LTRSEC-3554

Simplifying Multicloud Security with Cisco Multicloud Defense

Version 2.5

Eric Kostlan – Technical Marketing Engineer

Anubhav Swami – Principal Architect

Contents

Introduction

Cisco Multicloud Defense is a highly scalable, on-demand **"as-a-Service"** solution that provides agile, scalable, and flexible security to your multicloud infrastructure. It unifies security controls across cloud environments, protects workloads from every direction, and drives operational efficiency by leveraging secure cloud networking.

Cisco Multicloud Defense uses a common principle in public clouds and software-defined networking (SDN) which decouples the control and data plane, translating to the ***Multicloud Defense Controller*** and the ***Multicloud Defense Gateways***.

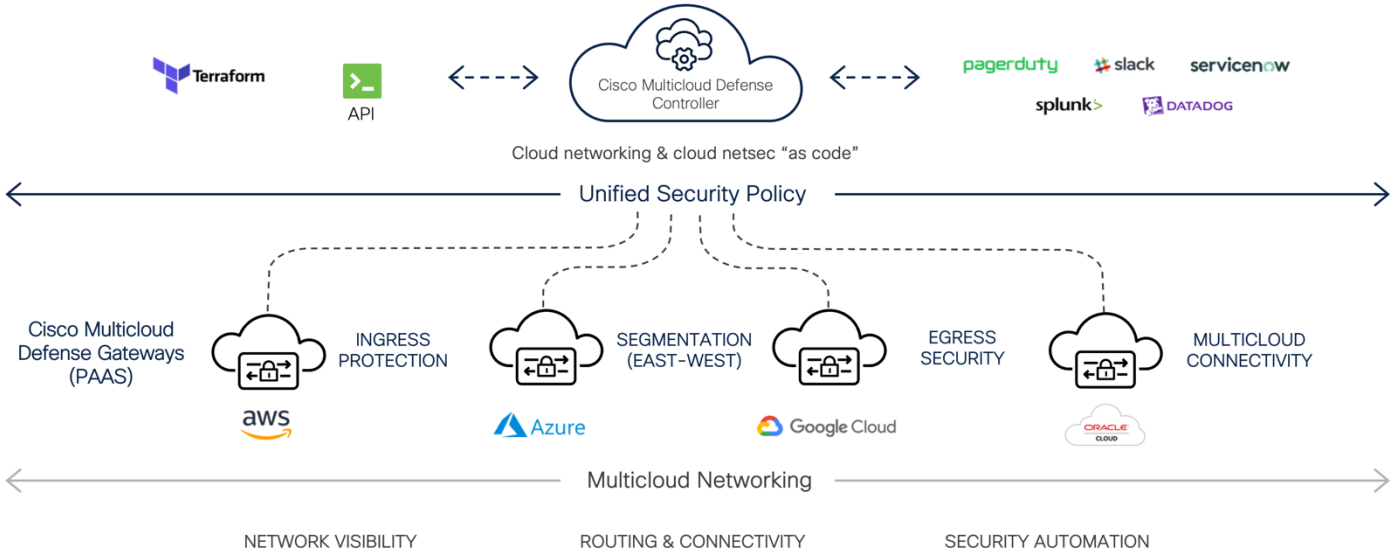


Figure 1: Cisco Multicloud Defense Overview

- **Multicloud Defense Controller (Software-as-a-Service):** The Multicloud Defense Controller is a highly reliable and scalable centralized controller (control plane) that automates, orchestrates, and secures multicloud infrastructure. It runs as a Software-as-a-Service (SaaS) and is fully managed by Cisco.
- **Multicloud Defense Gateway (Platform-as-a-Service)**: The Multicloud Defense Gateway is an auto-scaling fleet of security software with a patented flexible, single-pass pipelined architecture. These gateways are deployed as Platform-as-a-Service (PaaS) into the customer's public cloud account(s) by the Multicloud Defense Controller, providing advanced, inline security protections to defend against external attacks, block egress data exfiltration, and prevent the lateral movement of attacks.

Cisco Multicloud Defense Lab

This lab covers how Cisco Multicloud Defense provides cloud-agnostic security for multi-cloud infrastructure. The Cisco Multicloud Defense Gateway Provides the following security capabilities:

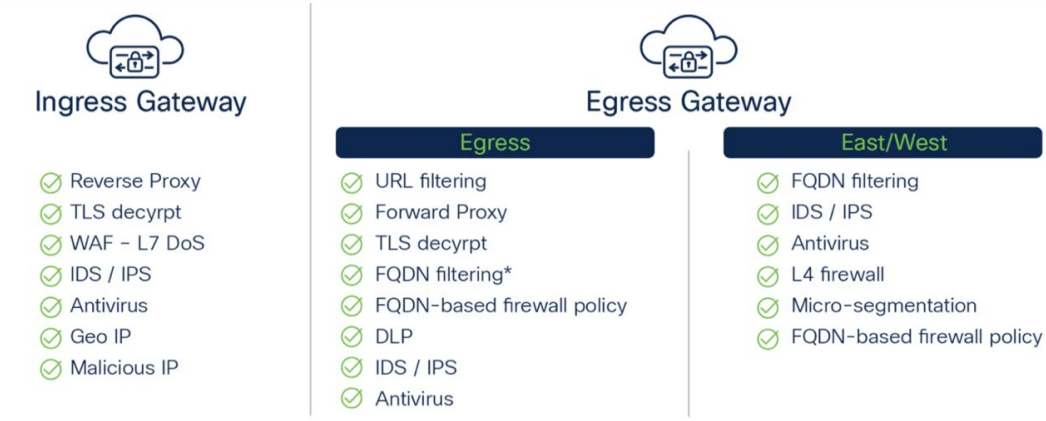


Figure 2: Cisco Multicloud Defense Gateway security capabilities

This lab focuses on securing AWS, Azure, and GCP infrastructure using distributed and centralized security models.

- **Centralized Security Model:** In the centralized security model, the Multicloud Defense Gateway(s) are deployed in a dedicated security VPC/VNet in the customer's account.
- **Distributed Security Model:** In the distributed security model, the Multicloud Defense Gateway(s) are deployed in the application VPC/VNet.

Figure 3 shows centralized and distributed security models. In this lab, you will deploy centralized and distributed (AWS) security models.

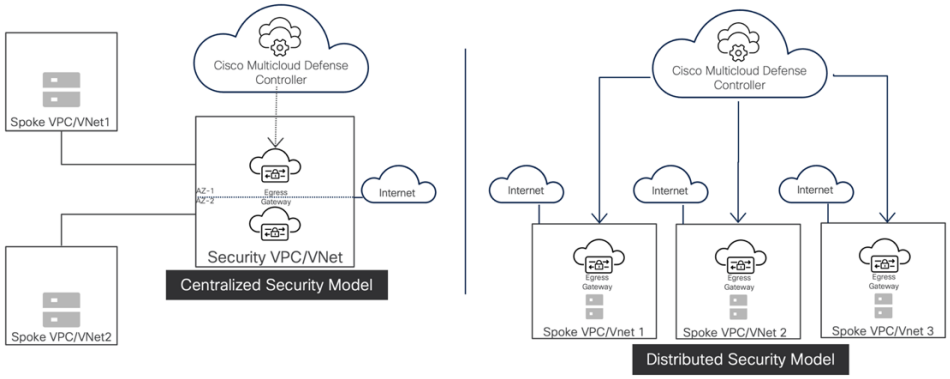


Figure 3: Cisco Multicloud Defense Gateway security capabilities

The lab exercises

The lab exercises are divided into two parts: core exercises and optional exercises.

The core exercises cover the product at a high level, and cover AWS and Azure centralized security deployments in detail.

Exercise 1 – Verify lab access and the lab environment

Exercise 2 – Enable visibility

Exercise 3 – Create a security policy and attach it to a ruleset

Exercise 4 – Protect AWS infrastructure

Exercise 5 – Protect Azure infrastructure

Note: The order of these five exercises can be altered as long as:

1. You perform **Exercise 1** before any other exercise.
2. You perform **Exercise 3** before Exercise 4 or Exercise 5.

The optional exercises cover GCP centralized deployment. The last exercise exposes you to the Cisco Multicloud Defense distributed security model, but only for AWS.

Exercise 6 – Configure visibility and infrastructure protection in GCP

Exercise 7 – Distributed security model

To perform Exercise 6, you need to have a valid GCP account. Speak to the proctor so he can add this account to the lab project. You may skip Exercise 6 and perform Exercise 7 if you wish.

Part I – Core exercises

Exercise 1 – Verify lab access and lab environment

Task 1. Access your jump-host

For each lab attendee, we have created an account on the jump-host. You will use the jump-host to access the workloads in the public cloud and generate outbound and east-west traffic.

This jump-host contains all the private keys necessary to access the public cloud servers. Static routes from the jump-host to the servers will bypass the gateways installed in **Exercises 4 through 6.**

You really do not need to access the jump-host until **Exercise 4**. But by confirming access now, we can identify any issues with your pod early. So that when you get to **Exercise 4**, there should not be any hiccups.

1. Log into the jump-host using SSH. Use the IP address and password the proctor provided you. The username is **podx**, where **x** is your pod number.
2. Run the command:
   grep podx- /etc/hosts (where **x** is your pod number)
   Note that the public ip addresses of the six application servers (two per public-cloud provider) for your pod are present.
3. Run **pingtest podx-** (where **x** is your pod number) to confirm that all these six of these servers are up.
4. Run the command **ls** to confirm you have three private keys: **aws**, **azure** and **gcp**.
5. Run the command
   ssh -i aws ubuntu@podx-app1-aws (where **x** is your pod number)
   Accept the key when prompted.
6. Once logged into podx-app1-aws, try the following commands **(note: These commands will both fail because podx-app1-aws and podx-app2-aws are in separate VPCs with no inter-VPC connectivity.)**
   ping 10.[100+x].100.10 (where **x** is your pod number) – for example, if your pod number is 40 then ping 10.140.100.10
   curl 10.[100+x].100.10 (where **x** is your pod number) – for example, if your pod number is 40 then ping 10.140.100.10
7. You can leave this connection open for **Exercise 4, Task 3.**

Task 2. Accessing public cloud consoles (for reference)

You will not need to do much work in the public cloud consoles, but you should understand how to access them. This might be of interest if you want to see what is happening "under the hood" as you perform the lab exercises. **You do not need to log into these consoles at this point in the lab**.

- AWS console: https://aws.amazon.com/console/

Login as an IAM user to account **698990355236** with the username **podx** (where **x** is your pod number) and the same password you used for the jump-host.

- Azure console: https://portal.azure.com/

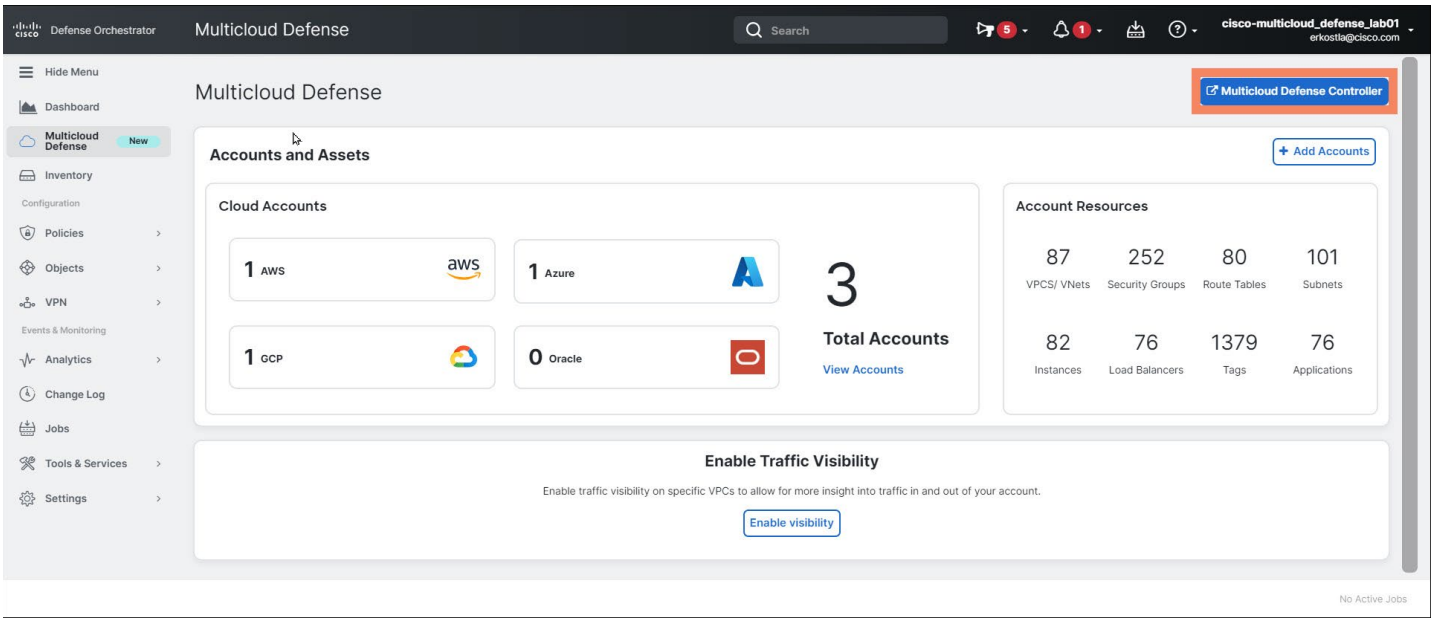
  You will need to log into this console to perform **Task 2 of Exercise 2**. Login instructions are included in that Task.

- GCP console: https://console.cloud.google.com/

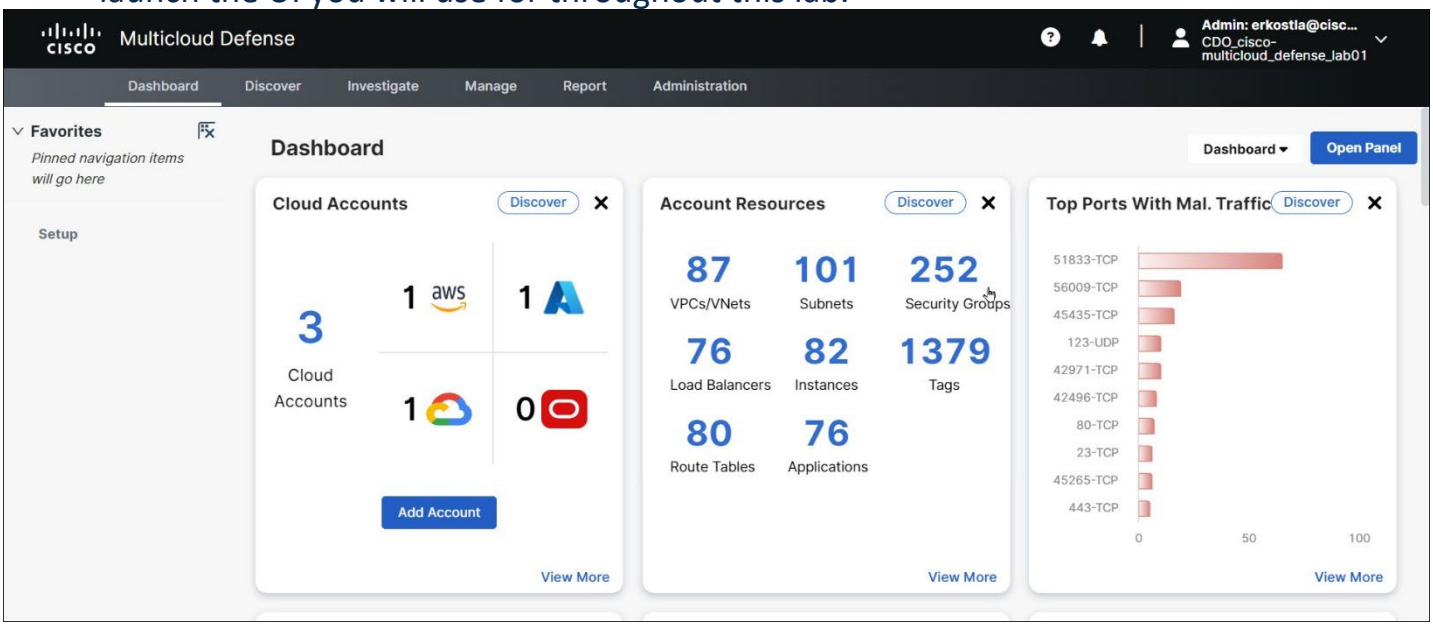
  This would only be used in optional **Exercise 6**. You must have a valid GCP account. If you are interested, let the proctor know, and you will be added to the project. **But please wait until you get to Exercise 6 to make this request.**

Task 3. Access Cisco Defense Orchestrator and the Cisco Multicloud Defense console

1. In your browser of choice, navigate to https://www.defenseorchestrator.com/ **(note: access CDO US region)**
2. Login as directed by the proctor.
3. Launch the Multicloud Defense portal from the Cisco Defense Orchestrator as shown in the below image.
4. Cloud accounts are pre-onboarded in this Cisco Defense Orchestration tenant.



5. In the upper right, click the button labeled Multicloud Defense Controller. This will launch the UI you will use for throughout this lab.



Task 4. Explore the lab inventory

The following infrastructure is pre-deployed for you.

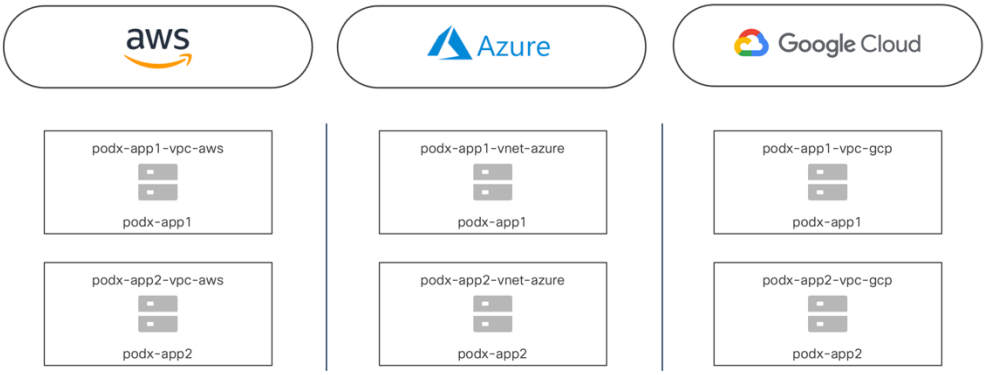


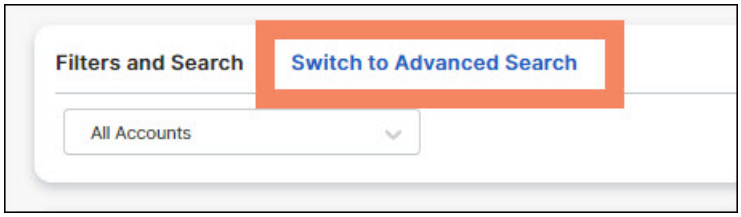
Figure: Application VPC – AWS, Azure, and GCP

We have pre-deployed the following resources in each pod.

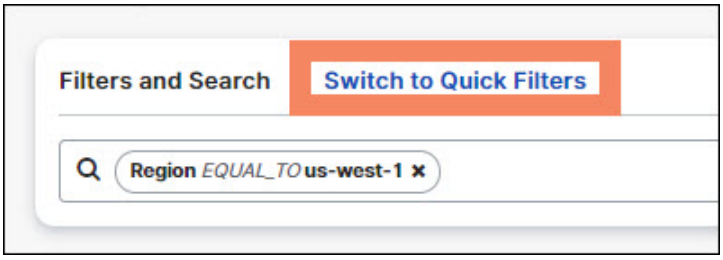
- **AWS:** podx-app1-vpc-aws, podx-app1, podx-app1-subnet, podx-mgmt-subnet, route table, routes, routable association, Internet gateway, and other required resources in the AWS account.
- **Azure**: podx-rg, podx-app1-vpc-azure, podx-app1, podx-app1-subnet, UDR, and other required resources in the Azure subscription.
- **GCP**: podx-app1-vpc-gcp, podx-app1, podx-app1-subnet, route table, routes, routable association, Internet gateway, and other required resources in the GCP account.

Using the Multicloud Defense console, you will explore this inventory.

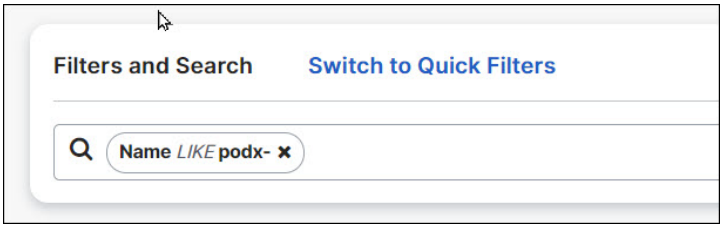
1. In the Multicloud Defense console, observe that there are three public cloud accounts onboarded: **one AWS account, one Azure account, and one GCP account.**
2. When accounts are onboarded, cloud resources are discovered.
   Note the high-level list of discovered resources.
3. In the Multicloud Defense Controller console, select **Discover > Summary**. **Note** that you can filter by account or show all the accounts resources together (the default)
4. Select **VPCs/VNets**. Note that for the AWS account, the default VPCs appear for all regions.
5. Using the filter, display the resource summary for each account separately.
6. Click **Switch to Advanced Search** to use the advanced search feature.



7. Click Switch to Quick Filters to switch back.



8. Return to the dashboard of the Multicloud Defense console.
9. Navigate to **Discover > Inventory > Instances**. Make sure you are showing the inventory for all accounts together.
10. Click **Switch to Advanced Search**. For the filter, select **Name** and then select **LIKE** and then type **podx-** (where **x** is your pod number). Confirm that you have six instances.



11. Navigate to **Discover > Inventory > Route Tables**. Make sure you are showing the inventory for all accounts together.
12. Drill down on a few route tables see the routes.
13. Navigate to **Discover > Inventory > Security Groups > Rules**. Make sure you are showing the inventory for all accounts together.
14. Observe that you can view the inbound and outbound security rules.

Exercise 2 – Enable visibility in AWS and Azure

In this exercise, you will enable visibility for the AWS and Azure account. Visibility for the GCP account is relegated to optional Exercise 6.

Cisco Multicloud defense uses the follow technologies to provide traffic visibility for you multi-cloud environment. A key benefit of Cisco Multicloud Defense is that you do not need to understand the details of how these technologies work or how to configure them.

- **AWS:** VPC flow logs and DNS query logs
- **Azure**: NSG flow logs
- **GCP**: VPC flow logs

In this exercise, you will enable visibility for the AWS and Azure account. Visibility for the GCP account is relegated to optional Exercise 6.

Task 1. Enable AWS visibility

1. In the Multicloud Defense Controller console, select the **Setup** tab. Note that there are three high-level configurations: Connect Account, Enable Visibility, and Secure Account. Since the accounts are already connected, move on to the second step by clicking **Enable Visibility**.

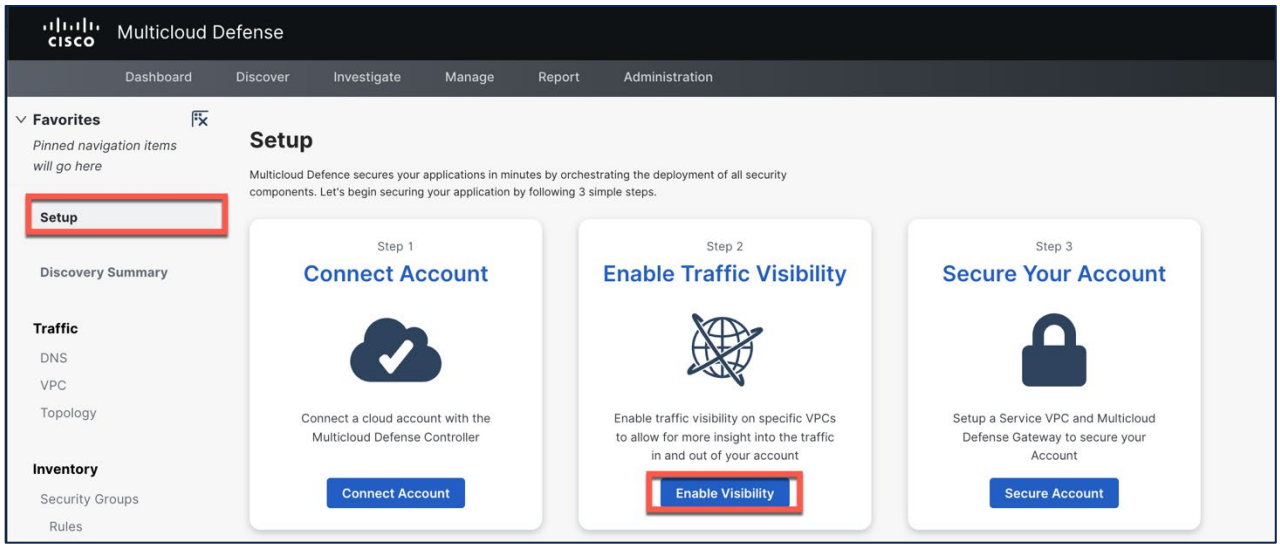


Figure: Enable visibility

2. Enter the information in the following figure (except use your pod number). Select both VPCs.
    a. CSP Account: **cisco-multicloud-defense-aws01**
    b. Region: **us-east-1**

c. Search in VPCs: **podx- (where x is your pod number)**

   i. Select **podx-app1-vpc** and **podx-app2-vpc**

d. Select S3 Bucket: **ciscomcd-cdo-cisco-multicloud-defense-lab-01-240129-201651**

e. Click **Next**.

Note: Use podx instead of using pod1-

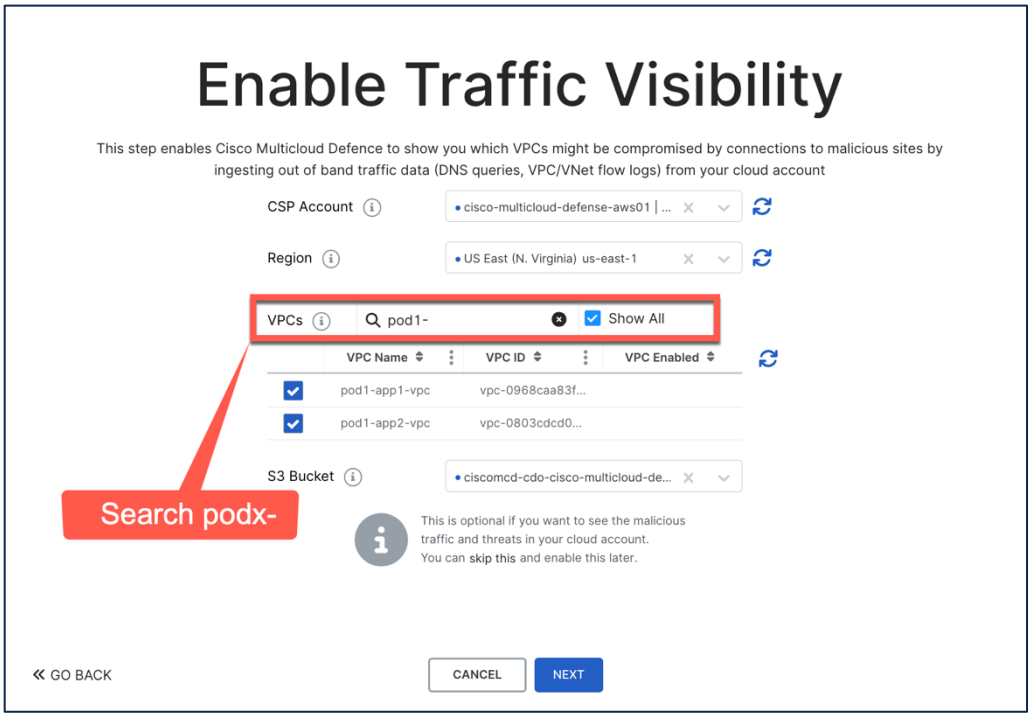


Figure: Configure visibility for AWS VPCs

3. Once visibility is configured, you will see the Success page.

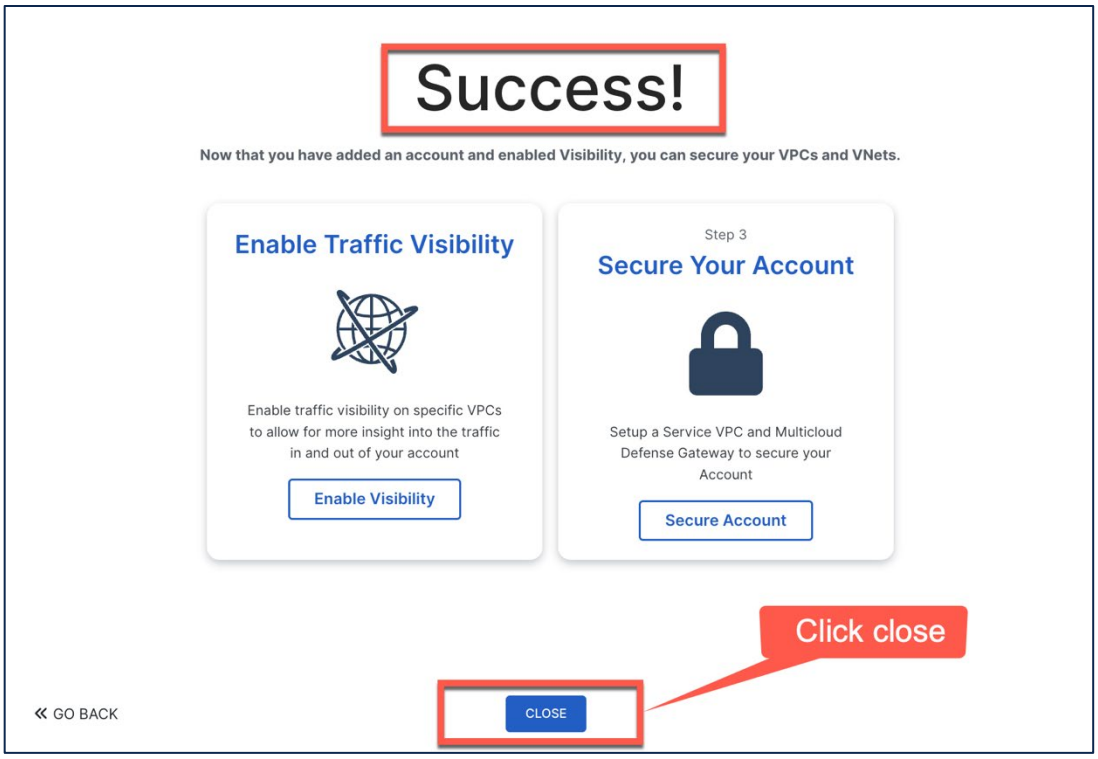


Figure: AWS visibility configuration complete

Task 2. Enable Azure visibility.

1. Click Enable Visibility.
2. Fill out all the fields in the box as shown below, except the storage account. Copy the BASH command to your clipboard.
   - **CSP account:** cisco-multicloud-defense-azure01
   - **Region:** eastus2 (name: East US 2 eastus2)
   - Copy the bash command on a notepad and replace **<storage account name>** with **podxazurestr** (where x is your pod number)
     - Example: If your pod number is 1 (original bash command)
       bash <(curl -Ls https://raw.githubusercontent.com/valtix-security/cli-azure-setup/master/discovery.sh) -l eastus2 -s **<storage account name>** -w https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/azure
     - Edited bash command (replace x with your pod number)

       bash <(curl -Ls https://raw.githubusercontent.com/valtix-security/cli-azure-setup/master/discovery.sh) -l eastus2 -s **pod1azurestr** -w https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/azure

Note: Use podx- instead of using pod1-

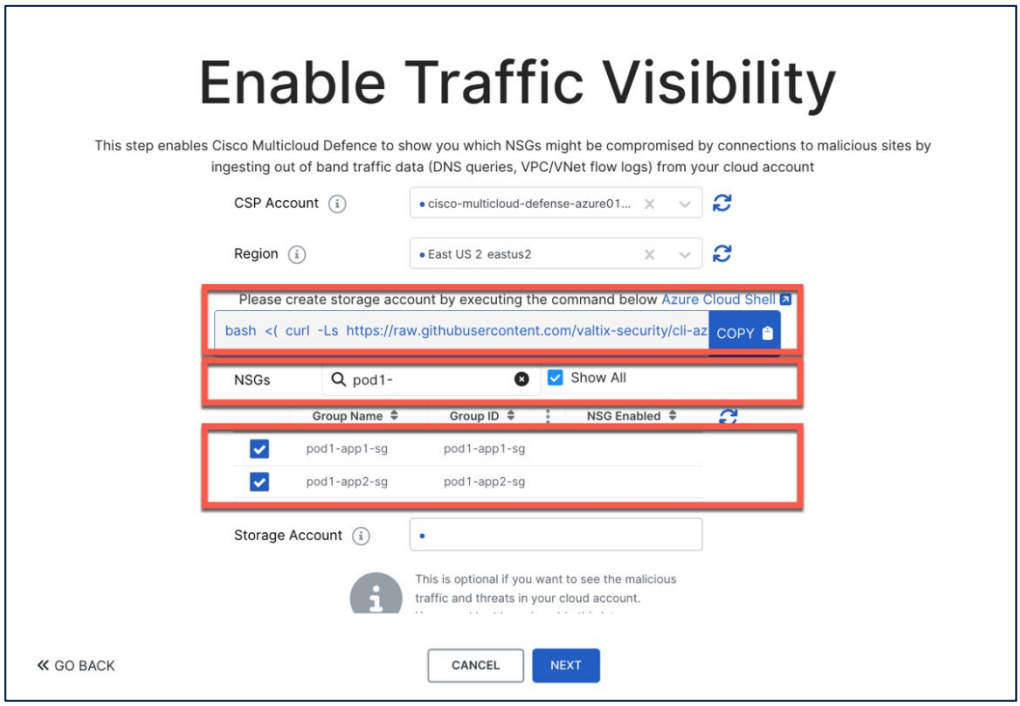
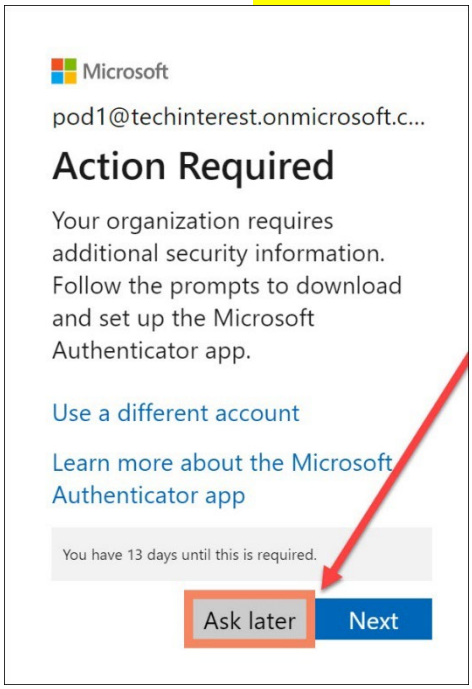


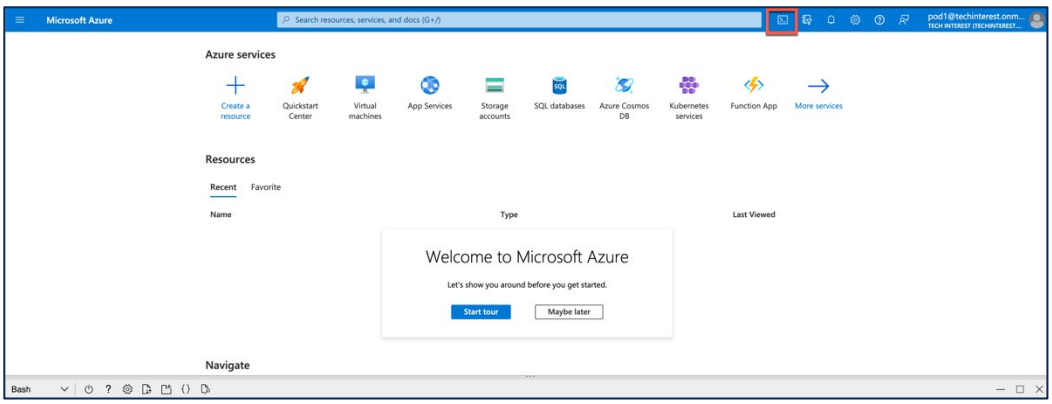
Figure: Configure visibility for Azure NGGs (before storage account creation)

3. Access the Azure portal using the URL https://portal.azure.com
   - **Username**: podx@techinterest.onmicrosoft.com (where x is your pod number)
   - **Password**: The same password as you used for the jump-host.
   - **For MFA**: Select **Ask later** to avoid setting up two factor authentication



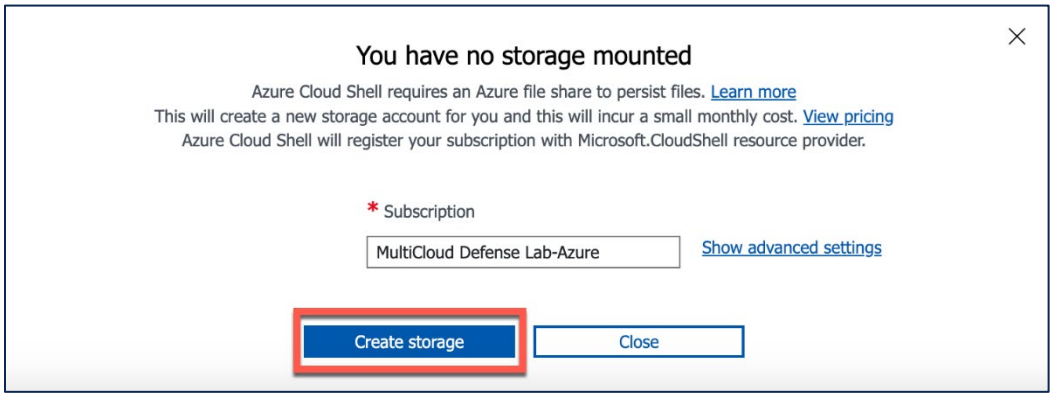
4. In the Azure portal, open the Azure Cloud Shell.

a. Select Bash



b. If you get the message **"You have no storage mounted"**

c. Select Create storage



5. Select BASH if prompted.

a. Paste the BASH command you copied in the step 2.

b. Hit **<Enter>** to run the BASH script.

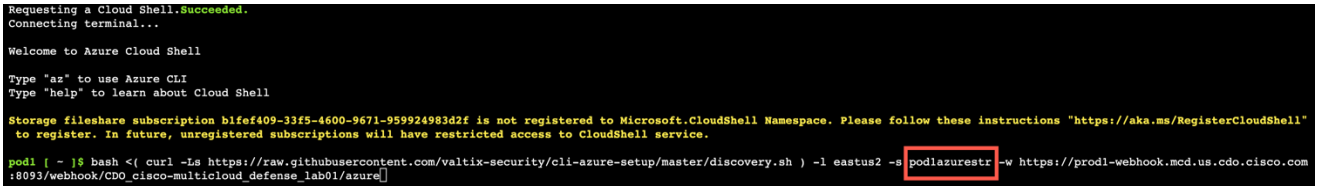


Figure: Storage account creation

c. Select y for the following

i. this the subscription you want to onboard to Multicloud Defense? [y/n] **y**

ii. Continue creating? [y/n] **y**

6. Once the script finishes, it will print out the storage account ID.

Figure: Storage account creation complete

7. Copy the storage account ID and paste it into the MULTICLOUD DEFENSE Controller UI. Select the two NSGs with associated with your pod.
8. Then click **Next**.

Note: Use podx instead of using pod1-

Figure: Configure visibility for Azure NSGs (after storage account creation)

9. Once visibility is configured, you will see the Success page.

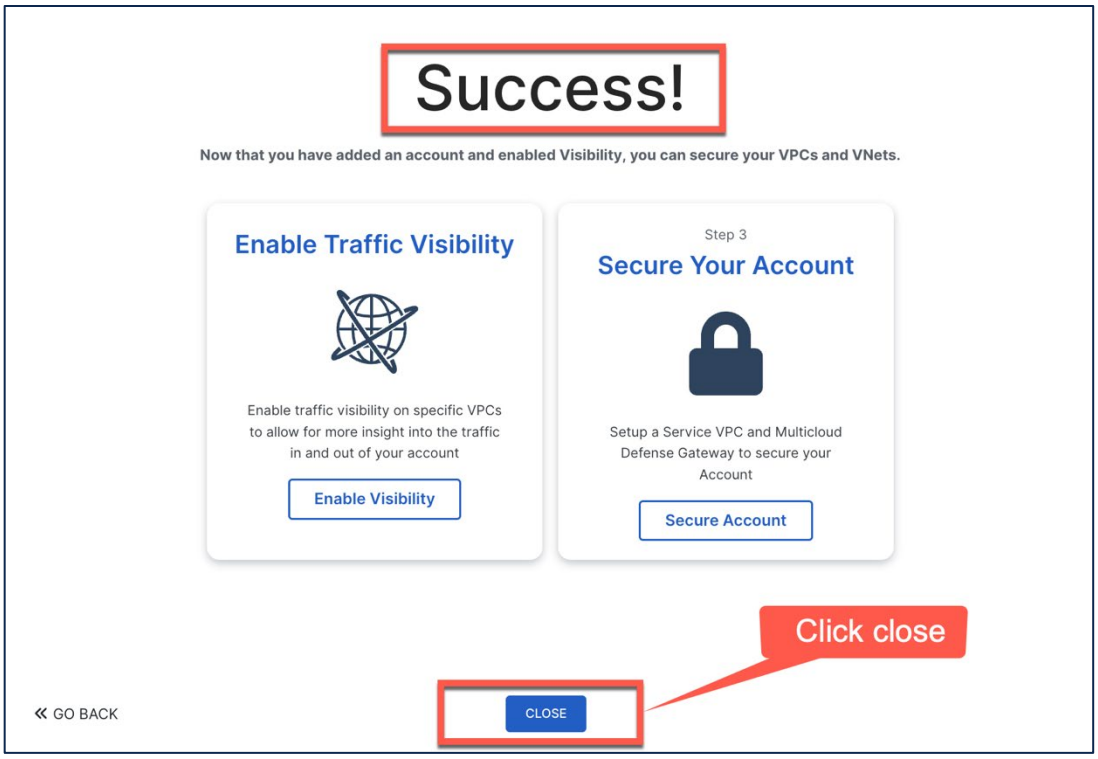


Figure: Azure visibility configuration complete

Task 3. Utilize visibility.

During this task, take a few minutes to familiarize yourself with the visibility features of Cisco Multicloud Defense. It will take a while for visibility to become interesting.

Below is an example of what AWS looked like a few hours after pod deployment.

1. In the Multicloud Defense Controller console, select **Discover > Traffic > VPC**. Scroll through the page and observe the available information.
2. In the Multicloud Defense Controller console, select **Discover > Traffic > Topology**.

Figure: Account traffic topology

3. Click on the **us-east-1 | cisco-multicloud-defense-aws-01** region for **AWS**.
4. Focus on inbound traffic. Since bots are constantly scanning the AWS public address space, you should see some inbound traffic. If you do not, generate inbound traffic by connecting to the public IP of one of the AWS instances with a browser.

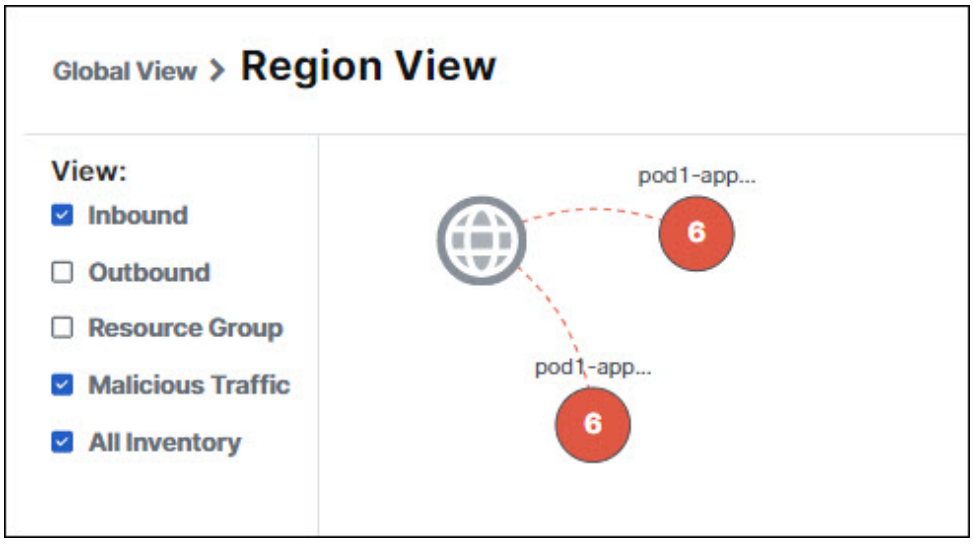


Figure: Inbound traffic

5. Click on one of the servers in the diagram. You should see some traffic details. Below is what this looks like for a server that has been on the Internet for several hours.

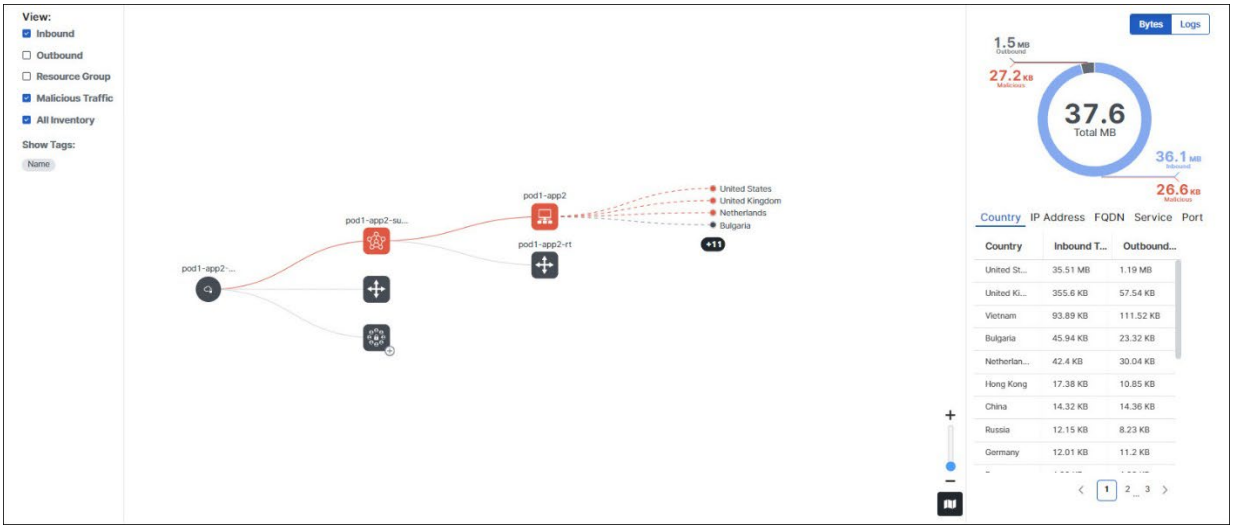


Figure: Flow details for one of the AWS instances

6. Mouse over the subnet and instance icons. Drill down to see the traffic details.

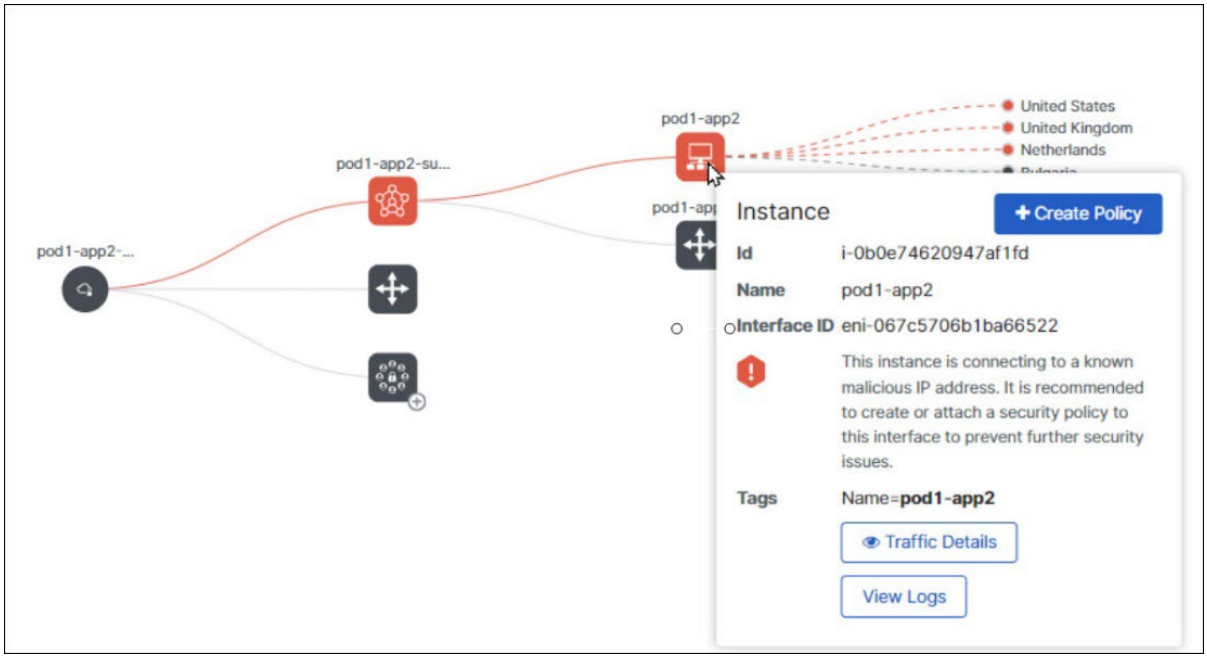


Figure: Traffic details and flow logs

7. Click on **View Logs** to see the traffic logs.

Exercise 3 – Create a security policy and attach it to a ruleset

In this exercise, you will create a tag-based policies to:

- prevent social security information from being exported from one of the spoke instances.
- allow podx-app1 to podx-app2 connection.
- Protect the webserver with IPS/WAF

Task 1. Create a rule set for egress and east-west traffic

In this task, you will create a rule set that will apply for both egress and east-west traffic. The first rule will be for egress traffic. The second rule will be for east-west traffic.

Before you create the rules, you will create two address objects and a data loss prevention profile to use in the rules.

Note: You will see health check rules when you start adding rules, do not delete health check rules. These health check rules are added for communication between the gateway and Multicloud Defense Controller.

1. Create address object for **app1 server** (**more info:** Create an address object that uses a user-defined tag for podx-app1 – These tags are dynamically updated in the real time)
    a. Navigate to **Manage > Security Policies > Addresses**
    b. Click **Create**.
    c. Select **Src/Dest**.
    d. Provide the name **podx-app1-egress** (where **x** with your pod number)
    e. Select the object type as **User Defined Tag**
    f. Under the Resource Tag table, select the key **role** and value **podx-prod**. **Leave everything else empty. (where x is your pod number)**
    g. Click **Save** to save the address object.

2. Create address object for **app2 server** (**more info:** Create an address object that uses user-defined tags for podx-app2 – These tags are dynamically updated in the real time)
    a. Navigate to **Manage > Security Policies > Addresses**
    b. Click **Create**.
    c. Select **Src/Dest**.
    d. Provide the name **podx-app2-egress**, (where **x** with your pod number).
    e. Select the object type as **User Defined Tag**
    f. Under the Resource Tag table, select the key **role** and value **podx-shared**. **Leave everything else empty. (where x is your pod number)**
    g. Click **Save** to save the address object.

3. Create data loss prevention profile (**more info:** Create a DLP filter list for SSN numbers. This filter list will be used later in the exercise for enabling DLP rule egress traffic)
    a. Go to **Manage > Profiles > Data Loss Prevention**.

b. Click **Create**
c. Provide the name **podx-block-ssn** – where x is your pod number
d. In the DLP Filter List table, select type **US Social Security Number** in the **Patterns** text column/field
e. In the DLP Filter List table, select type **US Social Security Number Without Dashes** in the **Patterns** text column/field
f. Set **1** in the **Count** (sending 1 or more SSNs in the traffic would trigger the action)
g. Select **Deny Log** as the Action
h. **Save** the profile

4. Now we have all the components to create a policy rule set (**more info:** Create an egress policy)
   a. Click **Manage > Security Policies > Rule Sets**
   b. Click on **Create** button
   c. Provide a name **podx-egress-policy** (where **x** is your pod number)
   d. Leave **Type** set to **Standalone**
   e. Click **Save**

5. Add the first rule to the rule set (**more info:** Create a forwarding rule in the egress policy – This policy allows egress traffic from podx-app1, this policy uses SNAT for outbound traffic, balanced IPS policy and DLP policy (podx-block-ssn))
   a. Click the text **podx-egress-policy** (where **x** is your pod number)
   b. Click **Add Rule** create a new rule. A new rule editor opens in the slide-over panel on the right
   c. Fill in the following information:

Parameter	Value
Name	rule1
Type	Forwarding
Service	sample-egress-forward-snat-tcp **(Info: SNAT and no decryption)**
Source	podx-app1-egress
Destination	internet
Action	Allow Log
Network Intrusion	ciscomcd-sample-ips-balanced-alert
Data Loss Prevention	podx-block-ssn

   d. Click **Save** to save the rule

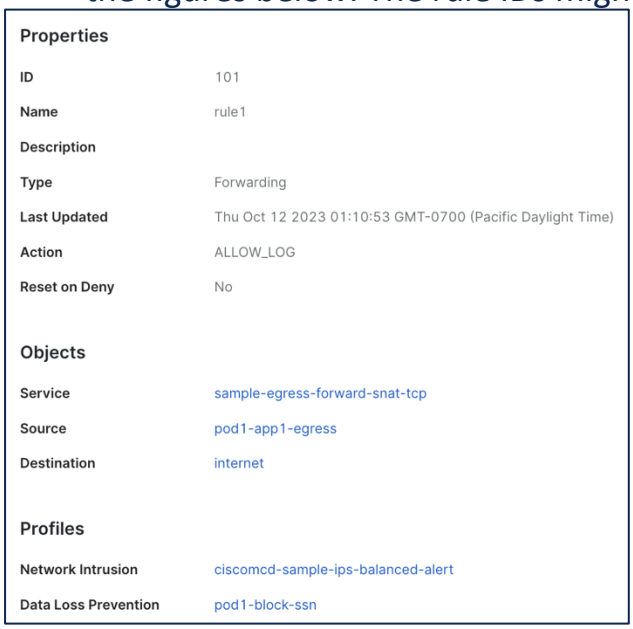
6. Add the second rule to the rule set (**more info:** Create a forwarding rule in the egress policy – This policy allows traffic from podx-app1 to podx-app2)
   a. Click **Add Rule** create a new rule. A new rule editor opens in the slide-over panel on the right
   b. Fill in the following information:

Parameter	Value
Name	rule2
Type	Forwarding
Service	sample-egress-forward-tcp **(Info: no SNAT and no decryption)**
Source	podx-app1-egress
Destination	podx-app2-egress

Action	Allow Log

c. Click **Save** to save the rule
d. Click **Save Changes** and then click **Save** to save the rule set

7. After saving the changes, click on each rule and confirm that they are configured as in the figures below. The rule IDs might be different.



Task 2. Create a rule set for ingress traffic.

1. Create address object (**more info:** Create an address object for podx-app1)
   a. Navigate to **Manage > Security Policies > Addresses**
   b. Click **Create**
   c. Select **Reverse Proxy Target**.
   a. Provide the name **podx-app1-ingress**, where x with your pod number
   d. Select the object type as **IP/FQDN**
   e. In the value field, enter in **10.x.100.10**, where x with your pod number
   f. Click **Save** to save the address object.
2. Create a service object (**more info:** Create a service object for podx-app1)
   a. Click on **Manage > Security Policies > Services**
   b. Click on **Create**
   c. Select **Reverse Proxy** as service type.
   d. Provide the name **podx-app1**, where x with your pod number
   e. In the service table, enter the following:
      i. Dst Port: **80**
      ii. Protocol: **TCP**
      iii. Target Backend Port: **80**
      iv. Protocol: **HTTP**
      i. Address: **podx-app1-ingress**, where x with your pod number
   g. Click **Save** to save the service object.
8. Create a policy rule set
   a. Click **Manage > Security Policies > Rule Sets**

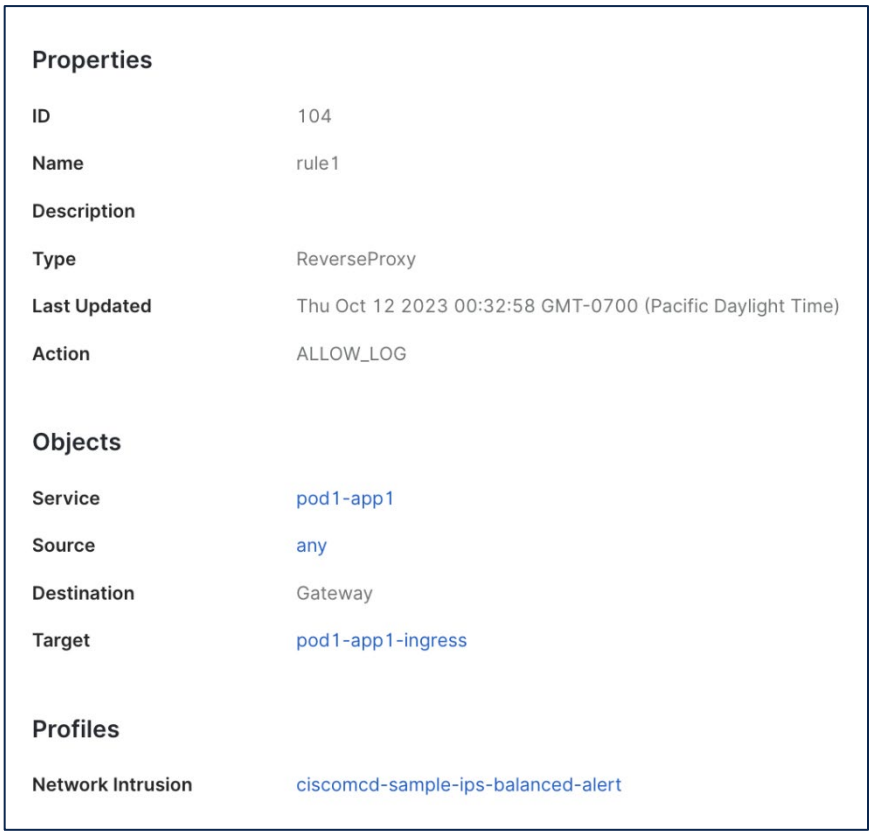
b. Click on **Create** button
c. Provide a name **podx-ingress-policy**, where x is your pod number
d. Leave **Type** set to **Standalone**
e. Click **Save**

3. Create a rule for ingress traffic
   a. Click the text **podx-ingress-policy**, where x is your pod number
   b. Click **Add Rule to** create a new rule. A new rule editor opens in the slide-over panel on the right
   c. Fill in the following information:

Parameter	Value
Name	rule1
Type	Reverse Proxy
Service	podx-app1 (where **x** is your pod number)
Source	any
Destination	Gateway
Action	Allow Log
Network Intrusion	ciscomcd-sample-ips-balanced-alert

   d. Click **Save** to save the rule
   e. Click **Save Changes** and then click **Save** to save the rule set

4. After saving the changes, click on each rule and confirm that they are configured as in the figures below. The rule IDs might be different.



Out-of-scope for this lab - If you want to test traffic the following traffic, create additional rules as shown in exercise 3. If you want to add more rule, please come back to this exercise after finishing other exercises.

- podx-app2 to podx-app
- podx to internet
- internet to podx-app2

Exercise 4 – Protect AWS infrastructure

Centralized security model

In this exercise, we will deploy a security VPC with Ingress and Egress gateway(s). This architecture is based on all the best practices listed by the cloud provider.

- **Ingress Gateway** (podx-ingress-gw-aws) inspects ingress traffic (i.e., traffic from the internet to the web servers). This architecture uses AWS Network Load Balancer as a frontend. The users on the internet uses AWS NLB endpoint (FQDN) to access podx-app1-aws
- **Egress Gateway** (podx-egress-gw-aws) inspects east/west traffic (i.e., from podx-app1-aws to podx-app2-aws) and it also inspects egress traffic (i.e. from podx-app1-aws, podx-app2-aws to internet). In this architecture, we use AWS Gateway Load Balancer (GWLB), Gateway Load Balancer Endpoint (GWLBe) and Geneve protocol.

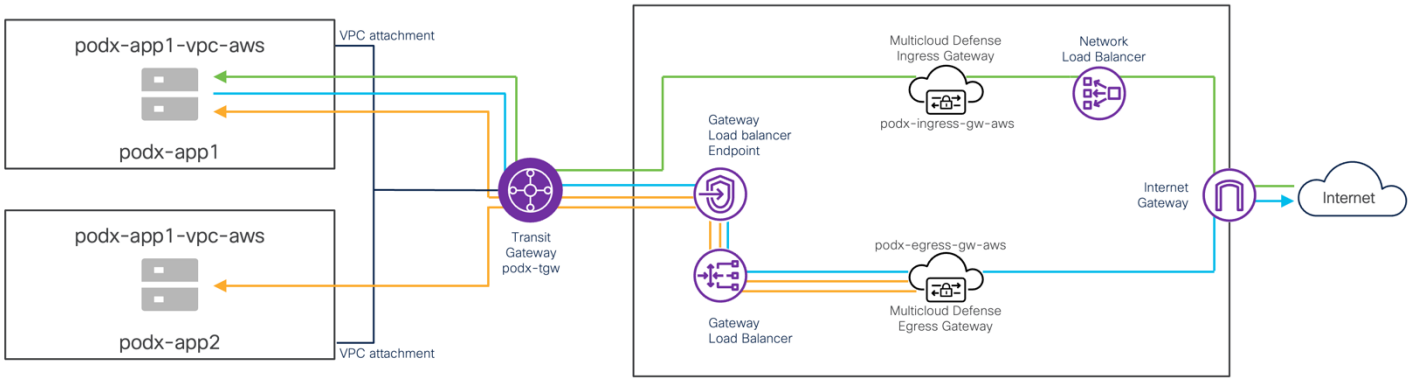


Figure: Centralized security model – AWS

The AWS transit gateway interconnects VPC, however, traffic is not routed directly from one VPC to another, it is forwarded to the Security VPC for inspection.

Task 1. Deploy the service VPC and gateways

1. In the Multicloud Defense Controller console, select the **Setup** tab. Click on **Secure Account**.

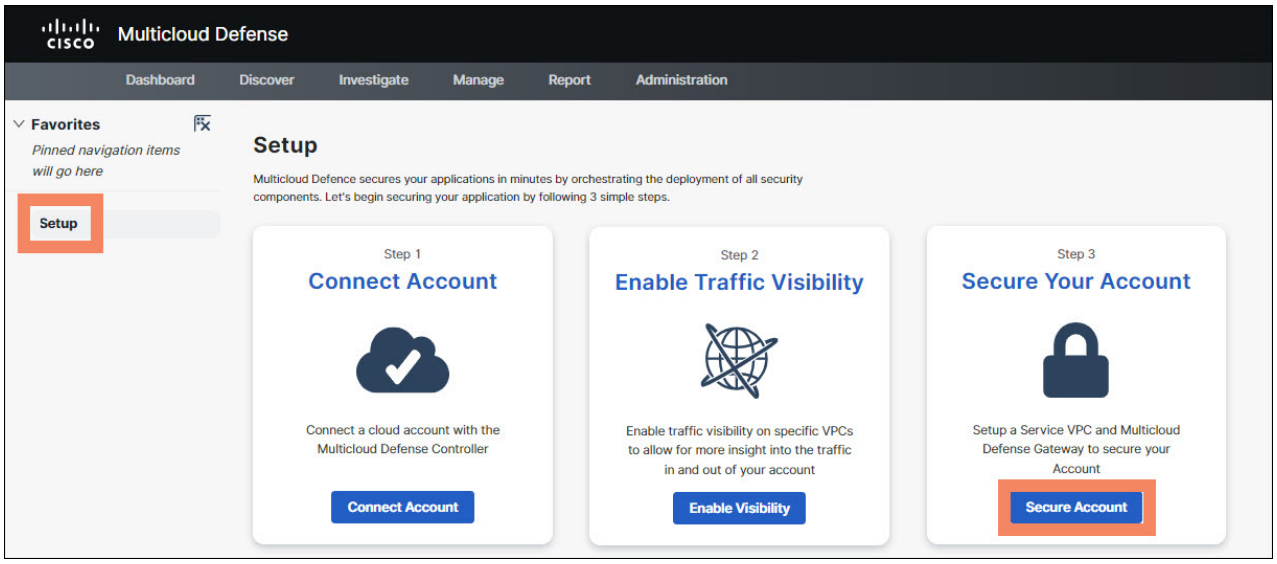


Figure: Secure Account

2. Note that **Centralized** is selected. Click **Next**.

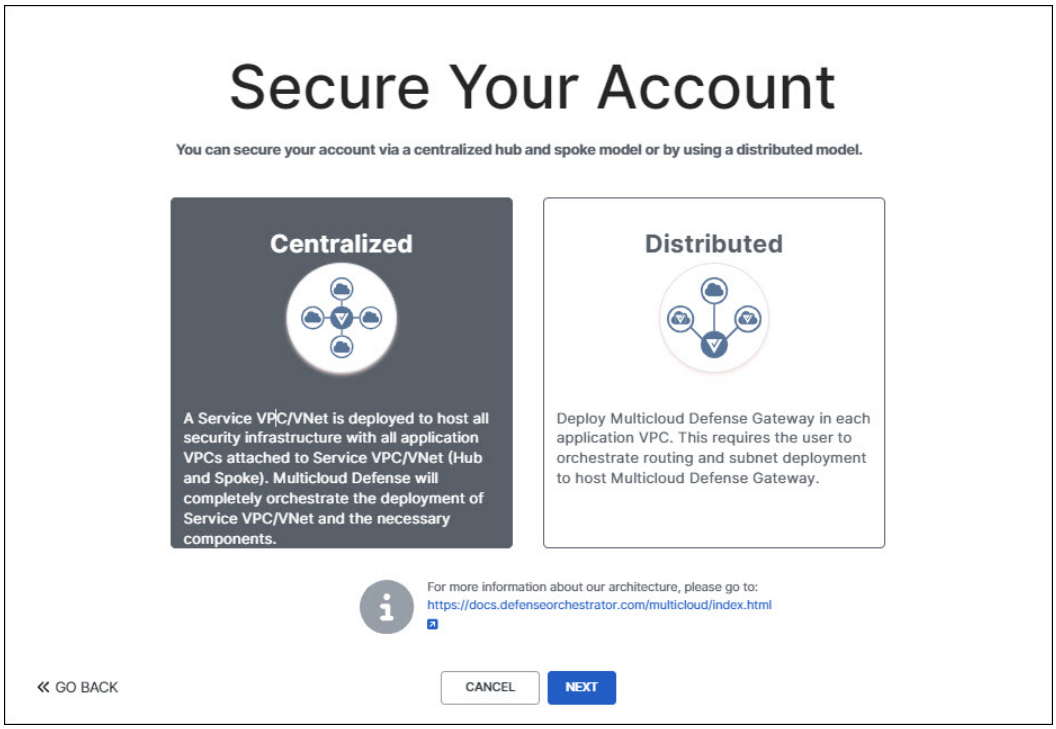


Figure: Select centralized security model

3. Configure the service VPC as follows.
    a. Name: **podx-svpc-aws** (where x is your pod number)
    b. CSP Account: the AWS account

c. Region: **East US (N. Virginia) us-east-1**

d. CIDR: **192.168.x.0/24** (where x is your pod number)

e. Availability Zones: **us-east-1a**

> In a production environment, you would want to specify more than one availability zone. This would result in gateways being deployed in each selected zone. However, for simplicity (and to reduce cost) you will only deploy in one availability zone.

f. Transit Gateway: **create_new**

g. Transit Gateway Name: **podx-tgw** (where x is your pod number)

h. Click auto accept shared attachments.

i. Click **Next**.

Note: Use podx instead of using pod1-

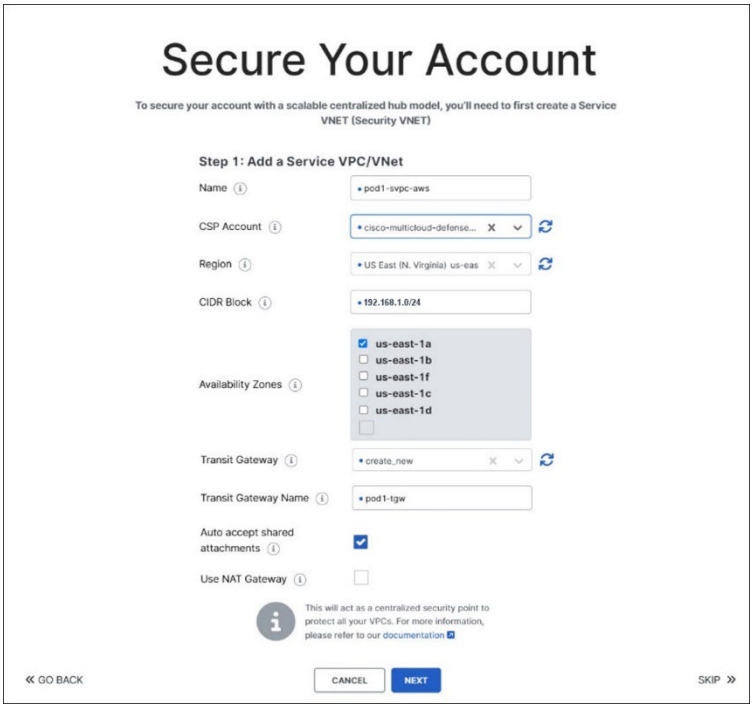
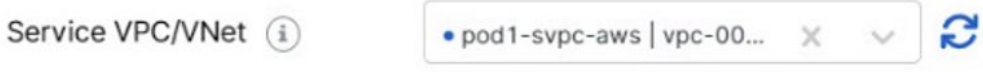


Figure: Configure service VPC

4. On deploy gateway screen, click on "Service VPC/VNet and search for podx- and wait this service VPC/VNet to become active v/s active pending. **If this doesn't change within two minutes, click the refresh button**



5. Add the gateways using the following information.

a. CSP Account: **cisco-multicloud-defense-aws01**

b. Service VPC/VNet: **podx-svpc-aws** (where **x** is your pod number)
Note: Wait for the service VPC to become active
Note: If the service VPC is stuck in active-pending for more than 1 minute, please let the proctor know. You may need to refresh the inventory.

c. Instance Type: 2 Virtual CPU

d. MCD Gateways: Check Ingress and East-West & Egress

e. Ingress Gateway Name: **podx-ingress-gw-aws**

f. Ingress Gateway Policy Ruleset: **podx-ingress-policy**

g. East/West Gateway Name: **podx-egress-gw-aws**

h. Egress Gateway Policy Ruleset: **podx-egress-policy**

i. IAM Gateway Role Name: ciscomcd-gateway-role

j. SSH Public Key: **podx-keypair** (where **x** is your pod number)

k. Keep the other settings as shown in the figure below but use your pod number instead of 1.

l. Click **Next**.

Note: Use podx instead of using pod1-

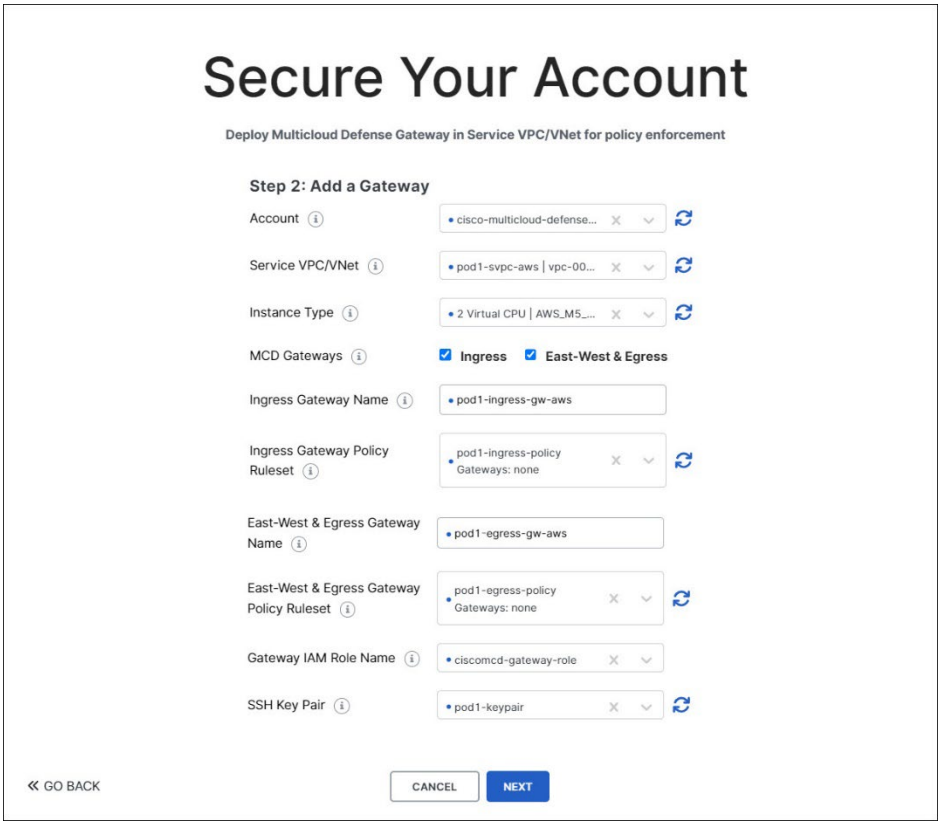


Figure: Create gateways

6. In the Multicloud Defense controller UI, navigate to **Manage > Gateways**. Confirm that two gateways are active.

Task 2. Attach application VPCs to service VPC through the transit gateway

1. In the Multicloud Defense Controller console, navigate to **Manage > Cloud Accounts > Inventory**.
2. Click on **VPCs/VNets**.
3. Filter the VPCs for **Region *EQUAL_TO* us-east-1**.
4. Find **podx-app1-vpc** (where **x** is your pod number). Click **Secure Now**.
    a. Service VPC: **podx-svpc-aws** (where **x** is your pod number)
    b. Select the **podx-app1-rt** route table (where **x** is your pod number).
    c. Inspect the route table and the changes that the controller will make.
        i. The default route next hop will be changed from the Internet gateway to the transit gateway attachment (and therefore through the egress gateway). This will apply to egress and east-west (inter-VPC) traffic.
        ii. The static routes to 20.12.187.121/32 and 52.9.113.154/32 remain unchanged. These routes were created so traffic from the jump host (and back-up jump host) would bypass the gateway. This makes testing from the jump host easier.
        iii. The local route was created by AWS and applies to intra-VPC traffic.
    d. Click **Save**.

Note: Please do not check podx-app1-mgmt-rt, this routable is used later in the exercise 7 (distributed security mode)

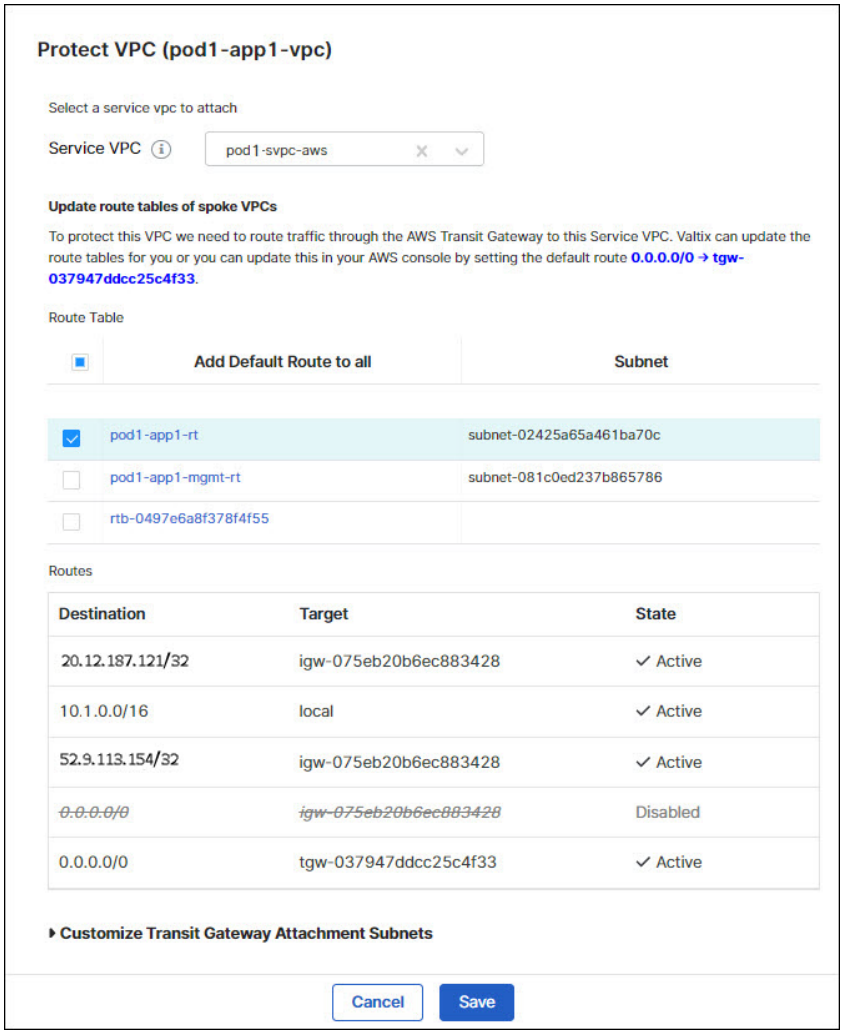


Figure. Secure a VPC

7. Repeat the previous step for **podx-app2-vpc** (where **x** is your pod number).

Task 3. Testing traffic filtering

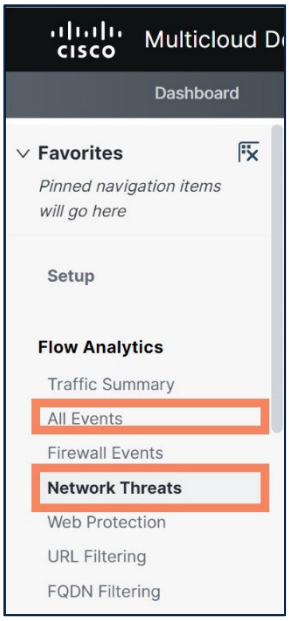
1. On the jump-host, you should still have an SSH session to the AWS App1 Linux server. If you do not, from the jump-host, type:
   ssh -i aws ubuntu@podx-app1-aws
   Where x is your pod number.
2. Test east-west traffic filtering.
   a. From podx-app1, curl podx-app2:
   ubuntu@ip-10-x-100-10:~$ **curl 10.[100+x].100.10/status; echo**
   where "x" is your pod number. This command should succeed. The policy rule set allows TCP.
   b. From podx-app1, ping podx-app2:
   ubuntu@ip-10-x-100-10:~$ **ping 10.[100+x].100.10**
   where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
3. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All Events**.

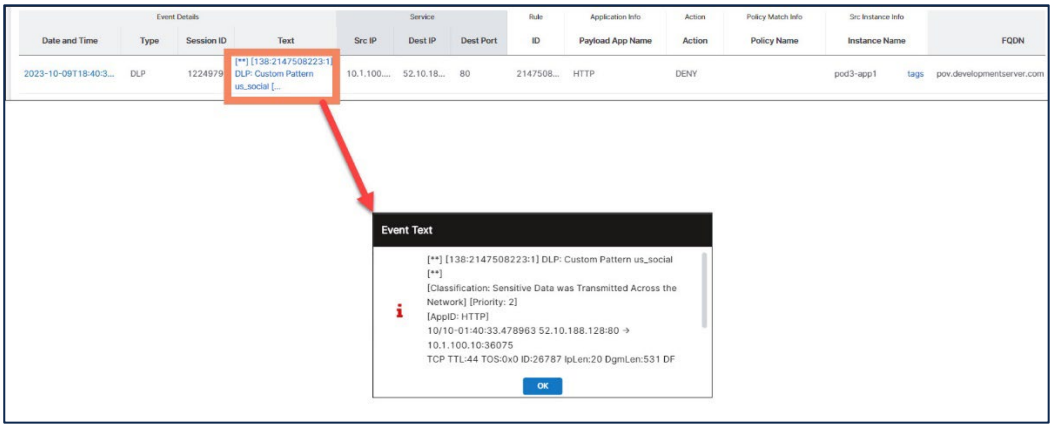
a. Filter by the account (AWS) and gateway (podx-egress-gw-aws (where **x** is your pod number)).
b. Confirm the east-west TCP traffic is being logged.
c. Click on the **Date and Time** field to get event details.

4. Test egress traffic filtering and data loss prevention.
   a. From podx-app1, curl google.com:
      ubuntu@ip-10-x-100-10:~$ **curl google.com**
      where "x" is your pod number. This command should succeed. The policy rule set allows TCP.
   b. From podx-app1, ping google.com:
      ubuntu@ip-10-x-100-10:~$ **ping google.com**
      where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
   c. Use curl to post some personal data:
      curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like the cloud"
      The command should succeed.
   d. Use curl to post some personal data containing a valid SSN:
      curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 555-55-5555"
      The command should fail.
   e. Try the following command:
      curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 123-45-6789"
      The command should succeed. The number 123-45-6789 is indeed **not** a valid SSN.

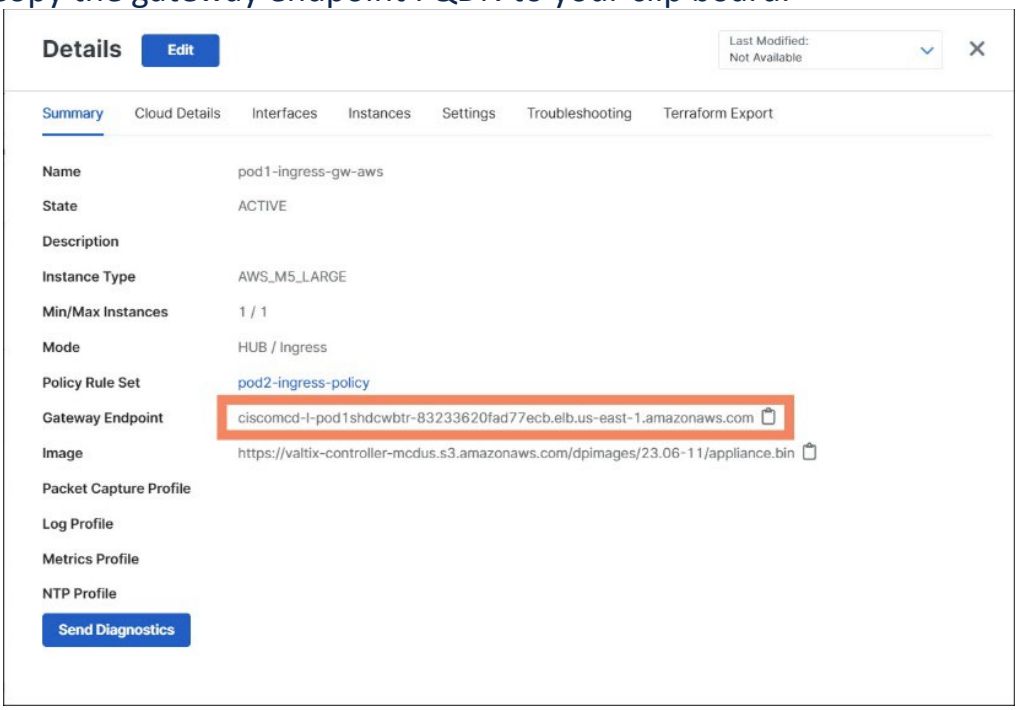
Note. If you have a machine with a browser to do this test, navigate to http://pov.developmentserver.com/share.html

5. In the Multicloud Defense console look for the egress traffic. You should not have to change the filter, but you may have to click the refresh button (Load latest events) in the UI (not the browser).
6. Toggle between All Events and Network Threats to focus on DLP. Click on event text to make is easier to read.





7. Click on the **Date and Time** field to get event details.
8. Test ingress forward proxy.
    a. In the Multicloud Defense console, navigate to **Manage > Gateways**.
    b. Click on **podx-ingress-gw-aws** (where **x** is your pod number).
    c. Copy the gateway endpoint FQDN to your clip board.



9. Paste this FQDN into a browser outside your pod. You should see the podx-app1 webpage.
10. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All**

Events.

a. Filter by the account (AWS) and gateway (podx-ingress-gw-aws, where **x** is your pod number).
b. Confirm the ingress HTTP traffic is being logged. Since bots are always scanning port 80 on AWS public IP space, you should see ingress traffic from unknown web sites as well as the traffic you generated.

Exercise 5 – Protect Azure infrastructure

In this exercise, we will deploy a security VPC with Ingress and Egress gateway(s). This architecture is based on all the best practices listed by the cloud provider.

- **Ingress Gateway** (podx-ingress-gw-azure) inspects ingress traffic (i.e., traffic from the internet to the web servers). This architecture uses Azure Public Load Balancer as a frontend. The user on the internet uses Azure Gateway endpoint (IP address) to access podx-app1-azure
- **Egress Gateway** (podx-egress-gw-azure) inspects east/west traffic (i.e., from podx-app1-azure to podx-app2-azure) and it also inspects egress traffic (i.e. from podx-app1-azure, podx-app2-azure to internet). In this architecture, we use Azure Internet Load Balancer (ILB) for east/west and egress traffic.

Centralized security model

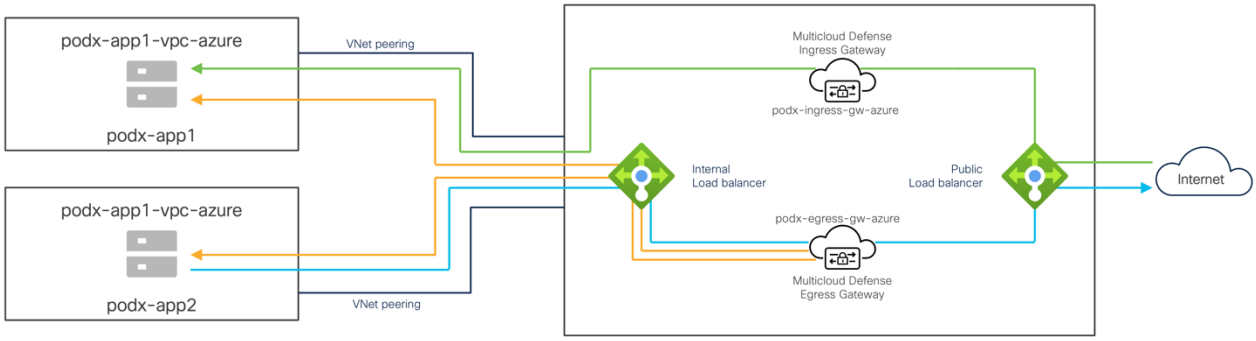


Figure: Centralized security model – Azure

The objective of this exercise is to protect your Azure environment with the MULTICLOUD DEFENSE centralized security model.

Note that MULTICLOUD DEFENSE in Azure can also use the distributed security model. But that is not covered in this lab.

Task 1. Deploy the service VNet and gateways

1. In the Multicloud Defense Controller console, select the **Setup** tab. Click on **Secure Account**.

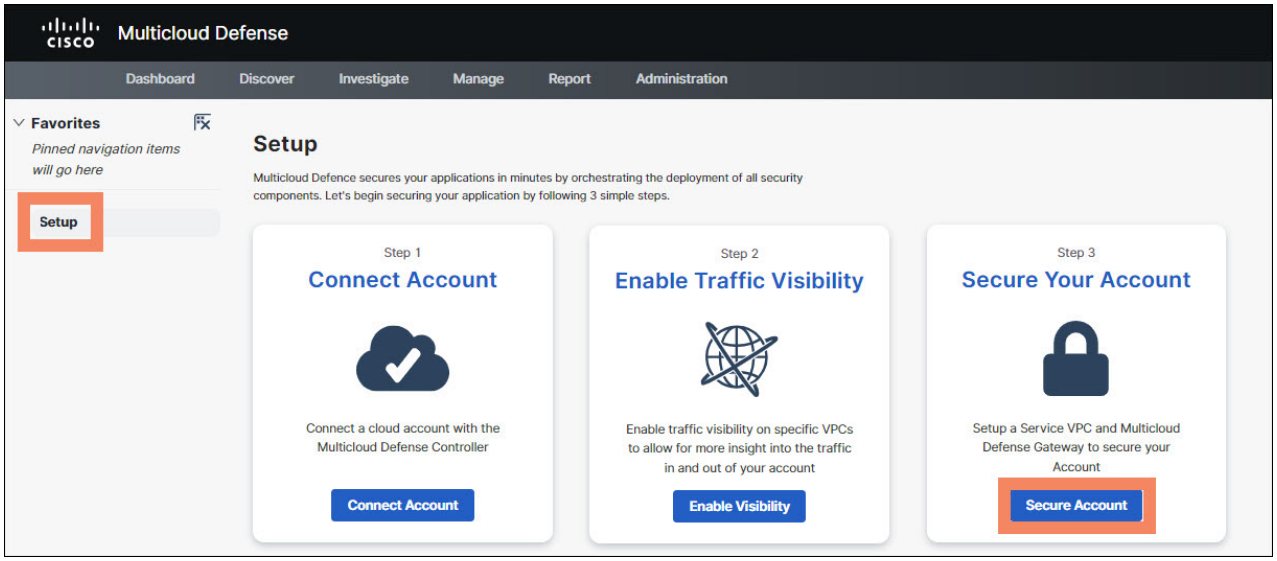


Figure: Secure Account

2. Note that **Centralized** is selected. Click **Next**.

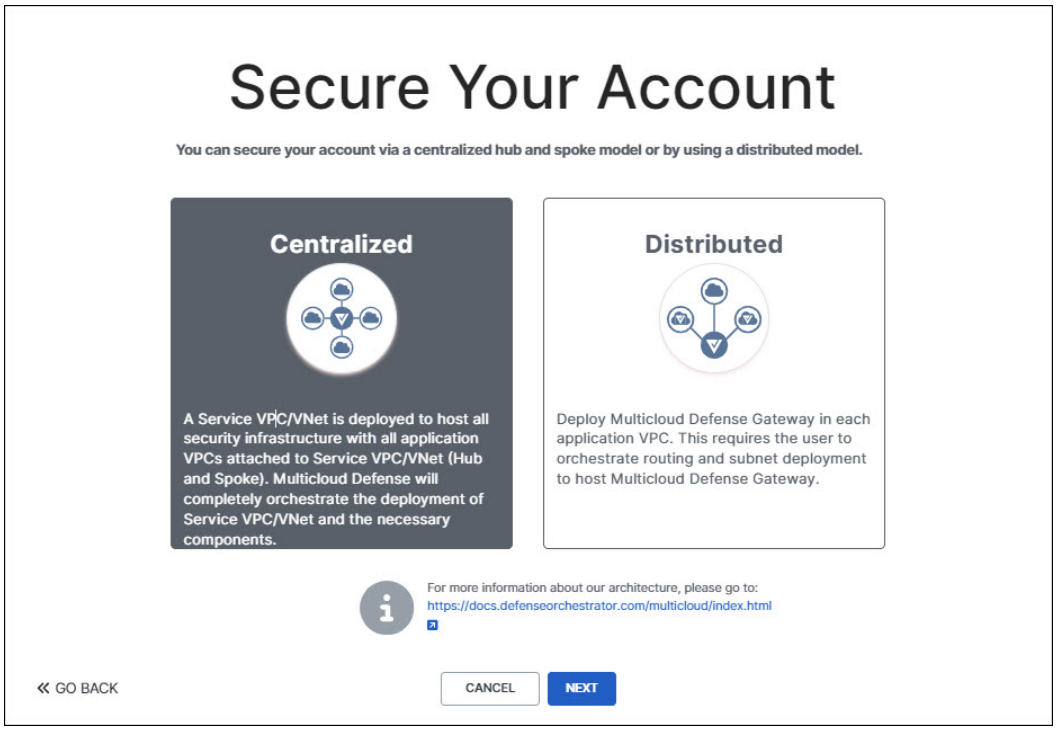


Figure: Select centralized security model

3. Configure the service VNet as follows.
    a. Name: **podx-svnet**, where **x** is your pod number
    b. CSP Account: **cisco-multicloud-defense-azure01**

c. Region: **East US 2 eastus-2**

d. CIDR: **192.168.x.0/24**, where **x** is your pod number

e. Availability Zones: **3**

> For this lab, you will deploy one gateway in **availability zone 3**. For the production environment, we recommend deploying multiple gateways in multiple availability zones.

f. Resource Group: **podx-rg** (where **x** is your pod number)

g. Click **Next**.

Note: Keep the settings as shown in the figure below but use your pod number instead of 1.

Note: Use podx instead of using pod1-

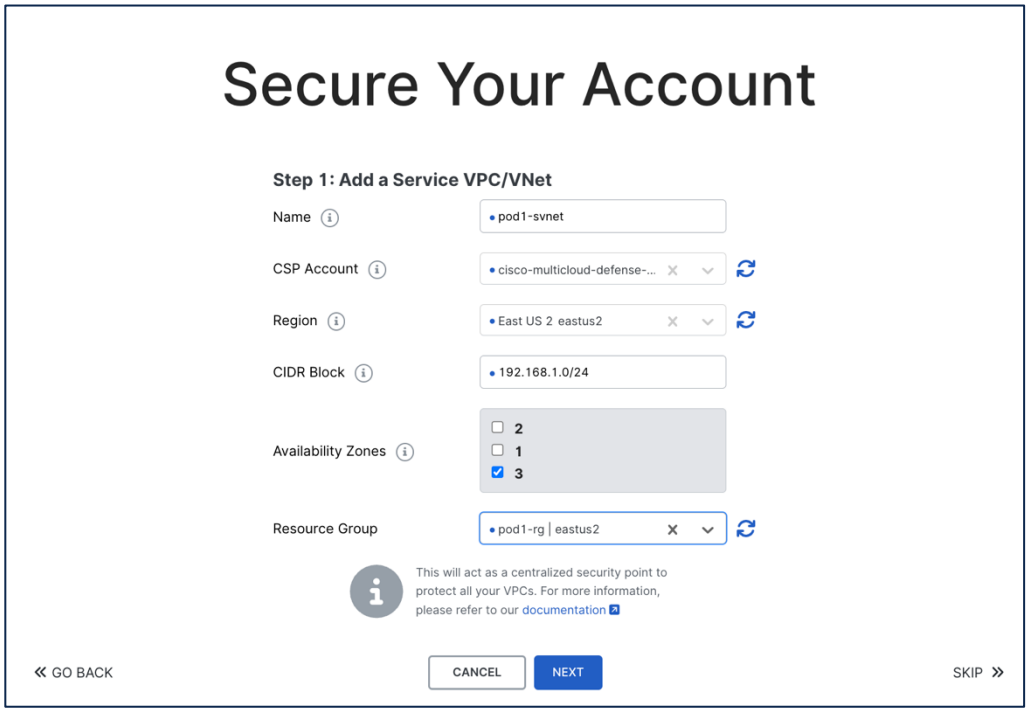
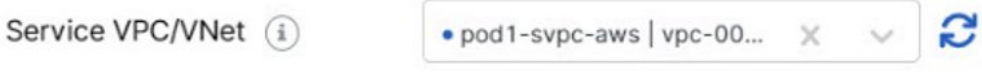


Figure: Configure service VNet

4. On the deploy gateway screen, click on “Service VPC/VNet and search for podx- and wait this service VPC/VNet to become active v/s active pending. If this doesn’t change within two minutes, click the refresh button



5. Add the gateways using the following information.
    a. CSP Account: **cisco-multicloud-defense-azure01**
    b. Service VNet: **podx-svnet**, where **x** is your pod number

    Note: Click the refresh button and wait for the above service-vnet to become active. Once active, then proceed to the next-step.

    c. Instance Type: **2 Virtual CPU**
    d. MCD Gateways: **Check Ingress and Egress**
    e. Ingress Gateway Name: **podx-ingress-gw-azure, where x is your pod number**
    f. Ingress Gateway Ruleset: **podx-ingress-policy, where x is your pod number**
    g. Egress Gateway Name: **podx-egress-gw-azure, where x is your pod number**
    h. Egress Gateway Ruleset: **podx-egress-policy, where x is your pod number**
    i. Resource Group: **podx-rg, where x is your pod number.**
    j. User Assigned Identity ID: Leave this blank. **This field should only be used in conjunction with Azure Vault.**
    k. SSH Public Key: choose **SSH Key Pair** and select **podx-keypair**, where x is your pod number

    Note: Keep the other settings as shown in the figure below but use your pod number instead of 1.

    l. Click **Next**.

    Note: Use podx instead of using pod1-

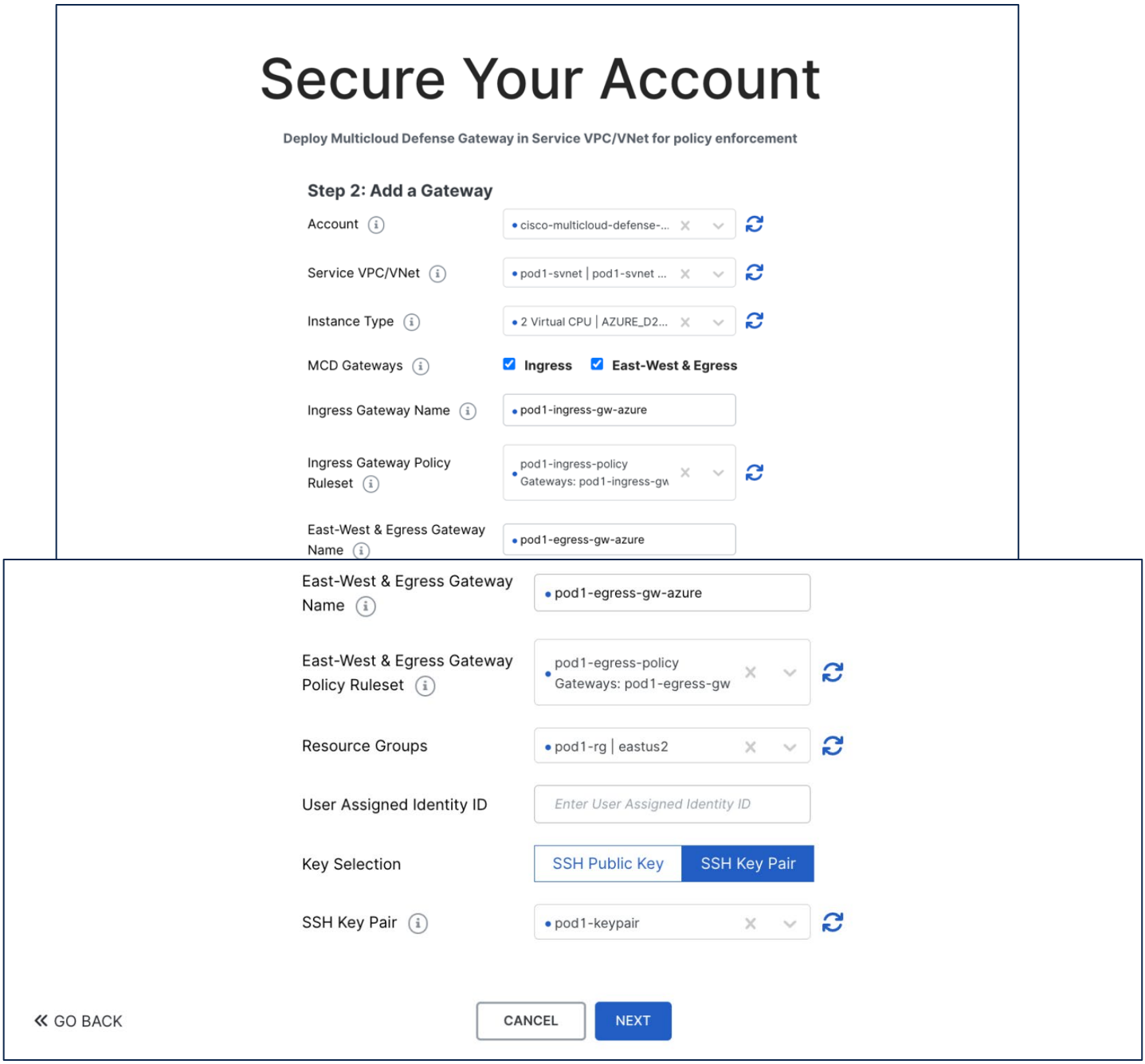


Figure: Create Ingress and Egress gateways in Azure

6. In the Multicloud Defense Controller UI, navigate to **Manage > Gateways**.
7. Search for **podx-** and confirm that four gateways are active – two for AWS and two for Azure. If gateways are active move to the next task.

Task 2. Peer application VNets to Service VNet

1. In the Multicloud Defense Controller console, navigate to **Manage > Inventory**.
2. Click on **VPCs/VNets**.

3. Filter podx- and notices to VNets in the eastus2 region (these are podx-app1-vnet and podx-app2-vnet).
4. Find **podx-app1-vnet**, where x is your pod number. Click **Secure Now**.
    a. Service VPC: **podx-svnet**, where **x** is your pod number
    b. Select the **podx-app1-rt** route table (where **x** is your pod number).
    a. Inspect the route changes that the controller will make.

    Inspect the route table and the changes that the controller will make.

    i. The default route will be added with the egress gateway as the next hop. Prior to this change, there was no need for a default route in the route table because traffic is forwarded by default to the Internet (using Azure effective routing). This will handle both egress and east-west (inter-VNet) traffic

    ii. The static routes to 20.12.187.121/32 and 52.9.113.154/32 remain unchanged. These routes were created so traffic from the jump host (and back-up jump host) would bypass the gateway. This makes testing from the jump host easier.

    c. Click **Save**.

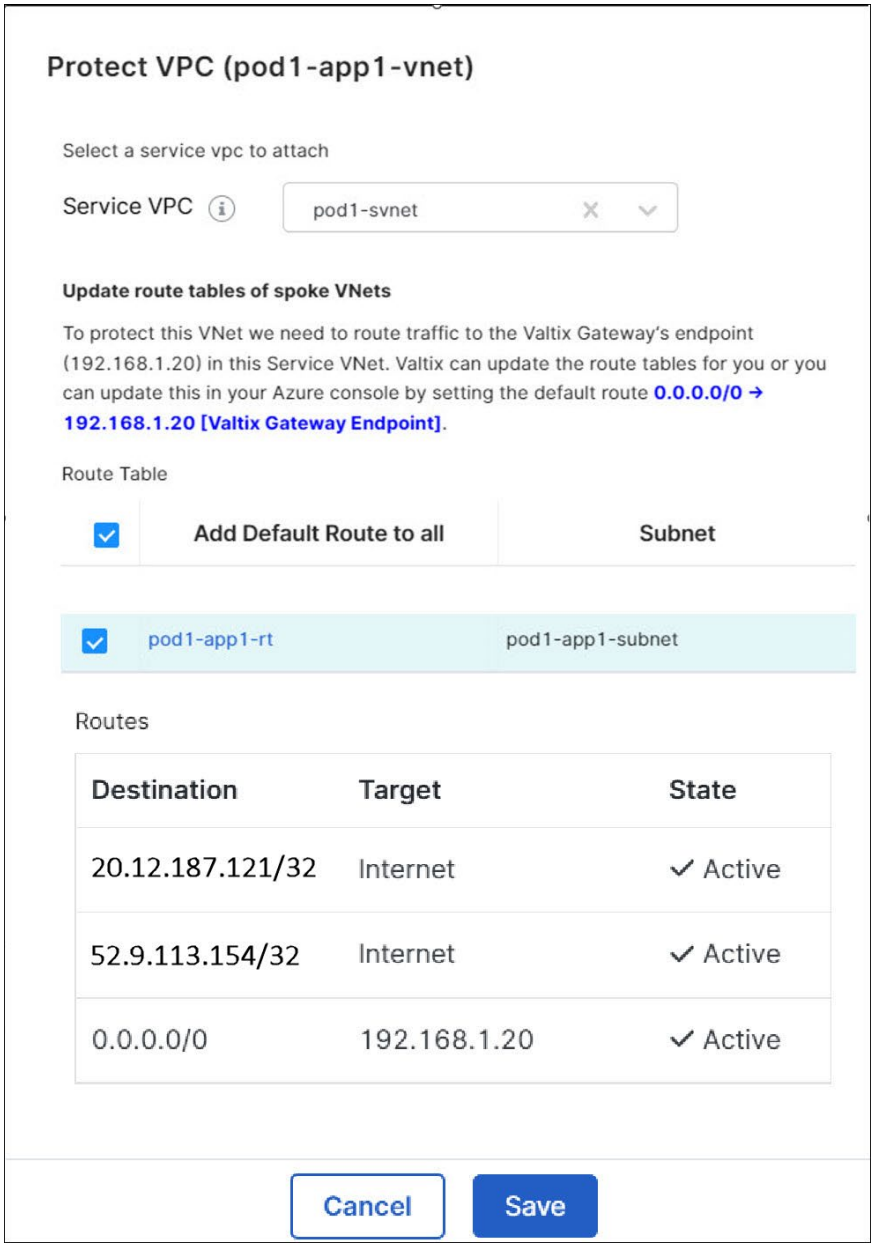


Figure. Secure podx-app1-vnet

5. Repeat the previous step for **podx-app2-vnet** (where **x** is your pod number).

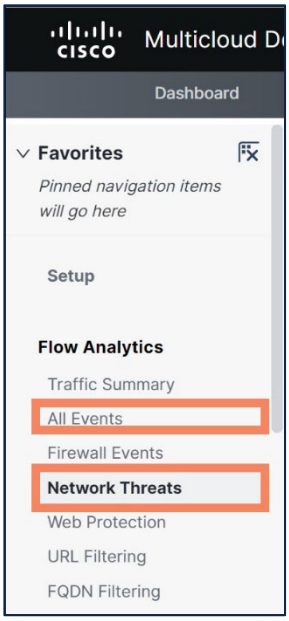
Task 3. Testing traffic filtering

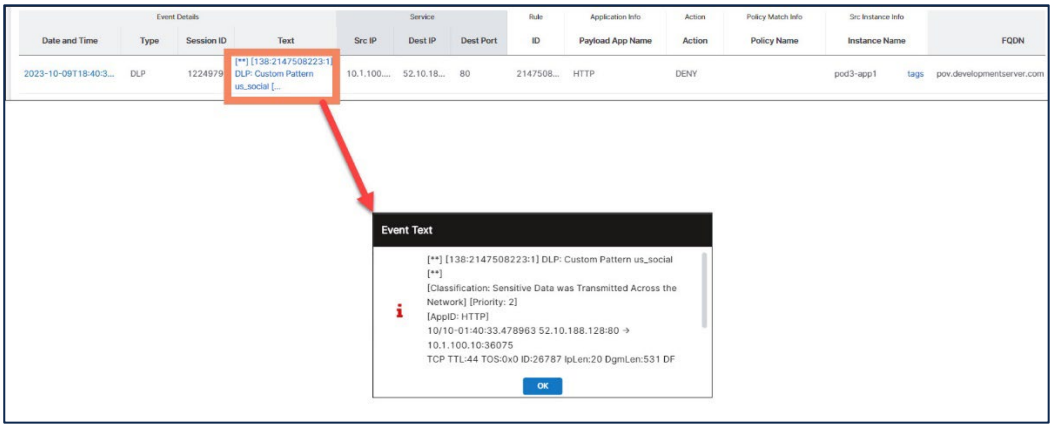
1. On the jump-host, start an SSH session to the Azure App1 Linux server.
    a. Open a new SSH session to the jump-host. Alternatively, you use the existing session by logging out of podx-app1-aws (where **x** is your pod number).
    b. From the jump-host, type:
    ssh -i azure ubuntu@podx-app1-azure
    Where x is your pod number.
2. Test east-west traffic filtering.
    a. From podx-app1, curl podx-app2:
    ubuntu@ip-10-x-100-10:~$ **curl 10.[100+x].100.10/status; echo**
    where **x** is your pod number. This command should succeed. The policy rule set allows TCP.
    b. From podx-app1, ping podx-app2:
    ubuntu@ip-10-x-100-10:~$ **ping 10.[100+x].100.10**
    where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
3. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All Events**.
    a. Filter by the account (Azure) and gateway (podx-egress-gw-azure (where x is your pod number)).
    b. Confirm the east-west TCP traffic is being logged.
    c. Click on the **Date and Time** field to get event details.
4. Test egress traffic filtering and data loss prevention.
    a. From podx-app1, curl google.com:
    ubuntu@ip-10-x-100-10:~$ **curl google.com**
    where "x" is your pod number. This command should succeed. The policy rule set allows TCP.
    b. From podx-app1, ping google.com:
    ubuntu@ip-10-x-100-10:~$ **ping google.com**
    where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
    c. Use curl to post some personal data:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like the cloud"
    The command should succeed.
    d. Use curl to post some personal data containing a valid SSN:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 555-55-5555"
    The command should fail.
    e. Try the following command:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 123-45-6789"

The command should succeed. The number 123-45-6789 is indeed **not** a valid SSN.

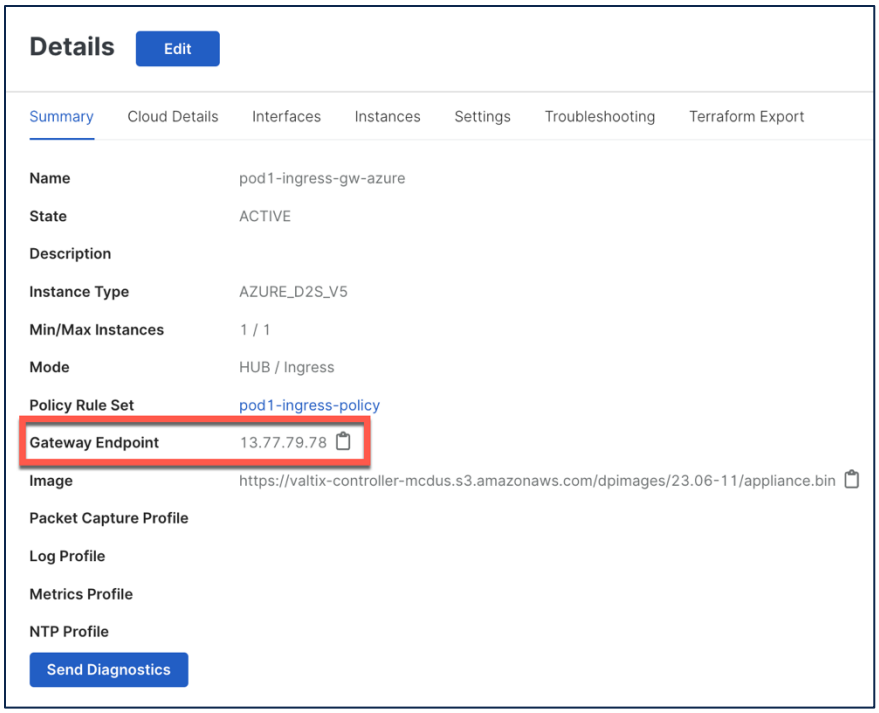
Note. If you have a machine with a browser to do this test, navigate to http://pov.developmentserver.com/share.html

5. In the Multicloud Defense console look for the egress traffic. You should not have to change the filter, but you may have to click the refresh button in the UI (not the brower).
6. Toggle between All Events and Network Treats to focus on DLP. Click on event text to make is easier to read.





7. Click on the **Date and Time** field to get event details.
8. Test ingress forward proxy.
    a. In the Multicloud Defense console, navigate to **Manage > Gateways**.
    b. Click on **podx-ingress-gw-azure**, where x is your pod number.
    c. Copy the gateway endpoint FQDN to your clip board.



9. Paste this gateway endpoint into a browser outside your pod. You should see the podx-app1-azure webpage.
10. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All Events**.
    a. Filter by the account (azure) and gateway (podx-ingress-gw-azure (where x is your pod number)).
    b. Confirm the ingress HTTP traffic is being logged. Since bots are always scanning port 80 on azure public IP space, you should see ingress traffic from unknown web sites as well as the traffic you generated.

Part II – Optional exercises

Exercise 6 – Configure visibility and infrastructure protection in GCP (optional)

Centralized security model

In this exercise, we will deploy a security VPC with Ingress and Egress gateway(s). This architecture is based on all the best practices listed by the cloud provider.

- **Ingress Gateway** (podx-ingress-gw-gcp) inspects ingress traffic (i.e., traffic from the internet to the web servers). This architecture uses GCP External Load Balancer as a frontend. The users on the internet uses GCP Gateway endpoint (IP address) to access podx-app1-gcp
- **Egress Gateway** (podx-egress-gw-gcp) inspects east/west traffic (i.e., from podx-app1-gcp to podx-app2-gcp) and it also inspects egress traffic (i.e. from podx-app1-gcp, podx-app2-gcp to internet). In this architecture, we use GCP Internet Load Balancer (ILB) for east/west and egress traffic.

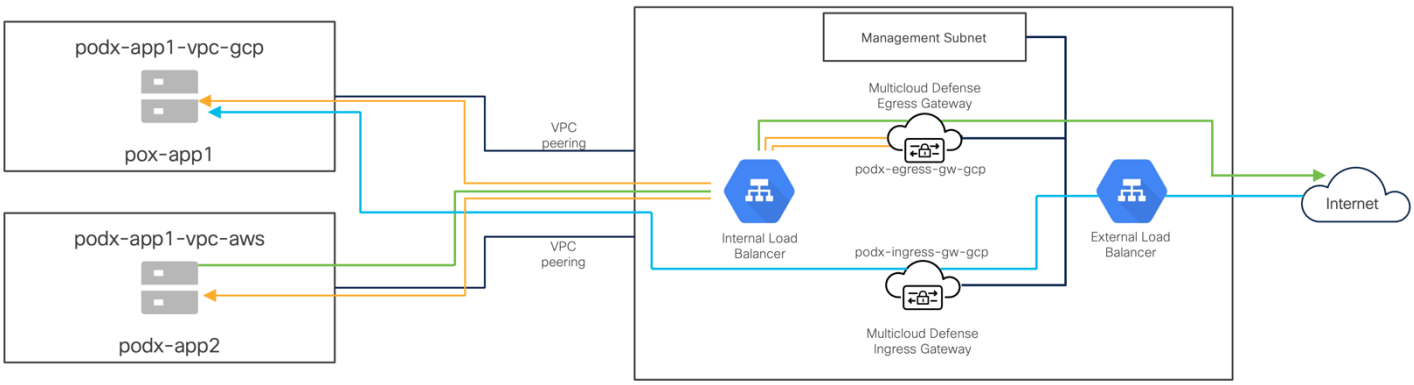


Figure: Centralized Security Model – GCP

Task 1. Enable GCP visibility.

You can only perform this exercise if you have access to a GCP account. If you are interested, first provide the proctor with your e-mail. Once added to the project, you will be sent an email. You need to accept the invite before getting access to the GCP project.

Before proceeding with this exercise, access the GCP portal using the URL https://console.cloud.google.com. Confirm that you have access to the project called. **MultiCloud Defense Lab-GCP**.

1. In the Multicloud Defense Controller console, select the **Setup** tab. Click **Enable Visibility**.
2. Enter the information in the following (except use your pod number).
    a. CSP account: **cisco-multicloud-defense-gcp01**
    b. Cloud Storage: **cisco-multicloud-defense-gcp-str** (select from the dropdown menu, where x is your pod number
    c. VPCs search: **podx-** (where **x** is your pod number)

d. Select VPCs **podx-app1-vpc & podx-app2-vpc** (where x is your pod number.

Note: Use podx- instead of using pod1-

Figure: Configure visibility for GCP VPCs

3. Copy the BASH command to your clipboard and replace **ciscomcd** with podx, where x is your pod number: (For better experience, copy this command on Windows Notepad and replace ciscomcd with podx, where x is your pod number)

   Example: Original bash command

   bash <(curl -sSL https://raw.githubusercontent.com/valtix-security/cli-gcp-setup/main/gcp-traffic-visibility.sh) -i gcp-multiclouddefen-nprd-45523 -p **ciscomcd** -s pod1-gcp-str -v pod1-app1-vpc,pod1-app2-vpc -w "https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/gcp/cloudstorage"

   Edited bash command (replace **ciscomcd** with **podx** – where x is your pod number)

   bash <(curl -sSL https://raw.githubusercontent.com/valtix-security/cli-gcp-setup/main/gcp-traffic-visibility.sh) -i gcp-multiclouddefen-nprd-45523 -p **podx** -s pod1-gcp-str -v pod1-app1-vpc,pod1-app2-vpc -w "https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/gcp/cloudstorage"

4. In the GCP, open the Google Cloud Shell. Paste the BASH command you copied in the previous step. Be sure the **-i** option is followed by the string **gcp-multiclouddefen-nprd-455223** (**not** the numerical project shown in the figure below).

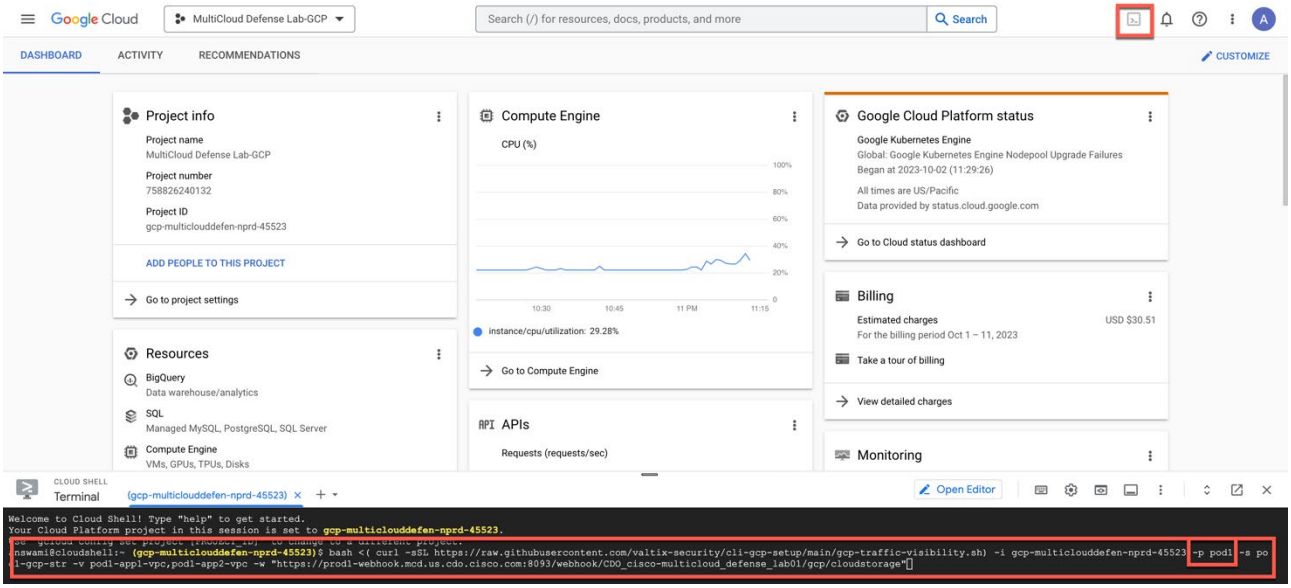


Figure: Running Google Cloud Shell script

5. When asked to authorize the cloud shell, click **AUTHORIZE**.

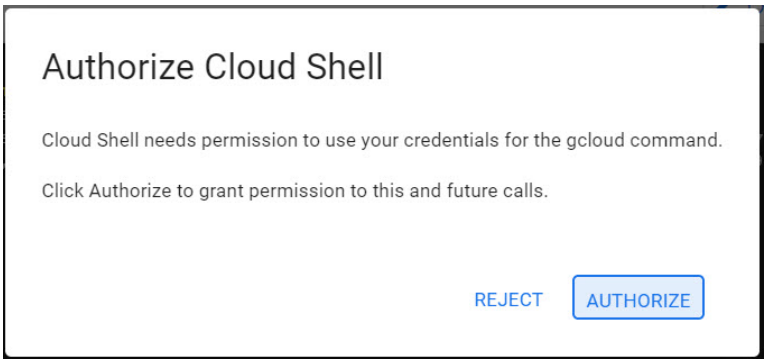


Figure: GCP visibility configuration complete

If you see the following error message, you need to delete visibility and re-enable it

ERROR: (gcloud.dns.policies.create) Resource in projects [gcp-multiclouddefen-nprd-45523] is the subject of a conflict: The resource 'entity.policy' named **'podx-dns-policy' already exists**

- Run the following command in the cloud shell **./delete-gcp-traffic-visibility.sh**
- Run the edited **bash command again** (use the up-arrow key to use the edited back command

Example: Original bash command

- bash <(curl -sSL https://raw.githubusercontent.com/valtix-security/cli-gcp-setup/main/gcp-traffic-visibility.sh) -i gcp-multiclouddefen-nprd-45523 -p **ciscomcd** -s pod1-gcp-str -v pod1-app1-vpc,pod1-app2-vpc -w "https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/gcp/cloudstorage"

Edited bash command (replace **ciscomcd** with **podx** – where x is your pod number)

- bash <(curl -sSL https://raw.githubusercontent.com/valtix-security/cli-gcp-setup/main/gcp-traffic-visibility.sh) -i gcp-multiclouddefen-nprd-45523 -p **pod1**-s pod1-gcp-str -v pod1-app1-vpc,pod1-app2-vpc -w https://prod1-webhook.mcd.us.cdo.cisco.com:8093/webhook/CDO_cisco-multicloud_defense_lab01/gcp/cloudstorage

6. click close on the pop window (don't run the bash command again)
7. Click next, you will see the Success page.

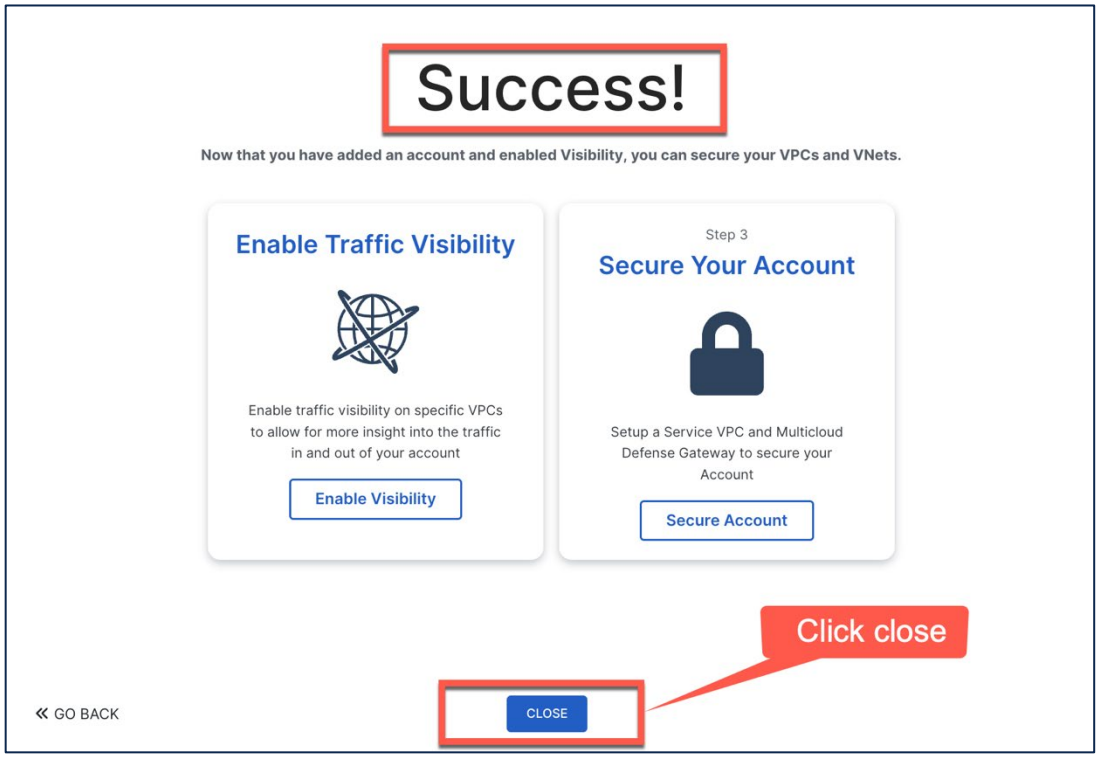


Figure: GCP visibility configuration complete

Task 2. Deploy the service VPC and gateways

1. In the Multicloud Defense Controller console, select the **Setup** tab. Click on **Secure Account**.

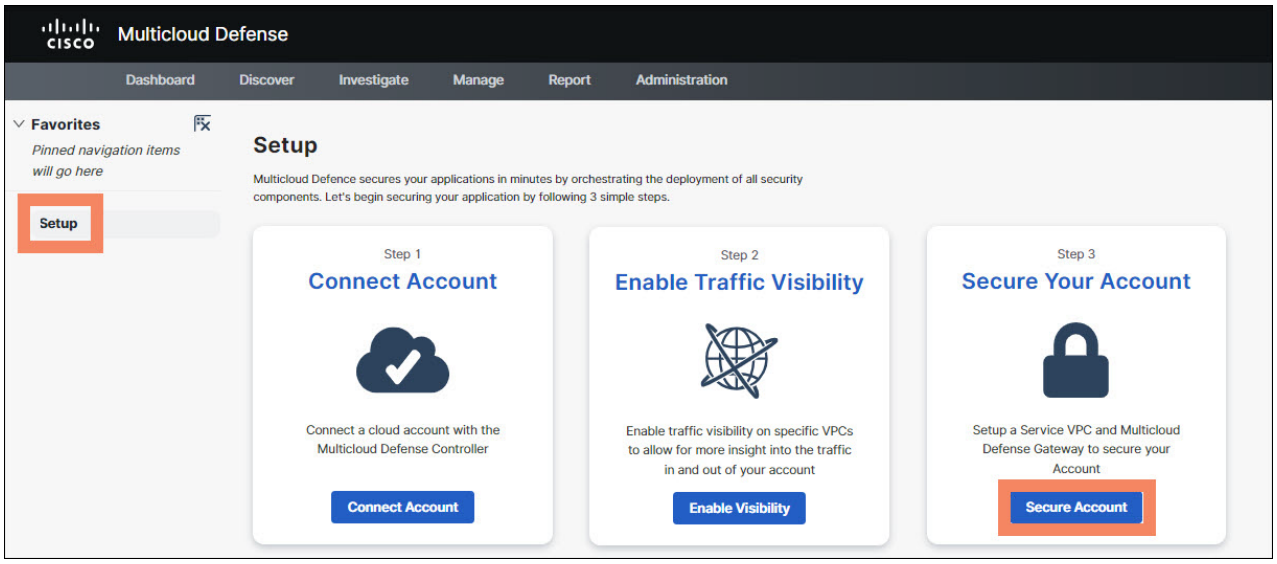


Figure: Secure Account

2. Note that **Centralized** is selected. Click **Next**.

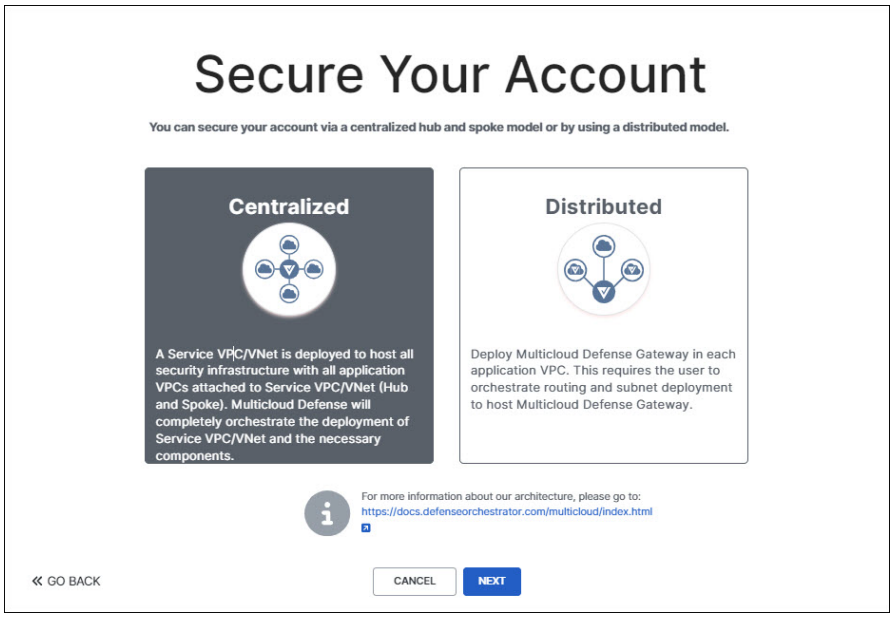


Figure: Select centralized security model

3. Configure the service VPC as follows.
    a. Name: **podx-svpc-gcp**, where x is your pod number
    b. CSP Account: **cisco-multicloud-defense-gcp01**
    c. Region: **Virginia us-east4**
    d. Datapath CIDR block: 192.168.x.0/24, where **x** is your pod number

e. Management CIDR block: 192.168.100+x.0/24 where x is your pod number **(example: if your pod number is 11, your management CIDR block is 192.168.111.0/24)**

f. Availability Zones: **us-east-4c**

In a production environment, you would want to specify more than one availability zone. This would result in gateways being deployed in each selected zone. However, for simplicity (and to reduce cost) you will only deploy in one availability zone.

g. Click **Next**.

Note: Use podx instead of using pod1-.

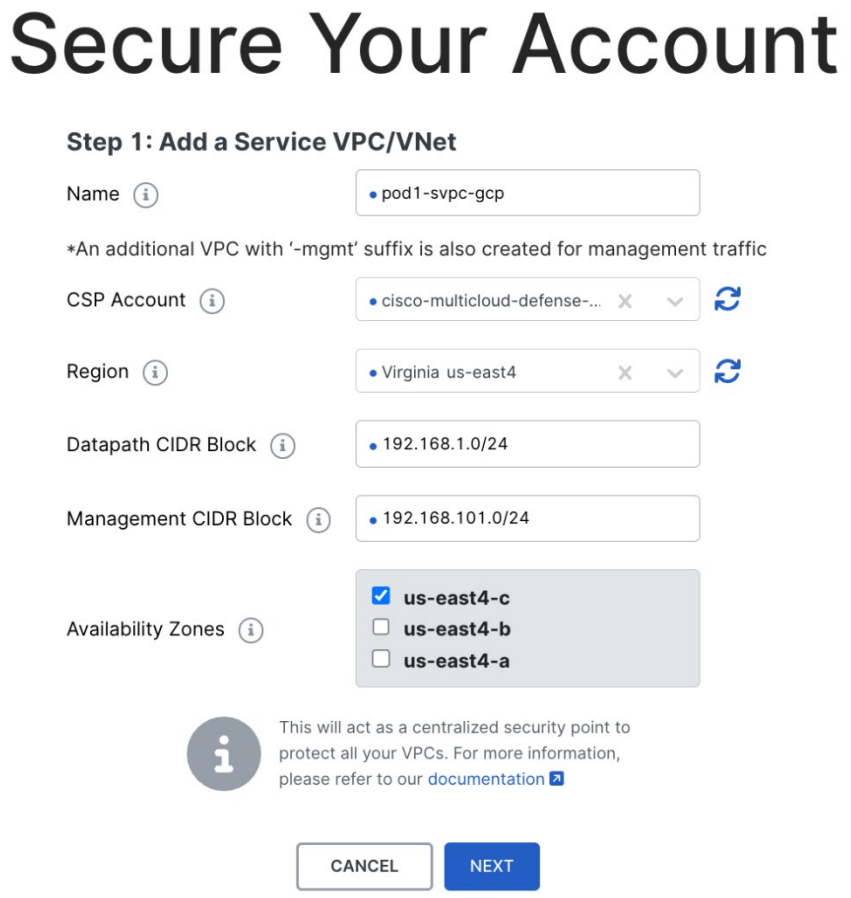
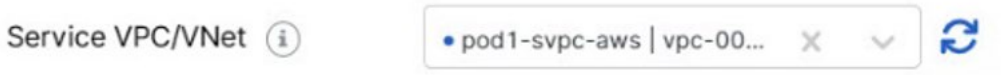


Figure: Secure Your Account

1. Deploy Gateways in service VPC
    i. podx-ingress-gw-gcp
    ii. podx-egress-gw-gcp
2. Account: cisco-multicloud-defense-gcp01

3. Service VPC/VNet: podx-svpc-gcp
4. Instance Type: 2 Virtual CPU
5. Gateway Service Gateway: ciscomcd-gateway@gcp-multiclouddefen-nprd-45523.iam.gserviceaccount.com

   Note: use this service account for all pods – sure there is no whitespace at the end of this link while copy-pasting)
6. MCD Gateways: Select Ingress and Egress Gateway
7. Ingress Gateway Name: podx-ingress-gw-gcp, where x is your pod number
8. Ingress Gateway Policy: podx-ingress-policy
9. Egress Gateway Name: podx-egress-gw-gcp, where x is your pod number
10. Egress Gateway Policy: podx-egress-policy
11. Click Next.
12. On the deploy gateway screen, click on “Service VPC/VNet and search for podx- and wait this service VPC/VNet to become active v/s active pending. If this doesn’t change within two minutes, click the refresh button



Note: Use podx instead of pod1-

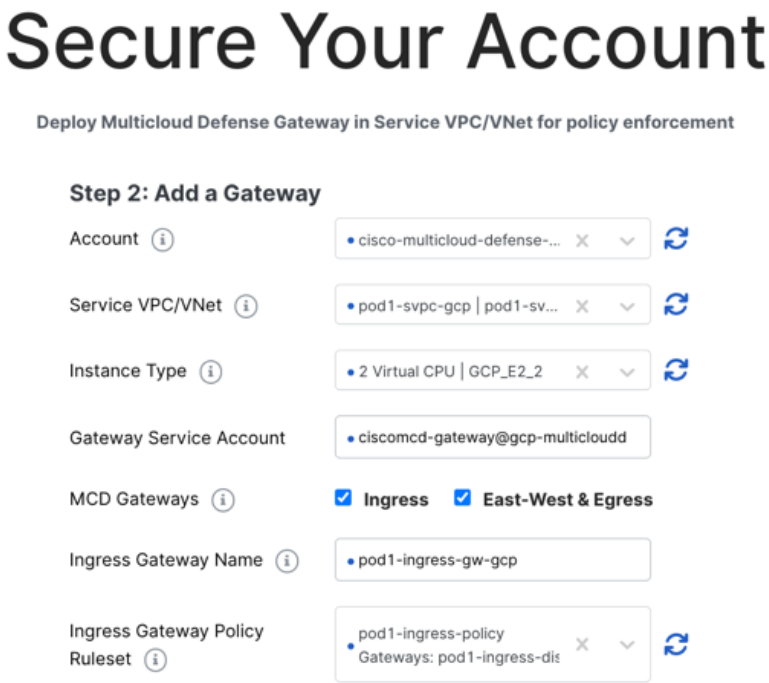


Figure: Create podx-ingress-gw-gcp and podx-ingress-gw-gcp gateways

13. In the Multicloud Defense Controller UI, navigate to **Manage > Gateways**. Confirm that two gateways are active. (deployment may take up to 10 minutes)

Task 3. Attach application VPCs to service VPC using VPC peering in GCP

1. In the Multicloud Defense Controller console, navigate to **Management > Inventory**.
2. Click on **VPCs/VNets**.
3. Filter the VPCs for **Region *EQUAL_TO* podx- (where x is your pod number)**
4. Find **podx-app1-vpc** (where **x** is your pod number). Look for VPC deployed in the account: **cisco-multicloud-defense-gcp01.**
5. Click **Secure Now**.
    a. Service VPC: **podx-svpc-gcp** (where **x** is your pod number)
    b. Select **"Send Traffic via Multicloud Defense Gateway"**
    c. Inspect the route table and the changes that the controller will make.
        i. The default route next hop will be changed from the Internet gateway to the service VPC peering (and therefore through the egress gateway). This will apply to egress and east-west (inter-VPC) traffic.
        ii. The static routes to 20.12.187.121/32 and 52.9.113.154/32 remain unchanged. These routes were created so traffic from the jump host (and back-up jump host) would bypass the gateway. This makes testing from the jump host easier.
        iii. The local route was created by GCP and applies to intra-VPC traffic.
    d. Click **Save**.

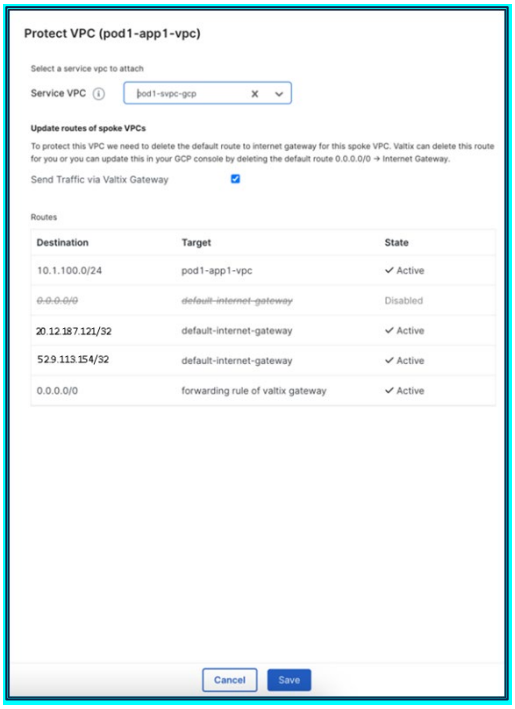


Figure: Protect VPC podx-app1-vpc

6. Repeat the previous step for **podx-app2-vpc** (where **x** is your pod number).

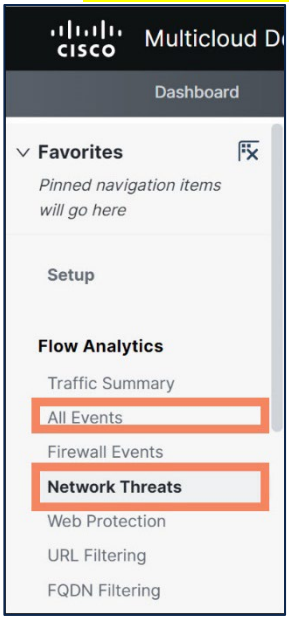
Task 4. Testing traffic filtering

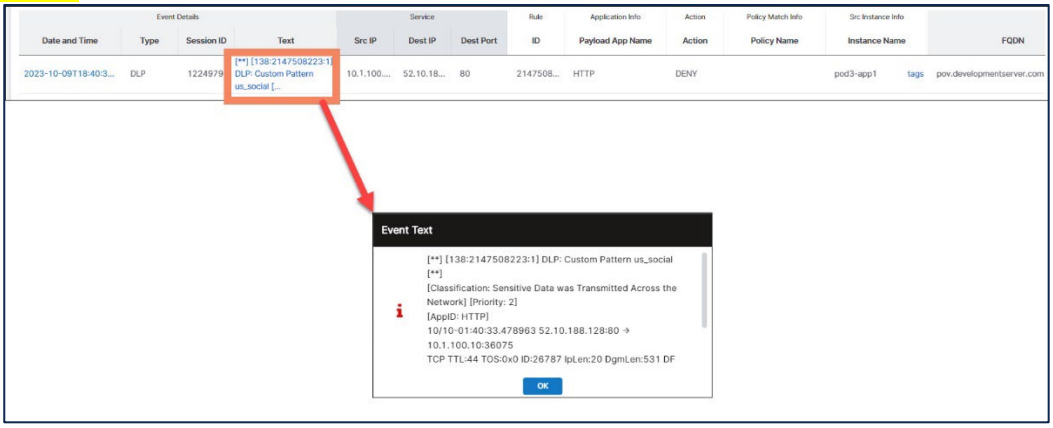
1. On the jump-host, start an SSH session to the GCP App1 Linux server.
    a. Open a new SSH session to the jump-host. Alternatively, you use the existing session by logging out of podx-app1-azure (where **x** is your pod number).
    b. From the jump-host, type:
    ssh -i gcp ubuntu@podx-app1-gcp
    Where **x** is your pod number.
2. Test east-west traffic filtering.
    a. From podx-app1, curl podx-app2:
    ubuntu@ip-10-x-100-10:~$ **curl 10.[100+x].100.10/status; echo**
    where **x** is your pod number. This command should succeed. The policy rule set allows TCP.
    b. From podx-app1, ping podx-app2:
    ubuntu@ip-10-x-100-10:~$ **ping 10.[100+x].100.10**
    where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
3. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All Events**.
    a. Filter by the account (gcp) and gateway (podx-egress-gw-gcp (where **x** is your pod number)).
    b. Confirm the east-west TCP traffic is being logged.
    c. Click on the **Date and Time** field to get event details.
4. Test egress traffic filtering and data loss prevention.
    a. From podx-app1, curl google.com:
    ubuntu@ip-10-x-100-10:~$ **curl google.com**
    where "x" is your pod number. This command should succeed. The policy rule set allows TCP.
    b. From podx-app1, ping google.com:
    ubuntu@ip-10-x-100-10:~$ **ping google.com**
    where "x" is your pod number. This command should fail. The policy rule set does **not** allow ICMP.
    c. Use curl to post some personal data:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like the cloud"
    The command should succeed.
    d. Use curl to post some personal data containing a valid SSN:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 555-55-5555"
    The command should fail.
    e. Try the following command:
    curl -X POST http://pov.developmentserver.com/cgi-bin/personal.cgi -d "personal_data=I like 123-45-6789"
    The command should succeed. The number 123-45-6789 is indeed **not** a valid

SSN.

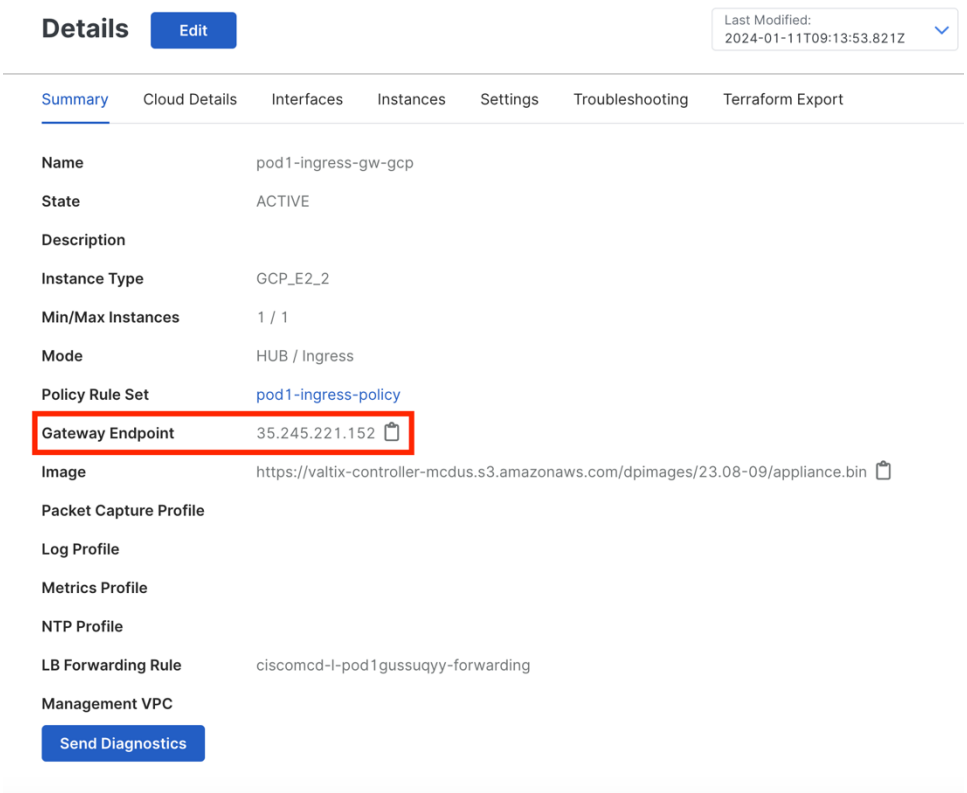
> Note. If you have a machine with a browser to do this test, navigate to http://pov.developmentserver.com/share.html

5. In the Multicloud Defense console look for the egress traffic. You should not have to change the filter, but you may have to click the refresh button in the UI (not the browser).
6. Toggle between All Events and Network Treats to focus on DLP. Click on event text to make is easier to read.





7. Click on the **Date and Time** field to get event details.
8. Test ingress forward proxy.
    a. In the Multicloud Defense console, navigate to **Manage > Gateways**.
    b. Click on **podx-ingress-gw-gcp**, where x is your pod number.
    c. Copy the gateway endpoint FQDN to your clip board.



9. Paste this gateway endpoint into a browser outside your pod. You should see the podx-app1-gcp webpage.
10. In the Multicloud Defense console, navigate to **Investigate > Flow Analysis > All Events**.
    a. Filter by the account (gcp) and gateway (podx-ingress-gw-gcp (where **x** is your pod number)).
    b. Confirm the ingress HTTP traffic is being logged. Since bots are always scanning port 80 on azure public IP space, you should see ingress traffic from unknown web sites as well as the traffic you generated.

Exercise7 – Distributed security model (optional)

Distributed security model

In this exercise, you will deploy the Cisco Multicloud Defense Ingress gateway (podx-ingress-dis-gw) in podx-app1-vpc-aws. This gateway has two interfaces:

- The Datapath interface is in ***podx-app1-subnet*** (same subnet as podx-app1 server)
- The management interface is in ***podx-mgmt-subnet***, this interface is used for connectivity between the gateway and Cisco Multicloud defense controller.

This podx1-ingress-dis-gw gateway is an ingress gateway that protects applications in the podx-app1-subnet.

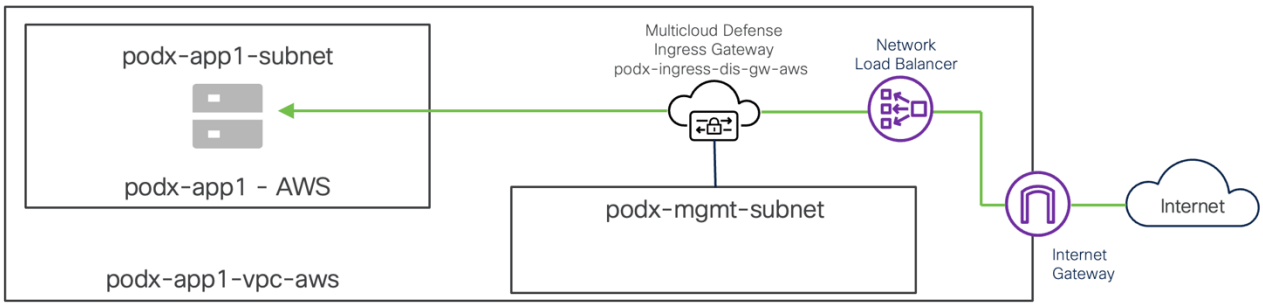


Figure: Distributed security model – AWS

Enable ingress protection

1. Since you are reusing a VPC for this exercise, you need to perform a step of cleanup. You would not need this step in a production environment.
    a. In a browser, navigate to https://aws.amazon.com/console/. Login as an IAM user to account **698990355236** with the username **podx** (where **x** is your pod number) and the same password you used for the jump-host.
    b. In the upper left corner of the screen, click on the **Services** and select **VPC**. In the left-hand navigation pane, select **Virtual private cloud > Route tables**.
    c. Enter **podx-** in the search field (where **x** is your pod number). Select the **podx-app1-rt** route table.
    d. Select **Edit Routes**.
    e. Change the **Target** of the **0.0.0.0/0** route to **Internet Gateway**. You will be asked to select an internet gateway, but there will only be one choice.
    f. Click **Save changes**.
2. In the Multicloud Defense Controller console, select the **Setup** tab. Click on **Secure Account**.
3. Select the **Distributed security model** (this exercise focuses on the distributed security model)
4. Now provide the following information

a. Select Account: **cisco-multicloud-defense-aws01**
b. Enter Name: **podx-ingress-dist-gw-aws** (where **x** is your pod number)
c. Select Instance Type: **2 Virtual CPU**
d. For autoscale, select **minimum 1** and **maximum 1**
e. Leave everything else on this page as the **default**
f. Click **Next**

5. Security: Select **Ingress**
6. For Gateway Image: **Leave the default values**
7. Select policy **podx-ingress-policy**
8. Select region: **us-east-1**
9. Select VPC: **podx-app1-vpc**
10. Specify keypair: **podx-keypair**
11. Select Gateway IAM: **ciscomcd-gateway-role**
    - Specify Mgmt. Security Group: **podx-app1-sg**
    - Specify Datapath Security Group: **podx-app1-sg**
12. Specify availability zone: **us-east-1a**
13. Specify Mgmt. Subnet name: **podx-mgmt-subnet**
14. Specify Datapath Subnet name: **podx-app1-subnet**
15. Click **Next**
16. Click Next again – **leave all the options as the default**
17. Click **Finish**
    Once you click on Finish, the controller deploys the ingress gateway in podx-app1-vpc, as part of this deployment the controller also deploys the AWS Network Load Balancer as a frontend device.
    Note: This deployment could take several minutes.
18. Go to **Manage** > **Gateways** and search for **podx-ingress-dis-gw**. If necessary, wait for the status of the gateway to become **Active**.
    a. Go to Multicloud defense Gateway click on **podx-ingress-dist-gw-aws**
    b. Copy gateway endpoint (FQDN) and open it in a browser. The FQDN will take you to podx-app1-aws application.
19. Go to Investigate All Events and check logs for inbound traffic.

Note: Since this is a lab setup, we have protected podx-app1-vpc using centralized VPC and a distributed model. In the real-world deployment, customer uses one of these models to protect their VPCs.

Conclusion

It is a multicloud world we live in, and organizations need a cloud-agnostic solution that unifies security controls across all environments while securing workloads at cloud speed and scale. With Cisco Multicloud Defense, organizations can leverage a simplified and unified security experience helping them navigate their multicloud future with confidence.

For more information on Cisco Multicloud Defends refer to cisco.com/go/multicloud-defense

Call-to-action

- Sign up for Cisco Defense Orchestrator free-tier
- Access Cisco Multicloud Defense from CDO tenant
- Onboard Accounts (AWS, Azure, and GCP)
- Enable visibility
- Create security policy
- Deploy Gateways in your account and enable enforcement

Resources

- Cisco Multicloud Defense page
- White Paper
- Technical Blog
- At-a-glance
- Ordering Guide

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at https://www.cisco.com/go/offices.