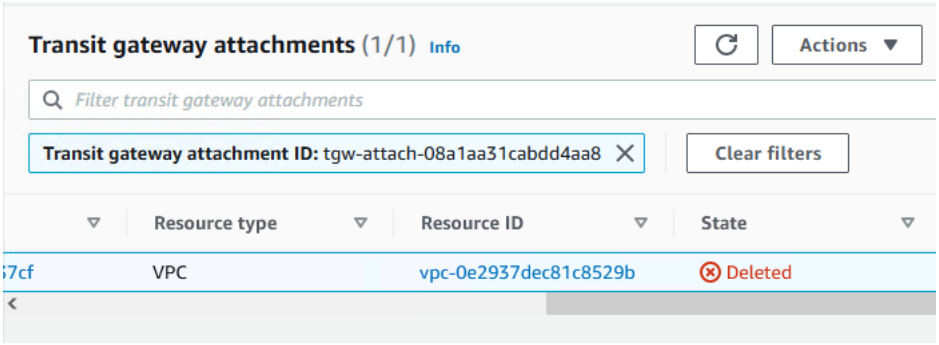
Step 1: Detach Transit Gateway

1. If you are finished with your lab and have no further questions, a built in clean up script can help the lab proctors clean up. Navigate to the VPC service via the search bar.
2. Click Transit Gateway attachments and delete the transit gateway attachment associated with your pod. It should be named “pod<x>-transit-attach”, where X is your pod number.



3. Wait until the state of the transit gateway is “Deleted”





4. Run the command "**ansible-playbook ise_in_aws.terminate.yaml".** Press Enter when prompted for the Project Name.

Expected Output:

(ISEonAWS):~/CiscoLive_ISE_in_AWS$ ansible-playbook ise_in_aws.terminate.yaml

[WARNING]: No inventory was parsed, only implicit localhost is available

[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

[WARNING]: While constructing a mapping from /home/ubuntu/CiscoLive_ISE_in_AWS/vars/main.yaml, line 3, column 1, found a duplicate dict key (ise_verify). Using last defined value only.

Project Name [ISEinAWS-pod31]:

PLAY [Terminate AWS EC2 Instances for Project "ISEinAWS-pod31"]
****

TASK [Gathering Facts]
****

ok: [localhost]

TASK [Get all EC2 Instances with tag project:"ISEinAWS-pod31"]
****

ok: [localhost]

```
TASK [Delete all EC2 instances with tag project:"ISEinAWS-pod31"] **************
ok: [localhost]

TASK [Get all VPCs with tag project:"ISEinAWS-pod31"] ************************
ok: [localhost]

TASK [Show vpcs] ************************************************************
ok: [localhost] =>
  vpcs:
    changed: false
    failed: false
    vpcs:
    - cidr_block: 10.31.0.0/16
      cidr_block_association_set:
      - association_id: vpc-cidr-assoc-0344c48663b0993ae
        cidr_block: 10.31.0.0/16
        cidr_block_state:
          state: associated
      classic_link_dns_supported: false
      classic_link_enabled: false
      dhcp_options_id: dopt-086ed054b34c26c91
      enable_dns_hostnames: true
      enable_dns_support: true
      id: vpc-0e2937dec81c8529b
      instance_tenancy: default
      is_default: false
      owner_id: '423620453332'
      state: available
      tags:
```

```
      Name: ISEinAWS-pod31
      project: ISEinAWS-pod31
      start_date: '2023-01-26'
    vpc_id: vpc-0e2937dec81c8529b

TASK [Find Dangling ENIs in the VPC] ****************************************
ok: [localhost]

TASK [Show enis] ************************************************************
ok: [localhost] =>
  enis:
    changed: false
    failed: false
    network_interfaces: []

TASK [Delete all dangling ENIs] *********************************************
skipping: [localhost]

TASK [Get All Subnets with tag project:"ISEinAWS-pod31"] ********************
ok: [localhost]

TASK [Show subnets] *********************************************************
ok: [localhost] =>
  subnets:
    changed: false
    failed: false
    subnets:
                    - assign_ipv6_address_on_creation: false
      availability_zone: us-east-2b
```

availability_zone_id: use2-az2

available_ip_address_count: 251

cidr_block: 10.31.0.0/24

default_for_az: false

enable_dns64: false

id: subnet-02fc0659508f6a720

ipv6_cidr_block_association_set: []

ipv6_native: false

map_customer_owned_ip_on_launch: false

map_public_ip_on_launch: false

owner_id: '423620453332'

private_dns_name_options_on_launch:

enable_resource_name_dns_a_record: false

enable_resource_name_dns_aaaa_record: false

hostname_type: ip-name

state: available

subnet_arn: arn:aws:ec2:us-east-2:423620453332:subnet/subnet-02fc0659508f6a720

subnet_id: subnet-02fc0659508f6a720

tags:

Name: zer0k_pod31_Private_Subnet

project: ISEinAWS-pod31

start_date: '2023-01-26'

vpc_id: vpc-0e2937dec81c8529b

TASK [Delete Subnets] ***

changed: [localhost] => (item={'availability_zone': 'us-east-2b', 'availability_zone_id': 'use2-az2', 'available_ip_address_count': 251, 'cidr_block': '10.31.0.0/24', 'default_for_az': False, 'map_public_ip_on_launch': False, 'map_customer_owned_ip_on_launch': False, 'state': 'available', 'subnet_id': 'subnet-02fc0659508f6a720', 'vpc_id': 'vpc-0e2937dec81c8529b', 'owner_id': '423620453332', 'assign_ipv6_address_on_creation': False, 'ipv6_cidr_block_association_set': [], 'tags': {'Name':

'zer0k_pod31_Private_Subnet', 'project': 'ISEinAWS-pod31', 'start_date': '2023-01-26'},
'subnet_arn': 'arn:aws:ec2:us-east-2:423620453332:subnet/subnet-02fc0659508f6a720',
'enable_dns64': False, 'ipv6_native': False, 'private_dns_name_options_on_launch':
{'hostname_type': 'ip-name', 'enable_resource_name_dns_a_record': False,
'enable_resource_name_dns_aaaa_record': False}, 'id': 'subnet-02fc0659508f6a720'})

TASK [Get All Route Tables with tag project:"ISEinAWS-pod31"] ******************

ok: [localhost]

TASK [Show rts] **

ok: [localhost] =>

rts:

changed: false

failed: false

route_tables:

- associations: []

id: rtb-03ac2a38ecabcd75a

owner_id: '423620453332'

propagating_vgws: []

route_table_id: rtb-03ac2a38ecabcd75a

routes:

- destination_cidr_block: 10.31.0.0/16

gateway_id: local

instance_id: null

interface_id: null

network_interface_id: null

origin: CreateRouteTable

state: active

tags:

Name: zer0k_pod31_RT_Private

project: ISEinAWS-pod31

start_date: '2023-01-26'

vpc_id: vpc-0e2937dec81c8529b

TASK [Delete Route Table] ***

changed: [localhost] => (item={'associations': [], 'propagating_vgws': [], 'route_table_id': 'rtb-03ac2a38ecabcd75a', 'routes': [{'destination_cidr_block': '10.31.0.0/16', 'gateway_id': 'local', 'origin': 'CreateRouteTable', 'state': 'active', 'instance_id': None, 'network_interface_id': None, 'interface_id': None}], 'vpc_id': 'vpc-0e2937dec81c8529b', 'owner_id': '423620453332', 'tags': {'project': 'ISEinAWS-pod31', 'Name': 'zer0k_pod31_RT_Private', 'start_date': '2023-01-26'}, 'id': 'rtb-03ac2a38ecabcd75a'})

TASK [Get All Internet Gateways with tag project:"ISEinAWS-pod31"] *************

ok: [localhost]

TASK [Show igws] **

ok: [localhost] =>

igws:

changed: false

failed: false

internet_gateways: []

TASK [Delete Internet Gateway] **

ok: [localhost] => (item={'cidr_block': '10.31.0.0/16', 'dhcp_options_id': 'dopt-086ed054b34c26c91', 'state': 'available', 'vpc_id': 'vpc-0e2937dec81c8529b', 'owner_id': '423620453332', 'instance_tenancy': 'default', 'cidr_block_association_set': [{'association_id': 'vpc-cidr-assoc-0344c48663b0993ae', 'cidr_block': '10.31.0.0/16', 'cidr_block_state': {'state': 'associated'}}], 'is_default': False, 'tags': {'start_date': '2023-01-26', 'project': 'ISEinAWS-pod31', 'Name': 'ISEinAWS-pod31'}, 'classic_link_enabled': False, 'classic_link_dns_supported': False, 'enable_dns_support': True, 'enable_dns_hostnames': True, 'id': 'vpc-0e2937dec81c8529b'})

TASK [Get all Security Groups with tag project:"ISEinAWS-pod31"] ***************

ok: [localhost]

```
TASK [Show sgs] ****************************************************************
ok: [localhost] =>
  sgs:
    changed: false
    failed: false
    security_groups:
    - description: zer0k_pod31_SG-ISE
      group_id: sg-0e2ec6c057793b79e
      group_name: zer0k_pod31_SG-ISE
      ip_permissions:
      - ip_protocol: '-1'
        ip_ranges:
        - cidr_ip: 10.31.0.0/16
          description: Allow all traffic within VPC
        - cidr_ip: 172.16.0.0/12
          description: Allow all traffic from on-premises
        ipv6_ranges: []
        prefix_list_ids: []
        user_id_group_pairs: []
      - from_port: 22
        ip_protocol: tcp
        ip_ranges:
        - cidr_ip: 0.0.0.0/0
          description: Allow SSH from anywhere
        ipv6_ranges: []
        prefix_list_ids: []
                              to_port: 22
        user_id_group_pairs: []
```

```
    ip_permissions_egress:
    - ip_protocol: '-1'
      ip_ranges:
      - cidr_ip: 0.0.0.0/0
        description: Allow All
      ipv6_ranges: []
      prefix_list_ids: []
      user_id_group_pairs: []
    owner_id: '423620453332'
    tags:
      Name: ISEinAWS-pod31
      project: ISEinAWS-pod31
    vpc_id: vpc-0e2937dec81c8529b


TASK [Delete Security Groups in VPC] ******************************************

changed: [localhost] => (item={'description': 'zer0k_pod31_SG-ISE', 'group_name':
'zer0k_pod31_SG-ISE', 'ip_permissions': [{'ip_protocol': '-1', 'ip_ranges': [{'cidr_ip': '10.31.0.0/16',
'description': 'Allow all traffic within VPC'}, {'cidr_ip': '172.16.0.0/12', 'description': 'Allow all
traffic from on-premises'}], 'ipv6_ranges': [], 'prefix_list_ids': [], 'user_id_group_pairs': []},
{'from_port': 22, 'ip_protocol': 'tcp', 'ip_ranges': [{'cidr_ip': '0.0.0.0/0', 'description': 'Allow SSH
from anywhere'}], 'ipv6_ranges': [], 'prefix_list_ids': [], 'to_port': 22, 'user_id_group_pairs': []}],
'owner_id': '423620453332', 'group_id': 'sg-0e2ec6c057793b79e', 'ip_permissions_egress':
[{'ip_protocol': '-1', 'ip_ranges': [{'cidr_ip': '0.0.0.0/0', 'description': 'Allow All'}], 'ipv6_ranges': [],
'prefix_list_ids': [], 'user_id_group_pairs': []}], 'tags': {'Name': 'ISEinAWS-pod31', 'project':
'ISEinAWS-pod31'}, 'vpc_id': 'vpc-0e2937dec81c8529b'})


TASK [Delete VPCs with tag project:"ISEinAWS-pod31"] ***************************

changed: [localhost] => (item={'cidr_block': '10.31.0.0/16', 'dhcp_options_id': 'dopt-
086ed054b34c26c91', 'state': 'available', 'vpc_id': 'vpc-0e2937dec81c8529b', 'owner_id':
'423620453332', 'instance_tenancy': 'default', 'cidr_block_association_set': [{'association_id': 'vpc-
cidr-assoc-0344c48663b0993ae', 'cidr_block': '10.31.0.0/16', 'cidr_block_state': {'state':
'associated'}}], 'is_default': False, 'tags': {'start_date': '2023-01-26', 'project': 'ISEinAWS-pod31',
'Name': 'ISEinAWS-pod31'}, 'classic_link_enabled': False, 'classic_link_dns_supported': False,
'enable_dns_support': True, 'enable_dns_hostnames': True, 'id': 'vpc-
0e2937dec81c8529b'})
```

**TASK [Delete ISEinAWS-pod31 by Name] **

ok: [localhost]

TASK [Delete ~/.ssh/ISEinAWS-pod31.pem] *************************************

ok: [localhost]

**PLAY RECAP **

localhost : ok=23 changed=4 unreachable=0 failed=0 skipped=1 rescued=0 ignored=0

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!