

# Clustering: Public Cloud

Clustering lets you group multiple Firewall Threat Defense Virtuals together as a single logical device. A cluster provides all the convenience of a single device (management, integration into a network) while achieving the increased throughput and redundancy of multiple devices. You can deploy Firewall Threat Defense Virtual clusters in a public cloud using the following public cloud platforms:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)

Currently, only routed firewall mode is supported.

**Note** Some features are not supported when using clustering. See Unsupported Features and Clustering.


- About Threat Defense Virtual clustering in the public cloud, on page 2
- Licenses for Threat Defense Virtual clustering, on page 4
- Requirements and prerequisites for Threat Defense Virtual clustering, on page 5
- Guidelines for Threat Defense Virtual clustering, on page 7
- Deploy the cluster in AWS, on page 8
- Deploy the cluster in Azure, on page 39
- Firewall Threat Defense Virtual clustering autoscale solution in Azure, on page 59
- Deploy the cluster in GCP, on page 84
- Threat Defense Virtual clustering with autoscale solution in GCP, on page 92
- Add the cluster to the Management Center (manual deployment), on page 106
- Edit cluster health monitor settings, on page 113
- Manage cluster nodes, on page 116
- Monitor cluster status, on page 119
- Troubleshoot the cluster, on page 125
- Upgrade the cluster, on page 127
- Reference for clustering, on page 129
- History for Threat Defense Virtual clustering in the public cloud, on page 141

# About Threat Defense Virtual clustering in the public cloud

This section describes the clustering architecture and how it works.

## How the cluster fits into your network

The cluster consists of multiple firewalls acting as a single device. To act as a cluster, the firewalls need the following infrastructure:

- Isolated network for intra-cluster communication, known as the cluster control link, using VXLAN interfaces. VXLANs, which act as Layer 2 virtual networks over Layer 3 physical networks, let the Firewall Threat Defense Virtual send broadcast or multicast messages over the cluster control link.
- Load Balancer(s)—For external load balancing, you have the following options depending on your public cloud:
  - AWS Gateway Load Balancer

    The AWS Gateway Load Balancer combines a transparent network gateway and a load balancer that distributes traffic and scales virtual appliances on demand. The Firewall Threat Defense Virtual supports the Gateway Load Balancer centralized control plane with a distributed data plane (Gateway Load Balancer endpoint) using a Geneve interface single-arm proxy.
  - Azure Gateway Load Balancer

    In an Azure service chain, Firewall Threat Defense Virtuals act as a transparent gateway that can intercept packets between the internet and the customer service. The Firewall Threat Defense Virtual defines an external interface and an internal interface on a single NIC by utilizing VXLAN segments in a paired proxy.
  - Native GCP load balancers, internal and external
  - Equal-Cost Multi-Path Routing (ECMP) using inside and outside routers such as Cisco Cloud Services Router

    ECMP routing can forward packets over multiple “best paths” that tie for top place in the routing metric. Like EtherChannel, a hash of source and destination IP addresses or source and destination ports, or both, can be used to send a packet to one of the next hops. If you use static routes for ECMP routing, then the Firewall Threat Defense failure can cause problems; the route continues to be used, and traffic to the failed Firewall Threat Defense will be lost. If you use static routes, be sure to use a static route monitoring feature such as Object Tracking. We recommend using dynamic routing protocols to add and remove routes, in which case, you must configure each Firewall Threat Defense to participate in dynamic routing.

**Note** Layer 2 Spanned EtherChannels are not supported for load balancing.

## Individual interfaces

You can configure cluster interfaces as individual interfaces.

Individual interfaces are normal routed interfaces, each with their own local IP address. The IP address for the interface will be configured automatically via DHCP. Static IP configuration is not supported.

# Control and data node roles

When you add a cluster in the Firewall Management Center , you choose one firewall to be the control node, and all additional firewalls to be data nodes. When you first create the cluster, the control node you specify will become the control node simply because it is the first node added to the cluster. Later, if multiple cluster nodes come online at the same time, the control node is determined by the priority setting; the priority is set between 1 and 100, where 1 is the highest priority.

When you create the cluster and deploy the cluster configuration, the Firewall Management Center deploys a special bootstrap configuration to each node that includes fundamental settings such as the cluster control link configuration. The bootstrap configuration lets each node join the cluster. Most bootstrap settings are defined in the cluster wizard; however, you define the cluster control link interface hardware configuration (for example, creating an EtherChannel or setting the Ethernet speed) on the control node before you create the cluster. The cluster control link settings are copied to the bootstrap configuration for each node.

All nodes in the cluster share the same policy configuration. The node that you initially specify as the control node will overwrite the configuration on the data nodes when they join the cluster, so you only need to perform initial configuration on the control node before you form the cluster. After you create the cluster, all configuration changes are made at the cluster level and are shared by all nodes.

Some features do not scale in a cluster, and the control node handles all traffic for those features.

# Cluster Control Link

Each node must dedicate one interface as a VXLAN (VTEP) interface for the cluster control link. For more information about VXLAN, see Configure VXLAN interfaces .

### VXLAN Tunnel Endpoint

VXLAN tunnel endpoint (VTEP) devices perform VXLAN encapsulation and decapsulation. Each VTEP has two interface types: one or more virtual interfaces called VXLAN Network Identifier (VNI) interfaces, and a regular interface called the VTEP source interface that tunnels the VNI interfaces between VTEPs. The VTEP source interface is attached to the transport IP network for VTEP-to-VTEP communication.

### VTEP Source Interface

The VTEP source interface is a regular Firewall Threat Defense Virtual interface with which you plan to associate the VNI interface. You can configure one VTEP source interface to act as the cluster control link. The source interface is reserved for cluster control link use only. Each VTEP source interface has an IP address on the same subnet. This subnet should be isolated from all other traffic, and should include only the cluster control link interfaces.

### VNI Interface

A VNI interface is similar to a VLAN interface. It is a virtual interface that keeps network traffic separated on a given physical interface by using tagging. You can only configure one VNI interface. Each VNI interface has an IP address on the same subnet.

#### Peer VTEPs

Unlike regular VXLAN for data interfaces, which allows a single VTEP peer, Firewall Threat Defense Virtual clustering allows you to configure multiple peers.

### Cluster control link traffic overview

Cluster control link traffic includes both control and data traffic.

Control traffic includes:

- Control node election.
- Configuration replication.
- Health monitoring.

Data traffic includes:

- State replication.
- Connection ownership queries and data packet forwarding.

## Configuration replication

All nodes in the cluster share a single configuration. You can only make configuration changes on the control node (with the exception of the bootstrap configuration), and changes are automatically synced to all other nodes in the cluster.

## Management network

You must manage each node using the management interface. Management from a data interface is not supported with clustering.

# Licenses for Threat Defense Virtual clustering

You assign feature licenses to the cluster as a whole, not to individual nodes. However, each node of the cluster consumes a separate license for each feature. The clustering feature itself does not require any licenses.

**Note** APEX, PLUS, and VPN-Only licenses are not supported for clustering. Before you create a cluster, remove unsupported license assignments from the devices that will become cluster nodes.

#### Performance tier license requirements

Each Firewall Threat Defense Virtual cluster node requires the same performance tier license. We recommend using the same number of CPUs and memory for all members, or else performance will be limited on all nodes to match the least capable member. The throughput level will be replicated from the control node to each data node so they match.

When you add the control node to the Firewall Management Center, you can specify the feature licenses you want to use for the cluster. You can modify licenses for the cluster in the **Devices** > **Device Management**, **Cluster** > **License** area.

**Note** If you add the cluster before the Firewall Management Center is licensed (and running in Evaluation mode), then when you license the Firewall Management Center, you can experience traffic disruption when you deploy policy changes to the cluster. Changing to licensed mode causes all data units to leave the cluster and then rejoin.

# Requirements and prerequisites for Threat Defense Virtual clustering

This reference provides the requirements and prerequisites for Threat Defense Virtual clustering, including model specifications, user roles, hardware and software requirements, and MTU configuration.

### Model requirements

- FTDv5, FTDv10, FTDv20, FTDv30, FTDv50, FTDv100

**Note** FTDv5 and FTDv10 do not support Amazon Web Services (AWS) Gateway Load Balancer (GWLB) and Azure GWLB.

- Supported public cloud services:
  - Amazon Web Services (AWS)
  - Microsoft Azure
  - Google Cloud Platform (GCP)
- Maximum of 16 nodes

See also the general requirements for the Firewall Threat Defense Virtual in the Secure Firewall Threat Defense Virtual getting started guides.

### User roles

- Administrator
- Access Admin
- Network Admin

### Hardware and software requirements

All units in a cluster:

- Must be in the same performance tier. We recommend using the same number of CPUs and memory for all nodes, or else performance will be limited on all nodes to match the least capable node.
- Must run the identical software except at the time of an image upgrade. Hitless upgrade is supported.
- The Firewall Management Center access must be from the Management interface. Data interface management is not supported.
- Multiple Availability Zone deployment is supported from Firewall Threat Defense Virtual version 7.6.0 and later. The earlier versions supports only single Availability Zone deployment.
- Cluster requires a minimum of *four* interfaces: Management, Diagnostics, Data (Geneve), and CCL (Cluster Control Link).
- Cluster control link interfaces of all units must be in the same subnet.

### MTU

Make sure the ports connected to the cluster control link have the correct (higher) MTU configured. If there is an MTU mismatch, the cluster formation will fail. When a node joins the cluster, it checks MTU compatibility by sending a ping to the control node with a packet size matching the cluster control link MTU. If the initial ping fails, the node tries a ping using a smaller packet size (the MTU divided by 2, then by 4, then by 8) until a ping succeeds. A notification is generated so you can fix the MTU mismatch on connecting switches and try again.

The cluster control link MTU should be 154 bytes higher than the data interfaces. Because the cluster control link traffic includes data packet forwarding, the cluster control link needs to accommodate the entire size of a data packet plus cluster traffic overhead (100 bytes) plus VXLAN overhead (54 bytes).

For AWS with GWLB, the data interface uses Geneve encapsulation. In this case, the entire Ethernet datagram is being encapsulated, so the new packet is larger and requires a larger MTU. You should set the source interface MTU to be the network MTU + 326 bytes. So for the standard 1500 MTU network path, the source interface MTU should be 1826, and the cluster control link MTU should be +154, 1980.

For Azure with GWLB, the data interface uses VXLAN encapsulation. In this case, the entire Ethernet datagram is being encapsulated, so the new packet is larger and requires a larger MTU. You should set the cluster control link MTU to be the source interface MTU + 80 bytes.

The table shows the default values for the cluster control link MTU and the data interface MTU.



**Note**

- We do not recommend setting the cluster control link MTU between 2561 and 8362; due to block pool handling, this MTU size is not optimal for system operation.
- For Threat Defense Virtual clusters on Azure, modifying the cluster control link (CCL) MTU via FlexConfig in Management Center is not recommended, as it may cause cluster stability issues.

***Table 1: Default MTU***

| Public Cloud | Cluster Control Link MTU | Data Interface MTU |
|---|---|---|
| AWS with GWLB | 1980 | 1826 |

| Public Cloud | Cluster Control Link MTU | Data Interface MTU |
|---|---|---|
| AWS | 1654 | 1500 |
| Azure with GWLB | 1454 | 1374 |
| Azure | 1454 | 1300 |
| GCP | 1554 | 1400 |

# Guidelines for Threat Defense Virtual clustering

## High availability

High Availability is not supported with clustering.

## IPv6

The cluster control link is only supported using IPv4.

## Multi-zone clustering

Multi-zone clustering is supported for up to three zones.

## Additional guidelines

- When significant topology changes occur (such as adding or removing an EtherChannel interface, enabling or disabling an interface on the Firewall Threat Defense or the switch, adding an additional switch to form a VSS or vPC or VNet) you should disable the health check feature and also disable interface monitoring for the disabled interfaces. When the topology change is complete, and the configuration change is synced to all units, you can re-enable the interface health check feature.
- When adding a node to an existing cluster, or when reloading a node, there will be a temporary, limited packet/connection drop; this is expected behavior. In some cases, the dropped packets can hang your connection; for example, dropping a FIN/ACK packet for an FTP connection will make the FTP client hang. In this case, you need to reestablish the FTP connection.
- Do not power off a node without first disabling clustering on the node.
- For decrypted TLS/SSL connections, the decryption states are not synchronized, and if the connection owner fails, then decrypted connections will be reset. New connections will need to be established to a new node. Connections that are not decrypted (they match a do-not-decrypt rule) are not affected and are replicated correctly.
- Dynamic scaling is supported from Secure Firewall version 7.3.
- If you are using Secure Firewall versions 7.2 or 7.3, Stateful Target Failover is not supported when you deploy the cluster on AWS.
- Perform a global deployment after the completion of each maintenance window.
- Ensure that you do not remove more than one device at a time from the auto scale group (AWS) / instance group (GCP) / scale set (Azure). We also recommend that you run the **cluster disable** command on the

device before removing the device from the auto scale group (AWS) / instance group (GCP) / scale set (Azure).

- If you want to disable data nodes and the control node in a cluster, we recommend that you disable the data nodes before disabling the control node. If a control node is disabled while there are other data nodes in the cluster, one of the data nodes has to be promoted to be the control node. Note that the role change could disturb the cluster.
- In the customized day 0 configuration scripts given in this guide, you can change the IP addresses as per your requirement, provide custom interface names, and change the sequence of the CCL-Link interface.
- If you experience CCL instability issues, such as intermittent ping failures, after deploying a Threat Defense Virtual cluster on a cloud platform, we recommend that you address the reasons that are causing CCL instability. Also, you can increase the hold time as a temporary workaround to mitigate CCL instability issues to a certain extent. For more information on how to change the hold time, see Edit Cluster Health Monitor Settings.
- When you are configuring your security firewall rule or security group for the Management Center virtual, you must include both Private and Public IP addresses of the Firewall Threat Defense Virtual in the Source IP address range. Also, ensure to specify the Private and Public IP addresses of the Firewall Management Center Virtual in the security firewall rule or security group of the Firewall Threat Defense Virtual. This is important to ensure proper registration of nodes during clustering deployment.

### Defaults for clustering

- The cLACP system ID is auto-generated, and the system priority is 1 by default.
- The cluster health check feature is enabled by default with the holdtime of 3 seconds. Interface health monitoring is enabled on all interfaces by default.
- The cluster auto-rejoin feature for a failed cluster control link is unlimited attempts every 5 minutes.
- The cluster auto-rejoin feature for a failed data interface is 3 attempts every 5 minutes, with the increasing interval set to 2.
- Connection replication delay of 5 seconds is enabled by default for HTTP traffic.

# Deploy the cluster in AWS

You can deploy a cluster in AWS manually or by using CloudFormation templates to deploy a stack. You can use the cluster with AWS Gateway Load Balancer, or with a non-native load balancer such as the Cisco Cloud Services Router.

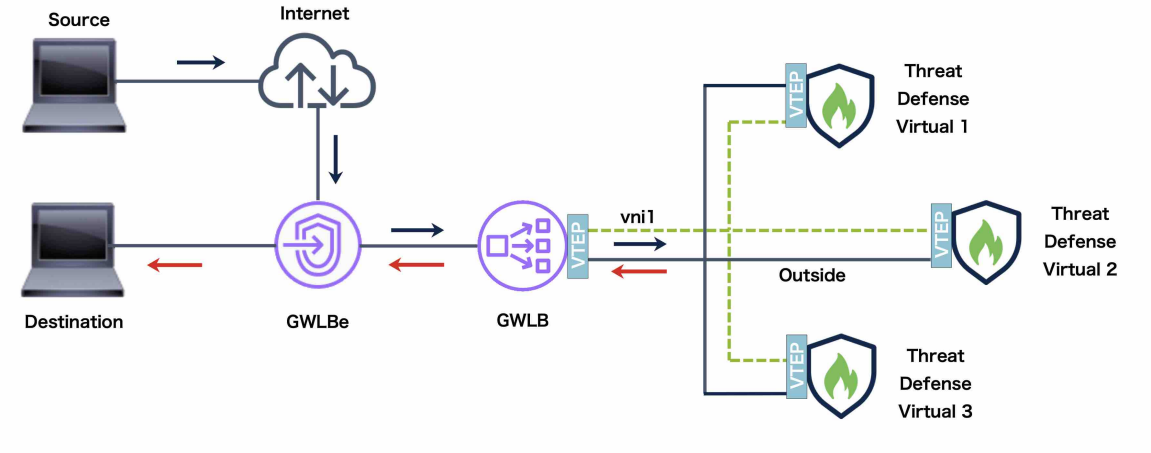
From version 10.0.0, the AWS Geneve clustering solution supports both single-arm and dual-arm deployment modes. This provides greater flexibility in network architecture.

## AWS gateway load balancer and Geneve single-arm proxy

This use case is the only currently supported use case for Geneve interfaces. The AWS Gateway Load Balancer combines a transparent network gateway and a load balancer that distributes traffic and scales virtual appliances on demand. The Threat Defense Virtual supports the Gateway Load Balancer centralized control plane with a distributed data plane (Gateway Load Balancer endpoint). The following figure shows traffic forwarded to

the Gateway Load Balancer from the Gateway Load Balancer endpoint. The Gateway Load Balancer balances traffic among multiple Threat Defense Virtuals, which inspect the traffic before either dropping it or sending it back to the Gateway Load Balancer (U-turn traffic). The Gateway Load Balancer then sends the traffic back to the Gateway Load Balancer endpoint and to the destination.

***Figure 1: Geneve Single-Arm Proxy***




**Note** Transport Layer Security (TLS) Server Identity Discovery is not supported with Geneve single-arm setup on AWS.

# Sample topologies

Sample topologies are given here for the following scenarios:

- Firewall Threat Defense Virtual clustering with autoscale in single and multiple availability zones of an AWS Region
- Firewall Threat Defense Virtual clustering with autoscale in single availability zone
- Firewall Threat Defense Virtual clustering with autoscale in multiple availability zones

### Firewall Threat Defense Virtual clustering with autoscale in single and multiple availability zones of an AWS region

An availability zone is a standalone data center or a set of independent data centers within an AWS region that operate independently. Each zone has its own networking infrastructure, connectivity, and power source, ensuring that a failure in one zone does not affect others. To improve redundancy and reliability, companies use multiple availability zones in their disaster recovery plans.

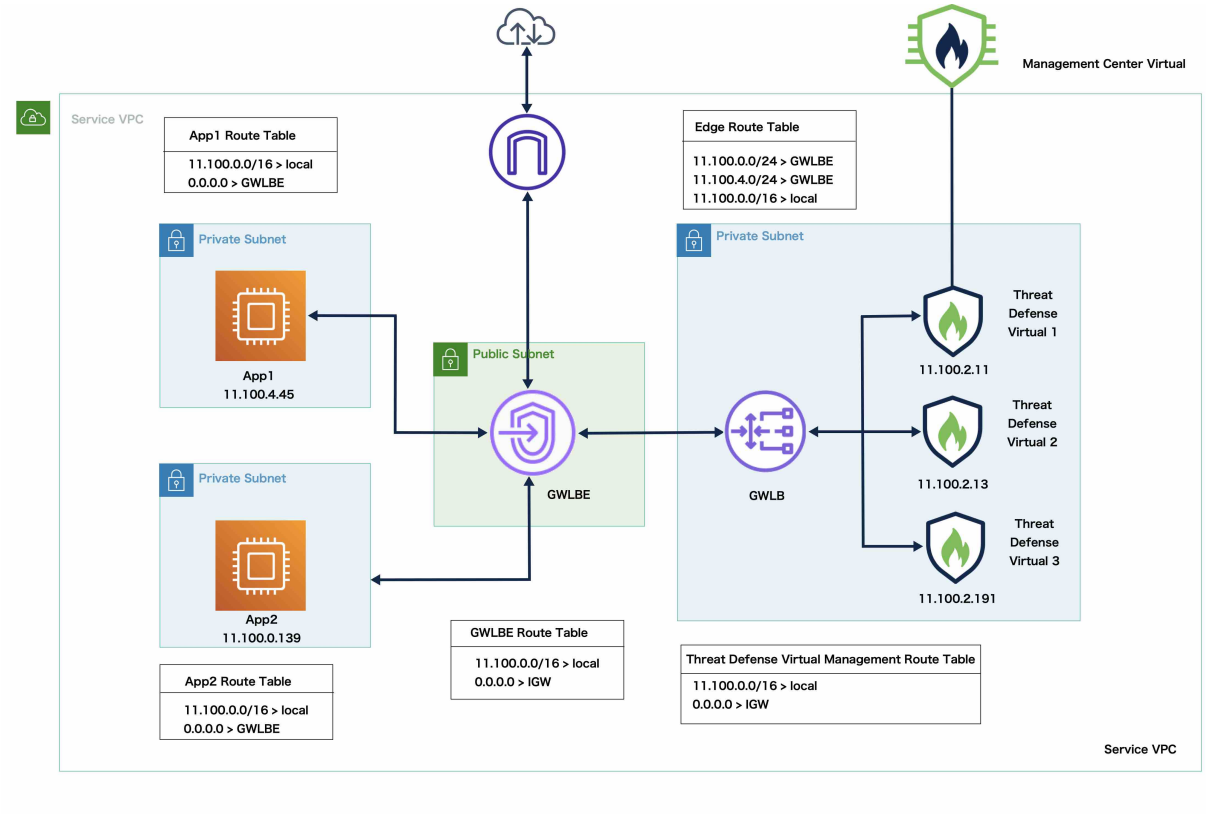
Deploying Firewall Threat Defense Virtual across multiple availability zones and configuring clustering with dynamic scaling can significantly enhance the availability and scalability of your infrastructure. Utilizing multiple availability zones in the same region can offer extra redundancy and guarantee high availability in the event of a failure.

You can modify the IP allocation mechanism of Cluster Control Link (CCL) to support both single and multiple availability zone deployments of Firewall Threat Defense Virtual clusters on AWS. The topologies depicted

here show both inbound and outbound traffic flow in single and multiple availability zones with autoscaling ability.

**Firewall Threat Defense Virtual clustering with autoscale in single availability zone**

There are two Firewall Threat Defense Virtual instances in the cluster that are connected to a GWLB.

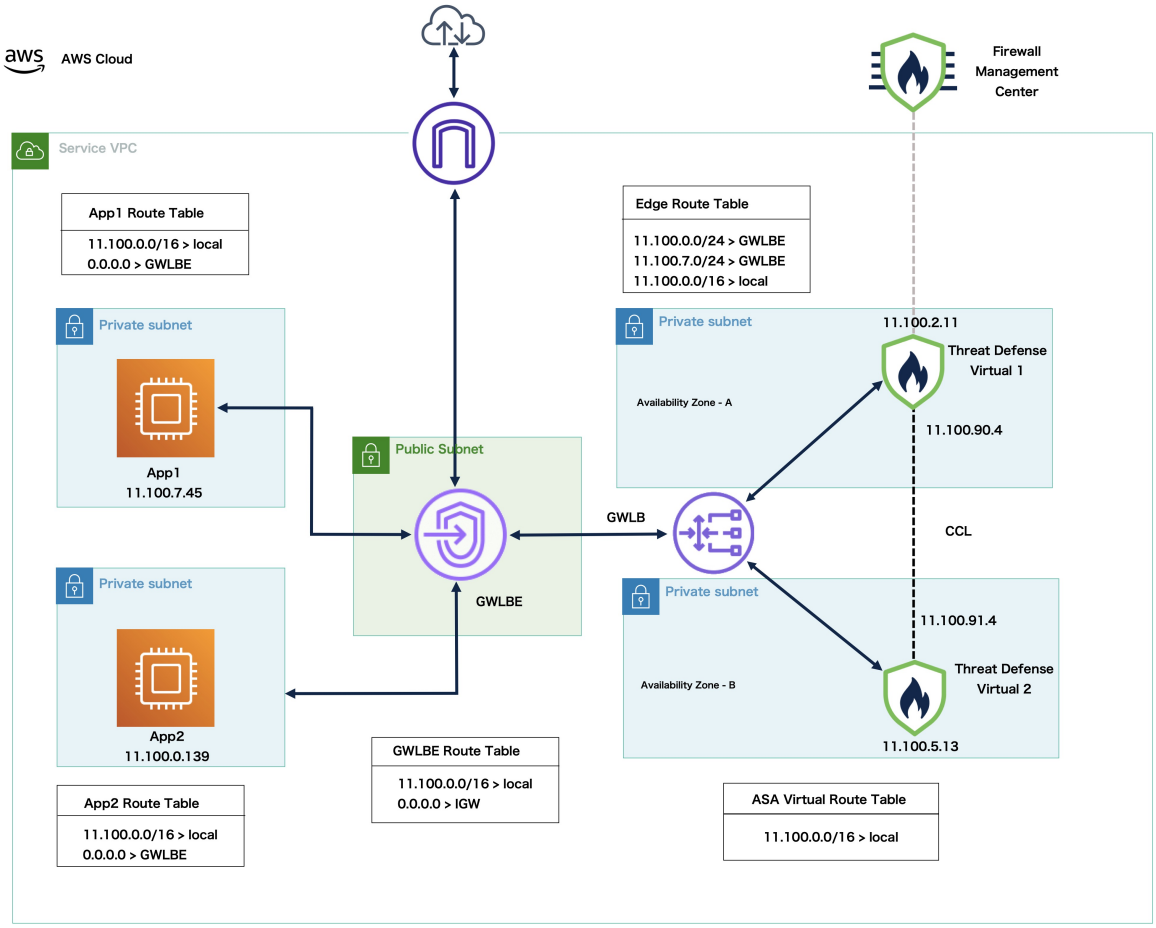



Inbound traffic from the internet goes to the GWLB endpoint, which is then transmits the traffic to the GWLB. Traffic is then forwarded to the Firewall Threat Defense Virtual cluster. After the traffic is inspected by an Firewall Threat Defense Virtual instance in the cluster, it is forwarded to the application VM, App1.

Outbound traffic from App1 is transmitted to the GWLB endpoint > GWLB > TDv > GWLB > GWLB Endpoint, which then sends it out to the internet.

**Firewall Threat Defense Virtual clustering with autoscale in multiple availability zones**

There are two Firewall Threat Defense Virtual instances in the cluster in different availability zones that are connected to a GWLB.



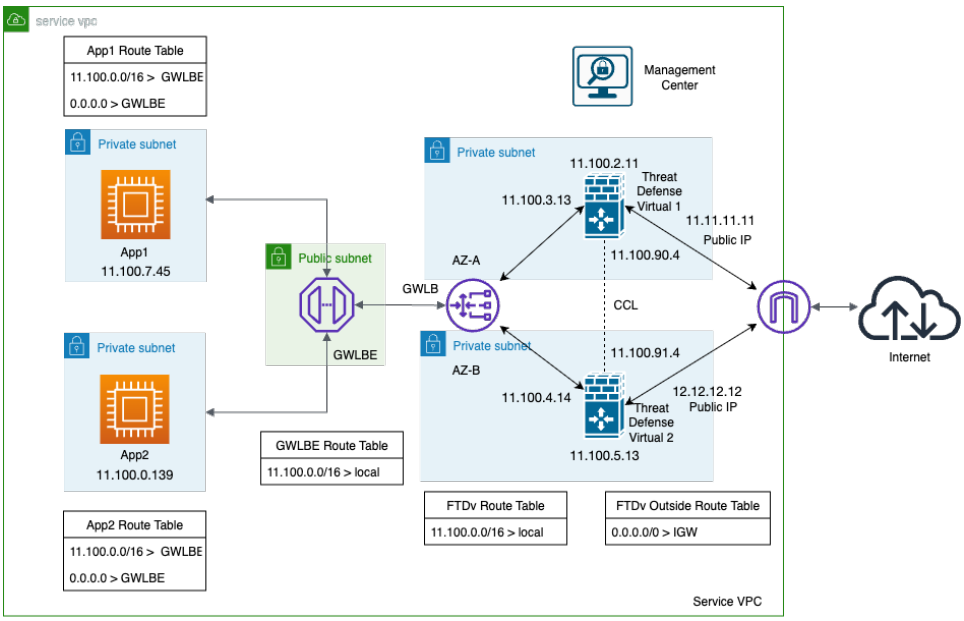

**Note** Multiple Availability Zone deployment is supported from Firewall Threat Defense Virtual Version 7.6.0 and later.

Inbound traffic from the internet goes to the GWLB endpoint, which then transmits the traffic to the GWLB. Based on the availability zone, the traffic is then routed to the Firewall Threat Defense Virtual cluster. After the traffic is inspected by an Firewall Threat Defense Virtual instance in the cluster, it is forwarded to the application VM, App1.

# AWS gateway load balancer and Geneve dual-arm proxy

A dual-arm proxy is a network deployment mode that enables the Threat defense Virtual to inspect traffic, applies Network Address Translation (NAT), and sends it directly from its outside interface to the Internet via the Internet Gateway. This direct egress path bypasses the GWLB and its endpoint on egress, streamlining traffic flow for greater efficiency. This approach is particularly effective in multi-VPC environments, where outbound traffic from multiple VPCs can share a single common egress point. This reduces infrastructure requirements, thus making the solution more cost-effective. Additionally, it supports clustering for more efficient traffic handling.

The purpose of this deployment is to inspect application traffic before it is sent to the Internet, using clustered Threat Defense Virtual instances deployed across separate Availability Zones in a dual-arm design, where the inside interface handles ingress traffic and the outside interface handles egress traffic. In this flow, traffic from the applications is routed to the GWLB, which forwards it to the inside interface of the Threat Defense Virtual for inspection. After applying NAT, the Threat Defense Virtual sends the traffic out through its outside interface directly to the Internet Gateway.

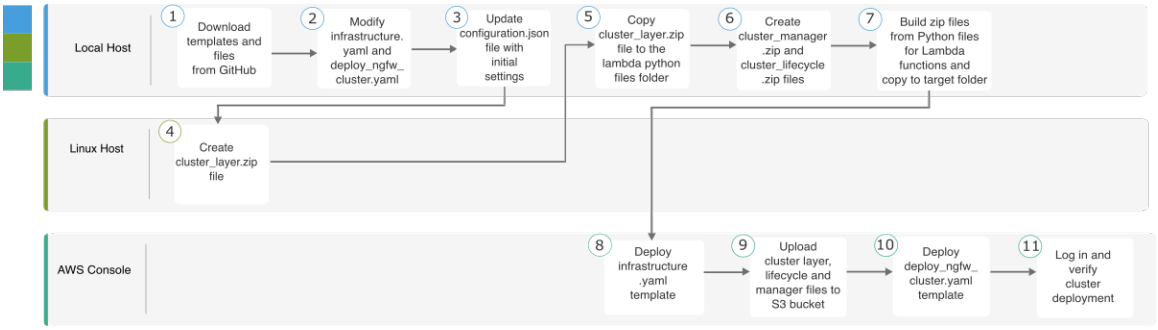



# End-to-end process for deploying Threat Defense Virtual cluster on AWS

Provides a workflow to help users successfully deploy and verify a Threat Defense Virtual cluster on AWS using either template-based or manual deployment processes.

### Template-based Deployment

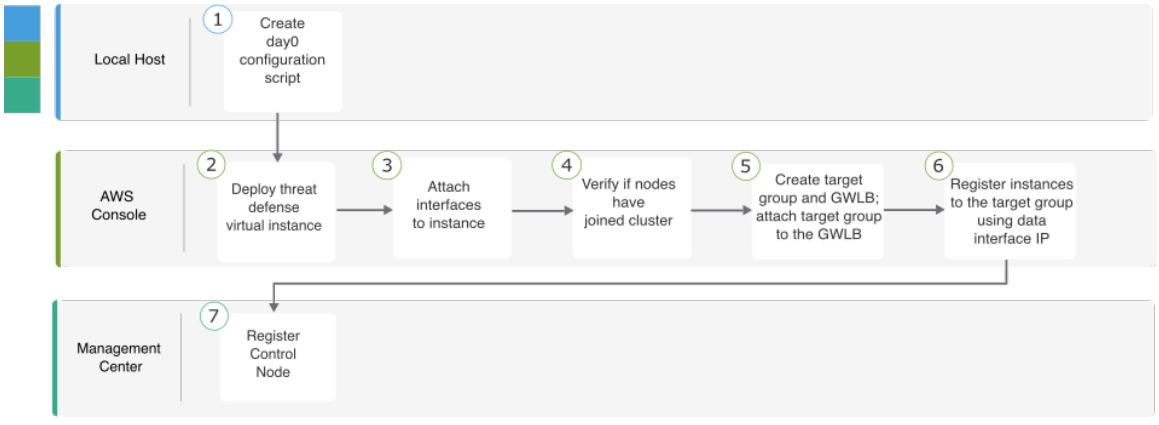
The following flowchart illustrates the workflow for template-based deployment of the Threat Defense Virtual cluster on AWS.



| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Clone the repository from GitHub |
| 2 | Local Host | Modify *infrastructure.yaml* and *deploy_ngfw_cluster.yaml* templates. |
| 3 | Local Host | Update the *Configuration.json* file with FMC object names. |
| 4 | Linux Host | Create *cluster_layer.zip* file. |
| 5 | Local Host | Copy *cluster_layer.zip* file to the Lambda python files folder. |
| 6 | Local Host | Create *cluster_manager.zip*, *custom_metrics_publisher.zip*, and *cluster_lifecycle.zip* files. |
| 7 | Local Host | Build zip files from Python files for Lambda functions and copy to target folder. |
| 8 | AWS Console | Deploy *infrastructure.yaml* template. |
| 9 | AWS Console | Upload *cluster_layer.zip*, *cluster_lifecycle.zip*, *custom_metrics_publisher.zip*, and *cluster_manager.zip* to the S3 bucket. |
| 10 | AWS Console | Deploy *deploy_ngfw_cluster.yaml* template. |
| 11 | AWS Console | Log in and verify cluster deployment. |

### Manual Deployment

The following flowchart illustrates the workflow for manual deployment of the Threat Defense Virtual cluster on AWS.



| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Create day 0 configuration script. |
| 2 | AWS Console | Deploy Threat Defense Virtual instance. |
| 3 | AWS Console | Attach interfaces to instance. |
| 4 | AWS Console | Verify if nodes have joined cluster. |
| 5 | AWS Console | Create target group and GWLB; attach target group to the GWLB. |
| 6 | AWS Console | Register instances with the target group using data interface IP. |
| 7 | Management Center | Register control node. |

# Templates

The templates are available in GitHub. The parameter values are self-explanatory with the parameter names, default values, allowed values, and description, given in the template.

- infrastructure.yaml – Template for infrastructure deployment.
- deploy_ngfw_cluster.yaml – Template for cluster deployment.

**Note** Ensure that you check the list of supported AWS instance types before deploying cluster nodes. This list is found in the *deploy_ngfw_cluster.yaml* template, under allowed values for the parameter InstanceType.

# Deploy the Stack in AWS Using a CloudFormation template

Deploy the stack in AWS using the customized CloudFormation template.

**Before you begin**

- You need an Amazon Linux virtual machine with Python 3.
- To allow the cluster to auto-register with the Firewall Management Center , you need to create two users with administrative privileges on the Firewall Management Center that can use the REST API. See the *Cisco Secure Firewall Management Center Administration Guide* .
- Add an access policy in the Firewall Management Center that matches the name of the policy that you specified in `Configuration.json` .

### Procedure

**Step 1** Prepare the template.

a) Clone the GitHub repository to your local folder. See https://github.com/CiscoDevNet/cisco-ftdv/tree/master/cluster/aws .

b) Modify **infrastructure.yaml** and **deploy_ngfw_cluster.yaml** with the required parameters.

c) Modify **cluster/aws/lambda-python-files/Configuration.json** with initial settings.

For example:

```
{
  "licenseCaps": ["BASE", "MALWARE", "THREAT"],
  "performanceTier": "FTDv50",
  "fmcIpforDeviceReg": "DONTRESOLVE",
  "RegistrationId": "cisco",
  "NatId": "cisco",
  "fmcAccessPolicyName": "AWS-ACL"
}
```

- Keep the fmcIpforDeviceReg setting as DONTRESOLVE.
- The fmcAccessPolicyName needs to match an access policy on the Firewall Management Center .

**Note** FTDv5 and FTDv10 tiers are not supported.

d) Create a file named **cluster_layer.zip** to provide essential Python libraries to Lambda functions.

We recommend using Amazon Linux with Python 3.9 installed to create the cluster_layer.zip file.

**Note** If you need an Amazon Linux environment, you can create an EC2 instance using Amazon Linux 2023 AMI or use AWS Cloudshell, which runs the latest version of Amazon Linux.

For creating the cluster-layer.zip file, you need to first create requirements.txt file that consists of the python library package details and then run the shell script.

1. Create the requirements.txt file by specifying the python package details.

   The following is the sample package details that you provide in the requirements.txt file:

   ```
   $ cat requirements.txt
   pycryptodome
   paramiko
   requests
   scp
   jsonschema
   cffi
   zipp
   importlib-metadata
   ```

2. Run the following shell script to create cluster_layer.zip file.

```
$ pip3 install --platform manylinux2014_x86_64
--target=./python/lib/python3.9/site-packages
--implementation cp --python-version 3.9 --only-binary=:all:
--upgrade -r requirements.txt
$ zip -r cluster_layer.zip ./python
```



**Note** If you encounter a dependency conflict error during installation, such as urllib3 or cryptography, it is recommended that you include the conflicting packages along with their recommended versions in the requirements.txt file. After that, you can run the installation again to resolve the conflict.

e) Copy the resulting **cluster_layer.zip** file to the lambda python files folder - `cluster/aws/lambda-python-files` .

f) Create the **cluster_layer.zip** , **custom_metrics_publisher.zip** , **cluster_manger.zip** and **lifecycle_ftdv.zip** files.

A make.py file can be found in the cloned repository ( `cluster/aws/make.py` ). This will zip the python files into a Zip file and copy to a target folder.

**python3 make.py build**



**Note** If you are using a private IP address for the Management Center Virtual registration, then make sure that you set USE_PUBLIC_IP_FOR_FMC_CONN to False in the `cisco-ftdv/cluster/aws/lambda-python-files/constant.py` file.

**Step 2** Deploy **infrastructure.yaml** and note the output values for cluster deployment. Before deploying the infrastructure stack, it is important to identify the AWS region and the availability zones that will be used. Each AWS region has a different set of availability zones and VPC infrastructure. Therefore, it is essential to select the correct region and availability zones for your deployment.

a) On the AWS Console, go to **CloudFormation** and click **Create stack** ; select **With new resources(standard)** .

b) Select **Upload a template file** , click **Choose file** , and select **infrastructure.yaml** from the target folder.

c) Click **Next** and provide the required information.

e

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| ClusterName | String | Enter unique Cluster name. |
| ClusterNumber | Number | Enter unique Cluster number. |
| DeploymentType | String | Specify whether "Single-arm" or "Dual-arm" deployment required.<br>**Note**<br>Dual-Arm is supported for versions 10.0.0 and later. |
| VpcCidr | String | Enter the CIDR block for a new VPC |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| NoOfAZs | Number | Select 2 or 3 Availability Zones (AZs) for releases 7.6.0 and above; for lower releases, Select 1AZ.<br>Management, Inside, Outside and CCL subnets will be distributed across the chosen AZs accordingly. |
| ListOfAZs | List | Select Availability Zones (Count should match with Number of Availability Zones) |
| MgmtSubnetNames | CommaDelimitedList | Management subnets name (With Internet GW as Route) |
| MgmtSubnetCidrs | CommaDelimitedList | Management subnets Cidr list |
| InsideSubnetNames | CommaDelimitedList | Inside subnets name (With Private Route) |
| InsideSubnetCidrs | CommaDelimitedList | Inside subnets Cidr list |
| OutsideSubnetNames | CommaDelimitedList | Outside subnet name (With Internet GW as Route) (Dual-Arm mode only) |
| OutsideSubnetCidrs | CommaDelimitedList | Enter Outside subnets Cidr list (Dual-Arm mode only) |
| CCLSubnetNames | CommaDelimitedList | Enter CCL subnet name |
| CCLSubnetCidrs | CommaDelimitedList | Enter CCL subnet CIDR |
| LambdaAZs | List | Select 2 Availability Zones for Lambda |
| LambdaSubnetNames | CommaDelimitedList | Enter Lambda Subnets name (With NAT GW as Route), for Lambda Functions |
| LambdaSubnetCidrs | CommaDelimitedList | Enter Lambda Subnet CIDRs |

d) Enter a unique **Cluster Name** and **Cluster Number** for the cluster.
e) Select the availability zone from the **Availability Zone** list. This field lists only availability zones based on the AWS region that you select for deploying the infrastructure stack using the ClusterFormation template.
f) Click **Next** , then **Create stack** .
g) After the deployment is complete, go to **Outputs** and note the S3 **BucketName** .

*Figure 2: Output of infrastructure.yaml*

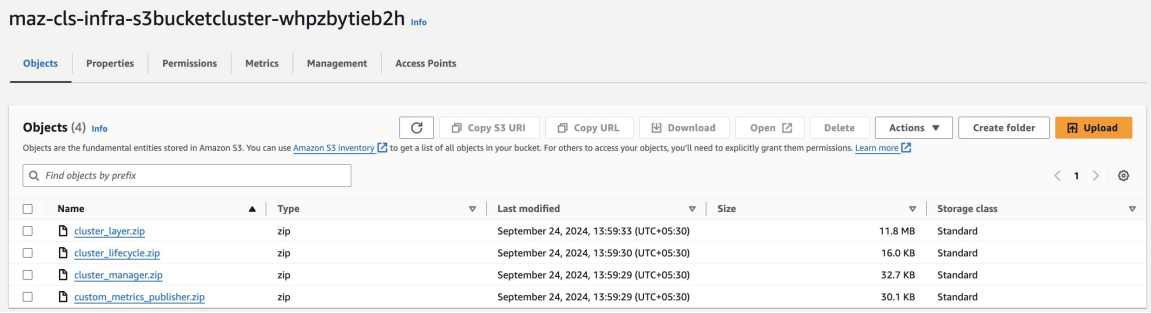
**Outputs** (13)

Search outputs

| Key | Value | Description | Export name |
|---|---|---|---|
| BucketName | maz-cls-infra-s3bucketcluster-whpzbytieb2h | Name of the Amazon S3 bucket | - |
| BucketUrl | http://maz-cls-infra-s3bucketcluster-whpzbytieb2h.s3-website-us-east-1.amazonaws.com | URL of S3 Bucket Static Website | - |
| CCLSubnetIds | subnet-0bc04e2cc9e53e5c0,subnet-0d7d046a0fca25615,subnet-03ef42bf527551569 | List of CCL subnet IDs (comma seperated) | - |
| EIPforNATgw | 3.218.44.132 | EIP reserved for NAT GW | - |
| FmcInstanceSGID | sg-076880aa64df2db5c | Security Group ID for FMC if user would like to launch in this VPC itself | - |
| InInterfaceSGId | sg-06ed933d6624fe51b | Security Group ID for Inside Interfaces | - |
| InsideSubnetIds | subnet-03d12cab8ee0eafff,subnet-0be9158b0970aebab,subnet-0b53c96fceb7c1f4d | List of Inside subnet IDs (comma seperated) | - |
| InstanceSGId | sg-0680b74be473186aa | Security Group ID for Instances Management Interface | - |
| LambdaSecurityGroupId | sg-057da2a9954e0d204 | Security Group ID for Lambda Functions | - |
| LambdaSubnetIds | subnet-03439803d989e6bdf,subnet-087488a9d6ffc95cd | List of lambda subnet IDs (comma seperated) | - |
| ListOfAZs | us-east-1a,us-east-1b,us-east-1c | Availability zones for NGFWv instances | - |
| MgmtSubnetIds | subnet-06f0bbbd3f207a504,subnet-0c339dc43688cddc9,subnet-0a67629632a655de7 | List of Mangement subnet IDs (comma seperated) | - |
| VpcName | vpc-09c2b0ad995e2fb24 | Name of the VPC created | - |

**Step 3** Upload **cluster_layer.zip** , **cluster_manager.zip** , **custom_metrics_publisher.zip** , and **cluster_lifecycle.zip** to the S3 bucket created by **infrastructure.yaml** .

*Figure 3: S3 Bucket*




**Note** Make sure that the Elastic IP address of the Lambda NAT Gateway is added to the security group associated with the Management Center Virtual.

**Step 4** Deploy **deploy_ngfw_cluster.yaml** .

a) Go to **CloudFormation** and click on **Create stack** ; select **With new resources(standard)** .

b) Select **Upload a template file** , click **Choose file** , and select **deploy_ngfw_cluster.yaml** from the target folder.

c) Click **Next** and provide the required information.

d) Provide the following cluster and infrastructure configuration information.

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| Cluster Configuration | | |
| ClusterGrpNamePrefix | String | This is the cluster name Prefix. The cluster number will be added as a suffix. |
| ClusterNumber | String | This is the cluster number. This will be suffixed to the cluster name (msa-ftdv-infra). For example, if this value is 1 , the group name will be msa-ftdv-infra-1.<br>It should be at least 1 digit, but not more than 3 digits. Default: 1. |
| ClusterSize | Numbers | This is the total number of Firewall Threat Defense Virtual nodes in a cluster.<br>Minimum: 1<br>Maximum:16 |
| DeploymentType | String | Specify whether Single-Arm or Dual-Arm deployment is required.<br>**Note**<br>Dual-Arm is supported for versions 10.0.0 and above. |
| DiagnosticInterface | String | Specify whether Diagnostic interface is required. |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| | | "ON" - Diagnostic interface will be attached.<br>"OFF" - Diagnostic interface will not be attached.<br>**Note**<br>"OFF" is supported for versions 10.0.0 and above. |
| Infrastructure Details | | |
| NoOfAZs | String | This is the total number of availability zones into which Firewall Threat Defense Virtual is deployed. (The number of availability zones varies from a Minimum 1 to Maximum 3 depending on a region).<br>The subnet will be created in these availability zones.<br>The availability zones available in this list is based on the region selected for deploying the cluster.<br>**Note**<br>Management, Inside, and Cluster Control Link (CCL) subnets are created across three availability zones based on this parameters. |
| AZ | String | The availability zone list is based on the region you plan to deploy.<br>In Availability Zone list, select the valid availability zone (1 availability zone or 2 availability zones or 3 availability zones).<br>Count should match with the value of Number of Availability Zones parameter. |
| NotifyEmailID | String | Email address to which cluster events email will be sent. You must accept a subscription email request to receive this email notification.<br>For example, admin@company.com. |
| VpcId | String | The VPC ID for the cluster group.<br>Type: AWS::EC2::VPC::Id |
| S3BktName | String | The S3 Bucket that contains the uploaded Lambda zip files. You must specify correct bucket name. |
| MgmtSubnetIds | List | Enter only one subnet per availability zone.<br>If you select multiple subnets from a same availability zone, then selecting an incorrect subnet may cause issues while deploying the Firewall Threat Defense Virtual instances.<br>Type: List<AWS::EC2::Subnet::Id> |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| InsideSubnetIds | List | Enter at least one subnet per availability zone.<br>If multiple subnets from the same Availability Zone are selected, then selecting an incorrect subnet may cause issues while deploying the Firewall Threat Defense Virtual instances.<br>Type: List<AWS::EC2::Subnet::Id> |
| OutsideSubnetIds | CommaDelimitedList | (Dual-Arm mode only)<br>Provide only one subnet per AZ. If multiple subnets from same AZ are chosen, wrong subnet selection will cause problems while deploying the NGFWv instances.<br>Make sure to add subnet from AZ provided. |
| LambdaSubnets | List | Enter at least two subnets for the Lambda functions. The two subnets you enter must have a NAT gateway to enable the Lambda functions to communicate with AWS services, which are public DNS.<br>Type: List<AWS::EC2::Subnet::Id> |
| CCLSubnetIds | String | Enter at least one subnet per availability zone.<br>If multiple subnets from the same Availability Zone are selected, then selecting an incorrect subnet may cause issues while deploying the Firewall Threat Defense Virtual instances.<br>Type: List<AWS::EC2::Subnet::Id> |
| CCLSubnetRanges | String | Enter IP addresses range of CCL subnets for different availability zones.<br>Exclude first 4 reserved IP addresses. IP address pool for Cluster Control Link (CCL).<br>IP address is allocated to the CCL interfaces of the Firewall Threat Defense Virtual instance from CCL IP address pool. |
| MgmtInterfaceSG | List | Select security group ID for the Firewall Threat Defense Virtual instances.<br>Type: List<AWS::EC2::SecurityGroup::Id> |
| InsideInterfaceSG | List | Select security group ID for the inside interface of Firewall Threat Defense Virtual instances.<br>Type: List<AWS::EC2::SecurityGroup::Id> |
| OutsideInterfaceSG | String | (Dual-Arm mode only) |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| | | Provide security group ID for NGFWv instances outside interface. |
| LambdaSG | List | Select a security group for the Lambda functions.<br>Ensure outbound connections are set to ANYWHERE .<br>Type: List<AWS::EC2::SecurityGroup::Id> |
| CCLInterfaceSG | List | Select a security group ID for CCL interface of the Firewall Threat Defense Virtual instances. |
| DualArmAppCidrList | CommaDelimitedList | (Dual-Arm mode only)<br>Enter dual-arm application CIDRs for East-West traffic. |
| GWLB Configuration | | |
| DeployGWLBE | String | Click Yes to deploy the GWLB endpoint.<br>By default, the value is set to No . |
| VpcIdLBE | String | Enter VPC to deploy Gateway Load Balancer Endpoint.<br>**Note**<br>Do not enter any value in this field if you are not deploying the GWLB endpoint. |
| GWLBESubnetId | String | Enter only one subnet ID.<br>**Note**<br>Do not enter any value in this field if you are not deploying the GWLB endpoint.<br>Ensure that the subnet belongs to the correct VPC, and the availability zones that you have specified. |
| TargetFailover | String | Enable Target Failover support when a target fails or deregisters. (By default, the value of this parameter is set to rebalance ).<br>• no_rebalance : Directs existing flows to failed targets and new flows to healthy targets, ensuring backward compatibility.<br>• rebalance : Redistributes existing flows while ensuring that new flows go to healthy targets.<br>rebalance is supported from Firewall Threat Defense Virtual Version 7.4.1 and later. |
| TgHealthPort | String | Enter Health Check Port for GWLB.<br>**Note**<br>By default, this port must not be used for traffic. |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| | | Ensure the value you provide is a valid TCP port. Default: 8080 |
| Cisco NGFWv Instance Configuration | | |
| InstanceType | String | Cisco Firewall Threat Defense Virtual EC2 instance type.<br>Ensure that the AWS Region supports Instance Type you select.<br>By default, **c5.xlarge** is selected.<br>**Note**<br>Dual-arm deployment supports only _.4xlarge when Diagnostic Interface is "ON". |
| LicenseType | String | Choose Cisco Firewall Threat Defense Virtual EC2 instance license type. Ensure that the AMI ID that you enter in **AMI-ID** parameter is of the same licensing type.<br>By default, **BYOL** is selected. |
| AssignPublicIP | String | Set the value as **true** to assign a public IP address for Firewall Threat Defense Virtual from the AWS IP address pool. |
| AmiID | String | Choose the correct AMI ID as per the region, version, and license type (BYOL or PAYG).<br>Firewall Threat Defense Virtual 7.2 and later support clustering, and Firewall Threat Defense Virtual Version 7.6 and later support the autoscaling and multiple availability zone enhancements.<br>Type: AWS::EC2::Image::Id |
| ngfwPassword | String | Firewall Threat Defense Virtual instance password.<br>All Firewall Threat Defense Virtual instances come up with a default password, which is in the Userdata field of the Launch Template (Cluster Group).<br>The password is activated after Firewall Threat Defense Virtual is accessible.<br>Minimum length must be 8 characters. The password can either be a plain text password or a KMS encrypted password. |
| KmsArn | String | Enter ARN of an existing KMS (AWS KMS key to encrypt at rest). |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| | | If you specify a value in this field, then the Firewall Threat Defense Virtual instance's admin password must be an encrypted password.<br>Example of generating an encrypted password: " aws kms encrypt --key-id <KMS ARN> --plaintext <password> "<br>The password encryption must be done using only the specified ARN. |
| FMC Automation Configuration | | |
| fmcDeviceGrpName | String | Enter a unique name for the cluster group in management center. |
| fmcPublishMetrics | String | Select **`true`** to create a Lambda Function to poll management center and publish specific device group metrics to AWS CloudWatch.<br>Allowed values:<br>• true<br>• false<br>By default, the value is set to true . |
| fmcMetricsUsername | String | Enter a unique internal user name for polling memory metrics from management center.<br>The user must have privileges of Network Admin and Maintenance User or more . |
| fmcMetricsPassword | String | Enter the password.<br>If you have mentioned KMS Master Key ARN parameter, ensure to provide an encrypted password.<br>Ensure to enter the correct password because entering incorrect password may result in failure of metrics collection. |
| fmcServer | String | The IP address can be an external IP address or the IP address reachable in Firewall Threat Defense Virtual management subnet in the VPC.<br>Minimum length: 7<br>Maximum length:15 |
| fmcOperationsUsername | String | Provide a unique internal user name for Firewall Management Center Virtual for CloudWatch. |

| Parameter | Allowed Values/Type | Description |
|---|---|---|
| | | The user must have Administrator privileges. |
| fmcOperationsPassword | String | Enter the password.<br>If you have mentioned KMS Master Key ARN parameter, ensure to provide an encrypted password. |
| Scaling Configuration | | |
| CpuThresholds | CommaDelimitedList | (Optional) Specifying non-zero lower and upper thresholds will create scale policies. If (0,0) is selected, no CPU scaling alarm or policies will be created. Evaluation points and data points are at default or recommended values.<br>By default, autoscaling is enabled in this template. Autoscale can be disabled after deployment. |
| MemoryThresholds | CommaDelimitedList | Specifying non-zero lower and upper threshold will create scale policies. If (0,0) is selected, no memory scaling alarm or policies will be created. Evaluation points and data points are at default or recommended values. |

e) Click **Next** .
f) Click to acknowledge all the AWS CloudFormation options.
g) Click **Submit** to deploy the cluster.
h) Click **Next** , then **Create stack** .

The Lambda functions manage the rest of the process, and the Firewall Threat Defense Virtual s will automatically register with the Firewall Management Center .

***Figure 4: Deployed Resources***

The status changes from CREATE_IN_PROGRESS to CREATE COMPLETE, indicating successful deployment.

**Step 5** Verify the cluster deployment by logging into any one of the nodes and using the **show cluster info** command.

***Figure 5: Cluster Nodes***

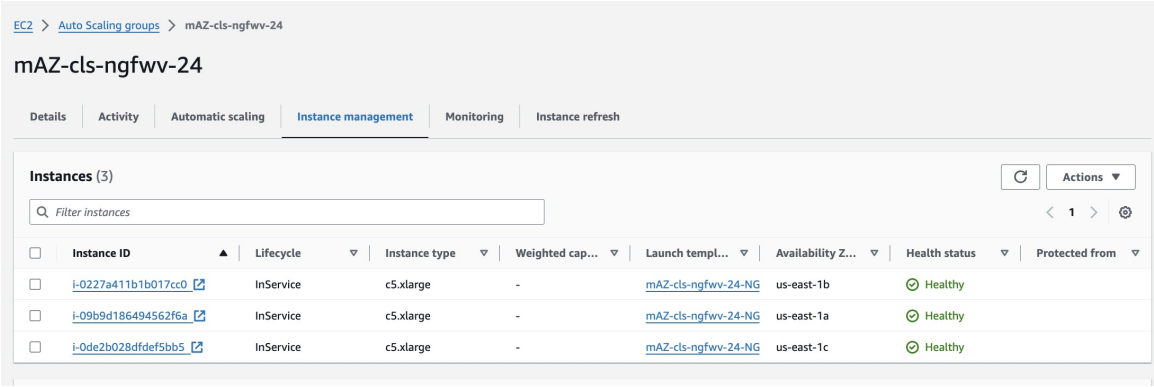
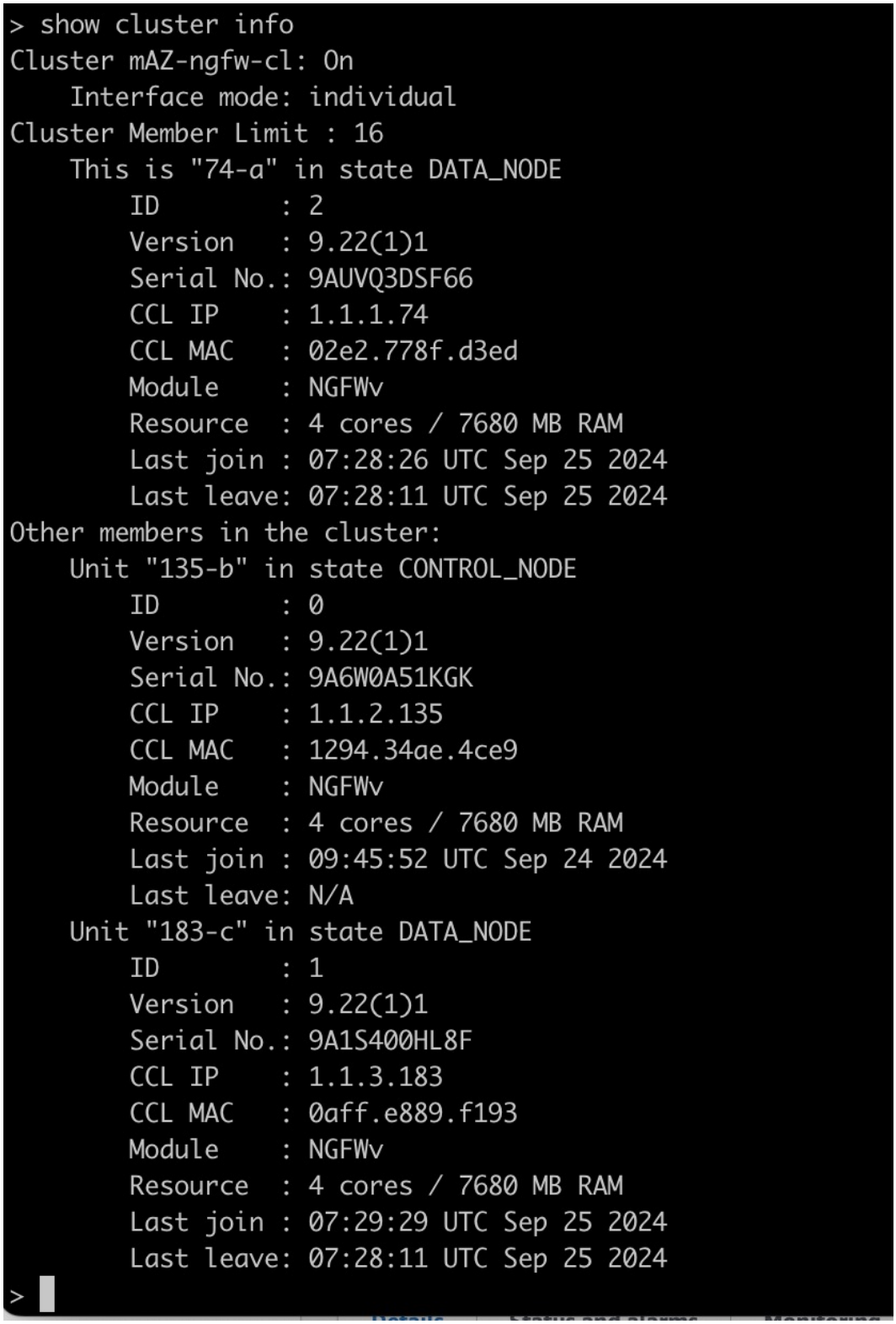


*Figure 6: show cluster info*



# Configure Management Center NAT for dual-arm deployment

The NAT configuration enables proper traffic forwarding in a dual-arm deployment where traffic from applications passes through firewall inspection and then routes directly to the Internet Gateway, bypassing the GWLB on egress.

In a dual-arm deployment, you need to configure security zones and NAT policies to handle traffic flow between the internal network, Geneve VNI tunnel interface, and external Internet Gateway. This configuration ensures that inspected traffic can reach its destination while maintaining security policy enforcement.

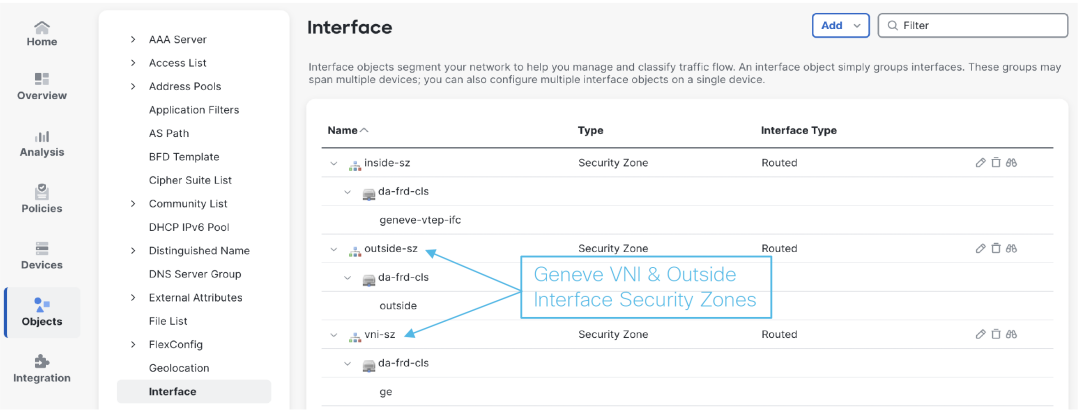
## Procedure

**Step 1** Create Security Zones.

Security zones allow you to apply access control, NAT, and inspection policies to a group of interfaces collectively, instead of configuring them individually.

In the Management Center, navigate to Objects > Interface > Add > Security Zones.

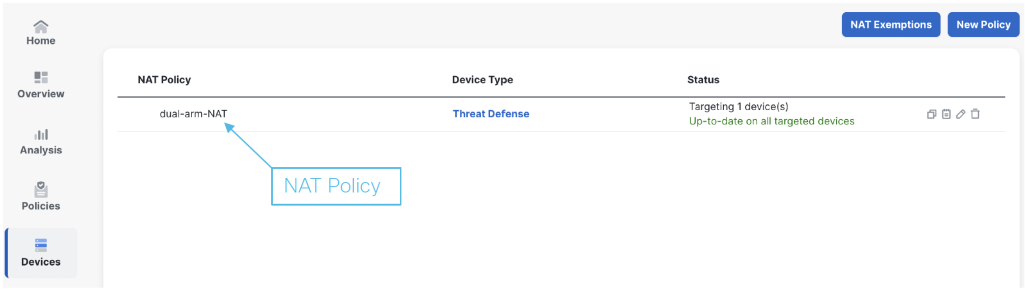
- **Inside Security Zone** → For the internal or ingress interface (traffic from applications).
- **VNI Security Zone** → For the Geneve VNI tunnel interface (traffic from GWLB).
- **Outside Security Zone** → For the egress interface toward the Internet Gateway.




**Step 2** Configure NAT policy.

The NAT policy (dual-arm-NAT) defines the source and destination address translation rules needed to forward traffic from the inside interface (after firewall inspection) directly to the outside interface for Internet access, bypassing the GWLB on egress.

Navigate to Devices > NAT > New Policy.

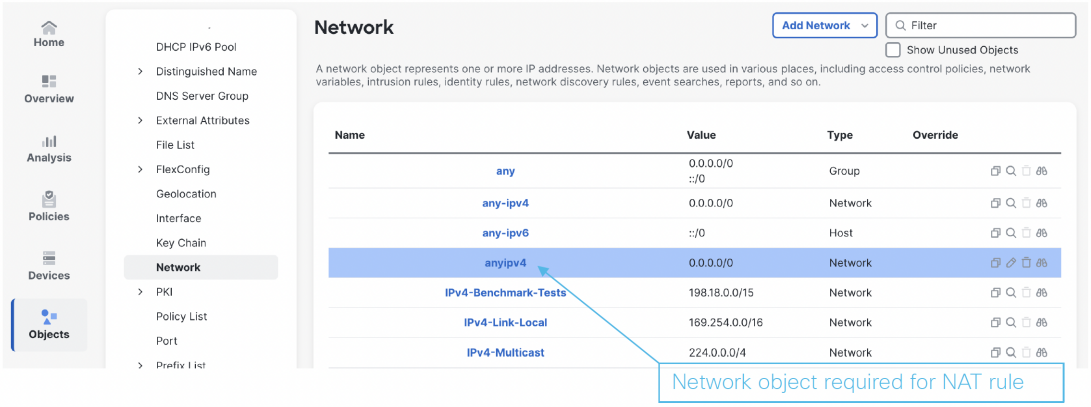



**Step 3** Create a Network Object for your source network.

Network objects represent one or more IP addresses or ranges. These objects are used in various places such as NAT rules, access control policies, and network discovery rules.

Navigate to Objects > Network > Add Network.

- **Name:** The logical name given to the network object.
- **Value:** The actual IP address range or network in CIDR format (for example, `0.0.0.0/0` for all IPv4 addresses).
- **Type:** Specifies whether the object is a Group, Network, or Host.

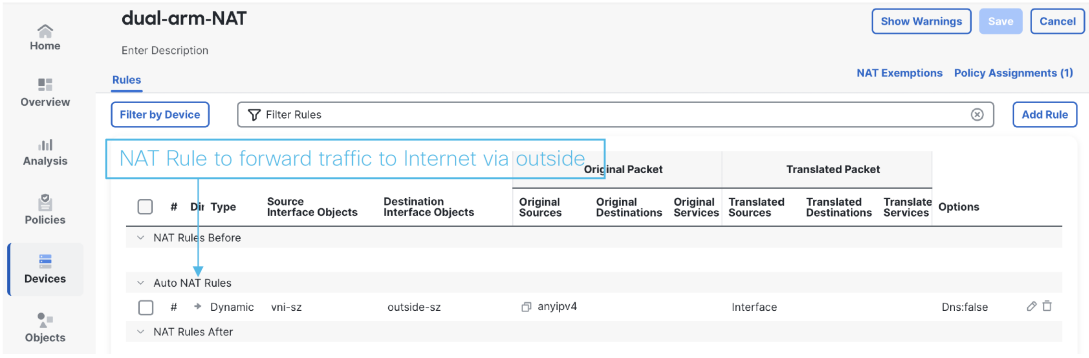



**Step 4** Add NAT Rule

NAT policy is configured for a dual-arm deployment to forward traffic to the internet through the outside interface.

Navigate to Devices > NAT > dual-arm-NAT > Add Rule

After completing the configuration, deploy the polices to the cluster group, following the path, Deploy > Select Cluster Group > Deploy All.

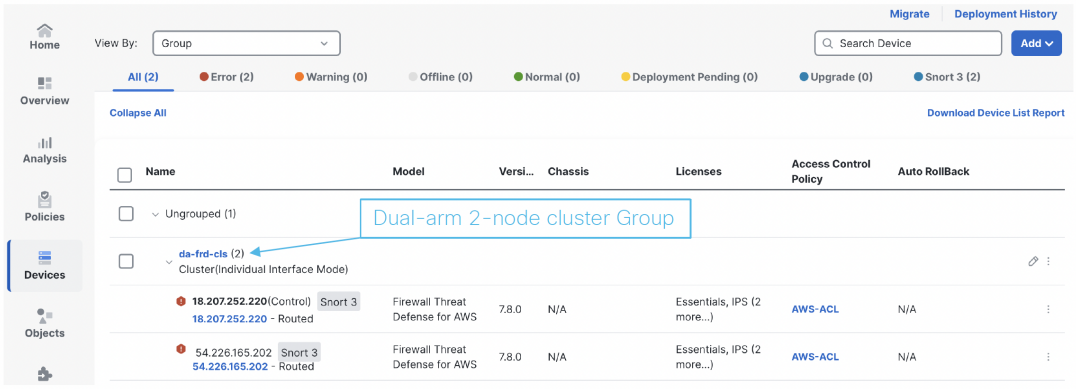



**Step 5** Post-deployment checks

a) Verify the Dual-Arm 2-node cluster group.

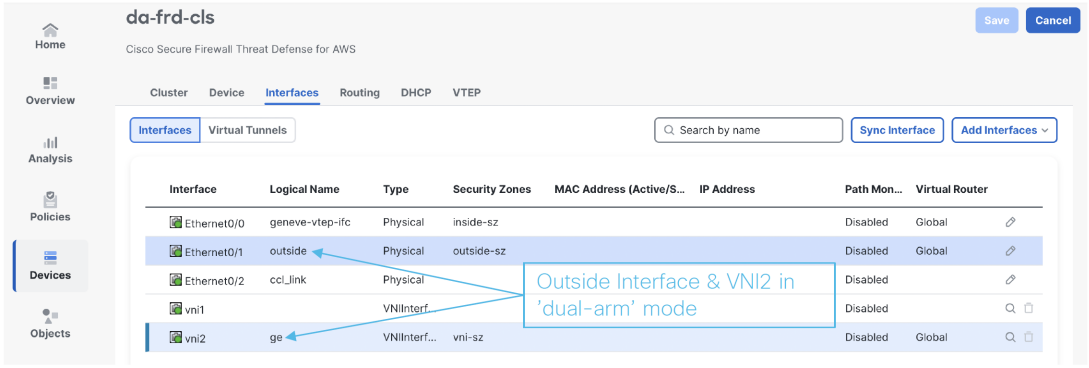
This step is to verify that the cluster consists of two Threat Defense Virtual instances operating in Dual-Arm mode, where each instance uses at least two dedicated network interfaces to handle separate traffic paths.

Navigate to Devices > Device Management and verify the cluster group.



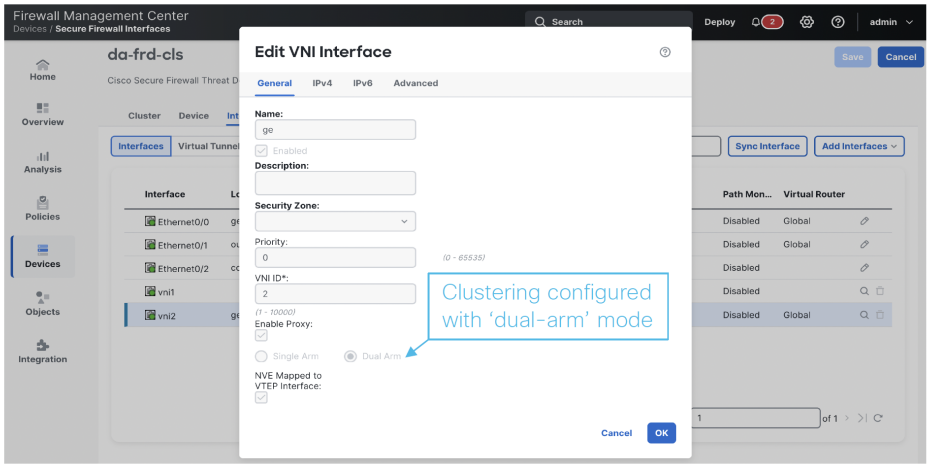

b) Verify the interfaces and configurations of the threat Defense Virtual cluster.

Navigate to Devices > Device Management > Cluster Group > Interfaces and verify the interfaces for the Dual-Arm deployment.




c) Verify the Dual-Arm configuration.

Navigate to Devices > Device Management > Cluster Group > Interfaces > vni2. Click on Edit VNI Interface to verify the Dual-Arm configuration.



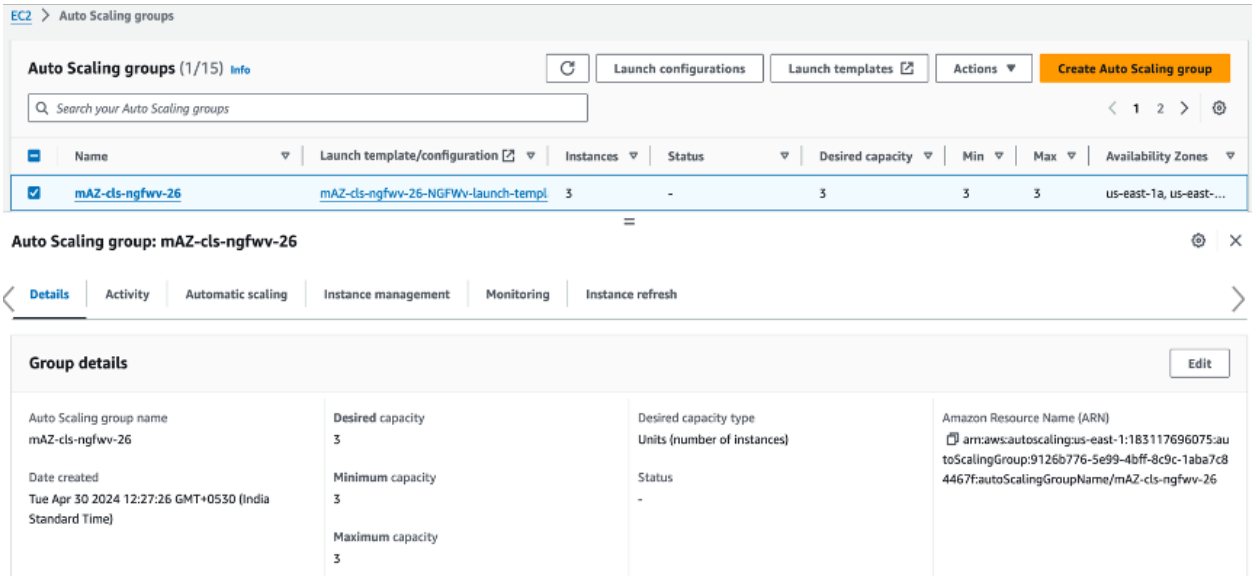

## Configure autoscale parameter

Configure the autoscale parameters to specify capacity settings and verify autoscale functionality for the Firewall Threat Defense Virtual deployment.

After the deployment is completed, specify the **Minimum**, **Maximum**, and **Desired** capacity of the Firewall Threat Defense Virtual autoscale group. After specifying the capacity values, verify the autoscale functionality.

### Procedure

**Step 1** From the AWS console, choose **Services** > **EC2** > **Auto Scaling groups** > **Created ClusterAutoscale group**.




**Step 2** Select the autoscale group check box.

**Step 3** Click **Actions** to edit the autoscaling group capacity.

**Step 4** Set the **Desired capacity**, and then configure the **Scaling limits** capacity.

**Step 5** Check if the CPU and Memory metric data is available and whether scaling is occurring as expected in AWS Cloudwatch alarms.

# Configure IMDSv2 Required Mode in Firewall Threat Defense Virtual clustering by updating Stack

This task enables IMDSv2 Required mode for existing Firewall Threat Defense Virtual clustering deployments.

You can configure the IMDSv2 Required mode for the Firewall Threat Defense Virtual autoscale group instances that are already deployed on the AWS.

**Before you begin**

IMDSv2 Required mode is only supported by Firewall Threat Defense Virtual version 7.6 and later. You must ensure that your existing instances version is compatible (upgraded to version 7.6) with IMDSv2 mode before configuring the IMDSv2 mode for your deployment.

Follow these steps to configure IMDSv2 Required mode in Firewall Threat Defense Virtual clustering by updating stack:

**Procedure**

**Step 1** On the AWS Console, go to **CloudFormation** and click **Stacks**.

**Step 2** Select the stack of the initially-deployed clustering instances.

**Step 3** Click **Update**.

**Step 4** On the **Update stack** page, click **Replace existing template**.

**Step 5** Under **Specify template** section, click **Upload a template file**.

**Step 6** Choose and upload the template which support IMDSv2.

**Step 7** Provide values for the input parameters in the template.

**Step 8** Update the stack.

# Manual cluster deployment in AWS

To deploy the cluster manually, prepare the day 0 configuration, deploy each node, and then add the control node to the Firewall Management Center.

On version 7.6.4-69 and from 10.0.0, each cluster node requires a unique AWS instance tag "cluster-node-id" with a value ranging from 1 to 16 for internal cluster configuration. Please ensure that the tag is added before the device boots up.

For example: key "cluster-node-id" -> value "1"

Additionally, make sure that "Allow tags in instance metadata" is set to "Enabled."

## Create the Day0 configuration for AWS

You can use either a fixed configuration or a customized configuration. We recommend using the fixed configuration.

### Create the Day0 configuration with a fixed configuration for AWS

The fixed configuration will auto-generate the cluster bootstrap configuration.

#### Single availability zone - Day0 configuration with a fixed configuration for AWS

```
{
    "AdminPassword": "password",
    "Hostname": "hostname",
    "FirewallMode": "Routed",
    "ManageLocally": "No",
    "Cluster": {
        "CclSubnetRange": "ip_address_start ip_address_end",
        "ClusterGroupName": "cluster_name",
        [For Gateway Load Balancer] "Geneve": "{Yes | No}",
        [For Gateway Load Balancer] "HealthProbePort": "port"
    }
}
```

For example:

```
{
 "AdminPassword": "Sup3rnatural",
 "Hostname": "ciscoftdv",
 "FirewallMode": "Routed",
 "ManageLocally": "No",
 "Cluster": {
             "CclSubnetRange": "10.5.90.4 10.5.90.30",
  "ClusterGroupName": "ftdv-cluster",
  "Geneve": "Yes",
  "HealthProbePort": "7777"
 }
}
```

### Multiple availability zone - Day0 configuration with a fixed configuration for AWS

```
{
    "AdminPassword": "password",
    "Hostname": "hostname",
    "FirewallMode": "Routed",
    "ManageLocally": "No",
    "Cluster": {
    "CclSubnetRange":[
    "ip_address_start_AZ1 ip_address_end_AZ1",
    "ip_address_start_AZ2 ip_address_end_AZ2",
    "ip_address_start_AZ3 ip_address_end_AZ3"
     ],
   "ClusterGroupName": "cluster_name",
        [For Gateway Load Balancer] "Geneve": "{Yes | No}",
        [For Gateway Load Balancer] "HealthProbePort": "port"
    }
}
```

For example: **Two Availability Zone**

```
{
 "AdminPassword": "Sup4rnatural",
 "Hostname": "ftdvcluster",
 "FirewallMode": "Routed",
 "ManageLocally": "No",
 "Cluster": {
        "CclSubnetRange": [
            "10.5.90.4  10.5.90.30",
            "10.5.91.4  10.5.91.30"
        ],
  "ClusterGroupName": "ftdv-cluster",
  "Geneve": "Yes",
  "HealthProbePort": "8080"
 }
}
```

For example: **Three Availability Zone**

```
{
 "AdminPassword": "Sup4rnatural",
 "Hostname": "ftdvcluster",
 "FirewallMode": "Routed",
 "ManageLocally": "No",
 "Cluster": {
        "CclSubnetRange": [
            "10.5.90.4  10.5.90.30",
            "10.5.91.4  10.5.91.30",
            "10.5.92.4  10.5.92.30"
        ],
  "ClusterGroupName": "ftdv-cluster",
  "Geneve": "Yes",
  "HealthProbePort": "8080"
 }
}
```

### Single or multiple availability zone Day0 configuration for Dual-Arm deployment

```
{
    "AdminPassword": "password",
```

```
    "Hostname": "hostname",
    "FirewallMode": "Routed",
    "ManageLocally": "No",
    "Diagnostic": "OFF",
    "Cluster": {
    "CclSubnetRange":[
    "ip_address_start_AZ1 ip_address_end_AZ1",
    "ip_address_start_AZ2 ip_address_end_AZ2",
    "ip_address_start_AZ3 ip_address_end_AZ3"
     ],
    "ProxyType": "dual-arm"
    "DualArmAppCidrList":[
    "CIDR_BLOCK_1",
    "CIDR_BLOCK_2",
    "CIDR_BLOCK_3"
     ],
   "ClusterGroupName": "cluster_name",
        [For Gateway Load Balancer] "Geneve": "{Yes | No}",
        [For Gateway Load Balancer] "HealthProbePort": "port"
    }
}
```

For example:

```
{
"AdminPassword": "FtDv_ClU3TeR44",
"Hostname": "ftdvcluster",
"FirewallMode": "routed",
"ManageLocally": "No",
"Diagnostic": "OFF",
"Cluster": {
"CclSubnetRange": [
"10.5.90.4 10.5.90.30",
"10.5.91.4 10.5.91.30",
"10.5.92.4 10.5.92.30"
],
"ProxyType": "dual-arm",
"DualArmAppCidrList": [
"10.0.0.0/8",
"172.16.0.0/12",
"192.168.0.0/16"
],
"Geneve": "Yes",
"HealthProbePort": "8080",
"ClusterGroupName": "ftdv-cluster"
}
}
```

### *Configure the cluster infrastructure subnet range*

Use a cluster infrastructure subnet range to allocate IP addresses within a /27 CIDR block. This range provides the usable IP addresses for your cluster infrastructure.

**Before you begin**

- Review the table to determine valid start and end IP addresses for the /27 CIDR blocks you plan to use.
- The first four IP addresses in each subnet (including the network address) are reserved by AWS. Configure the **CclSubnetRange** to begin with the fifth IP address in the subnet.

The table shows the valid start and end IP addresses for each /27 CIDR allocation.

*Table 2: **CclSubnetRange** mapping*

| CIDR | Start IP Address | End IP Address |
|---|---|---|
| x.y.z.0/27 | x.y.z.4 | x.y.z.30 |
| x.y.z.32/27 | x.y.z.36 | x.y.z.62 |
| x.y.z.64/27 | x.y.z.68 | x.y.z.94 |
| x.y.z.96/27 | x.y.z.100 | x.y.z.126 |
| x.y.z.128/27 | x.y.z.132 | x.y.z.158 |
| x.y.z.160/27 | x.y.z.164 | x.y.z.190 |
| x.y.z.192/27 | x.y.z.196 | x.y.z.222 |
| x.y.z.224/27 | x.y.z.228 | x.y.z.254 |

Complete these steps to configure the cluster infrastructure subnet range.

**Procedure**

**Step 1** Identify the /27 CIDR block assigned to your cluster infrastructure subnet.

**Step 2** Find the matching CIDR in the mapping table.

**Step 3** Replace "x.y.z" with the first three octets of your subnet. For example, 172.20.10 or 10.100.5.

**Step 4** Configure the range by entering the start and end IP addresses from the mapping table for your subnet.

- Example 1:
  - CIDR: 172.20.10.64/27
  - Subnet range: 172.20.10.68 to 172.20.10.94
- Example 2:
  - CIDR: 10.100.5.128/27
  - Subnet range: 10.100.5.132 to 10.100.5.158

## Deploy cluster nodes

Deploy cluster nodes to form a functional cluster that can handle traffic distribution and provide high availability for network security services.

Deploy the cluster nodes so they form a cluster.

**Procedure**

**Step 1** Deploy the Threat Defense Virtual instance by using the cluster day 0 configuration with the required number of interfaces - four interfaces if you are using Gateway Load Balancer (GWLB), or five interfaces if you are using non-native load balancer. To do this, in the **Configure Instance Details** > **Advanced Details** section, paste the cluster day 0 configuration.

**Note** Ensure that you attach interfaces to the instances in the order given below.

- AWS Gateway Load Balancer - four interfaces - management, diagnostic, inside, and cluster control link.
- Non-native load balancers - five interfaces - management, diagnostic, inside, outside, and cluster control link.

For more information on deploying Threat Defense Virtual on AWS, refer to Deploy the Threat Defense Virtual on AWS.

**Step 2** Repeat Step 1 to deploy the required number of additional nodes.

**Step 3** Use the **show cluster info** command on the Threat Defense Virtual console to verify if all nodes have successfully joined the cluster.

**Step 4** Configure the AWS Gateway Load Balancer.

a) Create a target group and GWLB.

b) Attach the target group to the GWLB.

**Note** Ensure that you configure the GWLB to use the correct security group, listener configuration, and health check settings.

c) Register the data interface (inside interface) with the Target Group using IP addresses.

For more information, refer to Create a Gateway Load Balancer.

**Step 5** Add the control node to the Management Center. Refer to Add the cluster to the Management Center (manual deployment), on page 106.

# Target failover for Secure Firewall Threat Defense Virtual clustering with GWLB in AWS

Threat Defense Virtual clustering in AWS utilizes the Gateway Load Balancer (GWLB) to balance and forward network packets for inspection to a designated Threat Defense Virtual node. The GWLB is designed to continue sending network packets to the target node in the event of a failover or deregistration of that node.

The Target Failover feature in AWS enables GWLB to redirect network packets to a healthy target node in the event of node deregistration during planned maintenance or a target node failure. It takes advantage of the cluster's stateful failover.

### Configuration methods

In AWS, you can configure Target Failover through the AWS Elastic Load Balancing (ELB) API or AWS console.

**Note** If a target node fails while the GWLB routes traffic using certain protocols such as SSH, SCP, CURL, and so on, then there may be a delay in redirecting traffic to a healthy target. This delay is due to rebalancing and rerouting of traffic flow.

- AWS API — In the AWS ELB API *modify-target-group-attributes*, you can define the flow handling behavior by modifying the parameters.
  - target_failover.on_unhealthy - It defines how the GWLB handles the network flow when the target becomes unhealthy.
  - target_failover.on_deregistration - It defines how the GWLB handles the network flow when the target is deregistered.

  This command shows the sample API parameter configuration of defining these two parameters.

```
aws elbv2 modify-target-group-attributes \
--target-group-arn arn:aws:elasticloadbalancing:…/my-targets/73e2d6bc24d8a067 \
--attributes \
Key=target_failover.on_unhealthy, Value=rebalance[no_rebalance] \
Key=target_failover.on_deregistration, Value=rebalance[no_rebalance]
```

  For more information, refer to TargetGroupAttribute in the AWS documentation.
- AWS Console — In the EC2 console, you can enable the Target Failover option on the Target Group page by configuring these options.
  - Edit Target Groups Attributes
  - Enable Target Failover
  - Verify Rebalance Flows

  For more information about how to enable Target Failover, refer to Enable target failover for Secure Firewall Threat Defense Virtual clustering in AWS, on page 38.

# Enable target failover for Secure Firewall Threat Defense Virtual clustering in AWS

This task enables target failover configuration in AWS to ensure network flow continuity when Firewall Threat Defense Virtual cluster targets fail or become deregistered.

The data interface of Firewall Threat Defense Virtual is registered to a target group of GWLB in AWS. In the Firewall Threat Defense Virtual clustering, each instance is associated with a Target Group. The GWLB

load balances and sends the traffic to this healthy instance identified or registered as a target node in the target group.

**Before you begin**

You must have deployed the cluster in AWS either by manual method or using CloudFormation templates.

If you are deploying a cluster using a CloudFormation template, you can also enable the **Target Failover** parameter by assigning the **rebalance** attribute that is available under **GWLB Configuration** section of the cluster deployment file, `deploy_ftdv_clustering.yaml`. In the template, by default, the value is set to **rebalance** for this parameter. However, the default value for this parameter is set to **no_rebalance** on the AWS console.

Where,

- **no_rebalance** - GWLB continues to send the network flow to the failed or deregistered target.
- **rebalance** - GWLB sends the network flow to another healthy target when the existing target is failed or deregistered.

For information on deploying stack in AWS, refer to:

- Deploy the Cluster in AWS Manually
- Deploy the Stack in AWS Using a CloudFormation Template

**Procedure**

**Step 1** On the AWS Console, go to **Services** > **EC2**

**Step 2** Click **Target Groups** to view the target groups page.

**Step 3** Select the target group to which the Firewall Threat Defense Virtual data interface IPs are registered. The target group details page is displayed, where you can enable the Target failover attributes.

**Step 4** Go to the **Attributes** menu.

**Step 5** Click **Edit** to edit the attributes.

**Step 6** Toggle the **Rebalance flows** slider button to the right to enable target failover to configure GWLB to rebalance and forward the existing network packets to a healthy target node in the event of target failover or deregistration.

# Deploy the cluster in Azure

You can use the cluster with the Azure Gateway Load Balancer (GWLB), or with a non-native load balancer. To deploy a cluster in Azure, use Azure Resource Manager (ARM) templates to deploy a Virtual Machine Scale Set.

## Sample topology for GWLB-based cluster deployment

Sample topologies for deploying Gateway Load Balancer (GWLB) clusters are given below.

*Figure 7: Inbound traffic use case and topology with GWLB*

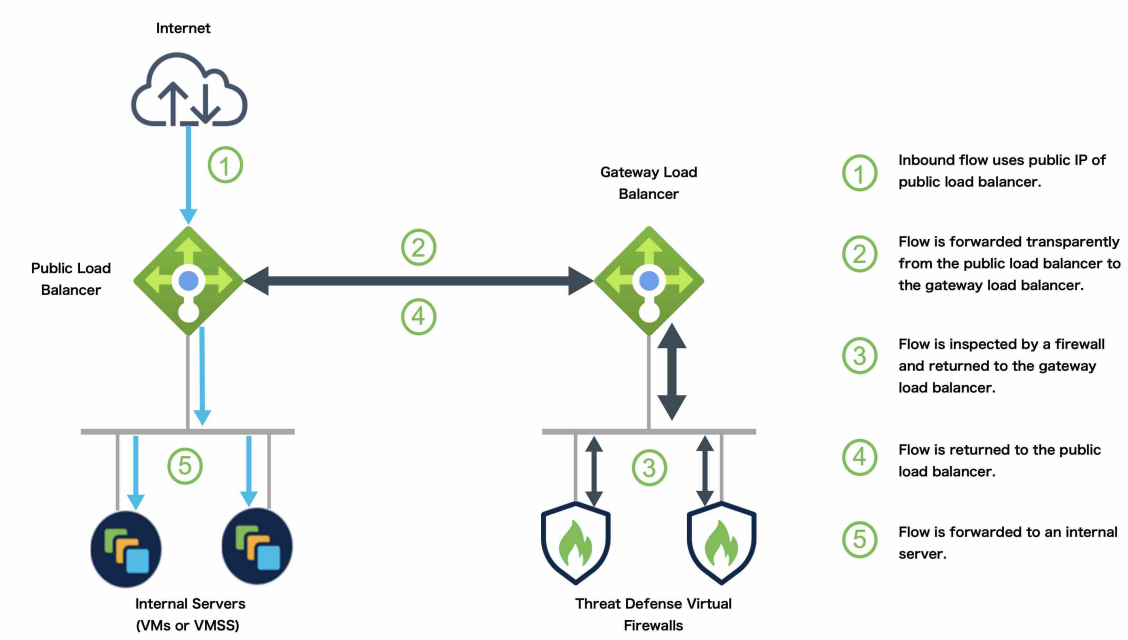
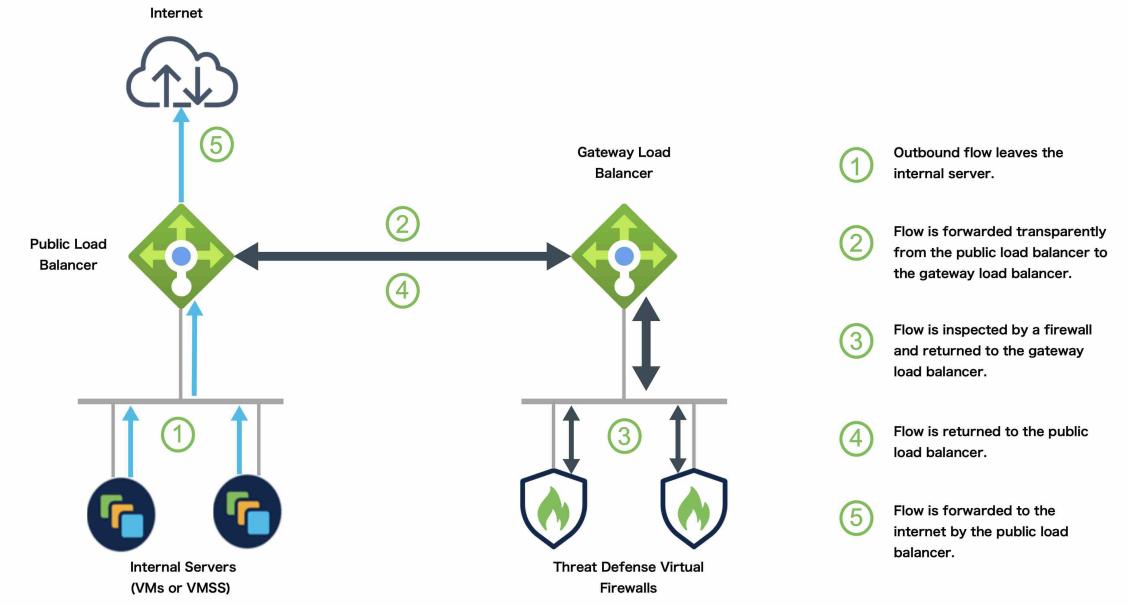



*Figure 8: Outbound traffic use case and topology with GWLB*

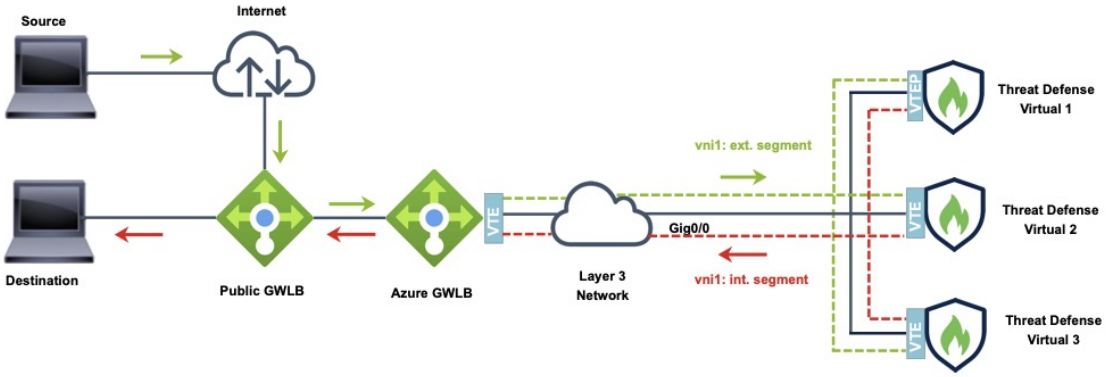


# Azure gateway load balancer and paired proxy

In an Azure service chain, Threat Defense Virtuals act as a transparent gateway that can intercept packets between the internet and the customer service. The Threat Defense Virtual defines an external interface and an internal interface on a single NIC by utilizing VXLAN segments in a paired proxy.

The figure shows traffic forwarded to the Azure Gateway Load Balancer from the Public Gateway Load Balancer on the external VXLAN segment. The Gateway Load Balancer balances traffic among multiple Threat Defense Virtuals, which inspect the traffic before either dropping it or sending it back to the Gateway Load Balancer on the internal VXLAN segment. The Azure Gateway Load Balancer then sends the traffic back to the Public Gateway Load Balancer and to the destination.

***Figure 9: Azure gateway load balancer with paired proxy***



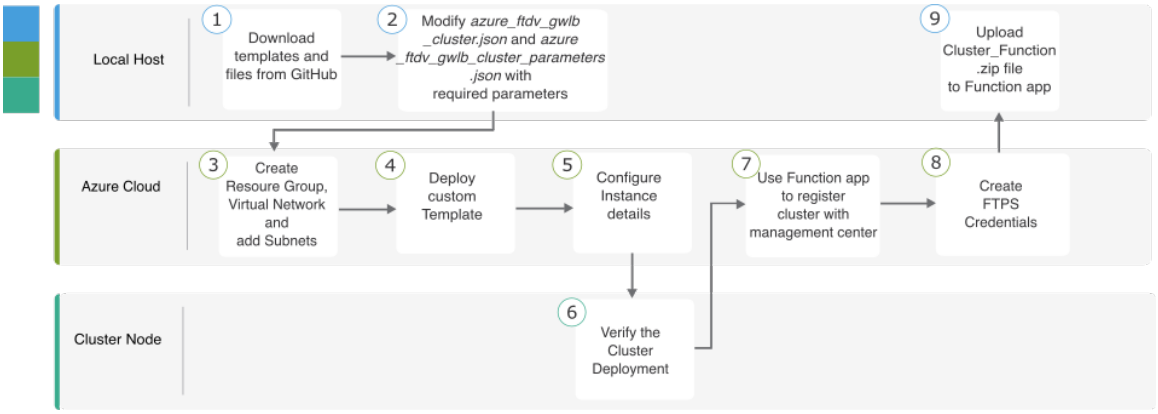

**Traffic flow between GWLBe to GWLB (Geneve Single-Arm Proxy) in Azure**

# End-to-end process for deploying Threat Defense Virtual cluster in Azure with GWLB

Provides a workflow to help users successfully deploy and verify a Threat Defense Virtual cluster on Azure with GWLB using either template-based or manual deployment processes.

### Template-based Deployment

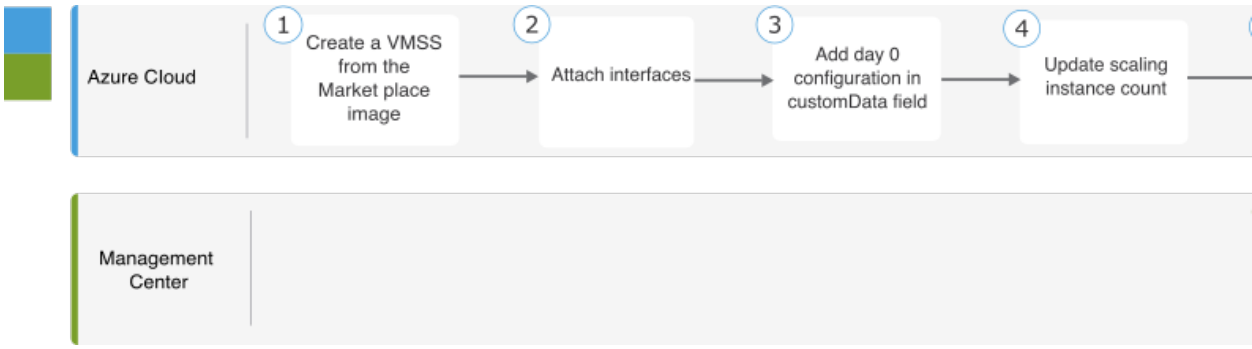
The following flowchart illustrates the workflow for template-based deployment of the Threat Defense Virtual cluster in Azure with GWLB.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Download templates and files from GitHub. |
| 2 | Local Host | Modify *azure_ftdv_gwlb_cluster.json* and *azure_ftdv_gwlb_cluster_parameters.json* with the required parameters. |
| 3 | Azure Cloud | Create the resource group, virtual network, and subnets. |
| 4 | Azure Cloud | Deploy custom template. |
| 5 | Azure Cloud | Configure instance details. |
| 6 | Cluster Node | Verify cluster deployment. |
| 7 | Azure Cloud | Use the Function app to register the cluster with the Management Center. |
| 8 | Azure Cloud | Create FTPS credentials. |
| 9 | Local Host | Upload *Cluster_Function.zip* file to the Function app. |

### Manual Deployment

The following flowchart illustrates the workflow of manual deployment of Threat Defense Virtual cluster in Azure with GWLB.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Create a VMSS from the Marketplace image. |
| 2 | Local Host | Attach interfaces. |
| 3 | Local Host | Add day 0 configuration in the customData field. |
| 4 | Local Host | Update scaling instance count. |
| 5 | Local Host | Configure GWLB. |
| 6 | Management Center | Add control node. |

# Templates

The templates are available in GitHub. The parameter values are self-explanatory with the parameter names and values given in the template.

- azure_ftdv_gwlb_cluster_parameters.json – Template to enter parameters for the Firewall Threat Defense Virtual cluster.
- azure_ftdv_gwlb_cluster.json – Template to deploy Firewall Threat Defense Virtual cluster.

# Prerequisites for Azure GWLB cluster deployment

Complete these prerequisite tasks before deploying the Azure GWLB cluster.

- To allow the cluster to auto-register to the management center, create a user with Network Admin & Maintenance User privileges on the management center. Users with these privileges can use REST API. Refer to the Cisco Secure Firewall Management Center Administration Guide.
- Add an access policy in the management center that matches the name of the policy that you will specify during template deployment.
- Ensure that the Management Center Virtual is licensed appropriately.

- Perform these steps after the cluster is added to the Management Center Virtual:

  1. Configure platform settings with the health check port number in the Management Center. For more information on configuring this, see Platform Settings.
  2. Create a static route for data traffic. For more information on creating a static route, see Add a Static Route.

     Sample static route configuration:

```
Network: any-ipv4
Interface: vxlan_tunnel
Leaked from Virtual Router: Global
Gateway: vxlan_tunnel_gw
Tunneled: false
Metric: 2
```

**Note** *vxlan_tunnel_gw* is the data subnet's gateway IP address.

# Deploy cluster on Azure with GWLB using an Azure resource manager template

This task creates a clustered deployment of Firewall Threat Defense Virtual instances on Azure using Gateway Load Balancer (GWLB) through Azure Resource Manager templates.

Deploy the Virtual Machine Scale Set for Azure GWLB using the customized Azure Resource Manager (ARM) template. The templates mentioned in the steps below are available on GitHub.

### Procedure

**Step 1** Prepare the template.

a) Clone the GitHub repository to your local folder. See https://github.com/CiscoDevNet/cisco-ftdv/tree/master/cluster/azure.

b) Modify azure_ftdv_gwlb_cluster.json and azure_ftdv_gwlb_cluster_parameters.json with the required parameters.

**Step 2** Log in to the Azure Portal: https://portal.azure.com.

**Step 3** Create a Resource Group.

a) In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.

b) Choose the required **Region**.

**Step 4** Create a virtual network with three subnets: Management, Data, and Cluster Control Link (CCL).

a) Create the virtual network.

1. In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.
2. Choose the required **Region**. Click **Next: IP addresses**.

In the **IP Addresses** tab, click **Add subnet** and add the following subnets – Management, Data, and Cluster Control Link.

b) Add the subnets.

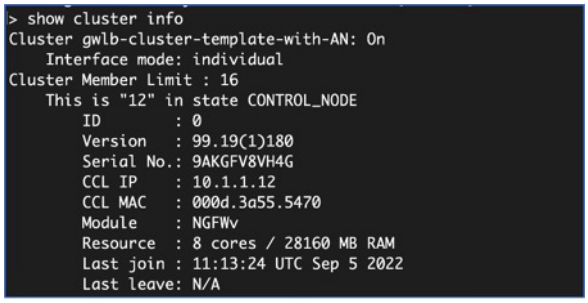
**Step 5** Deploy the custom template.

a) Click **Create** > **Template deployment (deploy using custom templates)**.
b) Click **Build your own template in the editor**.
c) Click **Load File**, and upload **azure_ftdv_gwlb_cluster.json**.
d) Click **Save**.

**Step 6** Configure the Instance details.

a) Enter the required values and then click **Review + create**.
b) Click **Create** after the validation is passed.
c) After the instance is running, verify the cluster deployment by logging into any one of the nodes and entering the **show cluster info** command.

***Figure 10: Show cluster info***




**Step 7** In the Azure Portal, click the Function app to register the cluster with the Firewall Management Center.

**Note** If you do not want to use the Function app, you can alternatively register the control node to the Firewall Management Center directly by using **Add** > **Device** (not **Add** > **Cluster**). The rest of the cluster nodes will register automatically.

**Step 8** Create FTPS Credentials by clicking **Deployment Center** > **FTPS credentials** > **User scope** > **Configure Username and Password**, and then click **Save**.

**Step 9** Upload the Cluster_Function.zip file to the Function app by executing the following **curl** command in the local terminal.

**curl -X POST -u** *username* **--data-binary @"Cluster_Function.zip" https://** *Function_App_Name***.scm.azurewebsites.net/api/zipdeploy**

**Note** The **curl** command might take few minutes (~2 to 3 minutes) to complete command execution.

The function will be uploaded to the Function app. The function will start, and you can see the logs in the storage account's outqueue. The device registration with the Management Center will be initiated.

*Figure 11: Functions*

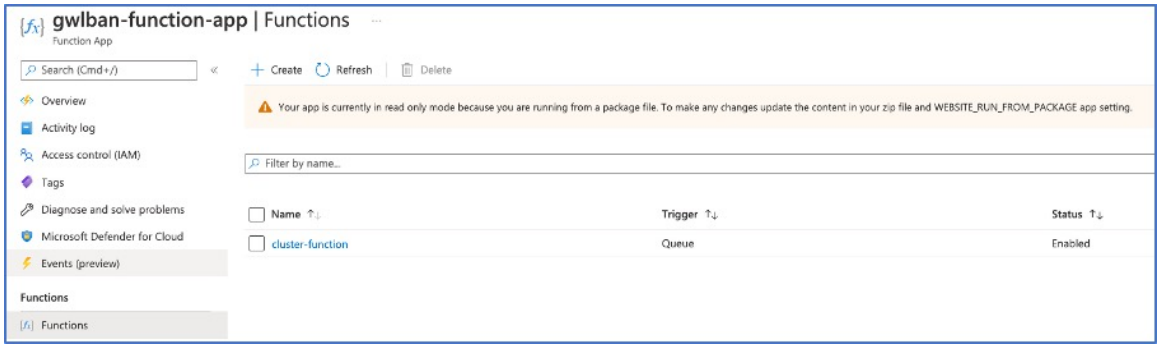



*Figure 12: Queues*

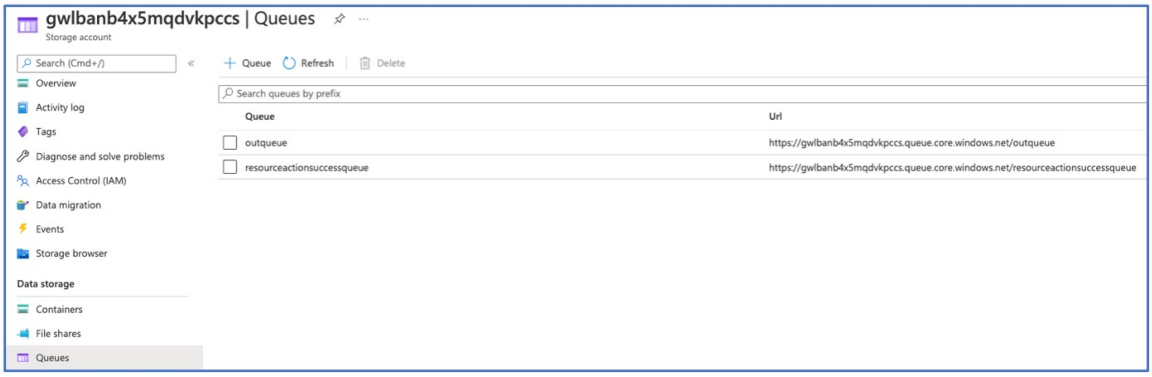
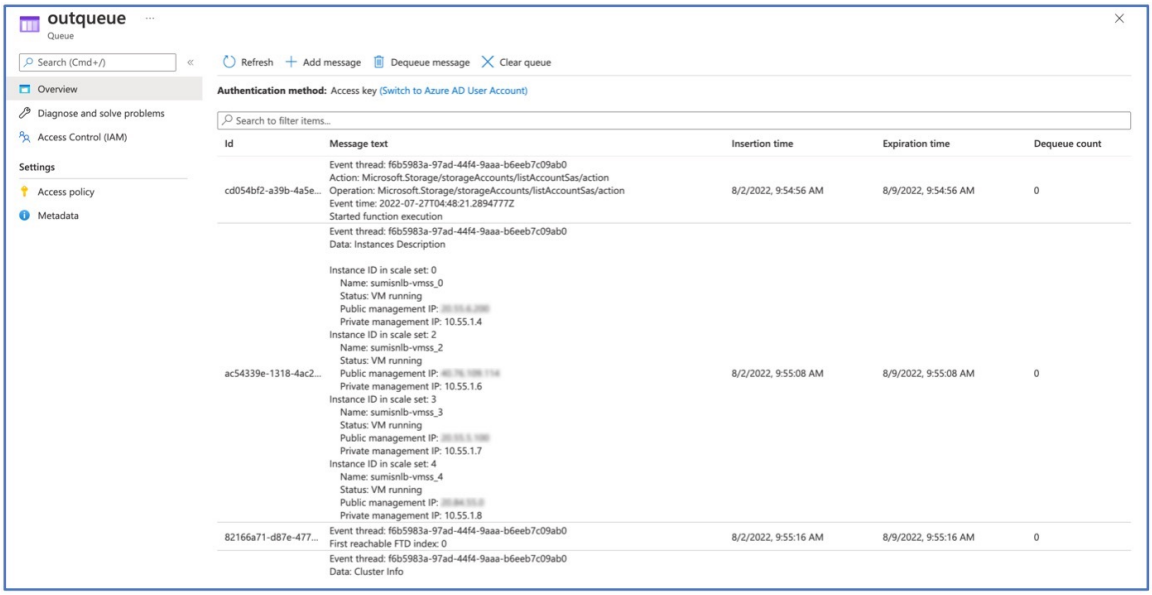



*Figure 13: Outqueue*



# Sample topology for NLB-based cluster deployment

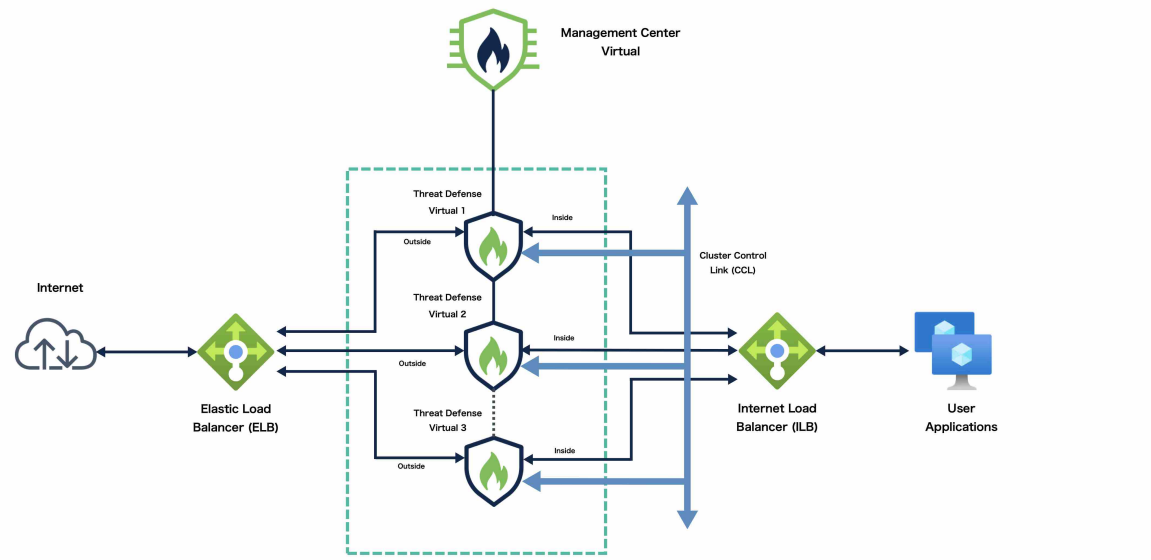
The sample topology provided here

- places the Threat Defense Virtual cluster between internal and external load balancers,
- uses a Management Center Virtual instance to manage the cluster, and
- depicts both inbound and outbound traffic flow patterns.

### Traffic flow patterns

This topology shows how traffic moves through the network components:

- **Inbound traffic**: Traffic from the internet goes to the external load balancer, which transmits the traffic to the Threat Defense Virtual cluster. After the traffic has been inspected by a Threat Defense Virtual instance in the cluster, it is forwarded to the application virtual machine (VM).
- **Outbound traffic**: Traffic from the application VM is transmitted to the internal load balancer. Traffic is then forwarded to the Threat Defense Virtual cluster and then sent out to the internet.

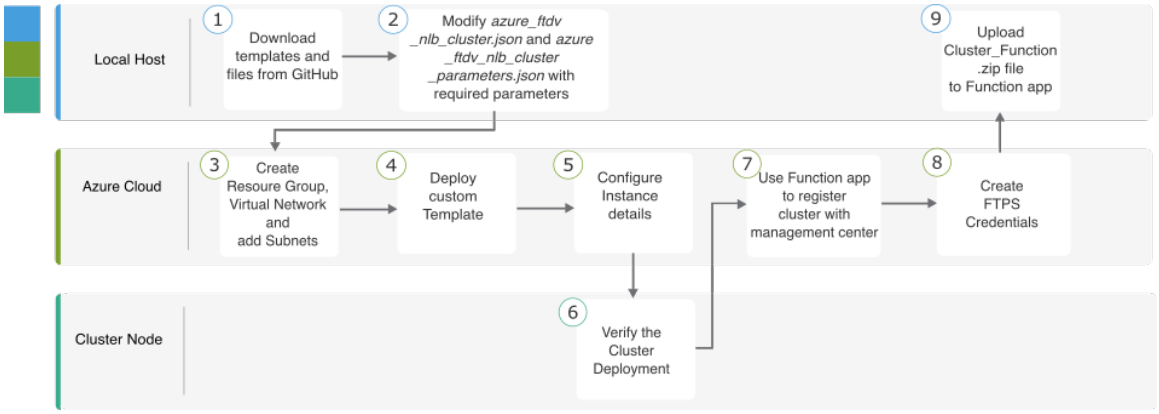



# End-to-end process for deploying Threat Defense Virtual cluster in Azure with NLB

Provides a workflow to help users successfully deploy and verify a Threat Defense Virtual cluster on Azure with NLB using either template-based or manual deployment processes.

### Template-based Deployment

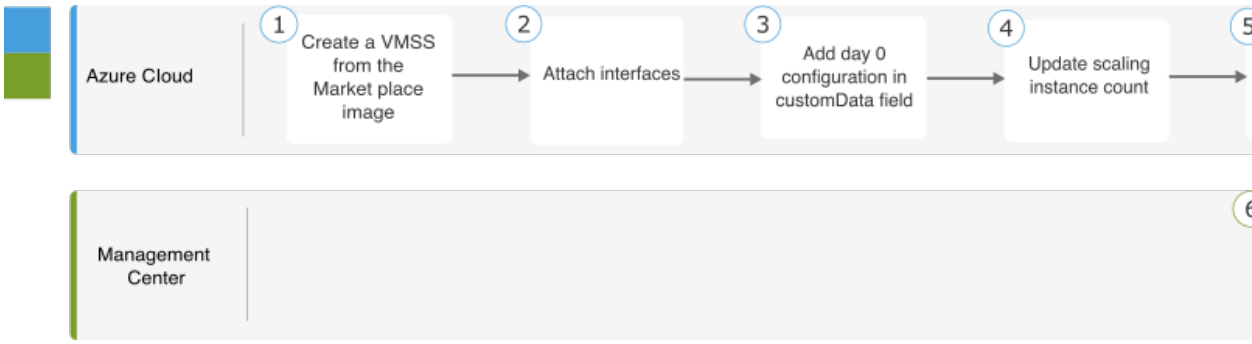
The following flowchart illustrates the workflow of template-based deployment of Threat Defense Virtual cluster in Azure with NLB.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Download templates and files from GitHub. |
| 2 | Local Host | Modify *azure_ftdv_nlb_cluster.json* and *azure_ftdv_nlb_cluster_parameters.json* with the required parameters. |
| 3 | Azure Cloud | Create the resource group, virtual network, and subnets. |
| 4 | Azure Cloud | Deploy custom template. |
| 5 | Azure Cloud | Configure instance details. |
| 6 | Cluster Node | Verify cluster deployment. |
| 7 | Azure Cloud | Use the Function app to register the cluster with the Management Center. |
| 8 | Azure Cloud | Create FTPS credentials. |
| 9 | Local Host | Upload *Cluster_Function.zip* file to the Function app. |

### Manual Deployment

The following flowchart illustrates the workflow of manual deployment of Threat Defense Virtual cluster in Azure with NLB.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Create a VMSS from the Marketplace image. |
| 2 | Local Host | Attach interfaces. |
| 3 | Local Host | Add day 0 configuration in the customData field. |
| 4 | Local Host | Update scaling instance count. |
| 5 | Local Host | Configure NLB. |
| 6 | Management Center | Add control node. |

# Templates

The templates are available in GitHub. The parameter values are self-explanatory with the parameter names and values given in the template.

These templates provide the necessary components for cluster deployment:

- azure_ftdv_nlb_cluster_parameters.json – Template to enter parameters for the Firewall Threat Defense Virtual cluster.
- azure_ftdv_nlb_cluster.json – Template to deploy Firewall Threat Defense Virtual cluster.

# Prerequisites for Azure NLB cluster deployment

Before deploying an Azure Network Load Balancer cluster, ensure that you complete these prerequisite tasks.

- To allow the cluster to auto-register with the Management Center, create a user with Network Admin & Maintenance User privileges on the Management Center. Users with these privileges can use REST API. Refer to the Cisco Secure Firewall Management Center Administration Guide.
- Add an access policy in the Management Center that matches the name of the policy that you will specify during template deployment.

- Ensure that the Management Center Virtual is licensed appropriately.
- After the cluster is added to the Management Center Virtual, complete these configuration steps:

1. Configure platform settings with the health check port number in the Management Center. For more information on configuring this, refer to Platform Settings.
2. Create static routes for traffic from outside and inside interfaces. For more information on creating a static route, refer to Add a Static Route.



**Important** Configure static routes for both interface gateway paths when Azure Load Balancer health probes are enabled on both inside and outside interfaces of FTDv cluster nodes. Do not rely on a single default route. Azure health probes may arrive on either interface, so FTDv must respond using the correct interface path. Missing routing configuration can cause probe drops and load balancer health check failures. If SSH on port 22 is used as the health probe and SSH is enabled on both interfaces through Platform Settings, verify return path routing for both interfaces. Similarly, if HTTP is used as the health probe and HTTP or HTTPS is enabled on both interfaces through Platform Settings, verify return path routing for both interfaces.



**Note** Ensure that you do not configure your virtual network using the Microsoft-owned IP address range that includes 168.63.129.16. This configuration causes a conflict with the health probe's IP address. For more information, refer to Microsoft Azure design guidance.

Sample static route configuration for the outside interface:

```
Network: any-ipv4
Interface: outside
Leaked from Virtual Router: Global
Gateway: ftdv-cluster-outside
Tunneled: false
Metric: 10
```



**Note** *ftdv-cluster-outside* is the outside subnet's gateway IP address.

Sample static route configuration for the inside interface:

```
Network: any-ipv4
Interface: inside
Leaked from Virtual Router: Global
Gateway: ftdv-cluster-inside-gw
Tunneled: false
Metric: 11
```

**Note** *ftdv-cluster-inside-gw* is the inside subnet's gateway IP address.

3. Configure NAT rule for data traffic. For more information on configuring NAT rules, refer to Network Address Translation.

# Deploy cluster on Azure with NLB using an Azure resource manager template

This task enables you to deploy a cluster for Azure NLB using the customized Azure Resource Manager (ARM) template for scalable network security.

Deploy the cluster for Azure NLB using the customized Azure Resource Manager (ARM) template. Note that the templates mentioned in the steps below are available on GitHub.

## Procedure

**Step 1** Prepare the template.

a) Clone the GitHub repository to your local folder. See https://github.com/CiscoDevNet/cisco-ftdv/tree/master/cluster/azure.

b) Modify azure_ftdv_nlb_cluster.json and azure_ftdv_nlb_cluster_parameters.json with the required parameters.

**Step 2** Log in to the Azure Portal: https://portal.azure.com.

**Step 3** Create a Resource Group.

a) In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.

b) Choose the required **Region**.

**Step 4** Create a virtual network with 5 subnets: Management, Diagnostic, Inside, Outside, and Cluster Control Link.

a) Create the virtual network.

1. In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.
2. Choose the required **Region**. Click **Next: IP addresses**.

b) Add the subnets.

In the **IP Addresses** tab, click **Add subnet** and add the following subnets – Management, Diagnostic, Inside, Outside, and Cluster Control Link.

**Step 5** Deploy the custom template.

a) Click **Create** > **Template deployment (deploy using custom templates)**.

b) Click **Build your own template in the editor**.

c) Click **Load File**, and upload **azure_ftdv_nlb_cluster.json**.

d) Click **Save**.

**Step 6** Configure the instance details.

a) Enter the required values and then click **Review + create**.

**Note** For the cluster control link starting and ending addresses, specify only as many addresses as you need (up to 16). A larger range can affect performance.

b) Click **Create** after the validation is passed.

c) After the instance is running, verify the cluster deployment by logging into any one of the nodes and using the **show cluster info** command.

***Figure 14: Show cluster info***

```
> show cluster info
Cluster gwlb-cluster-template-with-AN: On
    Interface mode: individual
Cluster Member Limit : 16
    This is "12" in state CONTROL_NODE
        ID          : 0
        Version     : 99.19(1)180
        Serial No.: 9AKGFV8VH4G
        CCL IP      : 10.1.1.12
        CCL MAC     : 000d.3a55.5470
        Module      : NGFWv
        Resource    : 8 cores / 28160 MB RAM
        Last join : 11:13:24 UTC Sep 5 2022
        Last leave: N/A
```

**Step 7** In the Azure Portal, click the Function app to register the cluster to the Firewall Management Center.

**Note** If you do not want to use the Function app, you can alternatively register the control node with the Management Center directly by using **Add** > **Device** (not **Add** > **Cluster**). The rest of the cluster nodes will register automatically.

**Step 8** Create FTPS Credentials by clicking **Deployment Center** > **FTPS credentials** > **User scope** > **Configure Username and Password**, and then click **Save**.

**Step 9** Upload the Cluster_Function.zip file to the Function app by executing the **curl** command in the local terminal.

**curl -X POST -u** *username* **--data-binary @"Cluster_Function.zip" https://** *Function_App_Name***.scm.azurewebsites.net/api/zipdeploy**

**Note** The **curl** command might take a few minutes (~2 to 3 minutes) to complete command execution.

The function will be uploaded to the Function app. The function will start, and you can see the logs in the storage account's outqueue. The device registration with the Management Center will be initiated.

# Manual cluster deployment in Azure

To deploy the cluster manually, prepare the day0 configuration, deploy each node, and then add the control node to the Firewall Management Center .

# Create the Day0 configuration for Azure

You can use either a fixed configuration or a customized configuration.

## Create the Day0 configuration with a fixed configuration for Azure

The fixed configuration will auto-generate the cluster bootstrap configuration.

```
"Cluster": {
"CclSubnetRange": "ip_address_start ip_address_end",
"ClusterGroupName": "cluster_name",
"HealthProbePort": "port_number",
"GatewayLoadBalancerIP": "ip_address",
"EncapsulationType": "vxlan",
"InternalPort": "internal_port_number",
"ExternalPort": "external_port_number",
"InternalSegId": "internal_segment_id",
"ExternalSegId": "external_segment_id"
}
```

### Example

A sample day 0 configuration is given below.

```
"Cluster": {
"CclSubnetRange": "10.45.3.4 10.45.3.30",     //mandatory user input
"ClusterGroupName": "ngfwv-cluster",          //mandatory user input
"HealthProbePort": "7777",                    //mandatory user input
"GatewayLoadBalancerIP": "10.45.2.4",         //mandatory user input
"EncapsulationType": "vxlan",
"InternalPort": "2000",
"ExternalPort": "2001",
"InternalSegId": "800",
"ExternalSegId": "801"
}
```

**Note** If you are copying and pasting the configuration given above, ensure that you remove **//mandatory user input** from the configuration

For the Azure health check settings, be sure to specify the **HealthProbePort** you set here.

For the **CclSubnetRange** variable, specify a range of IP addresses starting from x.x.x.4. Ensure that you have at least 16 available IP addresses for clustering. Some examples of start and end IP addresses are given below.

**Note** All cluster infrastructure subnets must use /27 CIDR

***Table 3: Examples of start and end IP addresses***

| CIDR | Start IP Address | End IP Address |
|---|---|---|
| 10.1.1.0/27 | 10.1.1.4 | 10.1.1.30 |

| CIDR | Start IP Address | End IP Address |
|---|---|---|
| 10.1.1.32/27 | 10.1.1.36 | 10.1.1.62 |
| 10.1.1.64/27 | 10.1.1.68 | 10.1.1.94 |
| 10.1.1.96/27 | 10.1.1.100 | 10.1.1.126 |
| 10.1.1.128/27 | 10.1.1.132 | 10.1.1.158 |
| 10.1.1.160/27 | 10.1.1.164 | 10.1.1.190 |
| 10.1.1.192/27 | 10.1.1.196 | 10.1.1.222 |
| 10.1.1.224/27 | 10.1.1.228 | 10.1.1.254 |

### Create the Day0 configuration with a customized configuration for azure

You can enter the entire cluster bootstrap configuration using commands.

```
"Cluster": {
"CclSubnetRange": "ip_address_start ip_address_end",
"ClusterGroupName": "cluster_name",
"HealthProbePort": "port_number",
"GatewayLoadBalancerIP": "ip_address",
"EncapsulationType": "vxlan",
"InternalPort": "internal_port_number",
"ExternalPort": "external_port_number",
"InternalSegId": "internal_segment_id",
"ExternalSegId": "external_segment_id"
}
```

## Deploy cluster nodes manually - GWLB-based deployment

This task allows you to deploy cluster nodes so they form a cluster in Azure using Gateway Load Balancer-based deployment.

Deploy the cluster nodes so they form a cluster.

### Procedure

**Step 1** Log in to the Azure Portal: https://portal.azure.com

**Step 2** Create a Resource Group.

**a.** In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.

**b.** Choose the required **Region**.

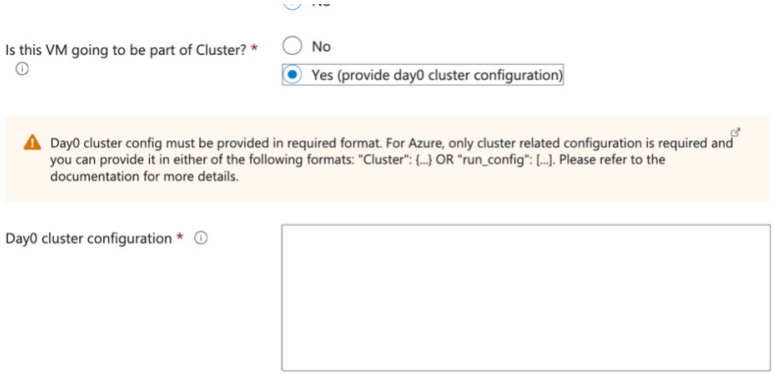
**Step 3** Create a Virtual Network with the necessary subnets: Management, Data and Cluster Control Link (CCL).

**Note** Configure the CCL with the smallest subnet mask as required. Wider subnets can impact performance.

Refer to the Azure document for creating the Virtual Network and subnet: https://learn.microsoft.com/en-us/azure/virtual-network/quickstart-create-virtual-network?tabs=portal

**Step 4** Go to the Marketplace and search for **Cisco Secure Firewall Threat Defense Virtual – BYOL and PAYG** and click **Create**.

**Step 5** Fill the required details and choose **Yes** for **Is this VM going to be part of Cluster?**




Paste the following cluster-related configuration in the text box.

```
"Cluster": {
"CclSubnetRange": "ip_address_start ip_address_end", //mandatory user input
"ClusterGroupName": "cluster_name", //mandatory user input
"HealthProbePort": "port_number", //mandatory user input
"GatewayLoadBalancerIP": "ip_address",  //mandatory user input
"EncapsulationType": "vxlan",
"InternalPort": "internal_port_number",
"ExternalPort": "external_port_number",
"InternalSegId": "internal_segment_id",
"ExternalSegId": "external_segment_id"
}
```

a) Click **Next** and select the **Virtual Network & Subnets**.

b) Click **Review + create**. Wait for the Threat Defense Virtual deployment to complete.

**Step 6** Connect to the Threat Defense Virtual device and use the **show cluster info** command to confirm the cluster formation is successful.

```
> show cluster info
Cluster ngfwv-cluster: On
    Interface mode: individual
Cluster Member Limit : 16
    This is "4" in state CONTROL_NODE
        ID        : 0
        Version   : 9.23(1)
        Serial No.: 9AC1VMGJKAQ
        CCL IP    : 169.254.200.4
        CCL MAC   : 6045.bda8.e07b
        Module    : NGFWv
        Resource  : 4 cores / 14336 MB RAM
        Last join : 05:22:55 UTC Jul 14 2025
        Last leave: N/A
```

```
Other members in the cluster:
    There is no other unit in the cluster
>
```

**Step 7** Configure the Azure Gateway Load Balancer. Refer to Auto Scale with Azure Gateway Load Balancer Use Case for more information.

**Step 8** Add the control node to the Firewall Management Center. Refer to Add the cluster to the Management Center (manual deployment), on page 106.

## Deploy cluster nodes manually - NLB-based deployment

Deploy cluster nodes manually to establish a functional cluster configuration in Azure using NLB-based deployment.

Deploy the cluster nodes so they form a cluster.

### Procedure

**Step 1** Log in to the Azure Portal: https://portal.azure.com

**Step 2** Create a Resource Group.

**a.** In the **Basics** tab, choose the **Subscription** and **Resource Group** from the drop-down lists.

**b.** Choose the required **Region**.

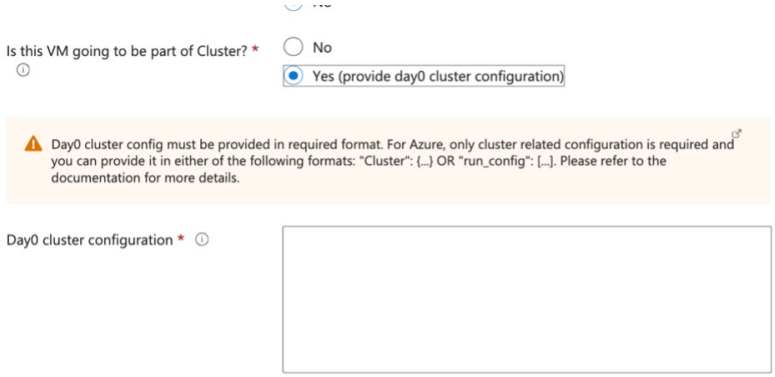
**Step 3** Create a Virtual Network with the necessary subnets: Management, Inside, Outside and Cluster Control Link (CCL).

**Note** Configure the CCL with the smallest subnet mask as required. Wider subnets can impact performance.

Refer to the Azure document for creating the Virtual Network and subnet: https://learn.microsoft.com/en-us/azure/virtual-network/quickstart-create-virtual-network?tabs=portal

**Step 4** Go to the Marketplace and search for **Cisco Secure Firewall Threat Defense Virtual – BYOL and PAYG** and click **Create**.

**Step 5** Fill the required details and choose **Yes** for **Is this VM going to be part of Cluster?**




Paste the following cluster-related configuration in the text box.

```
"Cluster": {
"CclSubnetRange": "ip_address_start ip_address_end", //mandatory user input
"ClusterGroupName": "cluster_name" //mandatory user input
}
```

**Step 6** Click **Next** and select the **Virtual Network & Subnets**.

**Step 7** Click **Review + create**. Wait until the Threat Defense Virtual deployment is completed.

**Step 8** Connect to the Threat Defense Virtual device and use the **show cluster info** command to confirm the cluster formation is successful.

```
> show cluster info
Cluster ngfwv-cluster: On
    Interface mode: individual
Cluster Member Limit : 16
    This is "4" in state CONTROL_NODE
        ID          : 0
        Version     : 9.23(1)
        Serial No.: 9AC1VMGJKAQ
        CCL IP      : 169.254.200.4
        CCL MAC     : 6045.bda8.e07b
        Module      : NGFWv
        Resource    : 4 cores / 14336 MB RAM
        Last join : 05:22:55 UTC Jul 14 2025
        Last leave: N/A
Other members in the cluster:
    There is no other unit in the cluster
>
```

**Step 9** Add the control node to the Management Center. Refer to Add the cluster to the Management Center (manual deployment), on page 106.

# Troubleshoot cluster deployment in Azure

This reference provides troubleshooting guidance for resolving common issues that may occur during cluster deployment in Azure, including traffic flow problems, cluster formation failures, and role assignment errors.

- Issue: No traffic flow

  Troubleshooting:

- Check if the health probe status of the Threat Defense Virtual instances deployed with a GWLB is healthy.
- If the Threat Defense Virtual instance's health probe status is unhealthy-
  - Check if the static route is configured in the Management Center Virtual.
  - Check if the default gateway is the data subnet's gateway IP address.
  - Check if the Threat Defense Virtual instance is receiving health probe traffic.
  - Check if the access list configured in the Management Center Virtual allows health probe traffic.

- Issue: Cluster is not formed

  Troubleshooting:

  - Check the IP address of the NVE-only cluster interface. Ensure that you can ping the NVE-only cluster interface of other nodes.
  - Check the IP address of the NVE-only cluster interfaces are part of the object group.
  - Ensure that the NVE interface is configured with the object group .
  - Ensure that the cluster interface in the cluster group has the right VNI interface. This VNI interface has the NVE with the corresponding object group.
  - Ensure that the nodes are pingable from each other. Since each node has its own cluster interface IP, these should be pingable from each other.
  - Check if the CCL subnet's start and end address mentioned during template deployment is correct. The start address should begin with the first available IP address in the subnet. For example, if the subnet is 192.168.1.0/27. The start address should be 192.168.1.4 (the three IP addresses at the start are reserved by Azure).
  - Check if the Management Center Virtual has a valid license.

- Issue: Role-related error while deploying resources again in the same resource group.

  Troubleshooting: Remove the roles given below by using these commands on the terminal.

  Error message:

```
"error": {
"code": "RoleAssignmentUpdateNotPermitted",
"message": "Tenant ID, application ID, principal ID, and scope are not allowed to be
updated."}
```

  - **az role assignment delete --resource-group** <*Resource Group Name*> **--role "Storage Queue Data Contributor"**
  - **az role assignment delete --resource-group** <*Resource Group Name*> **--role "Contributor"**

# Firewall Threat Defense Virtual clustering autoscale solution in Azure

A typical cluster deployment in an Azure region includes a defined number of Firewall Threat Defense Virtual instances (nodes). When the Azure region traffic varies, without dynamic scaling (autoscale) of the nodes, resource utilization in such cluster arrangement may underutilize the resources or cause latency. Cisco offers an autoscale solution for Firewall Threat Defense Virtual clustering in Version 7.7 and later that supports dynamic scaling of nodes in the Azure region. It allows you to scale in or scale out nodes from the cluster based on the network traffic. It uses logic based on the resource utilization statistics from Azure VMSS metrics such as CPU and memory metrics, to dynamically add or remove a node from a cluster.

### Autoscale solution features

The Firewall Threat Defense Virtual clustering with autoscale solution in Azure supports both Network Load Balancer (NLB or Sandwich topology) and Gateway Load Balancer (GWLB).

Cisco provides separate Azure Resource Manager (ARM) templates for deploying Firewall Threat Defense Virtual cluster with autoscale in Azure using NLB and GWLB, as well as infrastructure and configuration templates for deploying the Azure services such as Function App and Logic App.

## Sample topologies

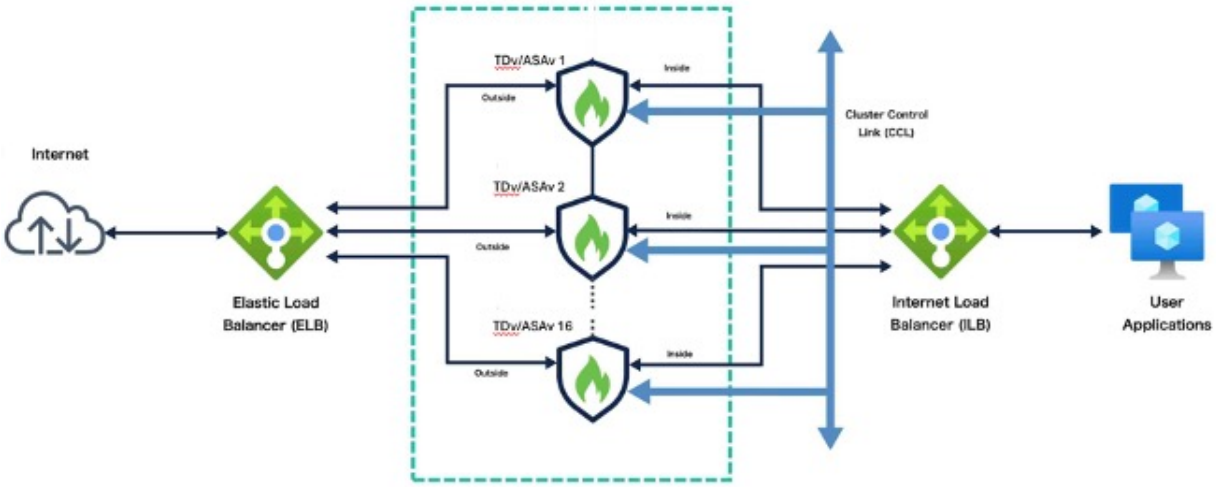
The sample topologies provided here

- demonstrate how Firewall Threat Defense Virtual clustering with autoscale functions in Azure environments
- provide reference implementations for different load balancing configurations, and
- show automated horizontal scaling solutions for network security deployment.

### Firewall Threat Defense Virtual clustering with autoscale using sandwich topology (Network Load Balancer)

The Firewall Threat Defense Virtual clustering with autoscale in Azure using sandwich topology (NLB) use case is an automated horizontal scaling solution that positions the Firewall Threat Defense Virtual scale set sandwiched between an Azure Internal load balancer (ILB) and an Azure External load balancer (ELB).

In this topology, the Firewall Threat Defense Virtual uses only four interfaces: management, inside, outside, and CCL subnets.




This describes high-level flow on how a Firewall Threat Defense Virtual cluster with autoscale in Azure using NLB functions:

- The ELB distributes traffic from the internet to the Firewall Threat Defense Virtual instances in the scale set, and then the firewall forwards traffic to the application.
- The ILB distributes outbound internet traffic from an application to Firewall Threat Defense Virtual instances in the scale set and then the firewall forwards traffic to the internet.
- A network packet will never pass through both internal and external load balancers in a single connection.
- The number of Firewall Threat Defense Virtual instances in the scale set will be scaled and configured automatically based on load conditions.

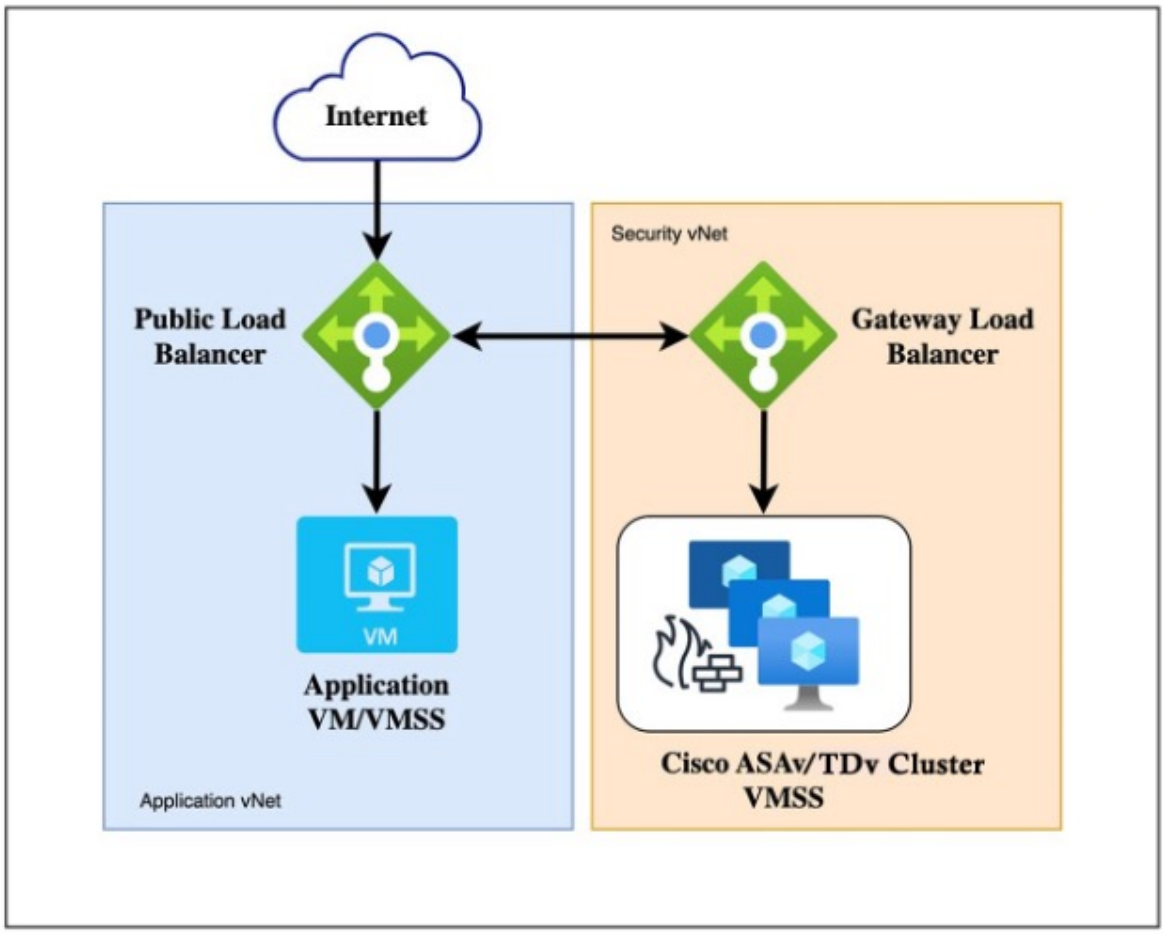
Firewall Threat Defense Virtual clustering with autoscale using Gateway Load Balancer

The integration of the Azure Gateway Load Balancer (GWLB) and Firewall Threat Defense Virtual cluster using autoscale solution simplifies deployment, management, and scaling of instances in the cluster setup. The Azure Gateway Load Balancer (GWLB) ensures that internet traffic to and from an Azure VM, such as an application server, is inspected by secure firewall without requiring any routing changes. This integration also reduces operational complexity and provides a single entry and exit point for traffic at the firewall. The applications and infrastructure can maintain visibility of source IP address, which is critical in some environments.

The Firewall Threat Defense Virtual uses only three interfaces: management, data, and CCL interface in this use case.

**Note**

- Network Address Translation (NAT) is not required if you are deploying the Azure GWLB.
- Only IPv4 is supported.




This describes high-level flow on how a Firewall Threat Defense Virtual cluster with autoscale in Azure using GWLB functions:

- Inbound traffic from the internet goes to the GWLB endpoint, which then transmits the traffic to the GWLB.
- The traffic is then routed to the Firewall Threat Defense Virtual cluster.
- After the traffic is inspected by the Firewall Threat Defense Virtual instance in the cluster, it is forwarded to the application VM.

# Prerequisites for Threat Defense Virtual clustering autoscale solution on Azure

Ensure that you have the Owner role in the Azure subscription.

Create the Azure Resource Group. Ensure that the Azure Virtual Network and the necessary subnets are created.

- Interfaces for the NLB-based cluster: Management, Diagnostic, Inside, Outside, CCL and the function app.

- Interfaces for the GWLB-based cluster: Management, Diagnostic, Data, CCL and the function app.

Complete the following Management Center configuration tasks:

- Ensure that Management Center Virtual is licensed correctly.
- Create the access control policy.
- Create the Security Zone (SZ) object for the interfaces. For the NLB-based cluster, create the Security Zone for the inside and outside interfaces. For the GWLB-based cluster, create the Security Zone for the data interface.
- Create a separate user name and password for the Azure function to add the Threat Defense Virtual instances to the Management Center Virtual and configure the instances.

Install the Azure CLI on your local system.

Download the Azure Clustering Autoscale repository from GitHub to your local computer and run the command **python3 make.py build** to create the Azure functions zip file.

# Autoscale logic for Firewall Threat Defense Virtual clustering in Azure

### Scaling Policy

In a cluster with autoscale, the scaling of nodes is determined based on the following policies:

- Scaling policy 1 - When one cluster node exceeds the resource utilization limits.
- Scaling policy 2 - Overall average resource utilization of all the nodes.

### Scale-out

Scale-out adds a new node to the cluster when the traffic load threshold exceeds the configured CPU or memory limit on any cluster node.

The following is the process of adding a new node to the cluster during scale-out:

1. A new Firewall Threat Defense Virtual instance is launched.
2. Appropriate configuration is applied to a Firewall Threat Defense Virtual .
3. Appropriate licenses are applied.
4. A new Firewall Threat Defense Virtual instance is added to the cluster.

If the configuration of the new Firewall Threat Defense Virtual instance fails (low probability) during the scale-out process, the failing instance is terminated, and a new instance is launched and configured.

### Scale-in

Scale-in removes a node from the cluster when both the configured scale-in threshold and the total number of cluster instances exceed the minimum cluster size.

The following is the process of terminating a node in the cluster during scale-in:

1. The Firewall Threat Defense Virtual instance with the least CPU or memory usage is identified using VMSS metrics.

2. If there is more than one instance with the same least utilization, then the instance with the higher VM index in VMSS is chosen for scale-in.
3. Any new connections to this instance are disabled by appropriate configuration and policies.
4. The instance is de-registered from smart licensing (applicable for BYOL).
5. The instance is terminated.

### Azure Functions (Function App)

The Function application enables the Firewall Threat Defense Virtual cluster and registers it with the management center. It also helps you select a hosting plan for Firewall Threat Defense Virtual clustering with autoscale deployment.

The following two types of hosting plans are offered:

- Consumption
  - This is the default hosting plan for Firewall Threat Defense Virtual clustering with autoscale.
  - This plan allows the Function app to connect to the Firewall Threat Defense Virtual instances by opening the SSH port to the Azure data center IP addresses of the region.
- Premium
  - You can select this hosting plan for the Function app during deployment.
  - This plan supports adding a Network Address Translation (NAT) gateway to the Function app to control the outbound IP address of the Function app. This plan allows SSH access to Firewall Threat Defense Virtual instances only from a fixed IP address of the NAT gateway thereby offering enhanced security.

For more information about overview on auto scale solution components, refer to Auto Scale Solution Components in Cisco Secure Firewall Threat Defense Virtual Getting Started Guide.

# Deployment and infrastructure templates on GitHub

Cisco provides Azure Resource Manager (ARM) templates and scripts for deploying an auto-scaling group of Firewall Threat Defense Virtual cluster using several Azure services, including Function App, Logic App, and auto-scaling groups.

The autoscale solution for Firewall Threat Defense Virtual cluster is an ARM template-based deployment that provides:

- Completely automated Firewall Threat Defense Virtual instance registration and de-registration with the management center using the Function App.
- Automatin application of NAT policy, access control policy, and routes, to the scaled-out threat defense virtual instances.
- Support for GWLB and NLB load balancers.
- Works only with the management center; the device manager is not supported.

### Firewall Threat Defense Virtual Clustering with Autoscale Solution Templates

#### Azure Resource Manager (ARM) templates

Two sets of templates are provided for autoscale solutions based on the (NLB or GWLB) load balancer you are using in Azure for the cluster.

The following templates are available on GitHub:

- Autoscale solution template for Firewall Threat Defense Virtual clustering using NLB: `azure_ftdv_nlb_cluster.json.json` available in the folder `arm-templates`.
- Autoscale solution template for Firewall Threat Defense Virtual clustering using GWLB: `azure_ftdv_gwlb_cluster.json` available in the folder `arm-templates`.

#### Setting up Azure infrastructure and configuration

- Function app to enable cluster on Firewall Threat Defense Virtual instances: `cluster_functions.zip`.
- Logic App code for the Firewall Threat Defense Virtual deployment, scale-in and scale-out workflow: `logic_app.txt`.

# Input parameters

The following table defines the template parameters and provides an example. Once you decide on these values, you can use these parameters to create the Firewall Threat Defense Virtual when you deploy the Azure Resource Manager (ARM) template into your Azure subscription. In the clustering with autoscale solution with GWLB for Azure, networking infrastructure is also created due to which additional input parameters have to be configured in the template. The parameter descriptions are self-explanatory.

***Table 4: Template Parameters***

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| resourceNamePrefix | String* (3-10 characters) | All the resources are created with name containing this prefix.<br>Note: Use only lowercase letters.<br>Example: ftdv | New |
| virtualNetworkRg | String | The virtual network resource group name.<br>Example: cisco-virtualnet-rg | Existing |
| virtualNetworkName | String | The virtual network name (already created).<br>Example: cisco-virtualnet | Existing |
| virtualNetworkCidr | CIDR format<br>x.x.x.x/y | CIDR of Virtual Network (already created) | Existing |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| mgmtSubnet | String | The management subnet name (already created).<br>Example: cisco-mgmt-subnet | Existing |
| dataSubnet | String | The data subnet name (already created)<br>Example: cisco-data-subnet | |
| cclSubnet | String | The cluster control link subnet name.<br>Example: cisco-ccl-subnet | |
| cclSubnetStartAddr | String | The starting range of CCL subnet IP address.<br>Example: 3.4.5.6 | |
| cclSubnetEndAddr | String | The ending range of CCL subnet IP address.<br>Example: 5.6.7.8 | |
| gwlbIP | String | GWLB is created in existing data subnet.<br>Example: 10.0.2.4 | |
| dataNetworkGatewayIp | String | The gateway IP address of the data subnet.<br>Example: 10.0.2.7 | |
| outsideSecurityZoneName | String | The security zone object Name created in the management center<br>Example: outside-sz | |
| TDvmManagementUserName | String | TDv management administrator username.<br>You are not allowed provide 'admin' as the username. | |
| diagSubnet | String | The diagnostic subnet name (already created).<br>Example: cisco-diag-subnet | Existing |
| insideSubnet | String | The inside Subnet name (already created).<br>Example: cisco-inside-subnet | Existing |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| internalLbIp | String | The internal load balancer IP address for the inside subnet (already created).<br>Example: 1.2.3.4 | Existing |
| insideNetworkGatewayIp | String | The inside subnet gateway IP address (already created). | Existing |
| outsideSubnet | String | The outside subnet name (already created).<br>Example: cisco-outside-subnet | Existing |
| outsideNetworkGatewayIp | String | The outside subnet gateway IP (already created). | Existing |
| deviceGroupName | String | Device group in Firewall Management Center (already created) | Existing |
| insideZoneName | String | Inside Zone name in the Firewall Management Center (already created) | Existing |
| outsideZoneName | String | Outside Zone name in the Firewall Management Center (already created) | Existing |
| softwareVersion | String | The Firewall Threat Defense Virtual Version (selected from drop-down list during deployment). | Existing |
| vmSize | String | Size of Firewall Threat Defense Virtual instance (selected from drop-down list during deployment). | N/A |
| ftdLicensingSku | String | Firewall Threat Defense Virtual Licensing Mode (PAYG/BYOL)<br>Note: PAYG is supported in Version 6.5+. | N/A |
| licenseCapability | Comma-separated string | BASE, MALWARE, URLFilter, THREAT | N/A |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| tdVmManagementUserName | String* | The Firewall Threat Defense Virtual VM management administrator user name.<br>This cannot be 'admin'. See Azure for VM administrator user name guidelines. | New |
| tdVmManagementUserPassword | String* | Password for the Firewall Threat Defense Virtual VM management administrator user.<br>Passwords must be 12 to 72 characters long, and must have: lowercase, uppercase, numbers, and special characters; and must have no more than 2 repeating characters.<br>**Note**<br>There is no compliance check for this in the template. | New |
| ftdAdminUserPassword | String | Firewall Threat Defense Virtual Admin user password.<br>**Note**<br>The criteria mentioned for the TDvmManagementUserPassword parameter is applicable to this parameter also. | |
| fmcIpAddress | String<br>x.x.x.x | The public IP address of the Firewall Management Center (already created) | Existing |
| fmcUserName | String | Firewall Management Center user name, with administrative privileges (already created) | Existing |
| fmcPassword | String | Firewall Management Center password for above Firewall Management Center user name (already created) | Existing |
| policyName | String | Security Policy created in the Firewall Management Center (already created) | Existing |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| clusterGroupName | String | The name of the cluster group to be used while registering the threat defense device to the management center .<br>Example: tdv-cluster | |
| healthCheckPortNumber | String | The health check port number used while creating the health probe in the Gateway Load balancer.<br>Example: 8080 | |
| functionHostingPlan | String | Function deployment hosting plan (consumption uses the consumption hosting plan, premium: uses the premium hosting plan).<br>Default: consumption | |
| functionAppSubnet | String | The function app subnet name (already created).<br>Example: tdv-fapp-subnet | |
| functionAppSubnetCIDR | String | The CIDR of the function app subnet (already created).<br>Example: 10.0.4.0/27 | |
| scalingMetricsList | String | The metrics used in determining the scaling the scaling decision.<br>Allowed: CPU & MEMORY | |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| scalingPolicy | POLICY-1 / POLICY-2 | POLICY-1 : Scale-Out will be triggered when the average load of any Firewall Threat Defense Virtual goes beyond the Scale Out threshold for the configured duration.<br>POLICY-2 : Scale-Out will be triggered when average load of all the Firewall Threat Defense Virtual devices in the VMSS goes beyond the Scale Out threshold for the configured duration.<br>In both cases Scale-In logic remains the same: Scale-In will be triggered when average load of all the Firewall Threat Defense Virtual devices comes below the Scale In threshold for the configured duration. | N/A |
| scalingMetricsList | String | Metrics used in making the scaling decision.<br>Allowed: CPU , MEMORY<br>Default: CPU | N/A |
| cpuScaleInThreshold | String | The scale-in threshold in percentage for CPU metrics .<br>Default: 10<br>When the Firewall Threat Defense Virtual metric goes below this value the scale-in will be triggered.<br>See Autoscale logic for Firewall Threat Defense Virtual clustering in Azure, on page 62 . | N/A |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| cpuScaleOutThreshold | String | The Scale-out threshold in percentage for CPU metrics .<br>Default: 80<br>When the Firewall Threat Defense Virtual metric goes above this value, the Scale-Out will be triggered.<br>The ‘ cpuScaleOutThreshold ’ should always be greater than the ‘ cpuScaleInThreshold ’.<br>See Autoscale logic for Firewall Threat Defense Virtual clustering in Azure, on page 62 . | N/A |
| memoryScaleInThreshold | String | The Scale-In threshold in percent for memory metrics.<br>Default: 0<br>When the Firewall Threat Defense Virtual metric goes below this value the Scale-In will be triggered.<br>See Autoscale logic for Firewall Threat Defense Virtual clustering in Azure, on page 62 . | N/A |
| memoryScaleOutThreshold | String | The Scale-Out threshold in percent for memory metrics.<br>Default: 0<br>When the Firewall Threat Defense Virtual metric goes above this value, the Scale-Out will be triggered.<br>The ‘memoryScaleOutThreshold’ should always be greater than the ‘memoryScaleInThreshold’.<br>See Autoscale logic for Firewall Threat Defense Virtual clustering in Azure, on page 62 . | N/A |
| minFtdCount | Integer | The minimum Firewall Threat Defense Virtual instances available in the scale set at any given time.<br>Example: 2 | N/A |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| maxFtdCount | Integer | The maximum Firewall Threat Defense Virtual instances allowed in the Scale set.<br>Example: 10<br>**Note**<br>This number is restricted by the Firewall Management Center capacity.<br>The Auto Scale logic will not check the range of this variable, hence fill this carefully. | N/A |
| metricsAverageDuration | Integer | Select from the drop-down.<br>This number represents the time (in minutes) over which the metrics are averaged out.<br>If the value of this variable is 5 (i.e. 5min) when the Auto Scale Manager is scheduled, it will check the past 5 minutes average of metrics and based on this, it will make a scaling decision.<br>**Note**<br>Only numbers 1, 5, 15, and 30 are valid due to Azure limitations. | N/A |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| initDeploymentMode | BULK / STEP | Primarily applicable for the first deployment, or when the Scale Set does not contain any Firewall Threat Defense Virtual instances.<br><br>BULK: The Auto Scale Manager will try to deploy ' minFtdCount ' number of Firewall Threat Defense Virtual instances in parallel at one time.<br><br>**Note**<br>The launch is in parallel, but registering with the Firewall Management Center is sequential due to Firewall Management Center limitations.<br><br>STEP: The Auto Scale Manager will deploy the ' minFtdCount ' number of Firewall Threat Defense Virtual devices one by one at each scheduled interval.<br><br>**Note**<br>The STEP option will take a long time for the 'minFtdCount' number of instances to be launched and configured with the Firewall Management Center and become operational, but useful in debugging.<br><br>The BULK option takes same amount of time to launch all 'minFtdCount' number of Firewall Threat Defense Virtual as one Firewall Threat Defense Virtual launch takes (because it runs in parallel), but the Firewall Management Center registration is sequential.<br><br>The total time to deploy 'minFtdCount' number of Firewall Threat Defense Virtual = (time to launch One Firewall Threat Defense Virtual + time to register/configure one Firewall Threat Defense Virtual * minFtdCount ). | |

| Parameter Name | Allowed Values/Type | Description | Resource Creation Type |
|---|---|---|---|
| *Azure has restrictions on the naming convention for new resources. Review the limitations or simply use all lowercase letters. Do not use spaces or any other special characters . | | | |

# Firewall Threat Defense Virtual Cluster with autoscale deployment - process and resources

Firewall Threat Defense Virtual cluster with autoscale deployment process on Azure includes these steps:

- Deploy the ARM template.
- Build and deploy the clustering function.
- Update and enable the Logic application.

### Azure Resource Manager template deployment resources

The following resources are created within a resource group when you deploy Firewall Threat Defense Virtual cluster with autoscale in Azure using the ARM template for Sandwich Topology (NLB) - `azure_ftdv_nlb_cluster_autoscale.json`

- Virtual Machine Scale Set (VMSS)
- External load balancer
- Internal load balancer
- Azure Function App
- Logic App
- Security groups for data and management interfaces

The following resources are created within a resource group when you deploy Firewall Threat Defense Virtual cluster with autoscale in Azure using the ARM template for GWLB - `azure_ftdv_gwlb_cluster_autoscale.json`

- Virtual Machine (VM) or Virtual Machine Scale Set (VMSS)
- Gateway Load Balancer (GWLB)
- Azure Function App
- Logic App
- Networking Infrastructure
- Security groups and other miscellaneous components needed for deployment.

# Deploy the Firewall Threat Defense Virtual cluster with autoscale solution

This task deploys the Firewall Threat Defense Virtual clustering with autoscale solution on Azure to provide automated scaling capabilities for network security infrastructure using Azure Resource Manager templates.

Based on the topology—Sandwich (NLB) or Gateway Load Balancer (GWLB) use case—you must download and configure the appropriate ARM template to deploy the Firewall Threat Defense Virtual clustering with autoscale solution on Azure.

**Before you begin**

**Download the Deployment Package from GitHub**

The Firewall Threat Defense Virtual clustering autoscale with NLB solution for Azure is an Azure Resource Manager (ARM) template-based deployment which makes use of the serverless infrastructure provided by Azure (Logic App, Azure Functions, Load Balancers, Virtual Machine Scale Set, and so on).

The Firewall Threat Defense Virtual clustering autoscale with GWLB solution for Azure is an ARM template-based deployment that creates the GWLB, networking infrastructure, threat defense virtual auto scaling group, serverless components, and other required resources.

The deployment procedures for both solutions are similar.

Download the files required to launch the Firewall Threat Defense Virtual clustering with autoscale solution for Azure.

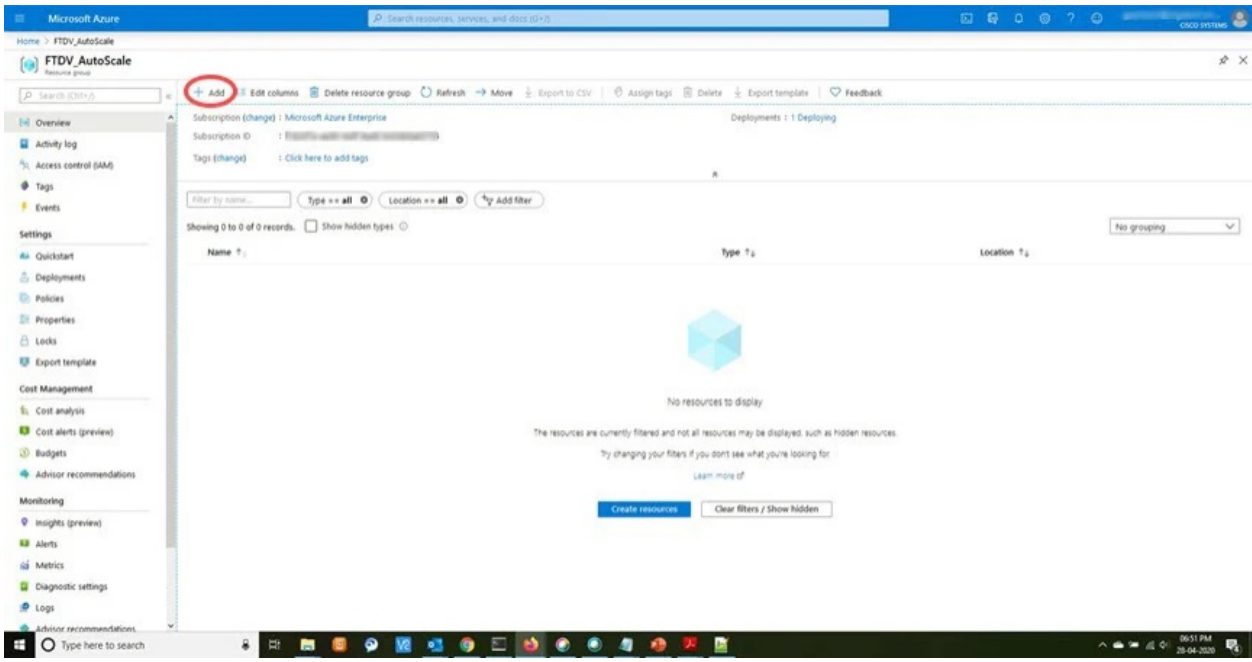
Deployment scripts and templates for your version are available in the GitHub repository.

Follow these steps to deploy the Firewall Threat Defense Virtual cluster with autoscale solution:

**Procedure**

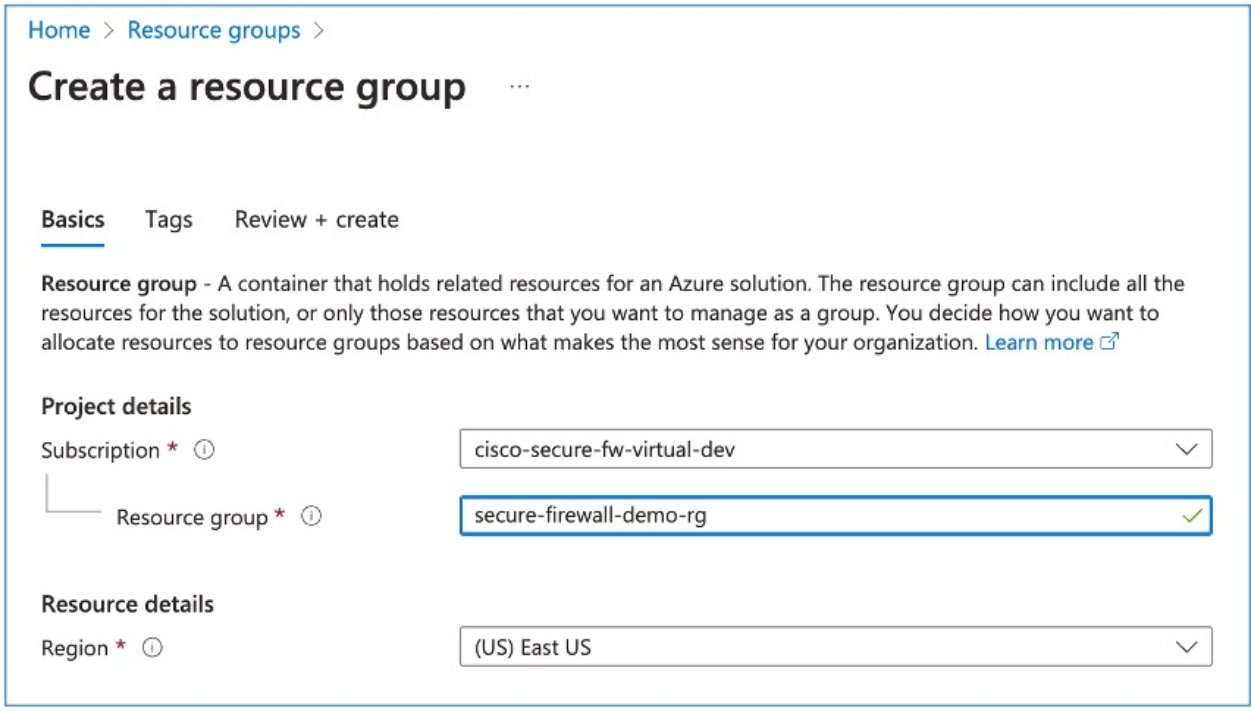
**Step 1** Log in to the Microsoft Azure portal (https://portal.azure.com) using your Microsoft account username and password.

**Step 2** Click **Resource groups** from the menu of services to access the **Resource Groups** blade. You will see all the resource groups in your subscription listed in the blade. Create a new resource group or select an existing, empty resource group. For example, **threat defense virtual_AutoScale**.




**Step 3** Click **Create a resource (+)** to create a new resource for template deployment.

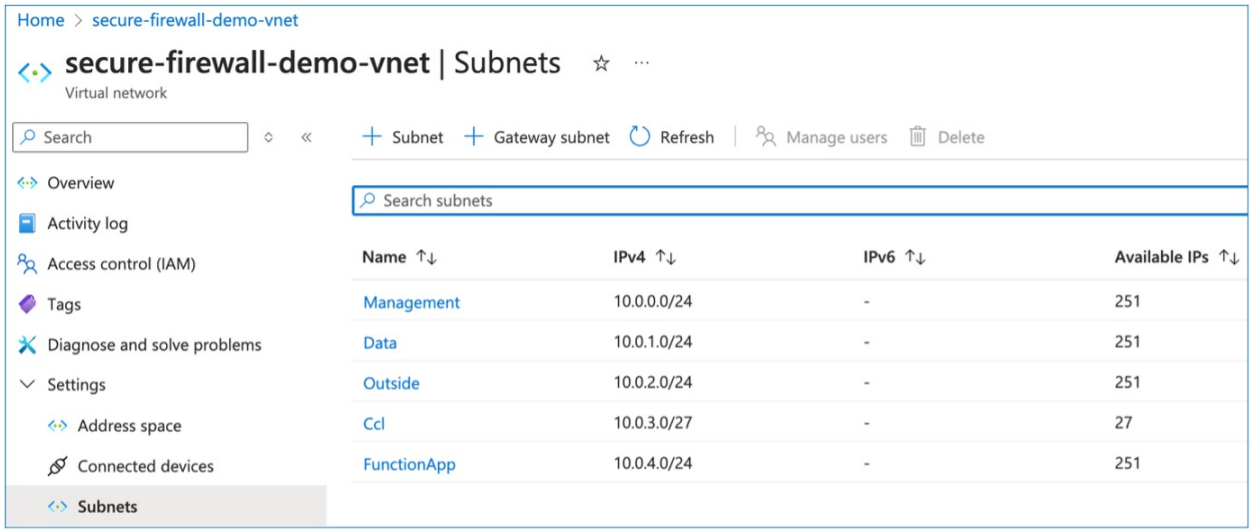
The **Create Resource Group** blade appears.




**Step 4** Click **Virtual Network** from the menu of services to access the Virtual network blade. Create a virtual network with subnets.

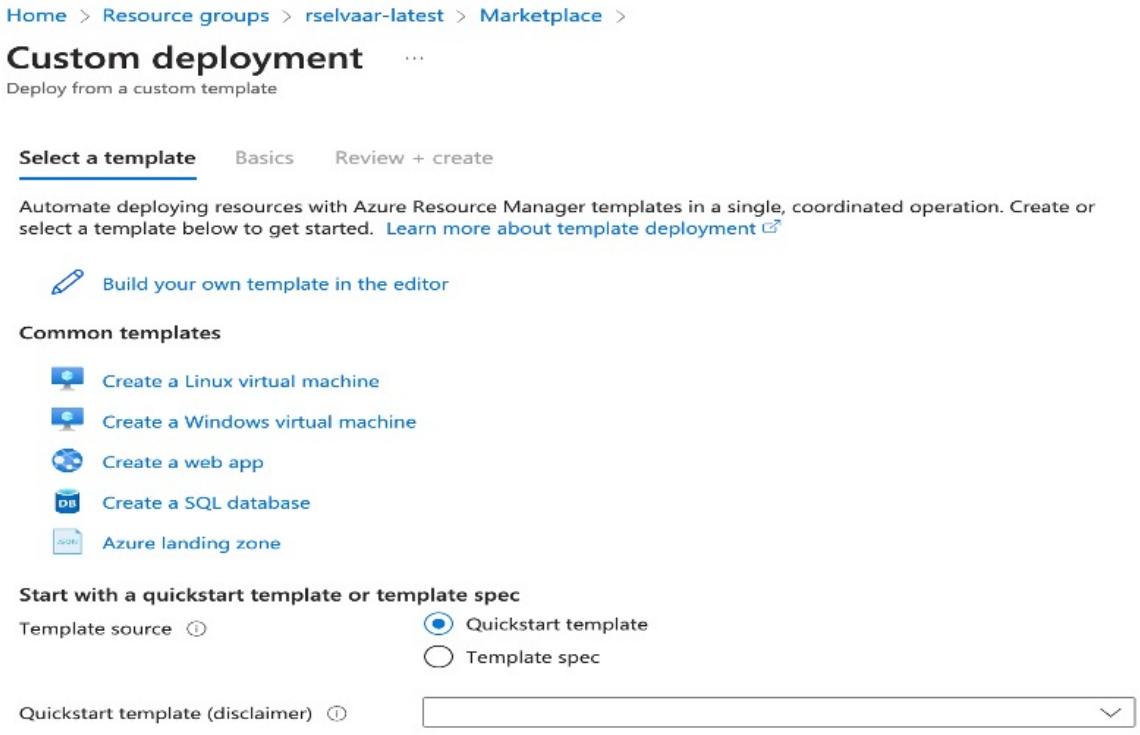
- For GWLB deployment, create virtual network with management, data, CCL subnets, and the function app.

- For NLB deployment, create virtual network with management, inside, outside, CCL subnets and the function app.



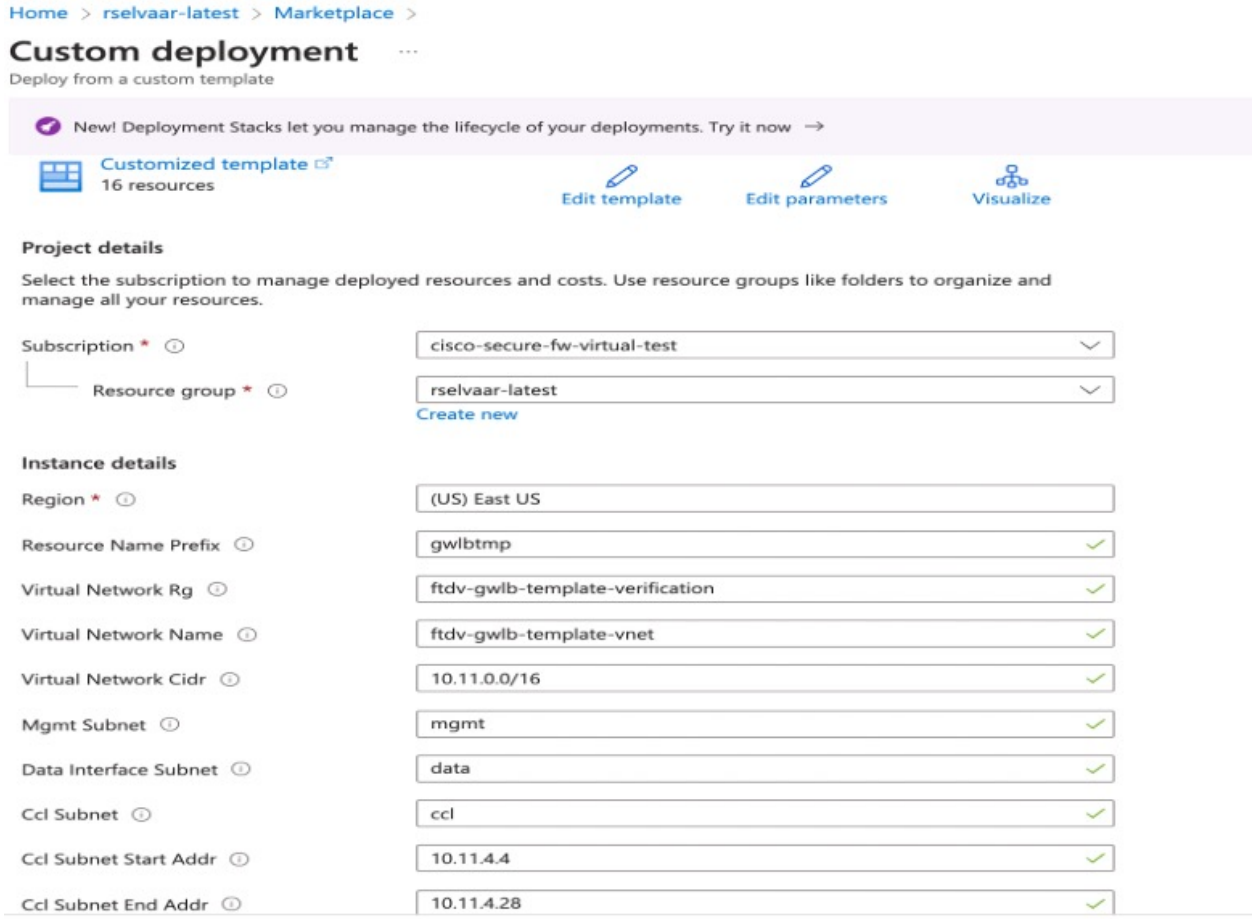

**Step 5** In **Search the Marketplace**, type **`Template deployment`** (deploy using custom templates), and then press **Enter**.

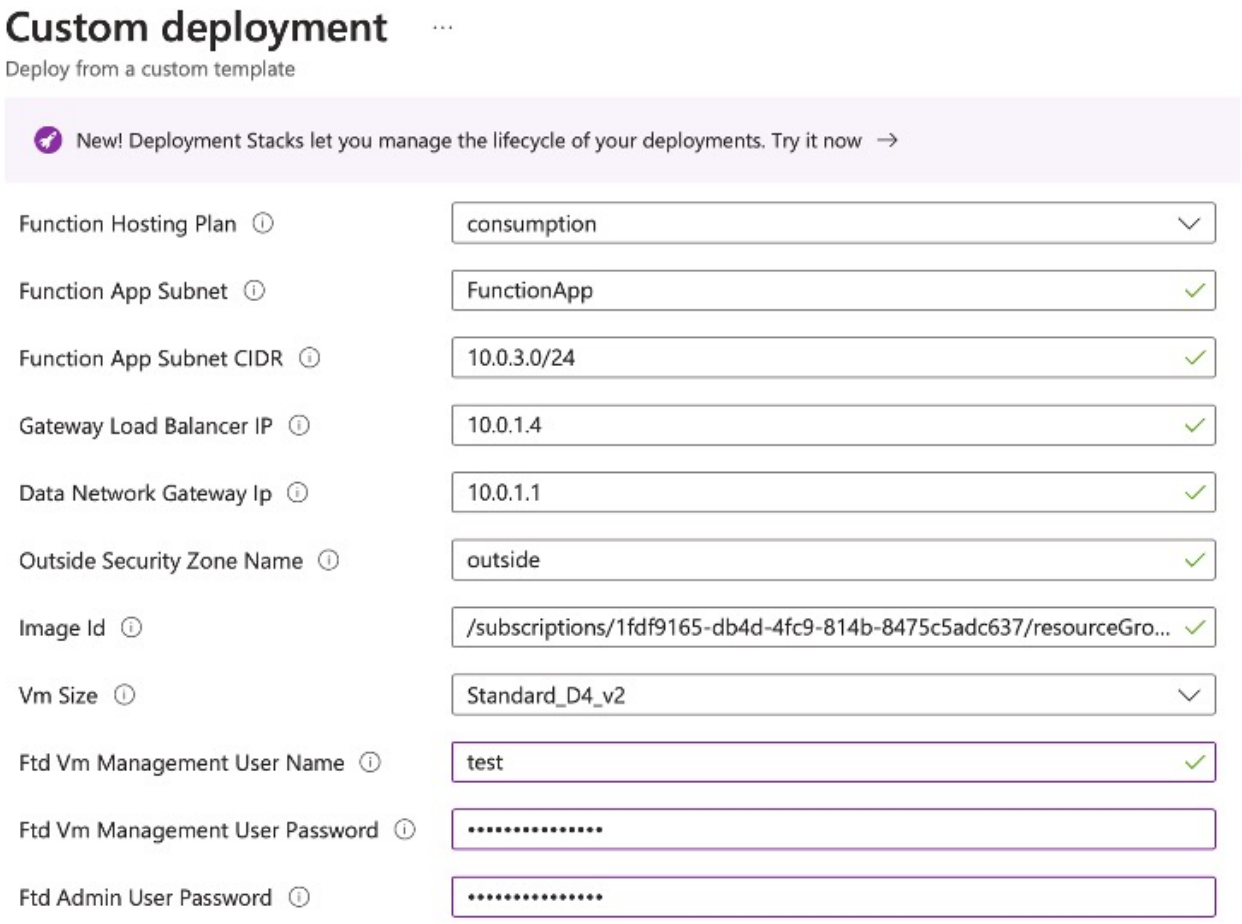
**Step 6** Click **Create**. There are several options for creating a template. Choose **Build your own template in editor**.

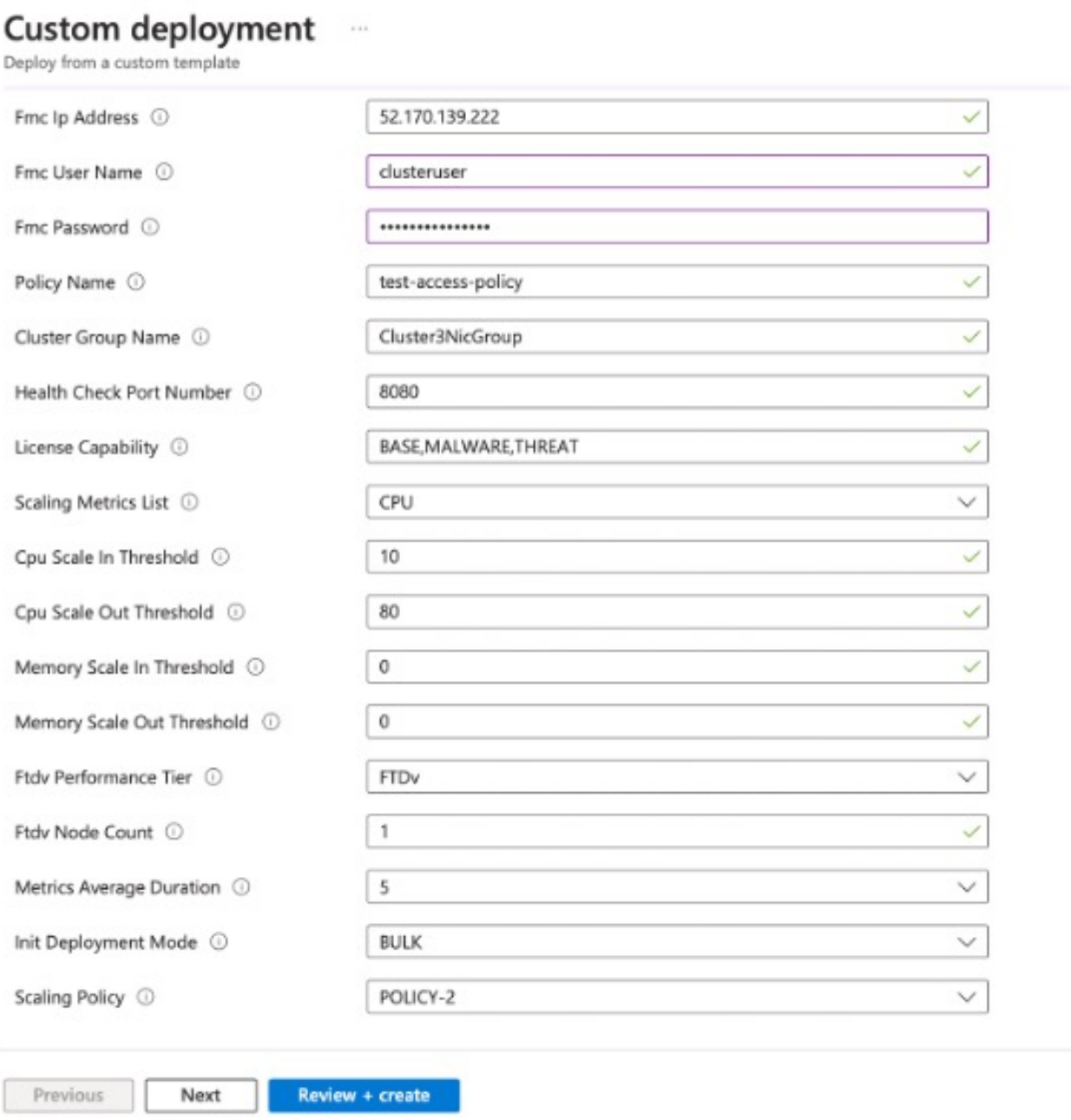



**Step 7** In the **Edit template** window, delete all the default content and copy the contents from the updated `azure_ftdv_gwlb_cluster_custom_image.json` or

`azure_ftdv_nlb_cluster_custom_image.json` (depending on the type of autoscale solution you are deploying on Azure) and click **Save**. Or Click **Load file** to browse and upload this file from your computer.




Custom deployment
Deploy from a custom template
New! Deployment Stacks let you manage the lifecycle of your deployments. Try it now →
Function Hosting Plan
consumption
Function App Subnet
FunctionApp
Function App Subnet CIDR
10.0.3.0/24
Gateway Load Balancer IP
10.0.1.4
Data Network Gateway Ip
10.0.1.1
Outside Security Zone Name
outside
Image Id
/subscriptions/1fdf9165-db4d-4fc9-814b-8475c5adc637/resourceGro...
Vm Size
Standard_D4_v2
Ftd Vm Management User Name
test
Ftd Vm Management User Password
Ftd Admin User Password




**Step 8** In the parameter field sections, fill all the parameters. Refer to Input Parameters for details about each parameter, then click **Review+Create**.

**Step 9** When a template deployment is successful, it creates all the required resources for the threat defense virtual auto scale for Azure solution. See the resources in the following figure. The **Type** column describes each resource, including the Logic App, VMSS, Load Balancers, Public IP address, etc.

### What to do next

Deploy Azure Functions App, on page 80.

# Deploy Azure Functions App

Deploy the cluster autoscale Azure Functions App to enable automatic scaling functionality for your Azure infrastructure.

When you deploy the ARM template, Azure creates the function app with the name **<resourceNamePrefix>-function-app**.

### Procedure

**Step 1** Go to the function app you created when you deployed the ARM template and do the following tasks:

Run the following command from your local computer to deploy the cluster autoscale Azure Functions to the Function app.

```
az functionapp deployment source config-zip -g <Resource Group Name>
-n <Function App Name> --src  <cluster_functions.zip> --build-remote true
```

**Step 2** After the deployment of the Azure Functions, you can view the uploaded Functions in the overview section of the function application.

# Update the Azure logic app

Configure the Logic App with the necessary information to function as the autoscale orchestrator for the autoscale functionality.

The Logic App acts as the orchestrator for the autoscale functionality. The Azure Resource Manager (ARM) template creates a skeleton Logic App, which you then need to update manually to provide the information necessary for it to function as the autoscale orchestrator.

### Procedure

**Step 1** From the repository, retrieve the file *LogicApp.txt* to the local system and edit as shown below.



**Important** Read and understand all of these steps before proceeding.

These manual steps are not automated in the ARM template so that only the Logic App can be upgraded independently in the future.

a) Required: Find and replace all the occurrences of "SUBSCRIPTION_ID" with your SUBSCRIPTION ID information.
b) Required: Find and replace all the occurrences of "RG_NAME" with your resource group NAME.

c) Required: Find and replace all of the occurrences of "FUNCTIONAPPNAME" to your function app NAME.

The following example shows a few of these lines in the *LogicApp.txt* file:

```
  "AutoScaleManager": {
      "inputs": {
          "function": {
              "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/AutoScaleManager"

          }
.
.
                      },
                      "Deploy_Changes_to_FTD": {
                          "inputs": {
                              "body": "@body('AutoScaleManager')",
                              "function": {
                                  "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/DeployConfiguration"

                              }
.
.
                      "DeviceDeRegister": {
                          "inputs": {
                              "body": "@body('AutoScaleManager')",
                              "function": {
                                  "id":
"/subscriptions/SUBSCRIPTION_ID/resourceGroups/RG_NAME/providers/Microsoft.Web/sites/FUNCTIONAPPNAME/functions/DeviceDeRegister"

                              }
                          },
                          "runAfter": {
                              "Delay_For_connection_Draining": [
```

d) (Optional) Edit the trigger interval, or leave the default value (5). This is the time interval at which the Autoscale functionality is periodically triggered. The following example shows these lines in the *LogicApp.txt* file:

```
        "triggers": {
            "Recurrence": {
                "conditions": [],
                "inputs": {},
                "recurrence": {
                    "frequency": "Minute",
                    "interval": 5
                },
```

e) (Optional) Edit the time to drain, or leave the default value (5). This is the time interval to drain existing connections from the Firewall Threat Defense Virtual before deleting the device during the Scale-In operation. The following example shows these lines in the *LogicApp.txt* file:

```
         "actions": {
              "Branch_based_on_Scale-In_or_Scale-Out_condition": {
                  "actions": {
                      "Delay_For_connection_Draining": {
                          "inputs": {
```

```
"interval": {
    "count": 5,
    "unit": "Minute"
}
```

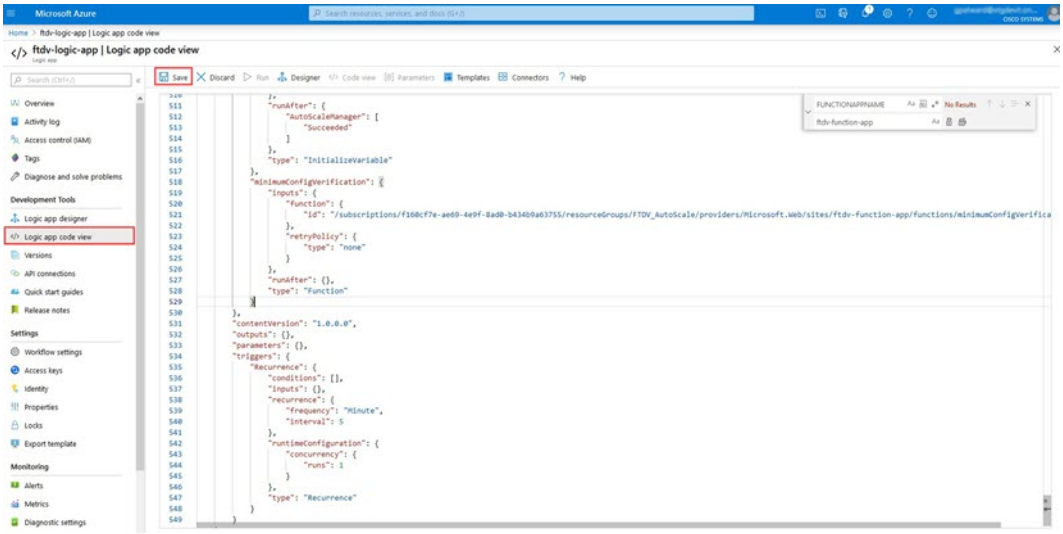
f) (Optional) Edit the cooldown time, or leave the default value (10). This is the time to perform no action after the Scale-Out is complete. The following example shows these lines in the *LogicApp.txt* file:

```
"actions": {
    "Branch_based_on_Scale-Out_or_Invalid_condition": {
        "actions": {
            "Cooldown_time": {
                "inputs": {
                    "interval": {
                        "count": 10,
                        "unit": "Second"
                    }
```

**Note** You can also perform these steps from the Azure portal. Consult the Azure documentation for more information.

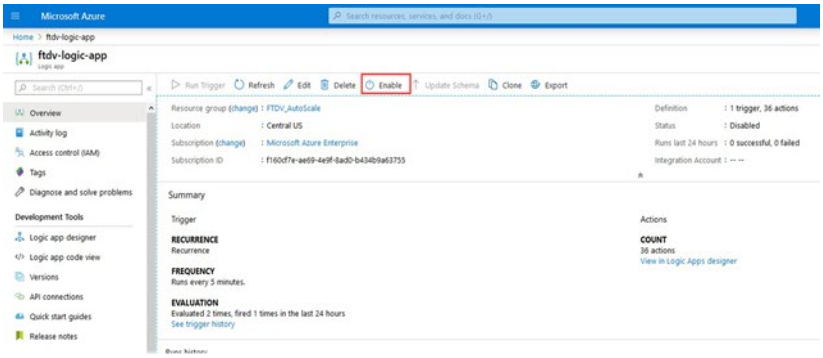
**Step 2** Go to the **Logic App code view**, delete the default contents, paste the contents from the edited *LogicApp.txt* file, and click **Save**.

***Figure 15: Logic app code view***



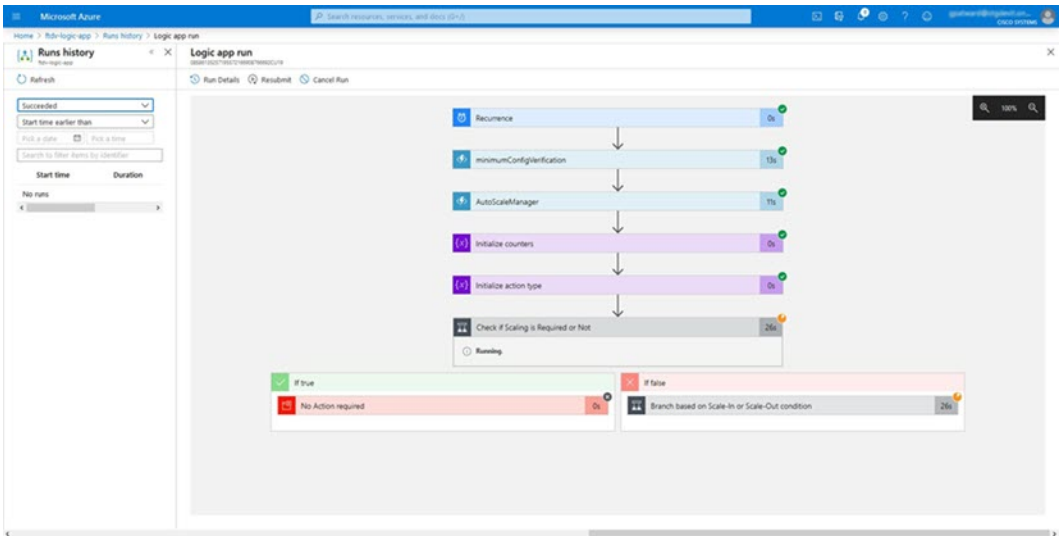

**Step 3** When you save the Logic App, it is in a 'Disabled' state. Click **Enable** when you want to start the autoscale manager.

*Figure 16: Enable logic app*



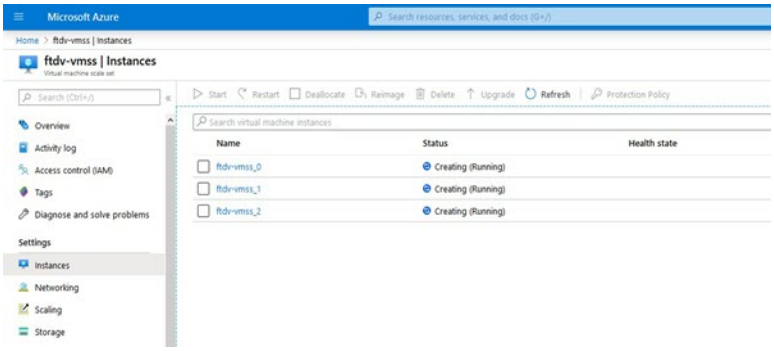

**Step 4** The tasks start running once enabled. Click the 'Running' status to see the activity.

*Figure 17: Logic app running status*




**Step 5** All deployment-related steps are complete once the Logic App starts. Verify in the virtual machine scale set (VMSS) that Firewall Threat Defense Virtual instances are being created.

*Figure 18: Threat Defense Virtual Instances Running*



In this example, three Firewall Threat Defense Virtual instances are launched because 'minFtdCount' was set to '3' and 'initDeploymentMode' was set to 'BULK' in the ARM template deployment.

# Deploy the cluster in GCP

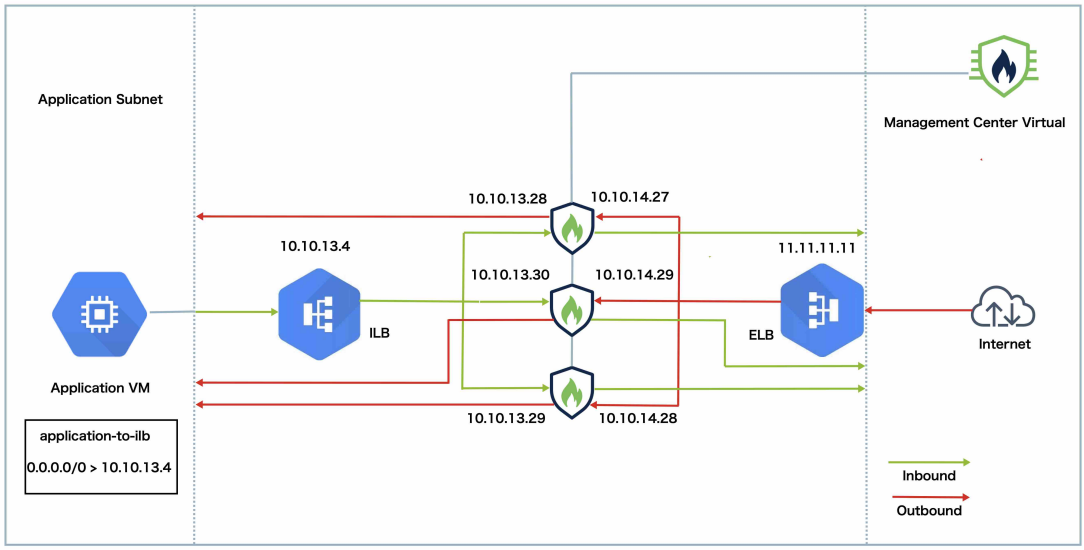
To deploy a cluster in GCP, you can either manually deploy or use an instance template to deploy an instance group. You can use the cluster with native GCP load balancers or non-native load balancers such as the Cisco Cloud Services Router.

**Note** Outbound traffic requires interface NAT and is limited to 64,000 connections.

## Sample topology for GCP clustering autoscale solution

Understand the architecture and traffic flow of a typical GCP clustering autoscale deployment.

***Figure 19: Sample Topology***




The topology shows both inbound and outbound traffic flow.

1. The Threat Defense Virtual cluster is placed between the internal and external load balancers. A Management Center Virtual instance is used to manage the cluster.
2. Inbound traffic from the internet goes to the external load balancer, which then transmits the traffic to the Threat Defense Virtual cluster.
3. The Threat Defense Virtual instance in the cluster inspects the traffic, and after inspection, forwards the traffic to the application VM.

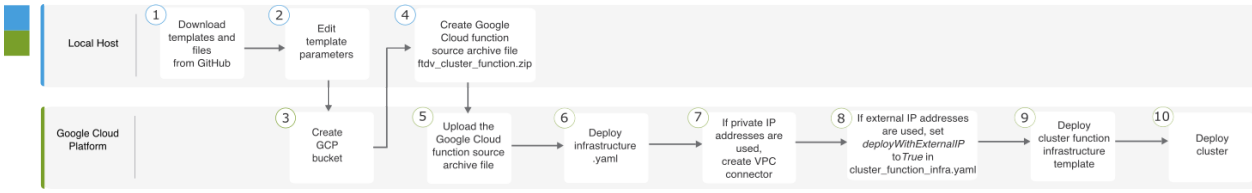
4. Outbound traffic from the application VM goes to the internal load balancer. The load balancer forwards this traffic to the Threat Defense Virtual cluster, which sends it to the internet.

# End-to-end process for deploying Threat Defense Virtual cluster in GCP

Provides a workflow to help users successfully deploy and verify a Threat Defense Virtual cluster on GCP using either template-based or manual deployment processes.

### Template-based Deployment

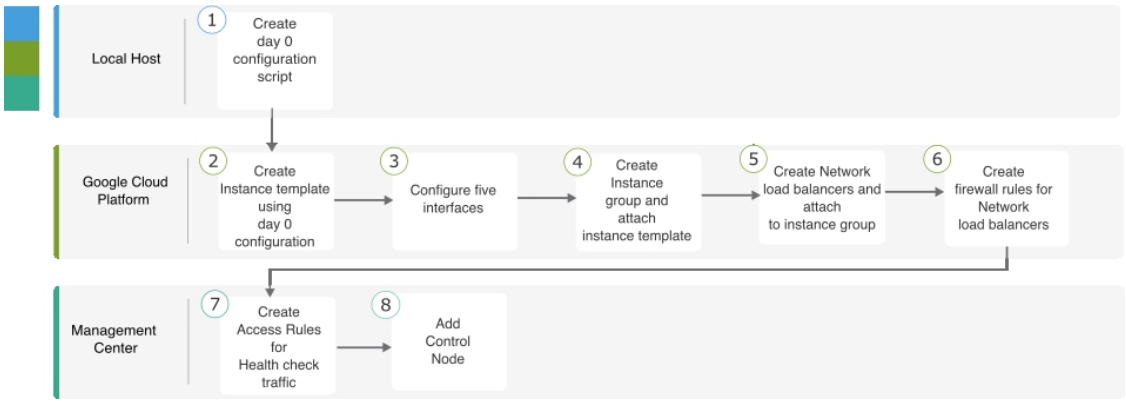
The following flowchart illustrates the workflow for template-based deployment of the Threat Defense Virtual cluster on GCP.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Download templates and files from GitHub. |
| 2 | Local Host | Edit template parameters. |
| 3 | Google Cloud Platform | Create GCP bucket. |
| 4 | Local Host | Create Google Cloud function source archive file *ftdv_cluster_function.zip*. |
| 5 | Google Cloud Platform | Upload the Google function source archive file. |
| 6 | Google Cloud Platform | Deploy infrastructure.yaml. |
| 7 | Google Cloud Platform | If private IP addresses are used, create VPC connector. |
| 8 | Google Cloud Platform | If external IP addresses are used, set *deployWithExternalIP* to *True* in *cluster_function_infra.yaml*. |
| 9 | Google Cloud Platform | Deploy cluster function infrastructure template. |
| 10 | Google Cloud Platform | Deploy cluster. |

### Manual Deployment

The following flowchart illustrates the workflow for manual deployment of the Threat Defense Virtual cluster on GCP.




| | Workspace | Steps |
|---|---|---|
| 1 | Local Host | Create day 0 configuration script. |
| 2 | Google Cloud Platform | Create instance template using day 0 configuration. |
| 3 | Google Cloud Platform | Configure the interfaces. |
| 4 | Google Cloud Platform | Create instance group and attach instance template. |
| 5 | Google Cloud Platform | Create NLB and attach to instance group. |
| 6 | Google Cloud Platform | Create firewall rules for NLB. |
| 7 | Management Center | Create access rules for health check traffic. |
| 8 | Management Center | Add control node. |

# Templates

The templates are available in GitHub. The parameter values are self-explanatory with the parameter names and values given in the template.

- Cluster deployment template for East-West traffic - deploy_ngfw_cluster_yaml
- Cluster deployment template for North-South traffic - deploy_ngfw_cluster.yaml

# Deploy the instance group in GCP using an instance template

Deploy a scalable instance group in Google Cloud Platform using pre-configured instance templates to establish a Secure Firewall cluster with automated management capabilities.

Deploy the instance group in GCP using an instance template.

### Before you begin

- Download the templates from GitHub to your local folder.
- Use Google Cloud Shell for deployment. Alternatively, you can use Google SDK on any macOS/Linux/Windows machine.
- To allow the cluster to auto-register with the Management Center, you need to create a user with administrative privileges on the Management Center that can use the REST API. See the *Cisco Secure Firewall Management Center Administration Guide*.
- Add an access policy in the Management Center that matches the name of the policy that you specified in *cluster_function_infra.yaml*.

### Procedure

**Step 1** Edit the **infrastructure.yaml**, **cluster_function_infra.yaml** and **deploy_ngfw_cluster.yaml** templates with the required *resourceNamePrefix* parameter (for example, ngfwvcls) and other required user inputs.

From Secure Firewall version 7.4.1, you can deploy the cluster without the diagnostic interface. To deploy the cluster with only the Outside, Inside, Management, and CCL interfaces, set the *withDiagnostic* variable to **False** in both the **infrastructure.yaml** and the **deploy_ngfw_cluster.yaml** files.

Note that there is a **deploy_ngfw_cluster.yaml** file in both the **east-west** and **north-south** folders in GitHub. Download the appropriate template as per your traffic flow requirement.

**Step 2** Create a bucket using Google Cloud Shell to upload the Google cloud function source archive file *ftdv_cluster_function.zip*.

**gsutil mb --pap enforced gs://***resourceNamePrefix***-ftdv-cluster-bucket/**

Ensure that the *resourceNamePrefix* variable here matches the *resourceNamePrefix* variable that you specified in **cluster_function_infra.yaml**.

**Step 3** Create an archive file for the cluster infrastructure.

**Example:**

```
zip -j ftdv_cluster_function.zip ./cluster-function/*
```

**Step 4** Upload the Google source archive that you created earlier.

**gsutil cp ftdv_cluster_function.zip gs://***resourceNamePrefix***-ftdv-cluster-bucket/**

**Step 5** Deploy infrastructure for the cluster.

**gcloud deployment-manager deployments create** *cluster_name* **--config infrastructure.yaml**

**Step 6** If you are using private IP addresses, perform the steps given below:

a) Launch and set up the Management Center Virtual with a Threat Defense Virtual management VPC.

b) Create a VPC connector to connect the Google Cloud functions with the Threat Defense Virtual management VPC.

**gcloud compute networks VPC-access connectors create** *VPC-connector-name* **--region us-central1 --subnet** *resourceNamePrefix***-ftdv-mgmt-subnet28**

**Step 7** If the Management Center is remote from the Threat Defense Virtual, and the Threat Defense Virtual needs an external IP address, ensure that you set **deployWithExternalIP** to **True** in **cluster_function_infra.yaml**.

**Step 8** Deploy the cluster function infrastructure.

**gcloud deployment-manager deployments create** *cluster_name* **--config cluster_function_infra.yaml**

**Step 9** Deploy the cluster.

**a.** For North-South topology deployment:

**gcloud deployment-manager deployments create** *cluster_name* **--config north-south/deploy_ngfw_cluster.yaml**

**b.** For East-West topology deployment:

**gcloud deployment-manager deployments create** *cluster_name* **--config east-west/deploy_ngfw_cluster.yaml**

# Manual cluster deployment in GCP

To deploy the cluster manually, prepare the day0 configuration, deploy each node, and then add the control node to the Firewall Management Center .

## Create the Day0 configuration with a fixed configuration for GCP

The fixed configuration will auto-generate the cluster bootstrap configuration.

```
{
    "AdminPassword": "password",
    "Hostname": "hostname",
    "FirewallMode": "Routed",
    "ManageLocally": "No",
                "Diagnostic": "OFF",       //Optional user input from version 7.4.1 - use
to deploy cluster without Diagnostic interface
    "Cluster": {
        "CclSubnetRange": "ip_address_start ip_address_end",
        "ClusterGroupName": "cluster_name"
    }
}
```

For example:

```
{
    "AdminPassword": "DeanW1nche$ter",
    "Hostname": "ciscoftdv",
    "FirewallMode": "Routed",
    "ManageLocally": "No",
    "Cluster": {
        "CclSubnetRange": "10.10.55.2 10.10.55.253",     //mandatory user input
        "ClusterGroupName": "ftdv-cluster"               //mandatory user input
    }
}
```

**Note** If you are copying and pasting the configuration given above, ensure that you remove **//mandatory user input** from the configuration.

For the **CclSubnetRange** variable, note that you cannot use the first two IP addresses and the last two IP addresses in the subnet. See Reserved IP addresses in IPv4 subnets for more information. Ensure that you have at least 16 available IP addresses for clustering. Some examples of start and end IP addresses are given below.

**Note** All cluster infrastructure subnets must use /27 CIDR.

*Table 5: Examples of start and end IP addresses*

| CIDR | Start IP Address | End IP Address |
|---|---|---|
| 10.1.1.0/27 | 10.1.1.2 | 10.1.1.29 |
| 10.1.1.32/27 | 10.1.1.34 | 10.1.1.61 |
| 10.1.1.64/27 | 10.1.1.66 | 10.1.1.93 |
| 10.1.1.96/27 | 10.1.1.98 | 10.1.1.125 |
| 10.1.1.128/27 | 10.1.1.130 | 10.1.1.157 |
| 10.1.1.160/27 | 10.1.1.162 | 10.1.1.189 |
| 10.1.1.192/27 | 10.1.1.194 | 10.1.1.221 |
| 10.1.1.224/27 | 10.1.1.226 | 10.1.1.253 |

## Deploy cluster nodes manually

Deploy cluster nodes to form a functional cluster on Google Cloud Platform (GCP) with proper network load balancing and management integration.

Deploy the cluster nodes so they form a cluster. For clustering on GCP, you cannot use the 4 vCPU machine type. The 4 vCPU machine type only supports four interfaces, and five are needed. Use a machine type that supports five interfaces, such as c2-standard-8.

### Procedure

**Step 1** Create an instance template using the day 0 configuration (in the **Metadata > Startup Script** section) with 5 interfaces: outside, inside, management, diagnostic, and cluster control link.

See Secure Firewall Threat Defense Virtual getting started guides.

**Step 2** Create an instance group, and attach the instance template.

**Step 3** Create GCP network load balancers (internal and external), and attach the instance group.

**Step 4** For GCP network load balancers, allow health checks in your security policy on the Management Center. Refer to Allow health checks for GCP network load balancers, on page 90.

**Step 5** Add the control node to the Management Center. Refer to Add the cluster to the Management Center (manual deployment), on page 106.

# Allow health checks for GCP network load balancers

Google Cloud provides health checks to determine if backends respond to traffic. This configuration allows health check traffic through firewall rules and access rules, redirects health check traffic to the Google metadata server using dynamic manual NAT rules, and sets up routing for GCP health checks across all interfaces used for health probes.

### Configuration requirements

Refer to https://cloud.google.com/load-balancing/docs/health-checks to create firewall rules for network load balancers. Then in the Firewall Management Center, create access rules to allow the health check traffic. See https://cloud.google.com/load-balancing/docs/health-check-concepts for the required network ranges. See Access Control Rules.

You also need to configure dynamic manual NAT rules to redirect the health check traffic to the Google metadata server at 169.254.169.254. See Configure dynamic manual NAT.

You can set up a route for GCP health checks across all interfaces that are used to configure their health probes. You can achieve this by creating a route with a higher metric on interfaces where a route for GCP health checks is not already available.

### North-South NAT rules sample configuration

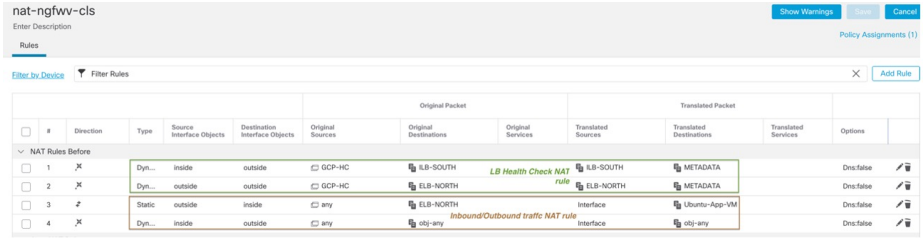
```
nat (inside,outside) source dynamic GCP-HC ILB-SOUTH destination static ILB-SOUTH METADATA
nat (outside,outside) source dynamic GCP-HC ELB-NORTH destination static ELB-NORTH METADATA

nat (outside,inside) source static any interface destination static ELB-NORTH Ubuntu-App-VM
nat (inside,outside) source dynamic any interface destination static obj-any obj-any

object network Metadata
 host 169.254.169.254

object network ILB-SOUTH
host <ILB_IP>
object network ELB-NORTH
host <ELB_IP>

object-group network GCP-HC
 network-object 35.191.0.0 255.255.0.0
 network-object 130.211.0.0 255.255.252.0
 network-object 209.85.204.0 255.255.252.0
 network-object 209.85.152.0 255.255.252.0
```



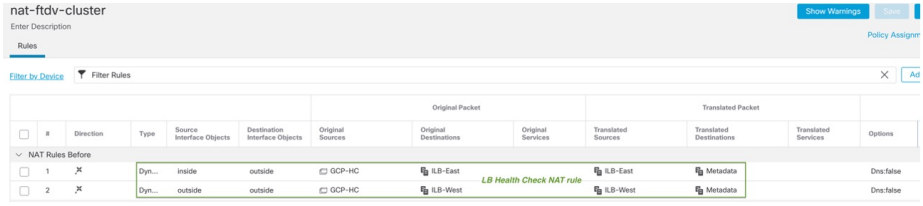

### East-West NAT rules sample configuration

```
nat (inside,outside) source dynamic GCP-HC ILB-East destination static ILB-East Metadata
nat (outside,outside) source dynamic GCP-HC ILB-West destination static ILB-West Metadata

object network Metadata
 host 169.254.169.254

object network ILB-East
host <ILB_East_IP>
object network ILB-West
host <ILB_West_IP>

object-group network GCP-HC
 network-object 35.191.0.0 255.255.0.0
 network-object 130.211.0.0 255.255.252.0
 network-object 209.85.204.0 255.255.252.0
 network-object 209.85.152.0 255.255.252.0
```




### North-South and East-West traffic routing configuration sample

```
route outside 0.0.0.0 0.0.0.0 <Outside_Gateway> 1
route inside 35.191.0.0 255.255.0.0 <Inside_Gateway> 1
route inside 130.211.0.0 255.255.252.0 <Inside_Gateway> 1
route inside 209.85.152.0 255.255.252.0 <Inside_Gateway> 1
route inside 209.85.204.0 255.255.252.0 <Inside_Gateway> 1
```

If a default route is not available, then policy-based routing can be used to route the traffic for health checks.

**Note** Set the NAT > Translated destination port to 80.

# Threat Defense Virtual clustering with autoscale solution in GCP

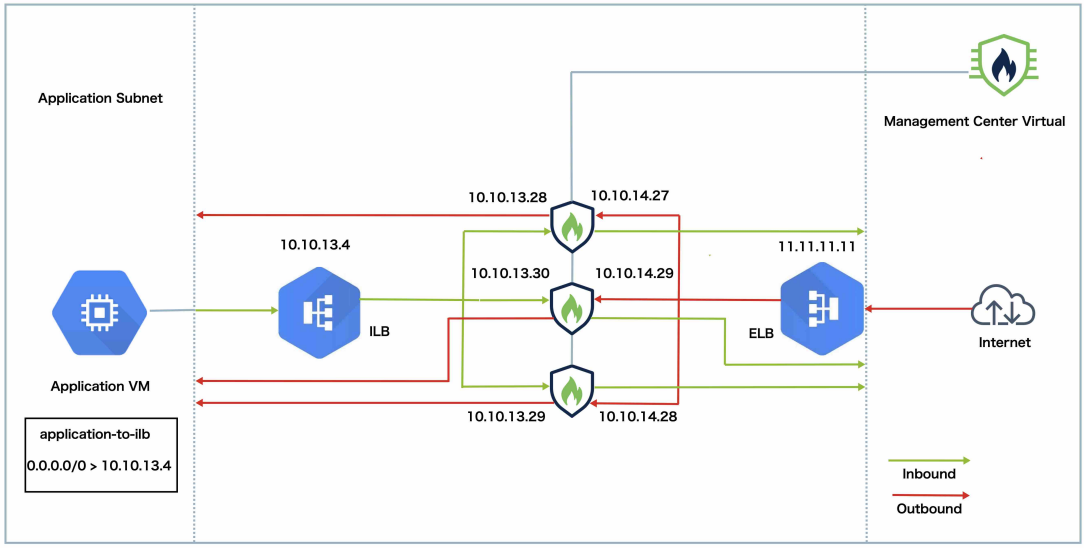
From version 10.0.0, the Threat Defense Virtual clustering solution in Google Cloud Platform (GCP) is enhanced with dynamic autoscaling capabilities. The autoscale solution is based on CPU utilization metrics and helps to achieve optimal resource usage. It is deployed using Terraform-based templates.

You can choose the scaling option as either dynamic clustering or fixed-node clustering, depending on your requirements. You can also define the CPU threshold value, beyond which scaling will be initiated.

## Sample topology for GCP clustering autoscale solution

Understand the architecture and traffic flow of a typical GCP clustering autoscale deployment.

***Figure 20: Sample Topology***




The topology shows both inbound and outbound traffic flow.

1. The Threat Defense Virtual cluster is placed between the internal and external load balancers. A Management Center Virtual instance is used to manage the cluster.
2. Inbound traffic from the internet goes to the external load balancer, which then transmits the traffic to the Threat Defense Virtual cluster.
3. The Threat Defense Virtual instance in the cluster inspects the traffic, and after inspection, forwards the traffic to the application VM.
4. Outbound traffic from the application VM goes to the internal load balancer. The load balancer forwards this traffic to the Threat Defense Virtual cluster, which sends it to the internet.

# Requirements and prerequisites for GCP clustering autoscale deployment

## Model requirements

Ensure that you meet the model and environment requirements for Threat Defense Virtual clustering autoscale solution on GCP.

- Supported Threat Defense Virtual models: FTDv20, FTDv30, FTDv50, and FTDv100.
- A valid Google Cloud Platform (GCP) account is required.
- Google Cloud SDK must be installed locally, or you should have access to GCP Cloud Shell.
- A Cisco Smart Licensing account is necessary for licensing the Management Center.

## Management center requirements

Configure the Management Center with the necessary policies and user access before deploying the autoscale solution.

- Make sure that an access control policy is configured.
- Make sure that the security zones for data interfaces are defined.
- A user with *Administrator* role is required for auto-registration.
- Make sure that the NAT rules are configured to respond to the health check of load balancers.

## Licensing

Use BYOL licensing and understand the license behavior for cluster deployments.

- The license entitlement of the control node applies to all data nodes in a cluster, regardless of their initial performance tier configuration during boot-up.
- After cluster formation, changing the performance tier of any individual Threat Defense Virtual is not allowed.
- By default, each unit includes a 100 kbps evaluation license.
- License behavior after deregistration:
  - Until reboot: The unit retains the existing throughput.
  - After reboot: The throughput reverts to the default 100 kbps.
- Only BYOL (Bring Your Own License) license type is supported.
- PAYG (Pay-As-You-Go) license type is not supported.

## Supported configurations

Use these supported configuration parameters when deploying the autoscale solution.

- Cluster size: 1-16 nodes.
- Traffic topologies: North-South and East-West.
- Deployment mode: Deployment using Terraform templates.

# Templates

The following templates are available on GitHub.

- infrastructure/main.tf – Template for infrastructure deployment.
- cluster_deployment/main.tf – Template for cluster deployment.

# Deploy Threat Defense Virtual clustering autoscale solution on GCP

There are two steps in the deployment process:

1. Infrastructure deployment (optional) – to deploy the required network infrastructure.
2. Cluster deployment – to deploy the Threat Defense Virtual cluster.



**Note** If you choose to use your own infrastructure for deploying the Threat Defense Virtual, then you can skip the infrastructure deployment step. However, it is essential to make sure that all the required resources are provided for the proper functioning of the solution.

# Infrastructure deployment parameters

These parameters are used for infrastructure deployment.

***Table 6: List of template parameters for infrastructure deployment (infrastructure_params.tfvars)***

| Parameter | Description | Example |
|---|---|---|
| project_id | GCP project ID | test-project-12345 |
| resource_name_prefix | String (lowercase alphabets only.) | demoftdv |
| region | GCP region to deploy | us-central1 |
| mgmt_ip_cidr_range | Mgmt Subnet CIDR | 10.112.0.0/27 |
| vpc_connector_ip_cidr_range | Mgmt VPC Connector CIDR /28 subnet only | 10.112.50.0/28 |
| with_diagnostic | Whether to enable diagnostic interface. | true or false (false is supported only starting with version 7.4.1.) |
| diag_ip_cidr_range | Diagnostic Subnet CIDR | 10.112.19.0/27 |
| inside_ip_cidr_range | Inside Subnet CIDR | 10.112.1.0/27 |
| outside_ip_cidr_range | Outside Subnet CIDR | 10.112.2.0/27 |
| ccl_ip_cidr_range | CCL Subnet CIDR (/27 subnet recommended) | 10.112.100.0/27 |

# Deploy infrastructure using terraform on GCP

Deploy and manage infrastructure resources on Google Cloud Platform using Terraform through the GCP Infrastructure Manager.

The GCP Infrastructure Manager facilitates Terraform deployments as a stack. From this interface, you can manage resources and perform cleanups.

## Procedure

**Step 1** Navigate to the GCP console and log in using your credentials.

**Step 2** Open Cloud Shell or local terminal.

If you are using local terminal, make sure that GCP CLI is configured on your system.

**Step 3** Create a new directory.

**Example:**

Use the command **mkdir infra** to create a directory named infra in Cloud Shell or your local terminal.

**Step 4** Navigate to the newly created directory.

**Example:**

Use the command **cd infra** to change to the new directory.

**Step 5** Copy or download the infrastructure_params.tfvars file from Cisco GitHub repository and save it to the new infra folder.

**Step 6** Open the infrastructure_params.tfvars file and update all the placeholder values as required. Refer to Infrastructure deployment parameters, on page 94.

**Step 7** Use the following commands after updating placeholder values to start the deployment.

```
gcloud infra-manager deployments \
apply
"projects/<project_id>/locations/<region>/deployments
/<deployment_name>" \
--location="<region>" \
--git-source-repo="<repo name>" \
--git-source-directory="infrastructure" \
--git-source-ref="<branch name>" \
--serviceaccount="
projects/<project_id>/serviceAccounts/<servi
ce_account_name>" \
--artifacts-gcs-bucket="gs://<bucket_name>/artifacts"
\
--inputs-file="/path/to/your/infra_params.tfvars"
```

***Table 7: Parameters description***

| Parameter | Description |
|---|---|
| deployment_name | Name of the Terraform deployment stack that will be shown in GCP Infrastructure Manager. |
| location | The location or region where the deployment will be applied. |

| Parameter | Description |
|---|---|
| git-source-repo | The name of the Git repository that contains the source code (Cisco GitHub Link). |
| git-source-directory | The directory in the Git repository where the infrastructure code is located. |
| git-source-ref | The branch name in the Git repository to be used for the deployment. |
| service-account | Name of the service account. |
| artifacts-gcs-bucket | The Google Cloud Storage bucket to store artifacts related to the deployment (optional). |
| inputs-file | The file path to the input parameters for the infrastructure configuration. |

**Note** If you do not use the template to deploy the infrastructure, you must provide the following resources for proper deployment and functioning of the solution.

- Management VPC
  - Management subnet
  - Management subnet for VPC connector
  - Management firewall rule name
- Inside VPC
  - Inside subnet
  - Inside firewall rule name
- Outside VPC
  - Outside subnet
  - Outside firewall rule name
  - Nat GW
- CCL VPC
  - CCL subnet
  - CCL firewall rule name.
- Diagnostic VPC (optional)
  - Diagnostic subnet
  - Diagnostic firewall rule name
- VPC connector with management subnet

# Cluster deployment parameters

The following input parameters are used to deploy the cluster.

***Table 8: List of template parameters for cluster deployment***

| Parameter | Description | Example |
|---|---|---|
| **General configuration** | | |
| type_of_deployment | Type of deployment (north_south or east_west) | north_south |

| Parameter | Description | Example |
|---|---|---|
| **Project information** | | |
| project_id | GCP project ID | test-project-12345 |
| region | Deployment region | us-east1 |
| zone1 | Deployment zone | a |
| zone2 | Deployment zone | b |
| zone3 | Deployment zone | c |
| resource_name_prefix | Prefix for naming resources | democluster |
| service_account_mail_id | Service account email | service-account-mail@example-project.iam.gserviceaccount.com |
| **VPC and subnet configuration** | | |
| mgmt_vpc_name | Management VPC name | <resource-name>-ftdv-mgmt-VPC |
| mgmt_subnet_name | Management subnet name | <resource-name>-ftdv-mgmt-subnet |
| inside_vpc_name | Inside VPC name | <resource-name>-ftdv-inside-VPC |
| inside_subnet_name | Inside subnet name | <resource-name>-ftdv-inside-subnet |
| outside_vpc_name | Outside VPC name | <resource-name>-ftdv-outside-VPC |
| outside_subnet_name | Outside subnet name | <resource-name>-ftdv-outside-subnet |
| diag_vpc_name | Diagnostics VPC name | <resource-name>-ftdv-diag-VPC |
| diag_subnet_name | Diagnostics subnet name | <resource-name>-ftdv-diag-subnet |
| ccl_vpc_name | CCL VPC name | <resource-name>-ftdv-CCL-VPC |
| ccl_subnet_name | CCL subnet name | <resource-name>-ftdv-CCL-subnet |
| **Firewall rules** | | |
| diag_firewall_rule_name | Firewall rule for diagnostics | <resource-name>-ftdv-diag-firewall-rule |
| ccl_firewall_rule_name | Firewall rule for CCL | <resource-name>-ftdv-CCL-firewall-rule |
| mgmt_firewall_rule_name | Firewall rule for management | <resource-name>-ftdv-mgmt-firewall-rule |
| inside_firewall_rule_name | Firewall rule for inside network | <resource-name>-ftdv-inside-firewall-rule |
| outside_firewall_rule_name | Firewall rule for outside network | <resource-name>-ftdv-outside-firewall-rule |
| inside_hc_firewall_rule_name | Firewall rule for inside network | <resource-name>-ftdv-inside-hc-firewall-rule |
| outside_hc_firewall_rule_name | Firewall rule for outside network | <resource-name>-ftdv-outside-hc-firewall-rule |

| Parameter | Description | Example |
|---|---|---|
| **Instance details** | | |
| machine_type | GCP supported machine type | n1-standard-8 |
| source_image_url | Source image location for THREAT Defense Virtual | projects/cisco-public/global/images/cisco-ftdv-10-0-0 |
| public_key | Public Key for SSH access | SSH-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC4v |
| **Autoscale details** | | |
| auto_scaling | Enable autoscaling | true/false |
| cpu_utilization_target | Target CPU utilization for autoscaling [0.0-1.0] | 0.60 |
| min_ftd_count | Minimum number of THREAT Defense Virtual instances | 0 |
| max_ftd_count | Maximum number of THREAT Defense Virtual instances | 2 |
| **THREAT Defense Virtual specific configuration** | | |
| ftd_password_secret_name | Name of the THREAT Defense Virtual secret for new Admin password in Secret Manager, device will use this password after first time login. | ftd-password-secret |
| hostname | Hostname for THREAT Defense Virtual | cisco-ngfwv |
| ccl_subnet_range | Subnet range for CCL, space separated | 10.112.100.2 10.112.100.30 |
| cluster_grp_name | Cluster group name for THREAT Defense Virtual | ftdv-cluster |
| with_diagnostic | Whether to enable diagnostics | true/false |
| assign_public_ip_to_mgmt | Whether to assign public IP to management interface | true/false |
| ftd_reg_via_public_ip | Whether to register THREAT Defense Virtual with public IP | true/false |
| **Management Center information and THREAT Defense Virtual configuration** | | |

| Parameter | Description | Example |
|---|---|---|
| reg_id | Management Center registration ID | cisco |
| nat_id | NAT ID for Management Center | cisco |
| policy_id | Initial policy ID created in Management Center | ftdv-ini-pol |
| fmc_ip | Management Center IP address | 10.112.0.2 / 34.113.15.29 |
| fmc_password_secret_name | Name of the FMC secret for new Admin password in Secret Manager, device will use this password after first time login. | FMC-password-secret |
| fmc_username | Management Center login username | restapi |
| license_caps | License capabilities | BASE, MALWARE, URLFilter, THREAT |
| performance_tier | Performance tier for THREAT Defense Virtual | FTDv20, FTDv30 |
| vpc_connector_name | Name of VPC connector | <resource-name>-connector |
| **Internal Load Balancer configuration** | | |
| ilb_frontend_protocol | Frontend protocol (TCP/UDP) | TCP |
| ilb_backend_protocol | Backend protocol | TCP |
| ilb_health_check_port | ILB Health-check balancer port, NAT required in Management Center | 8989 |
| ilb_timeout_sec | Load balancer timeout (seconds) | 5 |
| ilb_draining_timeout_sec | Timeout for draining connections (seconds) | 60 |
| ilb_check_interval_sec | Interval between health checks for ILB (seconds) | 10 |
| ilb_unhealthy_threshold | Number of failed health checks before marking unhealthy | 1 |
| **External Load Balancer specific configuration only in case of north_south deployment** | | |
| elb_frontend_protocol | Frontend protocol name for External Load Balancer (TCP/UDP) | TCP |

| Parameter | Description | Example |
|---|---|---|
| elb_backend_protocol | Backend protocol name for External Load Balancer (TCP/UDP/UNSPECIFIED) | TCP |
| elb_front_end_ports | List of ELB frontend (listener) ports | all or [22, 80, 443] |
| elb_health_check_port | ELB health-check port, NAT required in Management Center | 87878 |
| elb_timeout_sec | Load balancer timeout (seconds) | 5 |
| elb_unhealthy_threshold | Number of failed health checks before marking unhealthy | 2 |
| elb_check_interval_sec | Interval between health checks for ELB (seconds) | 10 |
| elb_draining_timeout_sec | Timeout for draining connections (seconds) | 60 |

# Deploy cluster using Terraform on GCP

Deploy a cluster using Terraform on Google Cloud Platform (GCP) through the Infrastructure Manager to manage resources and perform cleanups from a centralized interface.

The GCP Infrastructure Manager facilitates Terraform deployments as a stack. From this interface, you can manage resources and perform cleanups.

### Procedure

**Step 1** Navigate to the GCP console and log in using your credentials.

**Step 2** Open Cloud Shell or local terminal.

If you are using local terminal, make sure that GCP CLI is configured on your system.

**Step 3** Create a new directory.

**Example:**

Use the command **mkdir cluster** to create a directory named cluster in Cloud Shell or your local terminal.

**Step 4** Navigate to the newly created directory.

**Example:**

Use the command **cd cluster** to change to the new directory.

**Step 5** Copy or download the cluster_params.tfvars file from Cisco GitHub repository and save it to the new cluster folder.

**Step 6** Open the cluster_params.tfvars file and update all the placeholder values as required. Refer to Cluster deployment parameters, on page 97.

**Step 7** Use the following commands after updating placeholder values to start the deployment.

```
gcloud infra-manager deployments \
apply "projects/<project_id>/locations/<region>/deployments/<deployment_name>" \
--location="<region>" \
--git-source-repo="<repo name>" \
--git-source-directory="cluster_deployment" \
--git-source-ref="<branch name>" \
--service-account="projects/<project_id>/serviceAccounts/<service_account_name>" \
--artifacts-gcs-bucket="gs://<bucket_name>/artifacts" \
--inputs-file="/path/to/cluster_params.tfvars"
```

***Table 9: Parameters description***

| Parameter | Description |
|---|---|
| deployment_name | Name of the Terraform deployment stack that will be shown in GCP infrastructure manager. |
| location | The location or region where the deployment will be applied. |
| git-source-repo | The name of the Git repository that contains the source code (Cisco GitHub Link). |
| git-source-directory | The directory in the Git repository where the infrastructure code is located. |
| git-source-ref | The branch name in the Git repository to be used for the deployment. |
| service-account | Name of the service account. |
| artifacts-gcs-bucket | The Google Cloud Storage bucket to store artifacts related to the deployment, for example, Terraform state files. |
| inputs-file | The file path to the input parameters for the cluster deployment. |

# Deployment configurations after cluster deployment

This task creates a route inside the VPC to forward the required traffic through an internal load balancer on Google Cloud Shell.

This procedure is performed after cluster deployment as part of deployment configurations.

### Procedure

Run the following command to create the route:

**Example:**

```
gcloud compute routes create <ilb-route-name>
--network=<inside-vpc-name> --priority=1000 --destination-range=0.0.0.0/0
--next-hop-ilb=<ilb-forwarding-rule-name> --next-hop-ilb-region=<region>
```

**Note** You can create route from the GCP UI as well. To create a route from the GCP UI, follow these steps:

a. In the GCP UI, navigate to `Routes > Route Management > Create route`.

b. Fill in the Create a route dialog box.

1. **Name**: Specify a route name.
2. **Network**: Select **<inside-VPC-name>**.
3. **Route type**: Select **Static route**.
4. **Destination IPv4 range**: Enter **0.0.0.0/0**.
5. **Priority**: Set to **1000**.
6. **Next Hop**: Select **Specify a forwarding rule of internal passthrough Network Load balancer**.
7. **Forwarding rule name**: Select **<forwarding-rule-for-ilb>**.

c. Click **Create** to create the route.

# Deploy the cluster in GCP manually

To deploy the cluster manually, prepare the day0 configuration.

```
{
                "AdminPassword": "password",
                "Hostname": "hostname",
                "FirewallMode": "Routed",
                "ManageLocally": "No",
                "Diagnostic": "OFF",       //Optional user input from version 7.4.1
                "Cluster": {
                "CclSubnetRange": "ip_address_start ip_address_end",
                "ClusterGroupName": "cluster_name"
                }
                }
```

For example:

```
{
                "AdminPassword": "C15co@!23",
                "Hostname": "cisco-ftd",
                "FirewallMode": "routed",
                "Diagnostic": "OFF",
                "ManageLocally": "No",
                "Cluster": {
                "CclSubnetRange": "10.10.65.2 10.10.65.29",
                "ClusterGroupName": "gcp-cls-ftd"
                }
```

```
}
```

# Allow health checks for GCP network load balancers

Google Cloud provides health checks to determine if backends respond to traffic. This configuration allows health check traffic through firewall rules and access rules, redirects health check traffic to the Google metadata server using dynamic manual NAT rules, and sets up routing for GCP health checks across all interfaces used for health probes.

### Configuration requirements

Refer to https://cloud.google.com/load-balancing/docs/health-checks to create firewall rules for network load balancers. Then in the Firewall Management Center, create access rules to allow the health check traffic. See https://cloud.google.com/load-balancing/docs/health-check-concepts for the required network ranges. See Access Control Rules.

You also need to configure dynamic manual NAT rules to redirect the health check traffic to the Google metadata server at 169.254.169.254. See Configure dynamic manual NAT.

You can set up a route for GCP health checks across all interfaces that are used to configure their health probes. You can achieve this by creating a route with a higher metric on interfaces where a route for GCP health checks is not already available.

### North-South NAT rules sample configuration

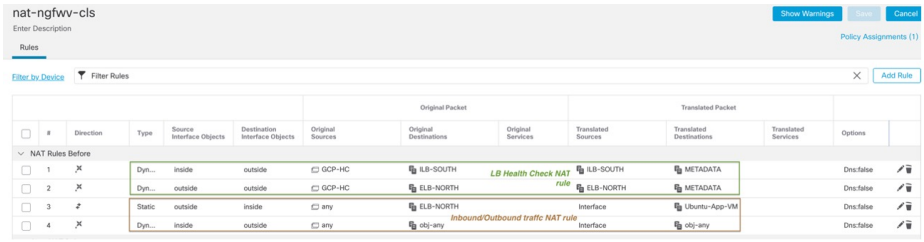
```
nat (inside,outside) source dynamic GCP-HC ILB-SOUTH destination static ILB-SOUTH METADATA
nat (outside,outside) source dynamic GCP-HC ELB-NORTH destination static ELB-NORTH METADATA

nat (outside,inside) source static any interface destination static ELB-NORTH Ubuntu-App-VM
nat (inside,outside) source dynamic any interface destination static obj-any obj-any

object network Metadata
 host 169.254.169.254

object network ILB-SOUTH
host <ILB_IP>
object network ELB-NORTH
host <ELB_IP>

object-group network GCP-HC
 network-object 35.191.0.0 255.255.0.0
 network-object 130.211.0.0 255.255.252.0
 network-object 209.85.204.0 255.255.252.0
 network-object 209.85.152.0 255.255.252.0
```



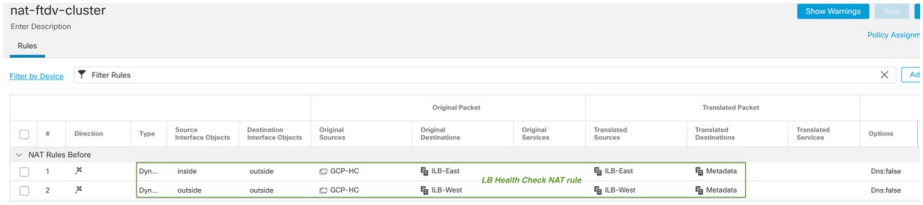
### East-West NAT rules sample configuration

```
nat (inside,outside) source dynamic GCP-HC ILB-East destination static ILB-East Metadata
nat (outside,outside) source dynamic GCP-HC ILB-West destination static ILB-West Metadata

object network Metadata
 host 169.254.169.254

object network ILB-East
host <ILB_East_IP>
object network ILB-West
host <ILB_West_IP>

object-group network GCP-HC
 network-object 35.191.0.0 255.255.0.0
 network-object 130.211.0.0 255.255.252.0
 network-object 209.85.204.0 255.255.252.0
 network-object 209.85.152.0 255.255.252.0
```




### North-South and East-West traffic routing configuration sample

```
route outside 0.0.0.0 0.0.0.0 <Outside_Gateway> 1
route inside 35.191.0.0 255.255.0.0 <Inside_Gateway> 1
route inside 130.211.0.0 255.255.252.0 <Inside_Gateway> 1
route inside 209.85.152.0 255.255.252.0 <Inside_Gateway> 1
route inside 209.85.204.0 255.255.252.0 <Inside_Gateway> 1
```

If a default route is not available, then policy-based routing can be used to route the traffic for health checks.

**Note** Set the NAT > Translated destination port to 80.

# Troubleshooting GCP clustering with autoscale

When the cluster fails to form, verify network connectivity and configuration of NVE-only cluster interfaces and associated object groups.

- Check the IP address of NVE-only cluster interface. Make sure you can ping the NVE-only cluster interface of other nodes.
- Make sure that the IP addresses of the NVE-only cluster interfaces are part of the object group.
- Make sure that the Network Virtualization Edge (NVE) interface is configured with the corresponding object group.
- Make sure that the cluster interface in the cluster group has the correct Virtual Network Identifier (VNI) interface. This VNI interface should have the NVE with the corresponding object group.

- Each node has its own cluster interface IP address. Make sure the nodes can ping each other to verify connectivity.

# Add the cluster to the Management Center (manual deployment)

This task registers a manually deployed cluster with the Firewall Management Center to enable centralized management and configuration of the cluster units.

Use this procedure to add the cluster to the Firewall Management Center if you manually deployed the cluster. If you used a template, the cluster will auto-register on the Firewall Management Center.

Add one of the cluster units as a new device to the Firewall Management Center; the Firewall Management Center automatically detects all other cluster members.

**Before you begin**

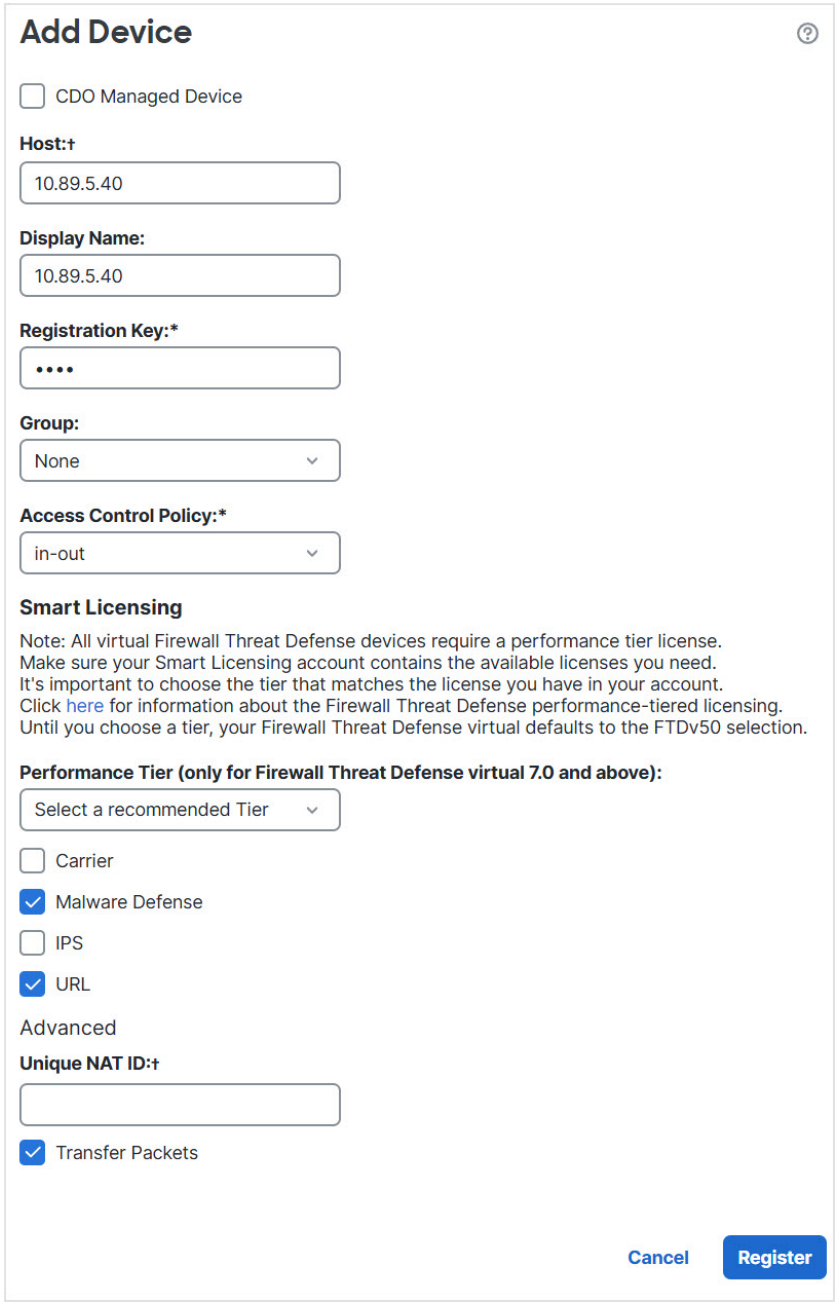
- All cluster units must be in a successfully formed cluster before adding the cluster to the Firewall Management Center. You should also check which unit is the control unit. Use the Firewall Threat Defense **show cluster info** command.

Follow these steps to add the cluster to the Management Center for manual deployment:

## Procedure

**Step 1** In the Firewall Management Center, choose **Devices** > **Device Management**, and then choose **Add** > **Add Device** to add the control unit using the unit's management IP address.

*Figure 21: Add device*




a) In the **Host** field, enter the IP address or hostname of the control unit.

   We recommend adding the control unit for the best performance, but you can add any unit of the cluster.

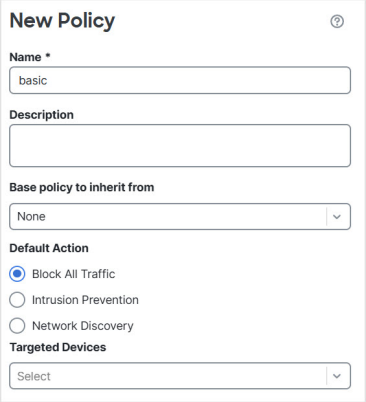
   If you used a NAT ID during device setup, you may not need to enter this field. For more information, see NAT environments.

b) In the **Display Name** field, enter a name for the control unit as you want it to display in the Firewall Management Center.

This display name is not for the cluster; it is only for the control unit you are adding. You can later change the name of other cluster members and the cluster display name.

c) In the **Registration Key** field, enter the same registration key that you used during device setup. The registration key is a one-time-use shared secret.
d) (Optional) Add the device to a device **Group**.
e) Choose an initial **Access Control Policy** to deploy to the device upon registration, or create a new policy.

If you create a new policy, you create a basic policy only. You can later customize the policy as needed.




f) Choose licenses to apply to the device.
g) If you used a NAT ID during device setup, expand the **Advanced** section and enter the same NAT ID in the **Unique NAT ID** field.
h) Select the **Transfer Packets** check box to allow the device to transfer event metadata and packet data associated with intrusion events to the Firewall Management Center.

This option is enabled by default. The transferred packet data is used for intrusion-event analysis. It does not necessarily contain the complete original packet as transmitted on the wire or the complete traffic session. For intrusion events generated by Snort inspection, the device can transfer only the Protocol Data Unit (PDU) or packet content associated with the intrusion-rule match.

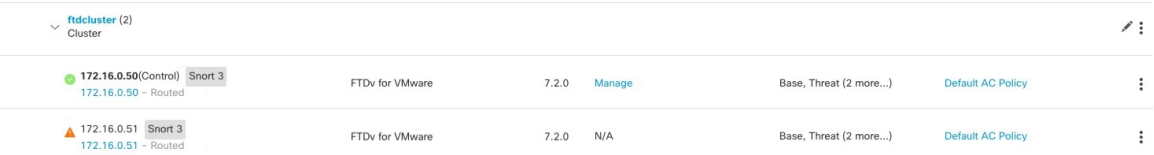
If you disable this option, the device sends only event information to the Firewall Management Center; packet data is not sent.

i) Click **Register**.

The Firewall Management Center identifies and registers the control unit, and then registers all data units. If the control unit does not successfully register, then the cluster is not added. A registration failure can occur if the cluster was not up, or because of other connectivity issues. In this case, we recommend that you try re-adding the cluster unit.

The cluster name shows on the **Devices** > **Device Management** page; expand the cluster to see the cluster units.

***Figure 22: Cluster management***

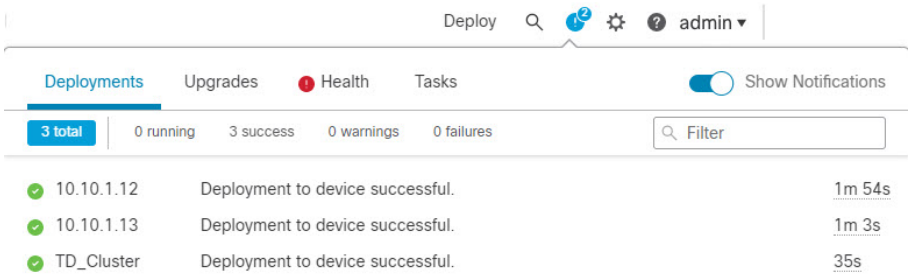


A unit that is currently registering shows the loading icon.

***Figure 23: Node registration***



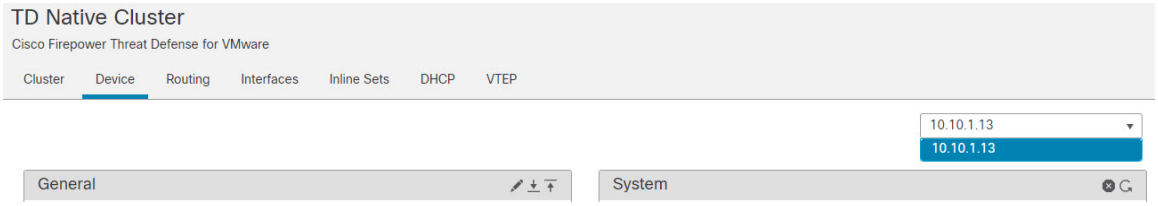

You can monitor cluster unit registration by clicking the **Notifications** icon and choosing **Tasks**. The Firewall Management Center updates the Cluster Registration task as each unit registers. If any units fail to register, see Reconcile Cluster Nodes, on page 117.




**Step 2** Configure device-specific settings by clicking the **Edit** (✎) for the cluster.

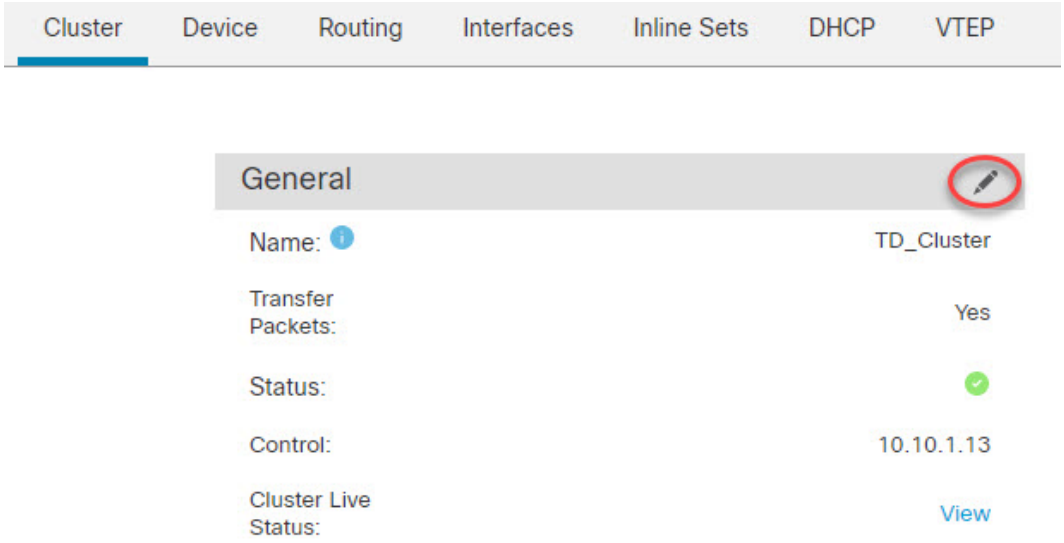
Most configuration can be applied to the cluster as a whole, and not nodes in the cluster. For example, you can change the display name per node, but you can only configure interfaces for the whole cluster.

**Step 3** On the **Devices** > **Device Management** and then choose **Add**, **Cluster** screen, you see **General**, **License**, **System**, and **Health** settings.

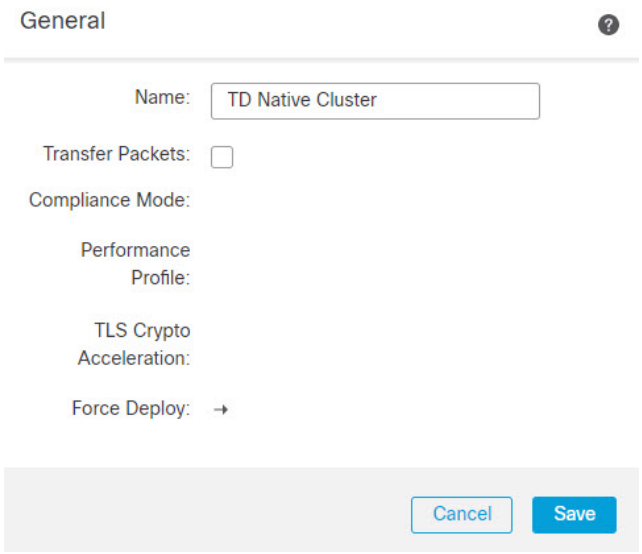



See these cluster-specific items:

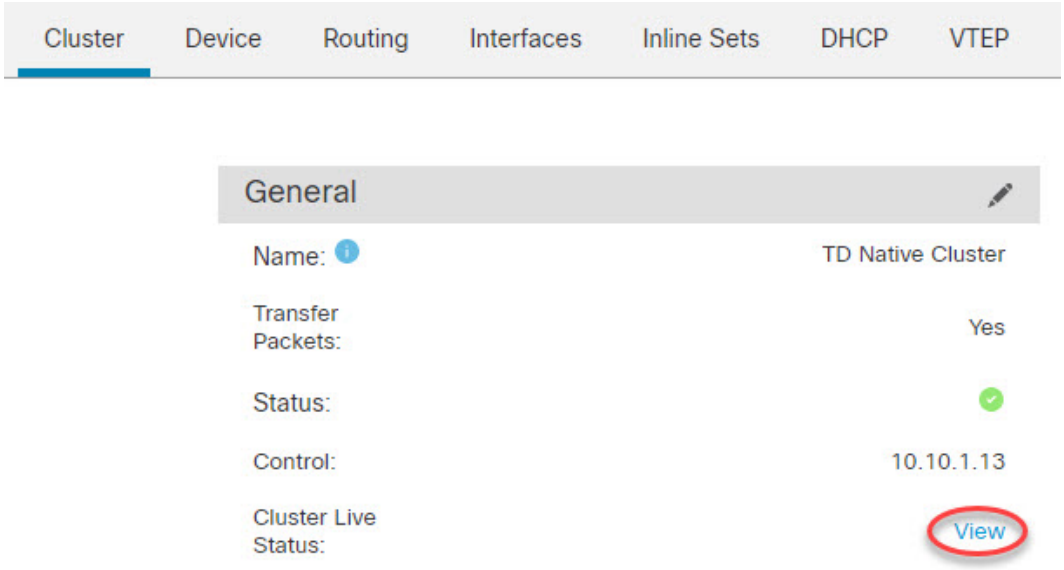
- **General > Name**—Change the cluster display name by clicking the **Edit** (✎).



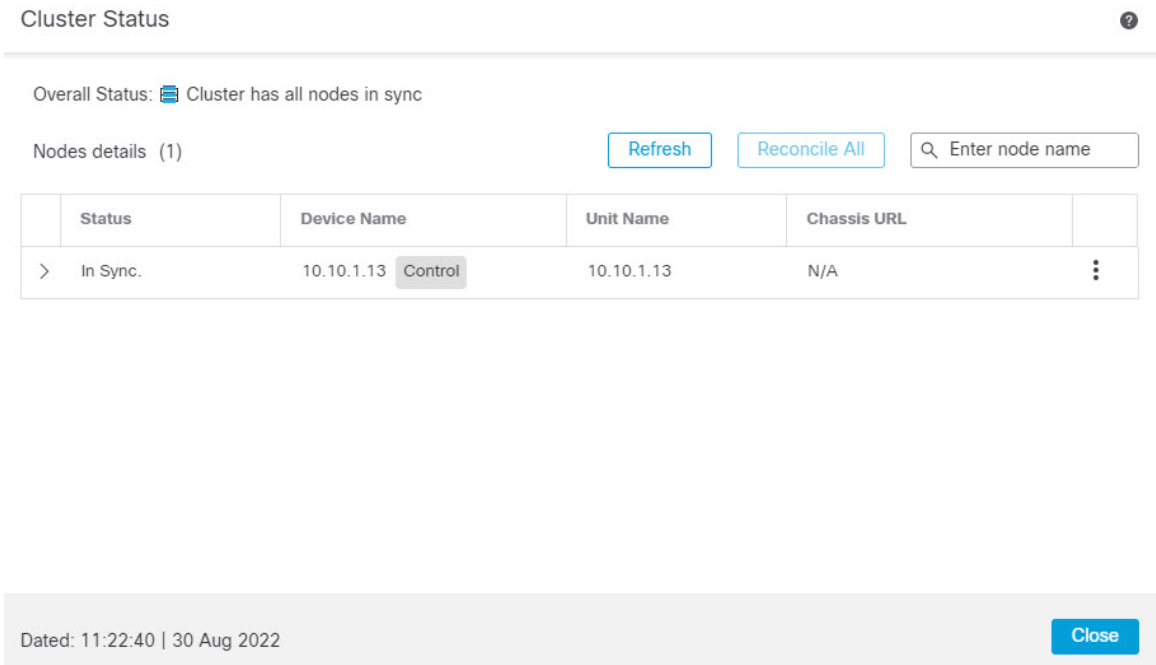

Then set the **Name** field.




- **General > Cluster Live Status**—Click the **View** link to open the **Cluster Status** dialog box.

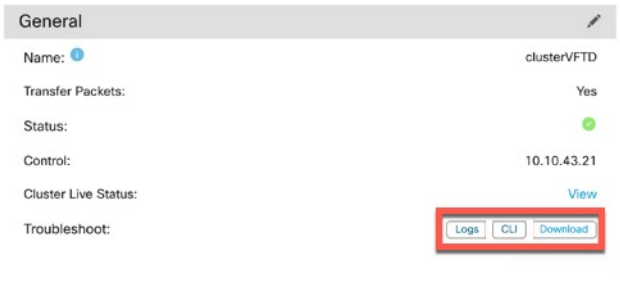



The **Cluster Status** dialog box also lets you retry data unit registration by clicking **Reconcile**.You can also ping the cluster control link from a node. See Perform a ping on the cluster control link.




- **General > Troubleshoot**—You can generate and download troubleshooting logs, and you can view cluster CLIs. See Troubleshoot the cluster.

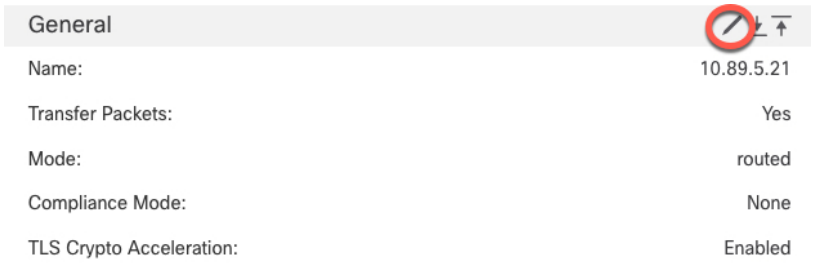
*Figure 24: Troubleshoot*



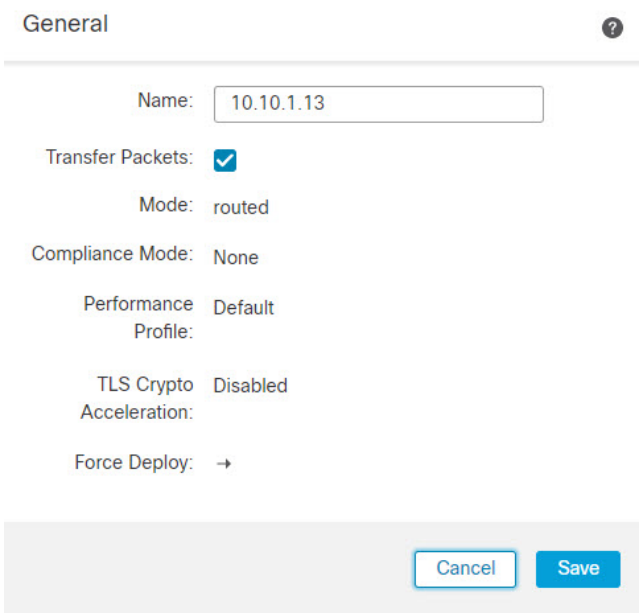

- **License**—Click **Edit** (✎) to set license entitlements.

**Step 4** On the **Devices** > **Device Management** and then click **Add** > **Device**, you can choose each member in the cluster from the top right drop-down menu and configure these settings.

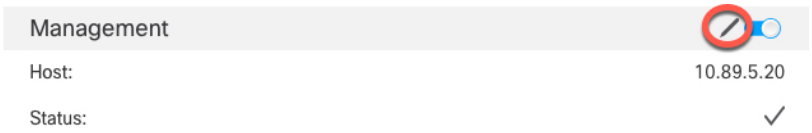
- **General > Name**—Change the cluster member display name by clicking the **Edit** (✎).




Then set the **Name** field.



- **Management > Host**—If you change the management IP address in the device configuration, you must match the new address in the Firewall Management Center so that it can reach the device on the network; edit the **Host** address in the **Management** area.




# Edit cluster health monitor settings

This task allows you to modify cluster health monitor settings to control how cluster nodes monitor system health and automatically rejoin the cluster after failures.

The **Cluster Health Monitor Settings** section of the **Cluster** page displays the settings for monitoring cluster node health. You can monitor any port-channel ID, any single physical interface ID, and the Snort and disk-full processes. Health monitoring is not performed on VLAN subinterfaces or virtual interfaces such as VNIs or BVIs. You cannot configure monitoring for the cluster control link. It is always monitored.

When a topology change occurs (such as adding or removing a data interface, enabling or disabling an interface on the node or the switch, or adding an additional switch to form a VSS or vPC or VNet) you should disable the system health check feature. Also, disable interface monitoring for any interfaces that are disabled. When the topology change is complete, and the configuration change is synced to all nodes, you can re-enable the system health check feature and monitored interfaces.

## Procedure

**Step 1** Choose **Devices** > **Device Management**.

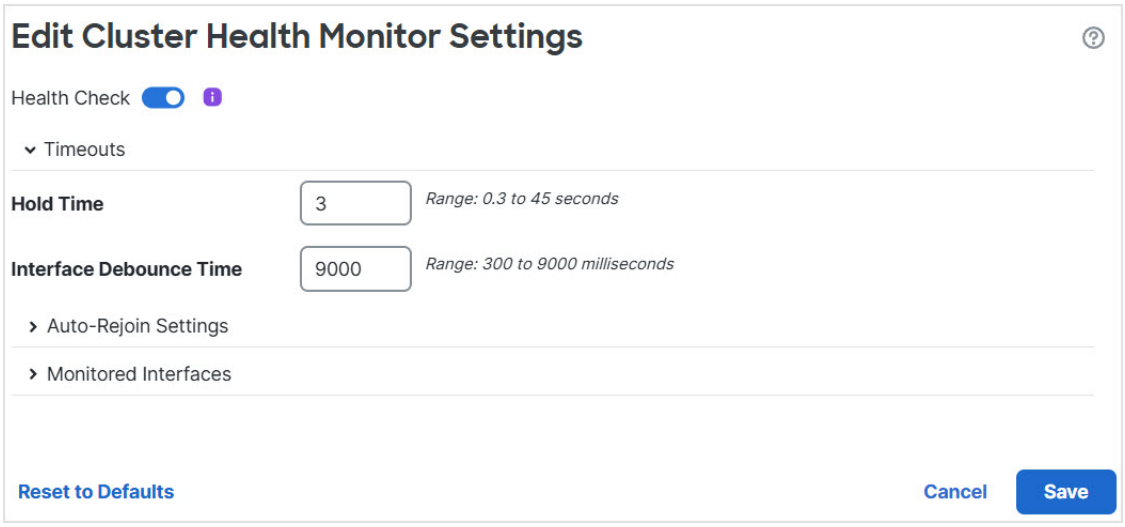
**Step 2** Next to the cluster you want to modify, click **Edit** (✎).

**Step 3** Click **Cluster**.

**Step 4** In the **Cluster Health Monitor Settings** section, click **Edit** (✎).

**Step 5** Disable the system health check by clicking the **Health Check** slider.

*Figure 25: Disable the system health check*



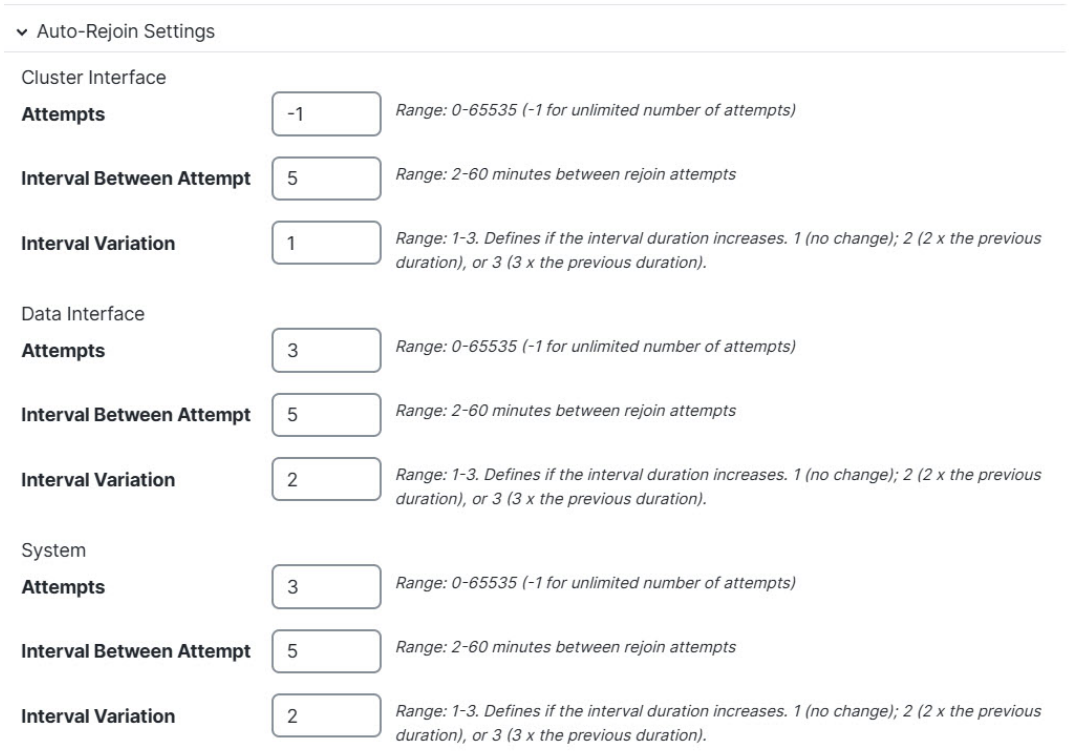

When any topology changes occur (such as adding or removing a data interface, enabling or disabling an interface on the node or the switch, or adding an additional switch to form a VSS or vPC or VNet) you should disable the system health check feature and also disable interface monitoring for the disabled interfaces. When the topology change is complete, and the configuration change is synced to all nodes, you can re-enable the system health check feature and monitored interfaces.

**Step 6** Configure the hold time and interface debounce time.

- **Hold Time**—Set the hold time to determine the amount of time between node heartbeat status messages, between 0.3 and 45 seconds. The default is 3 seconds.
- **Interface Debounce Time**—Set the debounce time between 300 and 9000 ms. The default is 500 ms. Lower values allow for faster detection of interface failures. Note that configuring a lower debounce time increases the chances of false-positives. When an interface status update occurs, the node waits the specified number of milliseconds before marking the interface as failed and removing the node from the cluster. If an EtherChannel transitions from a down state to an up state, such as after the switch reloads or enables an EtherChannel, a longer debounce time can prevent the interface from appearing failed on a cluster node if another cluster node bundles the ports more quickly.

**Step 7** Customize the auto-rejoin cluster settings after a health check failure.

*Figure 26: Configure Auto-Rejoin settings*

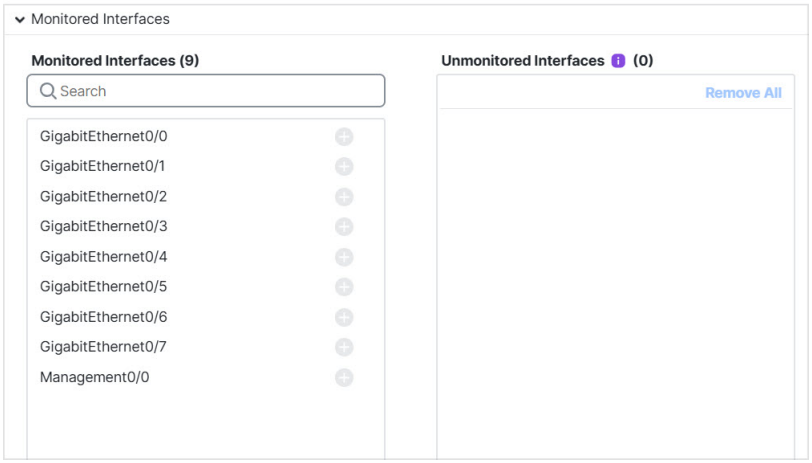



Set values for the **Cluster Interface**, **Data Interface**, and **System** (internal failures include: application sync timeout; inconsistent application statuses; and so on):

- **Attempts**—Sets the number of rejoin attempts, between -1 and 65535. **0** disables auto-rejoining. The default for the **Cluster Interface** is -1 (unlimited). The default for the **Data Interface** and **System** is 3.
- **Interval Between Attempts**—Defines the interval duration in minutes between rejoin attempts, between 2 and 60. The default value is 5 minutes. The maximum total time that the node attempts to rejoin the cluster is limited to 14400 minutes (10 days) from the time of last failure.
- **Interval Variation**—Defines if the interval duration increases. Set the value between 1 and 3: **1** (no change); **2** (2 x the previous duration), or **3** (3 x the previous duration). For example, if you set the interval duration to 5 minutes, and set the variation to 2, then the first attempt is after 5 minutes; the 2nd attempt is 10 minutes (2 x 5); the 3rd attempt 20 minutes (2 x 10), and so on. The default value is **1** for the **Cluster Interface** and **2** for the **Data Interface** and **System**.

**Step 8** Configure monitored interfaces by moving interfaces in the **Monitored Interfaces** or **Unmonitored Interfaces** window. You can also check or uncheck **Enable Service Application Monitoring** to enable or disable monitoring of the Snort and disk-full processes.

*Figure 27: Configure monitored interfaces*




The interface health check monitors for link failures. If all physical ports for a given logical interface fail on a particular node, but there are active ports under the same logical interface on other nodes, then the node is removed from the cluster. The time required to remove a member from the cluster depends on the type of interface and whether the node is established or is joining the cluster. By default, health check is enabled for all interfaces and the Snort and disk-full processes.

You might want to disable health monitoring of non-essential interfaces.

When any topology changes occur (such as adding or removing a data interface, enabling or disabling an interface on the node or the switch, or adding an additional switch to form a VSS or vPC or VNet) you should disable the system health check feature and also disable interface monitoring for the disabled interfaces. When the topology change is complete, and the configuration change is synced to all nodes, you can re-enable the system health check feature and monitored interfaces.

**Step 9** Click **Save**.

Deploy configuration changes; see Deploy configuration changes.

# Manage cluster nodes

Explains how to add, remove, and configure cluster nodes in your deployment.

## Disable clustering

Disable clustering to temporarily deactivate a node in preparation for deleting the node or for maintenance purposes.

You may want to deactivate a node to delete it or for temporary maintenance. This procedure temporarily deactivates a node, but the node will still appear in the Firewall Management Center device list. When a node becomes inactive, all data interfaces are shut down.

**Note** Do not power off the node without first disabling clustering.

**Procedure**

**Step 1** For the unit you want to disable, choose **Devices** > **Device Management**, click the **More** (⋮), and choose **Disable Node Clustering**.

**Step 2** Confirm that you want to disable clustering on the node.

The node will show **(Disabled)** next to its name in the **Devices** > **Device Management** list.

**Step 3** To reenable clustering, refer to Rejoin the Cluster.

# Rejoin the cluster

Rejoin a node to the cluster after it was removed because of failure or when clustering was manually disabled.

If a node was removed from the cluster, for example, due to a failed interface or if you manually disabled clustering, you must manually rejoin the cluster. Make sure the failure is resolved before you try to rejoin the cluster. Refer to Rejoining the cluster for more information about why a node can be removed from a cluster.

**Procedure**

**Step 1** For the unit you want to reactivate, choose **Devices** > **Device Management**, click **More** (⋮), and choose **Enable Node Clustering**.

**Step 2** Confirm that you want to enable clustering on the node.

# Reconcile Cluster Nodes

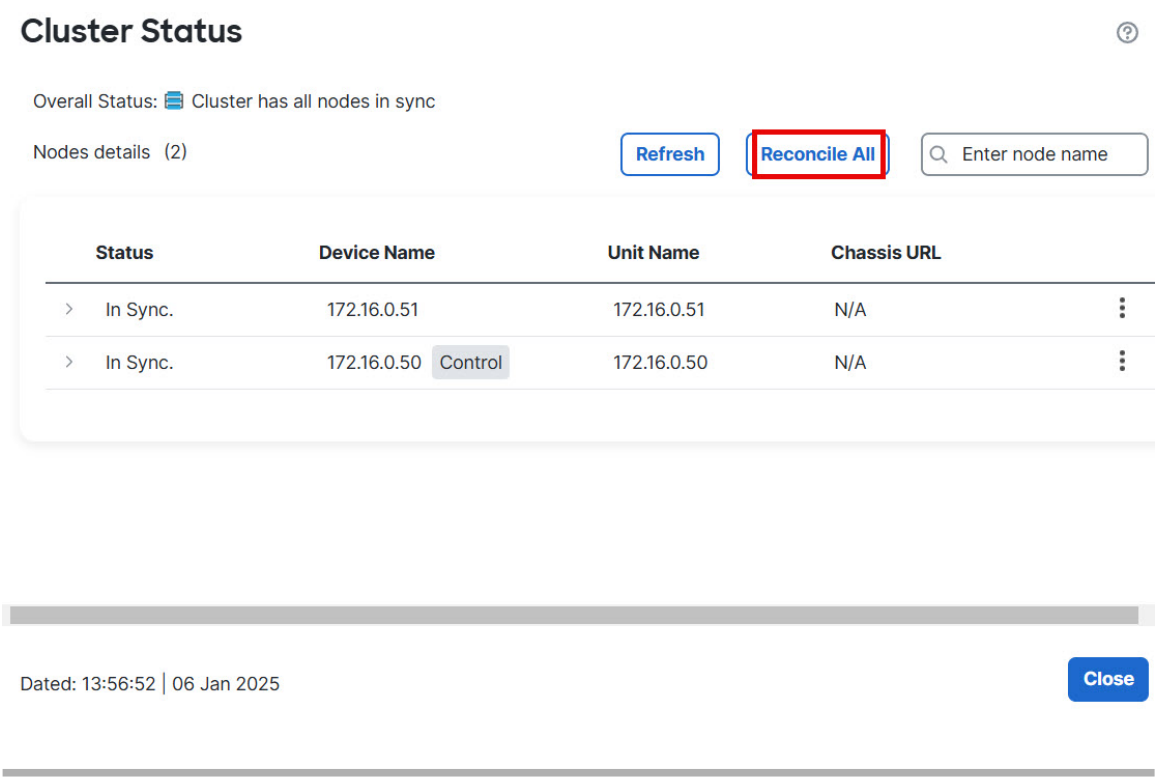
If a cluster node fails to register, you can reconcile the cluster membership from the device to the Firewall Management Center. For example, a data node might fail to register if the Firewall Management Center is occupied with certain processes, or if there is a network issue.

**Procedure**

**Step 1** Choose **Devices** > **Device Management More** (⋮) for the cluster, and then choose **Cluster Live Status** to open the **Cluster Status** dialog box.

**Step 2** Click **Reconcile All**.

*Figure 28: Reconcile All*




# Unregister the cluster or nodes and register with a new Firewall Management Center

This task allows you to move cluster nodes from one management center to another by safely removing them from their current management center and re-registering them with a new one.

You can unregister the cluster from the Firewall Management Center, which keeps the cluster intact. You might want to unregister the cluster if you want to add the cluster to a new Firewall Management Center.

You can also unregister a node from the Firewall Management Center without breaking the node from the cluster. Although the node is not visible in the Firewall Management Center, it is still part of the cluster, and it will continue to pass traffic and could even become the control node. You cannot unregister the current control node. You might want to unregister the node if it is no longer reachable from the Firewall Management Center, but you still want to keep it as part of the cluster while you troubleshoot management connectivity.

Unregistering a cluster:

- Severs all communication between the Firewall Management Center and the cluster.
- Removes the cluster from the **Device Management** page.
- Returns the cluster to local time management if the cluster's platform settings policy is configured to receive time from the Firewall Management Center using NTP.
- Leaves the configuration intact, so the cluster continues to process traffic.

  Policies, such as NAT and VPN, ACLs, and the interface configurations remain intact.

Registering the cluster again to the same or a different Firewall Management Center causes the configuration to be removed, so the cluster will stop processing traffic at that point; the cluster configuration remains intact so you can add the cluster as a whole. You can choose an access control policy at registration, but you will have to re-apply other policies after registration and then deploy the configuration before it will process traffic again.

**Before you begin**

This procedure requires CLI access to one of the nodes.

**Procedure**

**Step 1** Choose **Devices** > **Device Management**, click **More** (⋮) for the cluster or node, and choose **Unregister**.

**Step 2** You are prompted to unregister the cluster or node; click **Yes**.

**Step 3** You can register the cluster with a new (or the same) Firewall Management Center by adding one of the cluster members as a new device.

Add one of the cluster nodes as a device. The system will discover the remaining nodes.

a) Connect to the CLI of a cluster node, and identify the new Firewall Management Center using the **configure manager add** command.
b) Choose **Devices** > **Device Management**, and then click **Add Device**.

**Step 4** To re-add a deleted node, refer to Reconcile Cluster Nodes, on page 117.

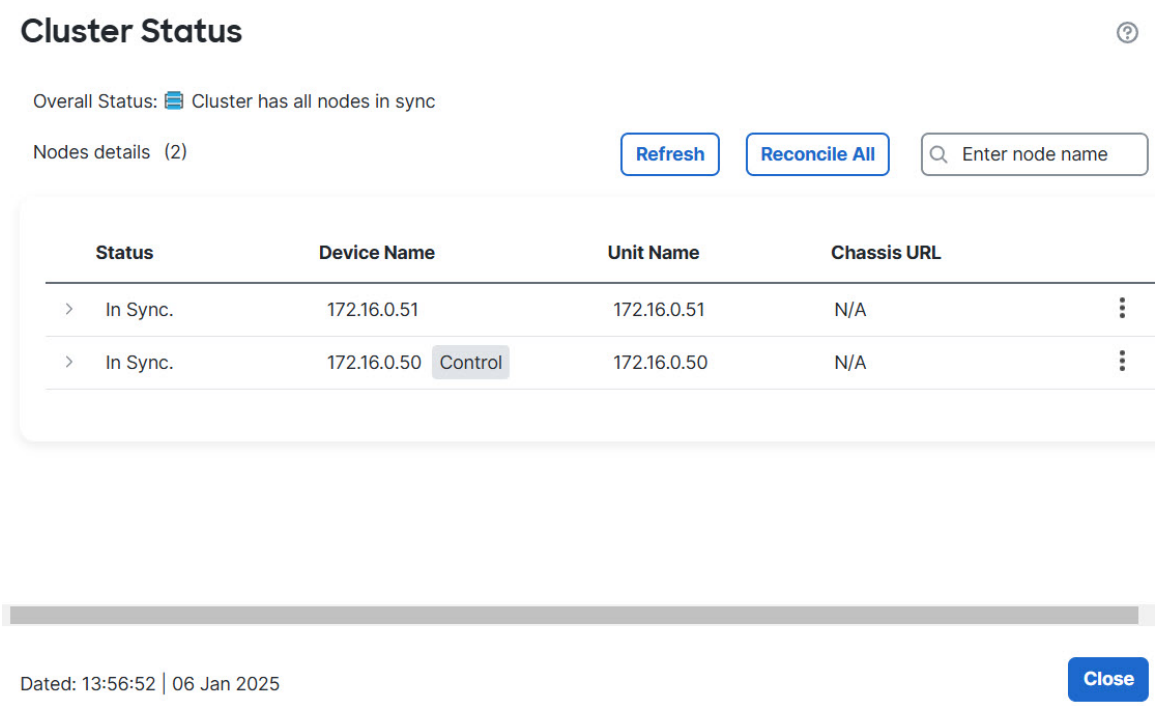
# Monitor cluster status

You can monitor the cluster in the Firewall Management Center and at the Firewall Threat Defense CLI.

Monitor cluster status through these methods:

- **Cluster Status** dialog box, which is available from the **Devices** > **Device Management**, **More** (⋮) icon or from the **Devices** > **Device Management**, click **Add**, choose the **Cluster** page **General** area **Cluster Live Status** link.

*Figure 29: Cluster status*




The Control node has a graphic indicator identifying its role.

Cluster member **Status** includes these states:

- In Sync—The node is registered with the Firewall Management Center.
- Pending Registration—The node is part of the cluster, but has not yet registered with the Firewall Management Center. If a node fails to register, you can retry registration by clicking **Reconcile All**.
- Clustering is disabled—The node is registered with the Firewall Management Center, but is an inactive member of the cluster. The clustering configuration remains intact if you intend to later re-enable it, or you can delete the node from the cluster.
- Joining cluster...—The node is joining the cluster on the chassis, but has not completed joining. After it joins, it will register with the Firewall Management Center.

For each node, you can view the **Summary** or the **History**.

*Figure 30: Node summary*

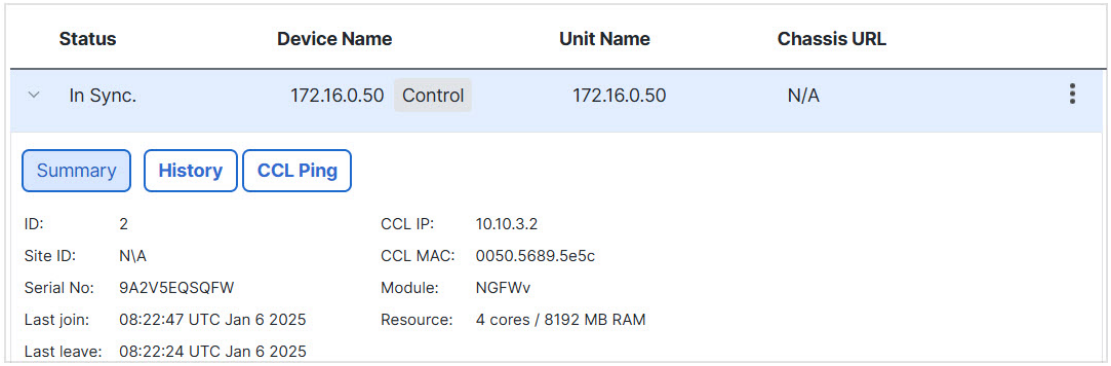
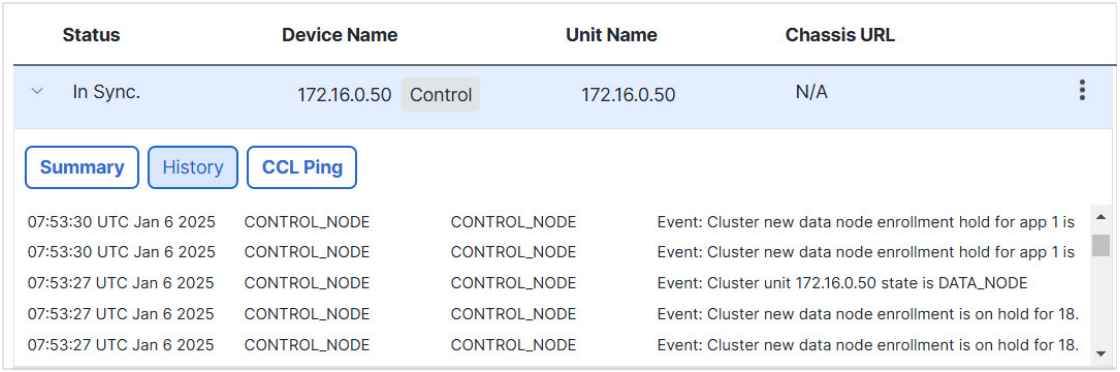



*Figure 31: Node history*




- **System** (⚙) > **Tasks** page.

  The **Tasks** page shows updates of the Cluster Registration task as each node registers.

- **Devices** > **Device Management** and then click **Add** > **Device** *cluster_name*.

  When you expand the cluster on the devices listing page, you can see all member nodes, including the control node shown with its role next to the IP address. For nodes that are still registering, you can see the loading icon.

- **show cluster** {**access-list** [*acl_name*] | **conn** [**count**] | **cpu** [**usage**] | **history** | **interface-mode** | **memory** | **resource usage** | **service-policy** | **traffic** | **xlate count**}

  To view aggregated data for the entire cluster or other information, use the **show cluster** command.

- **show cluster info** [**auto-join** | **clients** | **conn-distribution** | **flow-mobility counters** | **goid** [*options*] | **health** | **incompatible-config** | **loadbalance** | **old-members** | **packet-distribution** | **trace** [*options*] | **transport** { **asp** | **cp**}]

  To view cluster information, use the **show cluster info** command.

# Cluster health monitor

When a Firewall Threat Defense is the control node of a cluster, the Firewall Management Center collects various metrics periodically from the device metric data collector.

The cluster health monitor is comprised of these components:

- Overview dashboard: Displays information about the cluster topology, cluster statistics, and metric charts:
  - The topology section displays a cluster's live status, the health of individual threat defense, threat defense node type (control node or data node), and the status of the device. The status of the device could be *Disabled* (when the device leaves the cluster), *Added out of box* (in a public cloud cluster, the additional nodes that do not belong to the Firewall Management Center), or *Normal* (ideal state of the node).
  - The cluster statistics section displays current metrics of the cluster with respect to the CPU usage, memory usage, input rate, output rate, active connections, and NAT translations.



**Note** The CPU and memory metrics display the individual average of the data plane and snort usage.

  - The metric charts, namely, CPU Usage, Memory Usage, Throughput, and Connections, diagrammatically display the statistics of the cluster over the specified time period.

- Load Distribution dashboard: Displays load distribution across the cluster nodes in two widgets:
  - The Distribution widget displays the average packet and connection distribution over the time range across the cluster nodes. This data depicts how the load is being distributed by the nodes. Using this widget, you can easily identify any abnormalities in the load distribution and rectify it.
  - The Node Statistics widget displays the node level metrics in table format. It displays metric data on CPU usage, memory usage, input rate, output rate, active connections, and NAT translations across the cluster nodes. This table view enables you to correlate data and easily identify any discrepancies.
- Member Performance dashboard: Displays current metrics of the cluster nodes. You can use the selector to filter the nodes and view the details of a specific node. The metric data include CPU usage, memory usage, input rate, output rate, active connections, and NAT translations.
- CCL dashboard: Displays, graphically, the cluster control link data namely, the input, and output rate.
- Troubleshooting and Links: Provides convenient links to frequently used troubleshooting topics and procedures.
- Time range: An adjustable time window to constrain the information that appears in the various cluster metrics dashboards and widgets.
- Custom Dashboard: Displays data on both cluster-wide metrics and node-level metrics. However, node selection only applies for the threat defense metrics and not for the entire cluster to which the node belongs.

## View the cluster health monitor

The cluster health monitor provides a detailed view of the health status of a cluster and its nodes. This cluster health monitor provides health status and trends of the cluster in an array of dashboards.

**Before you begin**

Ensure you have created a cluster from one or more devices in the Firewall Management Center.

Follow these steps to view the cluster health monitor:

## Procedure

**Step 1** Choose **Troubleshooting** > **Health** > **Monitor**.

Use the Monitoring navigation pane to access node-specific health monitors.

**Step 2** In the device list, click **Expand**(>) and **Collapse** (V) to expand and collapse the list of managed cluster devices.

**Step 3** Click on the cluster name to view the cluster health statistics.

The cluster monitor reports health and performance metrics in several predefined dashboards by default. The metrics dashboards include:

- Overview: Highlights key metrics from the other predefined dashboards, including its nodes, CPU, memory, input and output rates, connection statistics, and NAT translation information.
- Load Distribution: Traffic and packet distribution across the cluster nodes.
- Member Performance: Node-level statistics on CPU usage, memory usage, input throughput, output throughput, active connection, and NAT translation.
- CCL: Interface status and aggregate traffic statistics.

You can navigate through the various metrics dashboards by clicking on the labels. For a comprehensive list of the supported cluster metrics, refer to Cisco Secure Firewall Threat Defense Health Metrics.

**Step 4** Configure the time range from the drop-down in the upper-right corner.

The time range can reflect a period as short as the last hour (the default) or as long as two weeks. Select **Custom** from the drop-down to configure a custom start and end date.

Click the refresh icon to set auto refresh to 5 minutes or to toggle off auto refresh.

**Step 5** Click on deployment icon for a deployment overlay on the trend graph, with respect to the selected time range.

The deployment icon indicates the number of deployments during the selected time-range. A vertical band indicates the deployment start and end time. For multiple deployments, multiple bands/lines appear. Click on the icon on top of the dotted line to view the deployment details.

**Step 6** View the **Health Alerts** for the node in the alert notification at the top of page, directly to the right of the device name.

Hover your pointer over the **Health Alerts** to view the health summary of the node. The popup window shows a truncated summary of the top five health alerts. Click on the popup to open a detailed view of the health alert summary.

**Step 7** View device monitor health and performance metrics in several predefined dashboards.

The metrics dashboards include:

- Overview: Highlights key metrics from the other predefined dashboards, including CPU, memory, interfaces, connection statistics; plus disk usage and critical process information.

- CPU: CPU utilization, including the CPU usage by process and by physical cores.
- Memory: Device memory utilization, including data plane and Snort memory usage.
- Interfaces: Interface status and aggregate traffic statistics.
- Connections: Connection statistics (such as elephant flows, active connections, peak connections, and so on) and NAT translation counts.
- Snort: Statistics that are related to the Snort process.
- ASP drops: Statistics related to the dropped packets against various reasons.

You can navigate through the various metrics dashboards by clicking on the labels. Refer to Cisco Secure Firewall Threat Defense Health Metrics for a comprehensive list of the supported device metrics.

**Step 8** Click the plus sign **Add New Dashboard**(+) in the upper right corner of the health monitor to create a custom dashboard by building your own variable set from the available metric groups.

For cluster-wide dashboard, choose Cluster metric group, and then choose the metric.

## Cluster metrics

The cluster health monitor tracks statistics that are related to a cluster and its nodes, and aggregate of load distribution, performance, and CCL traffic statistics.

***Table 10: Cluster Metrics***

| Metric | Description | Format |
|---|---|---|
| CPU | Average of CPU metrics on the nodes of a cluster (individually for data plane and snort). | percentage |
| Memory | Average of memory metrics on the nodes of a cluster (individually for data plane and snort). | percentage |
| Data Throughput | Incoming and outgoing data traffic statistics for a cluster. | bytes |
| CCL Throughput | Incoming and outgoing CCL traffic statistics for a cluster. | bytes |
| Connections | Count of active connections in a cluster. | number |
| NAT Translations | Count of NAT translations for a cluster. | number |
| Distribution | Connection distribution count in the cluster for every second. | number |
| Packets | Packet distribution count in the cluster for every second. | number |

# Troubleshoot the cluster

Use these methods to troubleshoot the devices and clusters:

- Use **Logs** to generates troubleshooting files for all nodes in the cluster. Use this option when you need one cluster-level collection instead of generating troubleshooting files from each node separately.
- Use **CLI** to run a predefined set of cluster diagnostic commands and displays the output. Use this option to check cluster configuration, member state, cluster health, control-plane transport statistics, packet drops, interface status, CPU usage, memory buffers, ARP entries, and reachability. The **CLI** option displays command output; it does not open an interactive CLI session.

| Commands |
| --- |
| **show running-config cluster**: Review the cluster group configuration, local unit, cluster interface, priority, health-check, auto-rejoin, and cLACP settings. |
| **show cluster info**: Confirm that clustering is enabled and identify the control node, member state, software version, serial number, CCL IP address, CCL MAC address, and last join or leave state. |
| **show cluster info health**: Check unit, interface, service, and overall cluster health. |
| **show cluster info transport cp**: Review control-plane transport counters, including reliable and unreliable message counters, errors, retransmits, drops, and buffer conditions. |
| **show version**: Capture platform, software, build, UUID, LSP, VDB, uptime, resource, and license context. |
| **show asp drop**: Identify accelerated security path packet or flow drop reasons and counters. |
| **show counters**: Review protocol, feature, and cluster CCL counters. |
| **show arp**: Verify ARP entries for interfaces that participate in cluster communication. |
| **show int ip brief**: Check interface IP addresses and administrative and line protocol status. |
| **show blocks**: Check buffer availability and failed allocations. |
| **show cpu detailed**: Check data path, control point, and total CPU usage. |
| **show interface** *ccl_interface*: Review interface state, MTU, MAC address, IP address, traffic counters, errors, and drops for the cluster-related interface. |
| **ping** *ccl_ip* **size** *ccl_mtu* **repeat 2**: Test reachability to a cluster peer by using the packet size selected by the tool. |

- Use the **CCL Ping** when you need to test cluster control link connectivity between joined cluster nodes.
- Troubleshooting files—If a node fails to join the cluster, a troubleshooting file is automatically generated. You can also generate and download troubleshooting files from the **Devices** > **Device Management** and then choose **Add**, **ClusterGeneral** area. For more information, refer to Generate troubleshooting files.

  You can also generate files from the **Device Management** page. Click **More** (⋮) and choose **Troubleshoot Files**.

- CLI output—Choose **Devices** > **Device Management** and then choose **Add**, **ClusterGeneral** area, you can view a set of pre-defined CLI outputs that can help you troubleshoot the cluster. These commands are automatically run for the cluster:

  You can also enter any **show** command in the **Command** field.

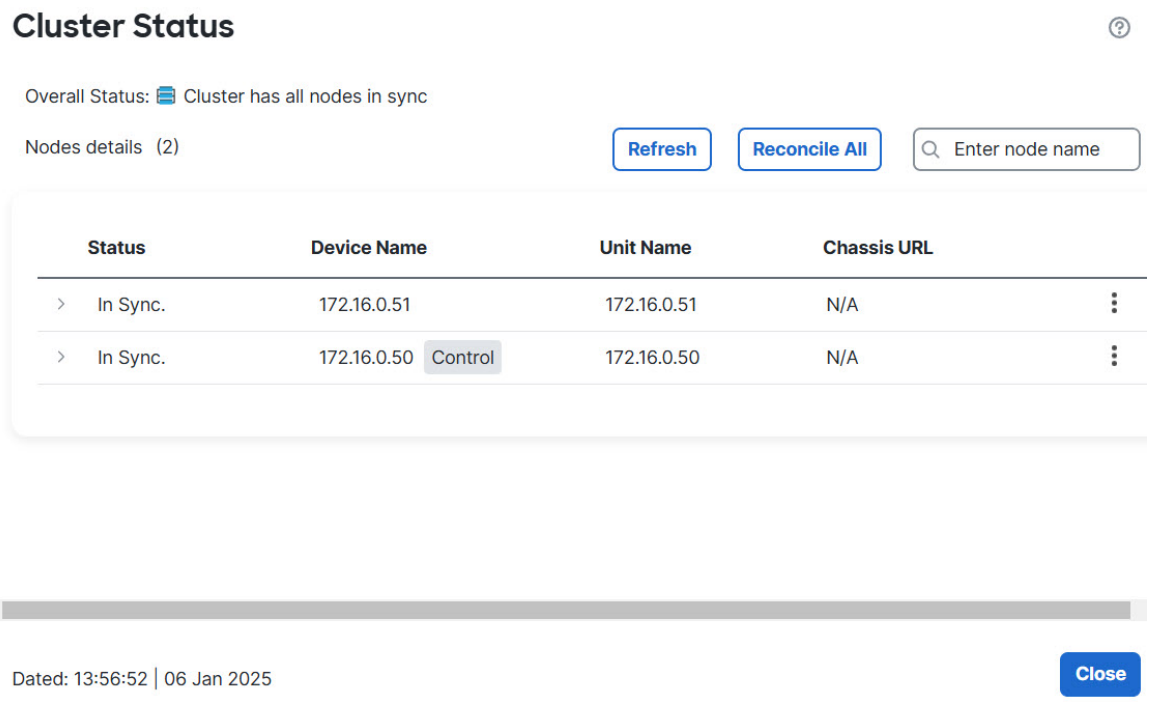
# Perform a ping on the cluster control link

When a node joins the cluster, it checks MTU compatibility by sending a ping to the control node with a packet size matching the cluster control link MTU. If the initial ping fails, the node tries a ping using a smaller packet size (the MTU divided by 2, then by 4, then by 8) until a ping succeeds. A notification is generated so you can fix the MTU mismatch on connecting switches and try again. This tool lets you manually ping all nodes that have already joined the cluster in case you are having cluster control link connectivity problems.

## Procedure

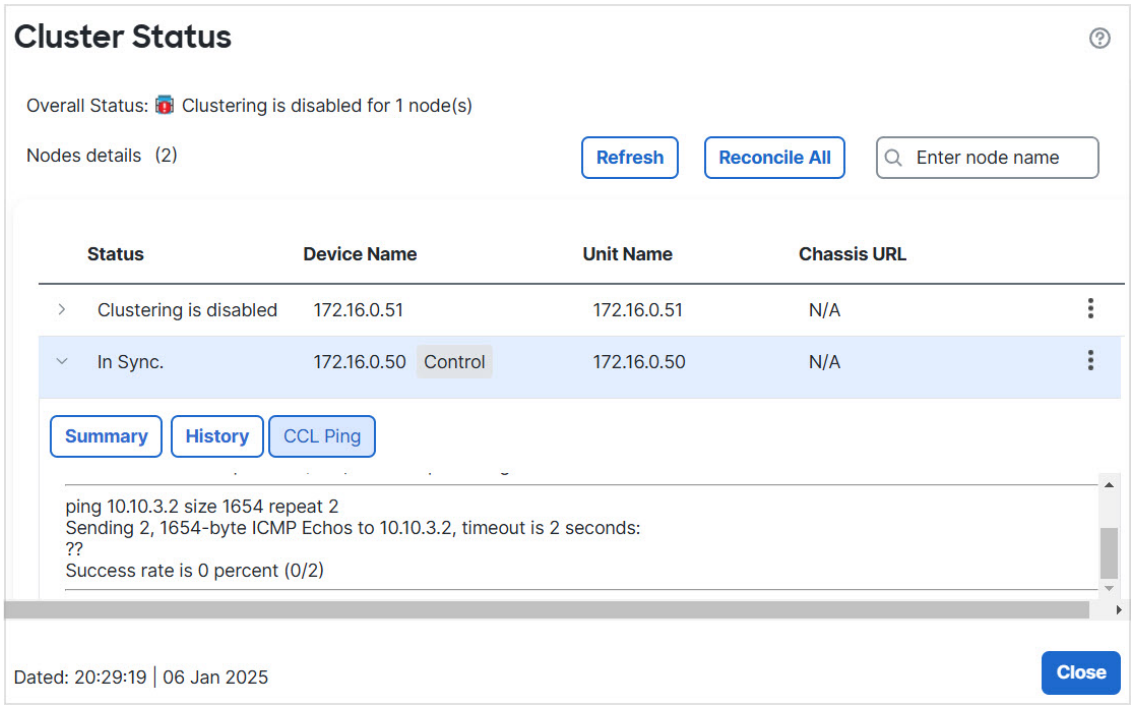
**Step 1** Choose **Devices** > **Device Management**, click the **More** (⋮) icon next to the cluster, and choose **Cluster Live Status**.

***Figure 32: Cluster status***




**Step 2** Expand one of the nodes, and click **CCL Ping**.

*Figure 33: CCL ping*




The node sends a ping on the cluster control link to every other node using a packet size that matches the maximum MTU.

# Upgrade the cluster

This task upgrades a Firewall Threat Defense Virtual cluster to a new software version by updating the cloud instance template and performing a managed upgrade.

The cluster upgrade process involves updating both the cloud infrastructure template and the software package through the management center. The upgrade is performed one node at a time to maintain cluster availability.

**Before you begin**

- Before you upgrade a cluster in the public cloud, copy the target version image to your cloud image repository and update the image ID in the cluster deployment template (we actually recommend replacing the existing template with a modified copy). This ensures that after the upgrade, new instances — for example, instances launched during cluster scaling — will use the correct version. If the marketplace does not have the image you need, such as when the cluster has been patched, create a custom image from a snapshot of a standalone Firewall Threat Defense Virtual instance running the correct version, with no instance-specific (day 0) configurations.

Follow these steps to upgrade a Firewall Threat Defense Virtual cluster:

**Procedure**

**Step 1** Upload the target image version to the cloud image storage.

**Step 2** Update the cloud instance template of the cluster with the updated target image version.

a) Create a copy of the instance template with the target image version.

b) Attach the newly created template to cluster instance group.



**Note** If the user wants to retain the old interface naming convention, use the "IfNamingConvention": "Old" key-value pair in the Day 0 configuration.

**Step 3** Upload the target image version upgrade package to the Firewall Management Center.

**Step 4** Perform a readiness check on the cluster that you want to upgrade.

**Step 5** After a successful readiness check, initiate the installation of the upgrade package.

**Step 6** The Firewall Management Center upgrades the cluster nodes one at a time.

**Step 7** The Firewall Management Center displays a notification after successful upgrade of the cluster.

There is no change to the serial number and UUID of the instance after the upgrade.

# Cluster replacement upgrade procedure

Secure Firewall 7.6.4-69 or later 7.6.x versions, and 10.0.0 or later versions, use a new cluster interface addressing convention. Existing clusters retain their legacy convention after an upgrade, but newly deployed members use the new convention and cannot join.

To add capacity, build a replacement cluster and migrate traffic:

**Procedure**

**Step 1** Keep the existing cluster serving traffic.

**Step 2** Deploy the replacement cluster at its final size, using a new CCL subnet. Use Secure Firewall version 7.6.4-69 or later 7.6.x versions, or version 10.0.0 or later. Keep all members on the same build, assign **cluster-node-id** tags from 1 to N, and enable metadata tags.

**Step 3** Confirm the new cluster is healthy. Register it with the Management Center, apply the equivalent policy, and deploy.

**Step 4** Edit the target group attributes before registering the replacement nodes. Under **Target configuration**, set **Target failover strategy** to **No rebalance**. Under **Target deregistration management**, set **Deregistration delay** (draining interval) to a value longer than your longest-lived flow. The maximum allowed value is 3600 seconds.

**Note**

- Set target failover first: Target groups are created with Rebalance existing flows. With rebalancing enabled, deregistering the old nodes moves their established flows to the remaining healthy targets in the replacement cluster. That cluster has no state for those flows and drops them. No rebalance allows existing flows to drain.
- Deregistration delay is a timer: The load balancer completes deregistration when the draining interval expires, regardless of whether connections remain. A connection count of zero does not extend the interval.

**Step 5** Register the new nodes with the existing target group, then deregister the old nodes. New traffic flows use the healthy replacement nodes. Existing flows on the old nodes drain.

**Step 6** After the old cluster's connection count reaches zero, remove its targets and instances, then unregister it from the Management Center.

**Note**

- Check connections: Use the **show conn count** command on the old devices. The load balancer reports flows for the entire target group and cannot show which cluster owns each flow while both are registered.
- Rollback: Keep the old cluster until the replacement is validated. If needed, restore traffic to the old cluster before deleting it.
- Record the build: Note the replacement cluster's release and build to select compatible versions for future nodes.
- Observe the version boundary: Use version 7.6.4-69 or later 7.6.x versions, or version 10.0.0 or later versions. Older 7.6.4 builds use the legacy addressing scheme.
- Plan for the overlap: Both clusters run concurrently. Confirm that Smart Licensing entitlements and vCPU quota cover the combined node count.

# Reference for clustering

This section includes more information about how clustering operates.

## Threat Defense features and clustering

Some Firewall Threat Defense features are not supported with clustering, and some are only supported on the control unit. Other features might have caveats for proper usage.

# Unsupported features and clustering

## Centralized features for clustering

These features are only supported on the control node, and are not scaled for the cluster.

**Note** Traffic for centralized features is forwarded from member nodes to the control node over the cluster control link.

If you use the rebalancing feature, traffic for centralized features may be rebalanced to non-control nodes before the traffic is classified as a centralized feature; if this occurs, the traffic is then sent back to the control node.

For centralized features, if the control node fails, all connections are dropped, and you have to re-establish the connections on the new control node.

**Note** To view FlexConfig features that are also centralized with clustering, for example RADIUS inspection, see the ASA general operations configuration guide. FlexConfig lets you configure many ASA features that are not present in the Firewall Management Center GUI. For more information, refer to FlexConfig policies.

- These application inspections are supported:
  - DCERPC and ESMTP
  - NetBIOS and PPTP
  - RSH
  - SQLNET and SUNRPC
  - TFTP
  - XDMCP
- Static route monitoring
- Site-to-site VPN
- IGMP multicast control plane protocol processing (data plane forwarding is distributed across the cluster)
- PIM multicast control plane protocol processing (data plane forwarding is distributed across the cluster)
- Dynamic routing

## Cisco Trustsec and clustering

Only the control node learns security group tag (SGT) information. The control node then populates the SGT to data nodes, and data nodes can make a match decision for SGT based on the security policy.

## Connection settings and clustering

Connection limits are enforced cluster-wide. Each node has an estimate of the cluster-wide counter values based on broadcast messages. Due to efficiency considerations, the configured connection limit across the cluster might not be enforced exactly at the limit number. Each node may overestimate or underestimate the cluster-wide counter value at any given time. However, the information will get updated over time in a load-balanced cluster.

## Dynamic routing and clustering

Dynamic routing and clustering are network configuration approaches that

- enable each node to run the routing protocol as a standalone router in Individual interface mode,
- allow routes to be learned by each node independently, and
- require separate router IDs for each node when communicating with external routers.

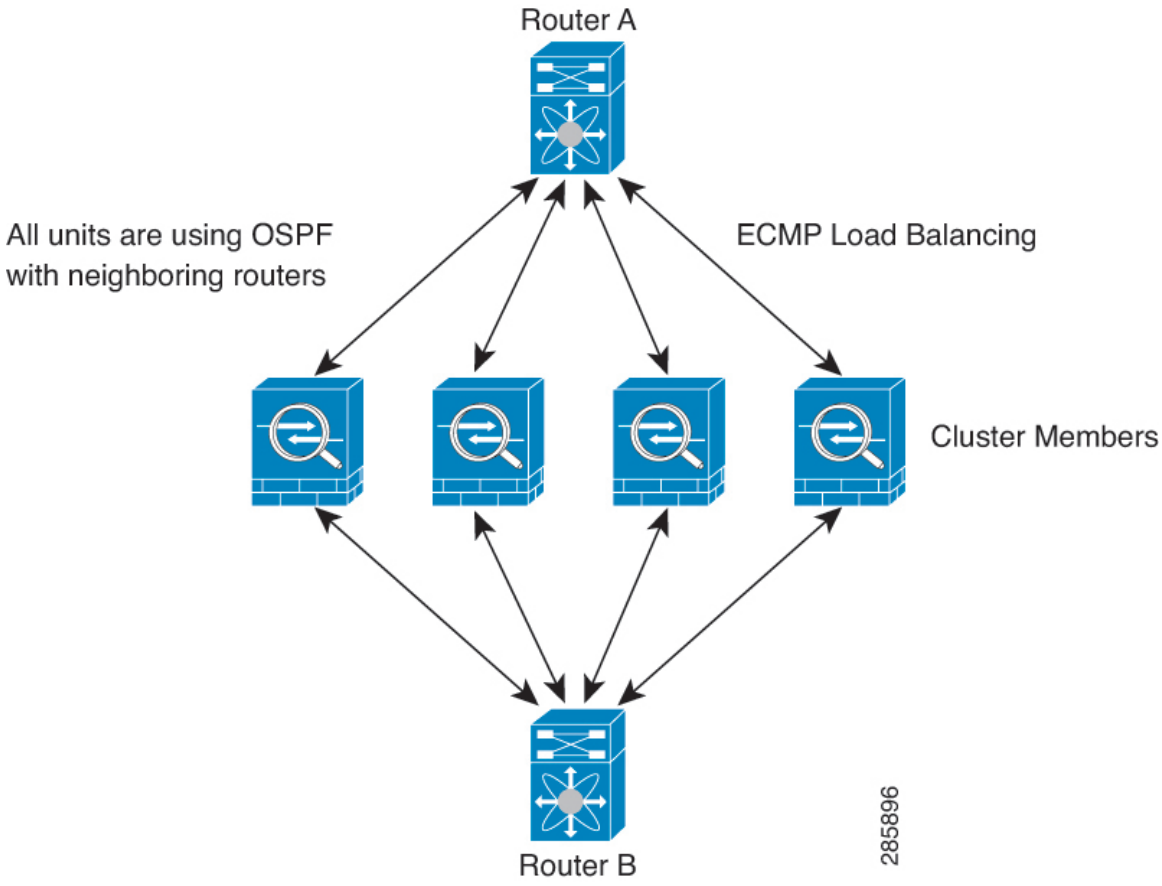
### Routing behavior and load balancing

In Individual interface mode, ECMP is used to load balance traffic between multiple equal-cost paths. External routers can discover multiple paths to destinations through different cluster nodes.

Router A learns that there are 4 equal-cost paths to Router B, each through a node. ECMP is used to load balance traffic between the 4 paths. Each node picks a different router ID when communicating with external routers.

You must configure a cluster pool for the router ID so that each node has a separate router ID.

*Figure 34: Dynamic routing in individual interface mode*




## FTP and clustering

- If FTP data channel and control channel flows are owned by different cluster members, then the data channel owner will periodically send idle timeout updates to the control channel owner and update the idle timeout value. However, if the control flow owner is reloaded, and the control flow is re-hosted, the parent/child flow relationship will not longer be maintained; the control flow idle timeout will not be updated.

## NAT and clustering

Provides information about NAT behavior and limitations in clustering deployments.

NAT can affect the overall throughput of the cluster. Inbound and outbound NAT packets can be sent to different Firewall Threat Defenses in the cluster, because the load balancing algorithm relies on IP addresses and ports, and NAT causes inbound and outbound packets to have different IP addresses and/or ports. When a packet arrives at the Firewall Threat Defense that is not the NAT owner, it is forwarded over the cluster control link to the owner, causing large amounts of traffic on the cluster control link. Note that the receiving node does not create a forwarding flow to the owner, because the NAT owner may not end up creating a connection for the packet depending on the results of security and policy checks.

If you still want to use NAT in clustering, then consider the following guidelines:

- No Proxy ARP—For individual interfaces, a proxy ARP reply is never sent for mapped addresses. This prevents the adjacent router from maintaining a peer relationship with an ASA that may no longer be in the cluster. The upstream router requires a static route or policy-based routing (PBR) with Object Tracking for the mapped addresses, which must point to the main cluster IP address.
- No interface PAT on an individual interface—Interface PAT is not supported for Individual interfaces.
- PAT with Port Block Allocation—See the following guidelines for this feature:
  - Maximum-per-host limit is not a cluster-wide limit, and is enforced on each node individually. Thus, in a 3-node cluster with the maximum-per-host limit configured as 1, if the traffic from a host is load-balanced across all 3 nodes, then it can get allocated 3 blocks with 1 in each node.
  - Port blocks created on the backup node from the backup pools are not accounted for when enforcing the maximum-per-host limit.
  - On-the-fly PAT rule modifications, where the PAT pool is modified with a completely new range of IP addresses, will result in xlate backup creation failures for the xlate backup requests that were still in transit while the new pool became effective. This behavior is not specific to the port block allocation feature, and is a transient PAT pool issue seen only in cluster deployments where the pool is distributed and traffic is load-balanced across the cluster nodes.
  - When operating in a cluster, you cannot simply change the block allocation size. The new size is effective only after you reload each device in the cluster. To avoid having to reload each device, we recommend that you delete all block allocation rules and clear all xlates related to those rules. You can then change the block size and recreate the block allocation rules.
- NAT pool address distribution for dynamic PAT—When you configure a PAT pool, the cluster divides each IP address in the pool into port blocks. By default, each block is 512 ports, but if you configure port block allocation rules, your block setting is used instead. These blocks are distributed evenly among the nodes in the cluster, so that each node has one or more blocks for each IP address in the PAT pool. Thus, you could have as few as one IP address in a PAT pool for a cluster, if that is sufficient for the number of PAT'ed connections you expect. Port blocks cover the 1024-65535 port range, unless you configure the option to include the reserved ports, 1-1023, on the PAT pool NAT rule.
- Same translation object in static and dynamic NAT rules—Do not use the same network object or object group as the translated address in both static NAT rules and dynamic NAT rules in a cluster. Similarly, avoid using network objects or object groups with overlapping address ranges across static and dynamic NAT rules. Use separate network objects or object groups for static translations and dynamic translations. Mixing these configurations can cause translation ownership inconsistencies in the cluster, which can result in traffic drops or intermittent connection failures.
- Reusing a PAT pool in multiple rules—To use the same PAT pool in multiple rules, you must be careful about the interface selection in the rules. You must either use specific interfaces in all rules, or "any" in all rules. You cannot mix specific interfaces and "any" across the rules, or the system might not be able to match return traffic to the right node in the cluster. Using unique PAT pools per rule is the most reliable option.
- No round-robin—Round-robin for a PAT pool is not supported with clustering.
- No extended PAT—Extended PAT is not supported with clustering.
- Dynamic NAT xlates managed by the control node—The control node maintains and replicates the xlate table to data nodes. When a data node receives a connection that requires dynamic NAT, and the xlate is not in the table, it requests the xlate from the control node. The data node owns the connection.

- Stale xlates—The xlate idle time on the connection owner does not get updated. Thus, the idle time might exceed the idle timeout. An idle timer value higher than the configured timeout with a refcnt of 0 is an indication of a stale xlate.
- No static PAT for the following inspections—
  - FTP
  - RSH
  - SQLNET
  - TFTP
  - XDMCP
  - SIP
- If you have an extremely large number of NAT rules, over ten thousand, you should enable the transactional commit model using the **asp rule-engine transactional-commit nat** command in the device CLI. Otherwise, the node might not be able to join the cluster.

## SIP inspection and clustering

A control flow can be created on any node (due to load balancing); its child data flows must reside on the same node.

## SNMP and clustering

You should always use the Local address, and not the Main cluster IP address for SNMP polling. If the SNMP agent polls the Main cluster IP address, if a new control node is elected, the poll to the new control node will fail.

When using SNMPv3 with clustering, if you add a new cluster node after the initial cluster formation, then SNMPv3 users are not replicated to the new node. You must remove the users, and re-add them, and then redeploy your configuration to force the users to replicate to the new node.

## Syslog and clustering

- Each node in the cluster generates its own syslog messages. You can configure logging so that each node uses either the same or a different device ID in the syslog message header field. For example, the hostname configuration is replicated and shared by all nodes in the cluster. If you configure logging to use the hostname as the device ID, syslog messages generated by all nodes look as if they come from a single node. If you configure logging to use the local-node name that is assigned in the cluster bootstrap configuration as the device ID, syslog messages look as if they come from different nodes.

# Performance scaling factor

When you combine multiple units into a cluster, you can expect the total cluster performance to be approximately 80% of the maximum combined throughput.

### Throughput calculation

For example, if your model can handle approximately 10 Gbps of traffic when running alone, then for a cluster of 8 units, the maximum combined throughput will be approximately 80% of 80 Gbps (8 units x 10 Gbps): 64 Gbps.

# Control node election

### Summary

The key components involved in the control node election process are:

- Cluster nodes: Communicate over the cluster control link and participate in the election process
- Election requests: Broadcast every 3 seconds by nodes when clustering is enabled
- Priority values: Set between 1 and 100 where 1 is the highest priority
- Control node: The elected node that coordinates cluster operations

### Workflow

These stages describe how cluster nodes elect a control node:

1. When clustering is enabled for a node (or when it first starts up with clustering already enabled), the node broadcasts an election request every 3 seconds.
2. Any other nodes with a higher priority respond to the election request. The priority is set between 1 and 100, where 1 is the highest priority.
3. If after 45 seconds, a node does not receive a response from another node with a higher priority, then it becomes the control node.



**Note** If multiple nodes tie for the highest priority, the cluster node name and then the serial number is used to determine the control node.

4. If a node later joins the cluster with a higher priority, it does not automatically become the control node. The existing control node always remains as the control node unless it stops responding, at which point a new control node is elected.
5. In a "split brain" scenario when there are temporarily multiple control nodes, the node with highest priority retains the role while the other nodes return to data node roles.



**Note** You can manually force a node to become the control node. For centralized features, if you force a control node change, then all connections are dropped, and you have to re-establish the connections on the new control node.

# High availability within the cluster

Clustering provides high availability by monitoring node and interface health and by replicating connection states between nodes.

## Node health monitoring

Each node periodically sends a broadcast heartbeat packet over the cluster control link. If the control node does not receive any heartbeat packets or other packets from a data node within the configurable timeout period, then the control node removes the data node from the cluster. If the data nodes do not receive packets from the control node, then a new control node is elected from the remaining nodes.

### Split brain scenario

If nodes cannot reach each other over the cluster control link because of a network failure and not because a node has actually failed, then the cluster may go into a "split brain" scenario where isolated data nodes will elect their own control nodes. For example, if a router fails between two cluster locations, then the original control node at location 1 will remove the location 2 data nodes from the cluster. Meanwhile, the nodes at location 2 will elect their own control node and form their own cluster. Note that asymmetric traffic may fail in this scenario. After the cluster control link is restored, then the control node that has the higher priority will keep the control node's role.

## Interface monitoring

Each node monitors the link status of all named hardware interfaces in use, and reports status changes to the control node.

All physical interfaces are monitored; only named interfaces can be monitored. You can optionally disable monitoring per interface.

A node is removed from the cluster if its monitored interfaces fail. The node is removed after 500 milliseconds.

## Status after failure

When a node in the cluster fails, the connections hosted by that node are seamlessly transferred to other nodes; state information for traffic flows is shared over the control node's cluster control link.

If the control node fails, then another member of the cluster with the highest priority (lowest number) becomes the control node.

The Firewall Threat Defense automatically tries to rejoin the cluster, depending on the failure event.



**Note** When the Firewall Threat Defense becomes inactive and fails to automatically rejoin the cluster, all data interfaces are shut down; only the Management interface can send and receive traffic.

## Rejoining the cluster

After a cluster member is removed from the cluster, how it can rejoin the cluster depends on why it was removed:

- Failed cluster control link when initially joining—After you resolve the problem with the cluster control link, you must manually rejoin the cluster by re-enabling clustering.

- Failed cluster control link after joining the cluster—The Firewall Threat Defense automatically tries to rejoin every 5 minutes, indefinitely.
- Failed data interface—The Firewall Threat Defense automatically tries to rejoin at 5 minutes, then at 10 minutes, and finally at 20 minutes. If the join is not successful after 20 minutes, then the Firewall Threat Defense application disables clustering. After you resolve the problem with the data interface, you have to manually enable clustering.
- Failed node—If the node was removed from the cluster because of a node health check failure, then rejoining the cluster depends on the source of the failure. For example, a temporary power failure means the node will rejoin the cluster when it starts up again as long as the cluster control link is up. The Firewall Threat Defense application attempts to rejoin the cluster every 5 seconds.
- Internal error—Internal failures include: application sync timeout; inconsistent application statuses; and so on.
- Failed configuration deployment—If you deploy a new configuration from Firewall Management Center, and the deployment fails on some cluster members but succeeds on others, then the nodes that failed are removed from the cluster. You must manually rejoin the cluster by re-enabling clustering. If the deployment fails on the control node, then the deployment is rolled back, and no members are removed. If the deployment fails on all data nodes, then the deployment is rolled back, and no members are removed.

## Data path connection state replication

Every connection has one owner and at least one backup owner in the cluster. The backup owner does not take over the connection in the event of a failure; instead, it stores TCP/UDP state information, so that the connection can be seamlessly transferred to a new owner in case of a failure. The backup owner is usually also the director.

Some traffic requires state information above the TCP or UDP layer. See the following table for clustering support or lack of support for this kind of traffic.

***Table 11: Features Replicated Across the Cluster***

| Traffic | State Support | Notes |
|---|---|---|
| Up time | Yes | Keeps track of the system up time. |
| ARP Table | Yes | — |
| MAC address table | Yes | — |
| User Identity | Yes | — |
| IPv6 Neighbor database | Yes | — |
| Dynamic routing | Yes | — |
| SNMP Engine ID | **No** | — |

# How the cluster manages connections

Connections can be load-balanced to multiple nodes of the cluster. Connection roles determine how connections are handled in both normal operation and in a high availability situation.

# Connection roles

See the following roles defined for each connection:

- Owner—Usually, the node that initially receives the connection. The owner maintains the TCP state and processes packets. A connection has only one owner. If the original owner fails, then when new nodes receive packets from the connection, the director chooses a new owner from those nodes.
- Backup owner—The node that stores TCP/UDP state information received from the owner, so that the connection can be seamlessly transferred to a new owner in case of a failure. The backup owner does not take over the connection in the event of a failure. If the owner becomes unavailable, then the first node to receive packets from the connection (based on load balancing) contacts the backup owner for the relevant state information so it can become the new owner.

  As long as the director (see below) is not the same node as the owner, then the director is also the backup owner. If the owner chooses itself as the director, then a separate backup owner is chosen.

  For clustering on the Firepower 9300, which can include up to 3 cluster nodes in one chassis, if the backup owner is on the same chassis as the owner, then an additional backup owner will be chosen from another chassis to protect flows from a chassis failure.
- Director—The node that handles owner lookup requests from forwarders. When the owner receives a new connection, it chooses a director based on a hash of the source/destination IP address and ports (see below for ICMP hash details), and sends a message to the director to register the new connection. If packets arrive at any node other than the owner, the node queries the director about which node is the owner so it can forward the packets. A connection has only one director. If a director fails, the owner chooses a new director.

  As long as the director is not the same node as the owner, then the director is also the backup owner (see above). If the owner chooses itself as the director, then a separate backup owner is chosen.

  ICMP/ICMPv6 hash details:

  - For Echo packets, the source port is the ICMP identifier, and the destination port is 0.
  - For Reply packets, the source port is 0, and the destination port is the ICMP identifier.
  - For other packets, both source and destination ports are 0.
- Forwarder—A node that forwards packets to the owner. If a forwarder receives a packet for a connection it does not own, it queries the director for the owner, and then establishes a flow to the owner for any other packets it receives for this connection. The director can also be a forwarder. Note that if a forwarder receives the SYN-ACK packet, it can derive the owner directly from a SYN cookie in the packet, so it does not need to query the director. (If you disable TCP sequence randomization, the SYN cookie is not used; a query to the director is required.) For short-lived flows such as DNS and ICMP, instead of querying, the forwarder immediately sends the packet to the director, which then sends them to the owner. A connection can have multiple forwarders; the most efficient throughput is achieved by a good load-balancing method where there are no forwarders and all packets of a connection are received by the owner.



**Note** We do not recommend disabling TCP sequence randomization when using clustering. There is a small chance that some TCP sessions won't be established, because the SYN/ACK packet might be dropped.

- Fragment Owner—For fragmented packets, cluster nodes that receive a fragment determine a fragment owner using a hash of the fragment source IP address, destination IP address, and the packet ID. All fragments are then forwarded to the fragment owner over the cluster control link. Fragments may be load-balanced to different cluster nodes, because only the first fragment includes the 5-tuple used in the switch load balance hash. Other fragments do not contain the source and destination ports and may be load-balanced to other cluster nodes. The fragment owner temporarily reassembles the packet so it can determine the director based on a hash of the source/destination IP address and ports. If it is a new connection, the fragment owner will register to be the connection owner. If it is an existing connection, the fragment owner forwards all fragments to the provided connection owner over the cluster control link. The connection owner will then reassemble all fragments.

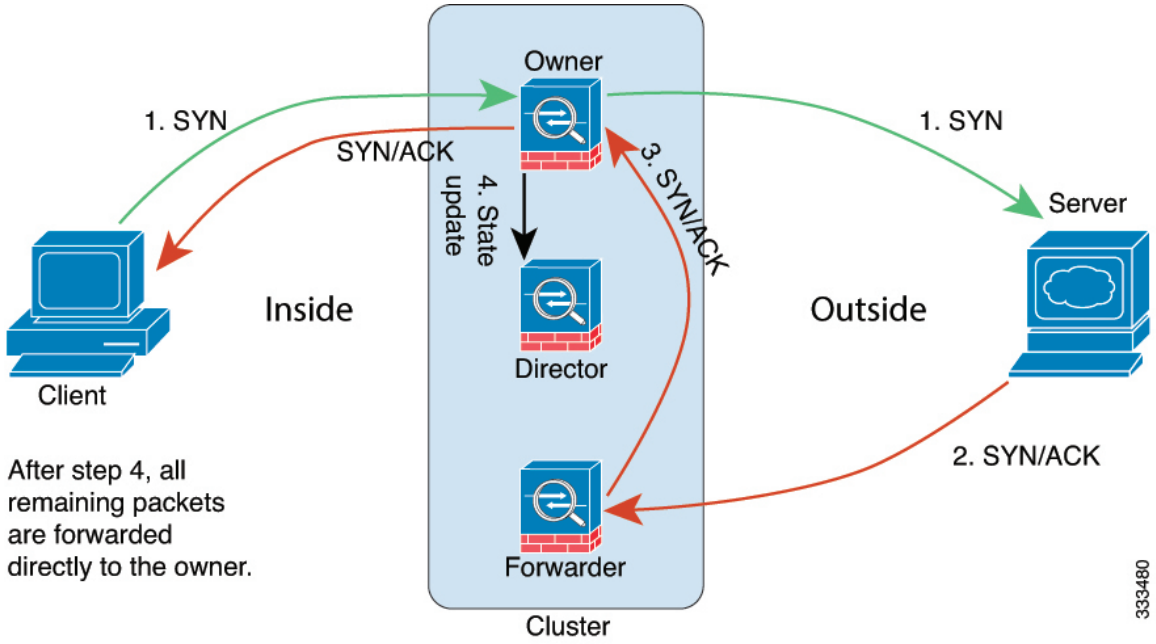
**Port Address Translation Connections**

## New connection ownership

Traffic redirection is not supported in this release. When a new connection is directed to a node of the cluster via load balancing, that node owns both directions of the connection. All the subsequent packets for the same connection should arrive the same node. If any connection packets arrive at a different node, they will be dropped. If a reverse flow arrives at a different node, it will be dropped as well. For centralized features, if the connections do not arrive on the control node, they will be dropped.

## Sample data flow for TCP

The following example shows the establishment of a new connection.



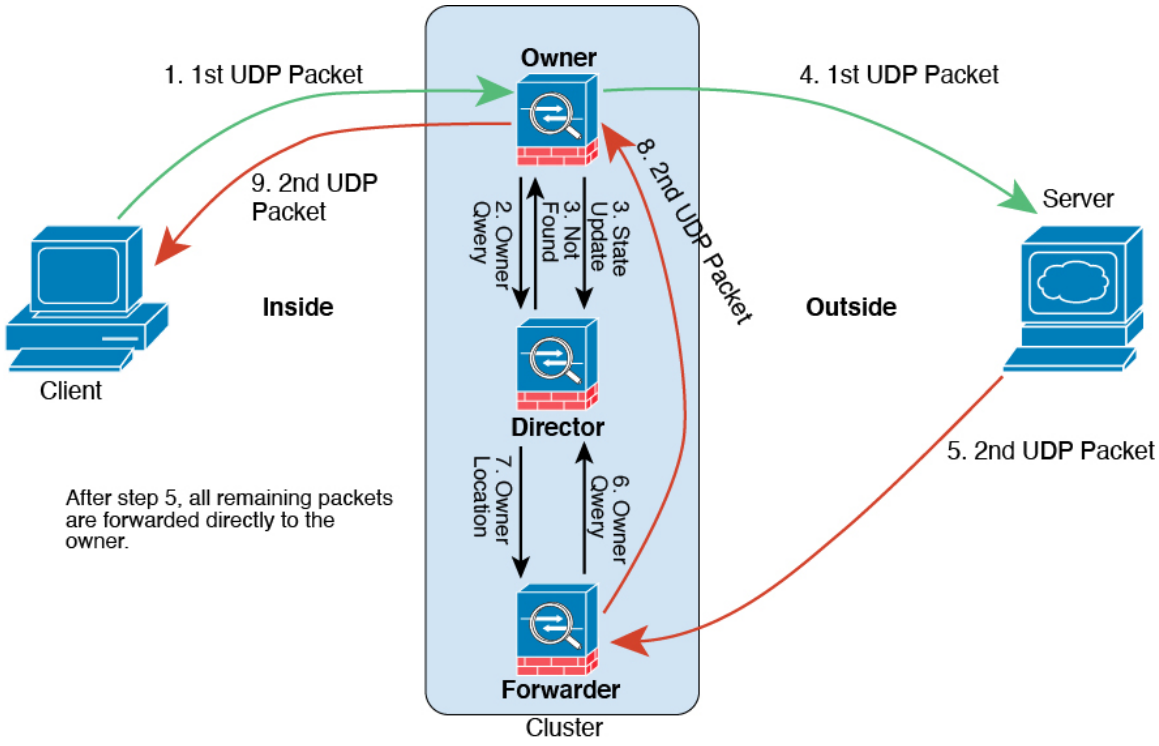

1. The SYN packet originates from the client and is delivered to one Firewall Threat Defense (based on the load balancing method), which becomes the owner. The owner creates a flow, encodes owner information into a SYN cookie, and forwards the packet to the server.
2. The SYN-ACK packet originates from the server and is delivered to a different Firewall Threat Defense (based on the load balancing method). This Firewall Threat Defense is the forwarder.

3. Because the forwarder does not own the connection, it decodes owner information from the SYN cookie, creates a forwarding flow to the owner, and forwards the SYN-ACK to the owner.
4. The owner sends a state update to the director, and forwards the SYN-ACK to the client.
5. The director receives the state update from the owner, creates a flow to the owner, and records the TCP state information as well as the owner. The director acts as the backup owner for the connection.
6. Any subsequent packets delivered to the forwarder will be forwarded to the owner.
7. If packets are delivered to any additional nodes, it will query the director for the owner and establish a flow.
8. Any state change for the flow results in a state update from the owner to the director.

## Sample data flow for ICMP and UDP

The following example shows the establishment of a new connection.

1. ***Figure 35: ICMP and UDP Data Flow***




The first UDP packet originates from the client and is delivered to one Firewall Threat Defense (based on the load balancing method).

2. The node that received the first packet queries the director node that is chosen based on a hash of the source/destination IP address and ports.
3. The director finds no existing flow, creates a director flow and forwards the packet back to the previous node. In other words, the director has elected an owner for this flow.
4. The owner creates the flow, sends a state update to the director, and forwards the packet to the server.

5. The second UDP packet originates from the server and is delivered to the forwarder.
6. The forwarder queries the director for ownership information. For short-lived flows such as DNS, instead of querying, the forwarder immediately sends the packet to the director, which then sends it to the owner.
7. The director replies to the forwarder with ownership information.
8. The forwarder creates a forwarding flow to record owner information and forwards the packet to the owner.
9. The owner forwards the packet to the client.

# History for Threat Defense Virtual clustering in the public cloud

This reference provides the historical timeline and version information for Threat Defense Virtual clustering features introduced across different software releases and deployment platforms in public cloud environments.

***Table 12: Feature history***

| Feature | Minimum Firewall Management Center | Minimum Firewall Threat Defense | Details |
|---|---|---|---|
| MTU ping test from data node on node join | 7.6.0 | 7.6.0 | When a node joins the cluster, it checks MTU compatibility by sending a ping to the control node with a packet size twice the cluster control link MTU. Previously, only the control node sent a ping. If the ping fails, a notification is generated so you can fix the MTU mismatch on connecting switches and try again.<br>Added/modified commands: **show cluster history**. |
| Cluster control link ping tool | 7.2.6/7.4.1 | Any | You can check to make sure all the cluster nodes can reach each other over the cluster control link by performing a ping. One major cause for the failure of a node to join the cluster is an incorrect cluster control link configuration; for example, the cluster control link MTU may be set higher than the connecting switch MTUs.<br>New/modified screens: **Devices** > **Device Management** > **More** > **Cluster Live Status.** |
| Troubleshooting file generation and download available from Device and Cluster pages | 7.4.1 | 7.4.1 | You can generate and download troubleshooting files for each device on the Device page and also for all cluster nodes on the Cluster page. For a cluster, you can download all files as a single compressed file. You can also include cluster logs for the cluster for cluster nodes. You can alternatively trigger file generation from the **Devices** > **Device Management** > **More** > **Troubleshoot Files** menu.<br>New/modified screens:<br>• **Devices** > **Device Management** > **Device** > **General**<br>• **Devices** > **Device Management** > **Cluster** > **General** |

| Feature | Minimum Firewall Management Center | Minimum Firewall Threat Defense | Details |
|---|---|---|---|
| View CLI output for a device or device cluster | 7.4.1 | Any | You can view a set of pre-defined CLI outputs that can help you troubleshoot the device or cluster. You can also enter any **show** command and see the output.<br>New/modified screens: **Devices** > **Device Management** > **Cluster** > **General** |
| Cluster health monitor settings. | 7.3.0 | Any | You can now edit cluster health monitor settings.<br>New/Modified screens: **Devices** > **Device Management** > **Cluster** > **Cluster Health Monitor Settings**<br>**Note**<br>If you previously configured these settings using FlexConfig, be sure to remove the FlexConfig configuration before you deploy. Otherwise the FlexConfig configuration will overwrite the management center configuration. |
| Cluster health monitor dashboard. | 7.3.0 | Any | You can now view cluster health on the cluster health monitor dashboard.<br>New/Modified screens: **System** > **Health** > **Monitor** |
| Clustering for the Firewall Threat Defense Virtual in Azure. | 7.3.0 | 7.3.0 | You can now configure clustering for up to 16 nodes the Firewall Threat Defense Virtual in Azure for the Azure Gateway Load Balancer or for external load balancers.<br>New/modified screens:<br>• **Devices** > **Device Management** > **Add Cluster**<br>• **Devices** > **Device Management** > **More** menu<br>• **Devices** > **Device Management** > **Cluster**<br>Supported platforms: Firewall Threat Defense Virtual in Azure |
| Clustering for the Firewall Threat Defense Virtual in the Public Cloud (Amazon Web Services and Google Cloud Platform). | 7.2.0 | 7.2.0 | The Firewall Threat Defense Virtual supports Individual interface clustering for up to 16 nodes in the public cloud (AWS and GCP).<br>New/Modified screens:<br>• **Devices** > **Device Management** > **Add Device**<br>• **Devices** > **Device Management** > **More** menu<br>• **Devices** > **Device Management** > **Cluster**<br>Supported platforms: Firewall Threat Defense Virtual in AWS and GCP |