

VMware NSX Advanced Load Balancer Installation Guide

VMware NSX Advanced Load Balancer 21.1.4

You can find the most up-to-date technical documentation on the VMware website at:

<https://docs.vmware.com/>

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 2022 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

Contents

1	NSX Advanced Load Balancer Overview	8
	Control Plane	9
	Data Plane	10
	Elastic HA for NSX Advanced Load Balancer Service Engines	11
2	Preparing For Installation	21
	System Requirements	21
	Kernel Supported Versions in Bare Metal - 21.1.x	25
	Kernel Supported Versions in Bare Metal - 20.1.x	33
	Software Version Compatibility	50
	OpenStack Support Matrix	51
	System Limits	52
	NSX Advanced Load Balancer IPAM and DNS	53
	Configuring NSX Advanced Load Balancer IPAM	54
	NSX Advanced Load Balancer DNS	64
	Configuring IPAM and DNS Support	69
	Networking Considerations	70
	Ports and Protocols	71
	Changing NSX Advanced Load Balancer Controller Cluster Configuration	74
	Deploying a Controller Cluster	74
	Dismantling the Controller Cluster	75
	Changing a Follower Node	77
	Replacing the Leader Node	78
	Replacing a Follower Node with a New Node	78
	NSX Advanced Load Balancer Controller Sizing	79
	Data Plane Development Kit (DPDK)	81
	Post-installation Configuration	82
	Creating Virtual Services in a VRF	84
	Creating VRF Contexts	85
3	Installing NSX Advanced Load Balancer in Google Cloud Platform	86
	Overview	87
	Networking Considerations	87
	Configuring Firewall Rules in GCP	87
	Deployment Prerequisites	96
	NSX Advanced Load Balancer Controller Installation	112
	Deployment Topology	112
	Creating the NSX Advanced Load Balancer Controller Image	113

Configuring the NSX Advanced Load Balancer Controller Cloud Connector	116
Creating Controller VM from GCP Console	116
Configuring NSX Advanced Load Balancer	118
Configuring GCP Cloud in the NSX Advanced Load Balancer Controller	119
Configuring Service Engine Group Properties	124
High Availability Options	125
NSX Advanced Load Balancer and Internal IPAM Configuration	139
Virtual Service Configuration	141
4 Installing NSX Advanced Load Balancer in Linux Server Cloud	145
Overview	145
Considerations for Deploying NSX Advanced Load Balancer in Linux Server Cloud	146
Preparing the Server	152
Installing NSX Advanced Load Balancer in a Linux Server Cloud	159
Configuring the NSX Advanced Load Balancer Cloud Connector	161
Configuring Linux Cloud using CLI	161
Configuring Linux Cloud using UI	163
Verifying NSX Advanced Load Balancer SE Creation	165
Additional Deployment Options	165
Configuring In-band Management for an NSX Advanced Load Balancer Service Engine	166
5 Installing NSX Advanced Load Balancer in OpenStack	168
Overview	168
Considerations for Deploying NSX Advanced Load Balancer with OpenStack	169
OpenStack Network Configuration for NSX Advanced Load Balancer Controller Cluster	171
Installing NSX Advanced Load Balancer in OpenStack	175
Installing Single Tenant and NSX Advanced Load Balancer-Managed Mode	175
Installing NSX Advanced Load Balancer OpenStack in No-Access OpenStack Cloud	184
Installing Heat Plugins in NSX Advanced Load BalancerOpenStack using Kolla Ansible	194
Configuring the NSX Advanced Load Balancer Cloud Connector	198
NSX Advanced Load Balancer OpenStack External Networks	199
Additional Deployment Options	200
NSX Advanced Load Balancer OpenStack Full Access Integration using Non-admin Privileges	200
6 Installing NSX Advanced Load Balancer in VMware NSX-T Environments	209
Overview of NSX-T Integration with NSX Advanced Load Balancer	209
Installing the Controller	210
Prerequisites to Install the Controller	210
Deploying the Controller OVA	215
Setting up the NSX Advanced Load Balancer Controller	217
Configuring the Cloud Connector	217

Prerequisites for Creating the NSX-T Cloud	217
Creating an NSX-T Cloud	219
7 Installing NSX Advanced Load Balancer on Linux KVM (No- Access mode)	221
Considerations for Deploying NSX Advanced Load Balancer in KVM based environments	221
Role Requirement	221
Hardware	225
Software	225
NIC Information	226
Preparing the server	227
Ansible Playbook	227
Virtual Functions (VF) from PF	229
Post Host Reboot	229
Installing the Controller	230
Additional Deployment Options	230
Disabling NSX Advanced Load Balancer Controller and Service Engines	231
8 Installing NSX Advanced Load Balancer in Microsoft Azure	233
Deployment Considerations	234
Minimum Requirements	234
Ports and Protocols	235
Azure Roles and Permissions	235
Authentication Modes	237
Service Engine Disc Encryption	241
NSX Advanced Load Balancer for Microsoft Azure - License Models	242
Microsoft Azure Resource Limits	243
Azure Virtual Network	244
Installing NSX Advanced Load Balancer Controller	245
Configuring the NSX Advanced Load Balancer Cloud Connector	246
Additional Deployment Options	249
Configuring Service Engine Disk Encryption	249
Multiple Azure Load Balancers for Microsoft Azure Cloud	250
9 Installing NSX Advanced Load Balancer in Cisco CSP	253
Considerations for Deploying NSX Advanced Load Balancer in Cisco CSP Environments	253
Prerequisites	253
Performance and Sizing for CSP	257
Installing NSX Advanced Load Balancer Controller and Service Engine in CSP	262
Installing NSX Advanced Load Balancer Controller in CSP	262
Installing NSX Advanced Load Balancer Service Engine in CSP	266
Configuring the NSX Advanced Load Balancer Cloud Connector	269

Additional Deployment Options	269
Configuring Dedicated Interfaces for HSM Communication on NSX Advanced Load Balancer Service Engines	270
Configuring Dedicated Interfaces for ASM Communication on NSX Advanced Load Balancer Service Engines	275
Configuring Dedicated Interfaces for both HSM and ASM (Sideband) Communication on the NSX Advanced Load Balancer Service Engine	279
Configuring Dedicated Interfaces for HSM Communication on the NSX Advanced Load Balancer Controller	284
10 Installing NSX Advanced Load Balancer in Cisco ACI based Environments	288
Overview	288
Networking Consideration	290
Software Requirements	290
Integration Options for NSX Advanced Load Balancer in ACI Fabric	290
11 Installing the NSX Advanced Load Balancer in Oracle Cloud	303
Considerations for Deploying the NSX Advanced Load Balancer in Oracle Cloud	303
Configuring OCI Credentials	303
Configuring OCI IPAM	305
Configuring the Cloud Connector	306
12 Installing NSX Advanced Load Balancer in Amazon Web Services	307
Deployment Considerations for NSX Advanced Load Balancer in AWS	307
Minimum Requirements	307
Ports and Protocols	308
Installation Options and Credential Methods	308
Encryption	325
Security Groups	332
Installing the NSX Advanced Load Balancer Controller	333
Deploying an EC2 Instance	333
NSX Advanced Load Balancer Controller Initial Configuration	336
Configuring NSX Advanced Load Balancer Controller Cluster in AWS	338
Configuring the NSX Advanced Load Balancer Cloud Connector	340
Configuring the NSX Advanced Load Balancer Cloud Connector Using AWS Shared Key	340
Configuring the NSX Advanced Load Balancer Cloud Connector Using IAM Roles	342
Configuring the NSX Advanced Load Balancer Cloud Connector Use Cross-Account AssumeRole	343
Multi AZ Support for AWS	344
Service Discovery Using IPAM and DNS	344
Configuring Security Groups	348
Additional Deployment Options	350

- Multi AZ Support for AWS 350
- DPDK support on AWS native AMI SE 352

13 Installing NSX Advanced Load Balancer in Nutanix Acropolis Based Environments 353

- Considerations for Deploying NSX Advanced Load Balancer in Nutanix Environments 353
 - Hardware Requirements for installing the NSX Advanced Load Balancer in Nutanix Acropolis 353
 - Software Requirements for installing the NSX Advanced Load Balancer in Nutanix Acropolis 354
- Installing the NSX Advanced Load Balancer in Nutanix Acropolis 354
 - Deploying the Controller through Prism 354
 - Deploying the Service Engine 356
 - Deploying VRF on Nutanix 358

14 Installing NSX Advanced Load Balancer in VMware vSphere Environments 359

- Overview 359
- Installation Considerations 361
 - vCenter Account Requirements 361
 - Software Requirement 380
 - Ports and Protocols 380
 - IP Address Requirements 380
 - VRF 381
 - IPAM and DNS 384
 - Controller and VMware Communication 384
- Deploying NSX Advanced Load Balancer Controller in VMware vCenter 384
 - Modes of Deployment 384
- Configuring NSX Advanced Load Balancer Cloud Connector 389
 - Configuring NSX Advanced Load Balancer Cloud Connector in Write Access Mode 389
 - Configuring NSX Advanced Load Balancer Cloud Connector in No Access Mode and Read Access Mode 392
 - Configuring IP Address Pools 396
- Additional Configuration Options 396
 - Configuring VRF 396
 - NSX Advanced Load Balancer IPAM and DNS 398

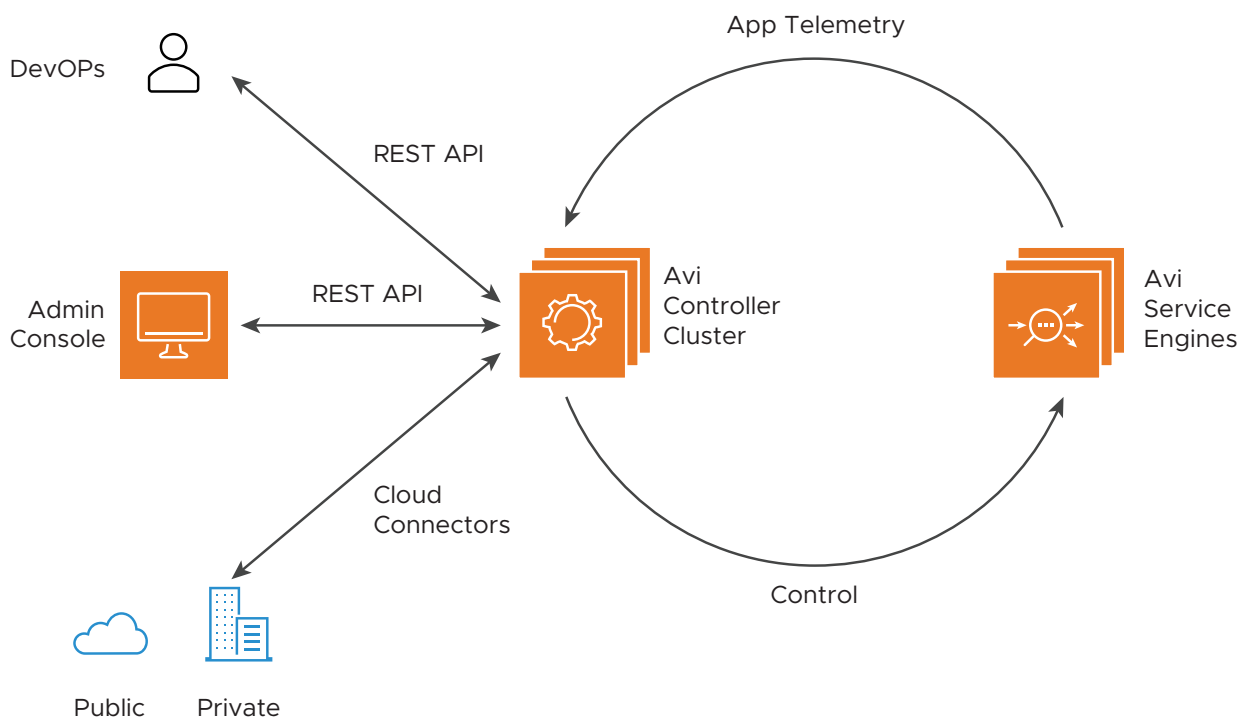
NSX Advanced Load Balancer Overview

1

NSX Advanced Load Balancer is built on software-defined principles, enabling a next-generation architecture to deliver the flexibility and simplicity expected by IT and lines of businesses.

The NSX Advanced Load Balancer architecture separates the data and control planes to deliver application services beyond load balancing, such as application analytics, predictive autoscaling, micro-segmentation, and self-service for application owners in on-premises or cloud environments.

The platform provides a centrally managed, dynamic pool of load balancing resources on commodity x86 servers, Virtual Machines (VMs), or containers, to deliver granular services close to individual applications. This allows network services to scale up without the added complexity of managing hundreds of disparate appliances.



NSX Advanced Load Balancer provides out-of-the-box integrations for on-premises or cloud deployments. These integrations with private cloud frameworks, software-defined network (SDN) controllers, container orchestration platforms, virtualized environments, and public clouds enable turnkey application services and automation.

NSX Advanced Load Balancer integrates with the following ecosystems:

- Bare Metal (Linux Server Cloud)
- VMware vCenter
- VMware NSX-T
- OpenStack
- Cisco CSP
- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Oracle Cloud Infrastructure

The NSX Advanced Load Balancer consists of:

- The NSX Advanced Load Balancer Controller or Controller Cluster that functions as the Control Plane
- The Service Engines(SEs) that function as the Data Plane

This chapter includes the following topics:

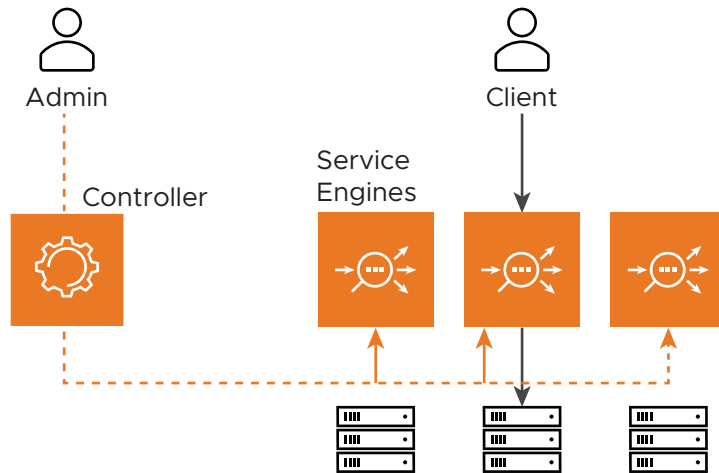
- [Control Plane](#)
- [Data Plane](#)

Control Plane

The NSX Advanced Load Balancer Controller is the single point of management and control. It is typically deployed as a three-node cluster to ensure high availability.

The Controller implements the control plane. A single NSX Advanced Load Balancer deployment is managed from this Controller Cluster (identified by FQDN and/or cluster IP address), regardless of the number of applications being load balanced or the number of NSX Advanced Load Balancer Service Engines (SEs) required.

The Controller places virtual services on SEs to load balance applications, regardless of whether the SEs were created automatically (write-access mode) or manually (read or no-access mode). For a higher degree of automation, a Controller is deployed in a write-access deployment, allowing it to work with the underlying orchestrator to launch new SEs in response to events such as creating a new virtual service or increase in application load above a threshold level.. The Controller's REST API provides visibility to all configured applications (virtual services) and enables complete application lifecycle automation.



Control Plane High Availability

In a production deployment environment, the best practice to ensure high availability is to deploy a set of three NSX Advanced Load Balancer Controllers as a High Availability (HA) cluster. In a cluster deployment, one of the Controllers is the leader and performs load balancing and configuration management for the cluster. The other two Controllers are the followers; they collaborate with the leader to perform data collection from SEs and process analytics data.

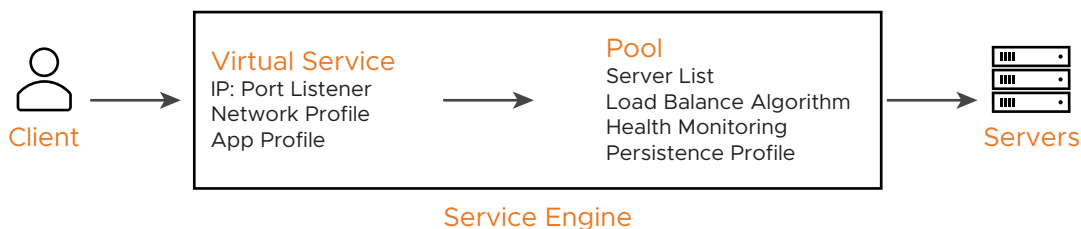
Data Plane

NSX Advanced Load Balancer SEs handle all data plane operations within the NSX Advanced Load Balancer by receiving and executing instructions from the Controller.

The SEs perform load balancing and all client and server-facing network interactions. They collect real-time telemetry data from application traffic flows.

In a typical load balancing scenario, a client will communicate with a virtual service, which is an IP address and port hosted in NSX Advanced Load Balancer by an SE. The virtual service internally passes the connection through a number of profiles. The SE can terminate and proxy the client TCP connection, terminate SSL/TLS, and proxy the HTTP request for HTTP traffic. Once the request is validated, it is forwarded internally to a pool to choose an available back-end server.

A new TCP connection then originates from the SE. This connection uses the IP address of the SE on the internal network as the client request's source IP address. Return traffic also follows the same path. The client communicates exclusively with the virtual service IP address and not the back-end server IP.



Data Plane High Availability

NSX Advanced Load Balancer SE groups support the following HA modes:

- **Elastic HA:** Provides fast recovery for individual virtual services following failure of the SE. Depending on the mode, the virtual service is already running on multiple SEs or is quickly placed on another SE. The following modes of cluster HA are supported:
 - Active/Active
 - N + M
- **Legacy HA:** Emulates a 2-device hardware active/standby HA configuration operation. The active SE carries all the traffic for a virtual service placed on it. The other SE in the pair is the standby for the VS, carrying no traffic for it when the active SE is healthy.

Elastic HA for NSX Advanced Load Balancer Service Engines

This section describes about Elastic HA for NSX Advanced Load Balancer SEs.

High Availability Modes

NSX Advanced Load Balancer supports two SE elastic HA modes which combine scale-out performance as well as high availability:

- N+M mode (the default)
- Active/Active

NSX Advanced Load Balancer also provides a third mode, [legacy HA](#) mode, which enables a smooth migration from legacy appliance-based load balancers.

Elastic HA N+M Mode

N+M is the default mode for elastic HA.

- In this mode, each virtual service is typically placed on just one SE.
- The “N” in “N+M” is the minimum number of SEs required to place virtual services in the SE group — this calculation is done by the NSX Advanced Load Balancer Controller based on Virtual Services per Service Engine parameter. N will vary over time, as virtual services are placed on or removed from the group.
- The “M” of “N+M” is the number of additional SEs the Controller spins up to handle up to M SE failures without reducing the capacity of the SE group. M appears in Buffer Service Engines field.

Note The buffer SE in N+M mode is the number of SE failures the system can tolerate for the Virtual Service to be operationally up (placed on at least 1 SE), but not to be at the same capacity. If there is specific minimum scale per Virtual Service set in the SE Group and if additional SE is required above that, then you should increase the buffer SE according to the calculations.

Edit Service Engine Group: Default-Group

Basic Settings
Advanced

Service Engine Group Name*
Default-Group

Metric Update Frequency ?
☐ Real-Time Metrics
Duration. Use 0 for Infinite.
min

• High Availability & Placement Settings •

High Availability Mode ?
Legacy HA
☐ Active/Standby
Elastic HA
☐ Active/Active
☒ N + M (buffer)

VS Placement across SEs ?
☒ Compact
☐ Distributed

Virtual Services per Service Engine ?
3
Maximum

☐ SE Self-Election ?

• Service Engine Capacity and Limit Settings •

Max Number of Service Engines ?
3
Maximum

• Memory Allocation •

☐ Host Geolocation Profile ?

Memory for Caching * ?
0
%

Available Memory for Connections and Buffers ?
100
%

Connections and Buffers Memory Distribution (slide the bar left or right) ?
Connections: 50%
Buffers: 50%

Cancel
Save

Edit Service Engine Group: Default-Group

Basic Settings
Advanced

• Advanced HA & Placement •

Buffer Service Engines ?
1

Scale per Virtual Service ?
1
Minimum
2
Maximum

☐ CPU socket Affinity ?
☐ Dedicated dispatcher CPU ?

• Security •

HSM Group
Select HSM Group

• Log Collection and Streaming Settings •

Significant Log Throttle ?
100
Logs/Second

UDF Log Throttle ?
100
Logs/Second

Non-significant Log Throttle ?
100
Logs/Second

Number of Streaming Threads ?
1
Threads

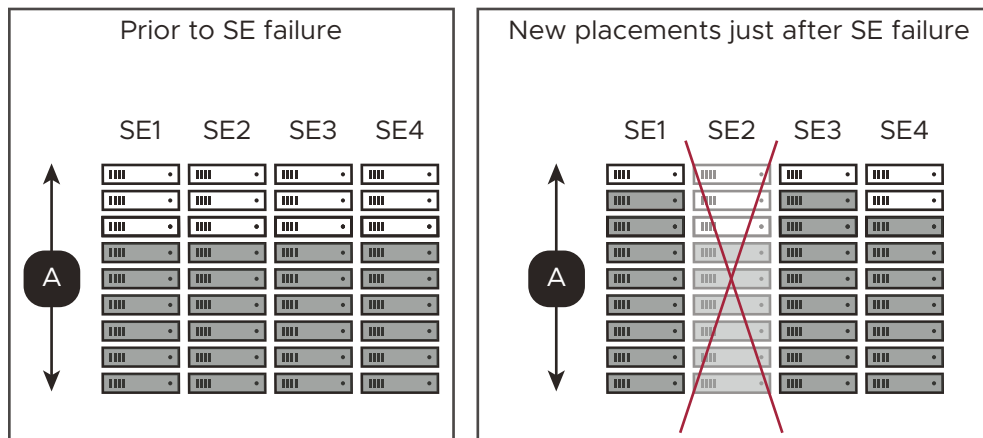
Cancel
Save

Elastic HA N+M Example

The left side of figure 2 shows 20 virtual service placements on an SE group. With Virtual Services per SE set to 8, N is 3 ($20/8 = 2.5$, which rounds to 3). With $M = 1$, a total of $N+M = 3 + 1 = 4$ SEs are required in the group 2.

Note No single SE in the group is completely idle. The Controller places virtual services on all available SEs. In N+M mode, the NSX Advanced Load Balancer ensures enough buffer capacity exists in aggregate to handle one ($M=1$) SE failure. In this example, each of the four SEs has five virtual services placed. A total of 12 spare slots are still available for additional virtual service placements, which is sufficient to handle one SE failure.

The right side of the figure below shows the SE group just after SE2 has failed. The five virtual services in SE2 have been placed onto spare slots found on surviving SEs: SE1, SE3, and SE4.



The imbalance in loading disappears over time if one or both of two things happens:

- 1 New virtual services are placed on the group. As many as 4 virtual services could be placed without compromising the $M=1$ condition. They would be placed on SE5 because NSX Advanced Load Balancer chooses the least-loaded SE first.
- 2 The Auto-Rebalance option is selected (As shown in the above image).

With M set to 1, the SE group is single-SE fault tolerant. Customers desiring multiple-SE fault tolerance set M higher. NSX Advanced Load Balancer permits M to be dynamically increased by the administrator without interrupting any services. Consequently, one may start with $M=1$ (typical of most N+M deployments), and increase it if conditions warrant.

If an N+M group has scaled out to Max Number of Service Engines and $N \times$ Virtual Services per SE have been placed, NSX Advanced Load Balancer will permit additional VS placements (into the spare capacity represented by M), but an HA_COMPROMISED event will be logged.

For a Write Access cloud, the NSX Advanced Load Balancer will attempt to recover the failed SE after five minutes by rebooting the virtual machine. After a further five minutes, the NSX Advanced Load Balancer will attempt to delete the failed SE virtual machine after which a new SE will be spun up to restore the configured buffer capacity.

Back to M = 1 state



As shown in above, with only 4 slots remaining just after the 5 re-placements, if NSX Advanced Load Balancer orchestrator mode is set to write access, NSX Advanced Load Balancer spins up SE5 to meet the M=1 condition, which in this case requires at least 8 slots available for re-placements.

Note To provide time to identify the cause of a failure, the first SE that fails in an SE group is not automatically deleted even after five minutes. You can then perform troubleshooting on the failed SE and delete the virtual machine manually if restoration is not possible. The Controller will delete the SE virtual machine after 3 days if you have not manually deleted the same.

Elastic HA Active/Active

In active/active mode, NSX Advanced Load Balancer places each virtual service on more than one SE, as specified by “Minimum Scale per Virtual Service” parameter — the default minimum is 2. If an SE in the group fails, then

- Virtual services that had been running are not interrupted. They continue to run on other SEs with degraded capacity until they can be placed once again.
- If NSX Advanced Load Balancer’s orchestrator mode is set to write access, a new SE is automatically deployed to bring the SE group back to its previous capacity. After waiting for the new SE to spin up, the Controller places on it the virtual services that had been running on the failed SE.

Edit Service Engine Group: Default-Group

Basic SettingsAdvanced

Service Engine Group Name*
Default-Group

Metric Update Frequency ⓘ
☐ Real-Time Metrics
Duration. Use 0 for Infinite. min

• High Availability & Placement Settings •

High Availability Mode ⓘ
Legacy HA
☐ Active/Standby
Elastic HA
☐ Active/Active ☒ N + M (buffer)

VS Placement across SEs ⓘ
☒ Compact ☐ Distributed

Virtual Services per Service Engine ⓘ
3Maximum

☐ SE Self-Election ⓘ

• Service Engine Capacity and Limit Settings •

Max Number of Service Engines ⓘ
10Maximum

• Memory Allocation •

☐ Host Geolocation Profile ⓘ

Memory for Caching * ⓘ
0%

Available Memory for Connections and Buffers ⓘ
100%

Connections and Buffers Memory Distribution (slide the bar left or right) ⓘ

Connections: 50%Buffers: 50%

• License •

License Type ⓘ
Cores

☐ Enable Per-app SE Mode ⓘ

SE Bandwidth Type ⓘ
SE Bandwidth Unlimited

Number of SE Data Paths ⓘ
Set numberMaximum

☒ Use Hyperthreading ⓘ

Cancel

Save

VMware, Inc.

15

Edit Service Engine Group: Default-Group

Basic Settings Advanced

• Advanced HA & Placement •

Buffer Service Engines 0

Scale per Virtual Service 2 Minimum 4 Maximum

☐ CPU socket Affinity ☐ Dedicated dispatcher CPU

• Security •

HSM Group
Select HSM Group

• Log Collection and Streaming Settings •

Significant Log Throttle 100 Logs/Second

UDF Log Throttle 100 Logs/Second

Non-significant Log Throttle 100 Logs/Second

Number of Streaming Threads 1 Threads

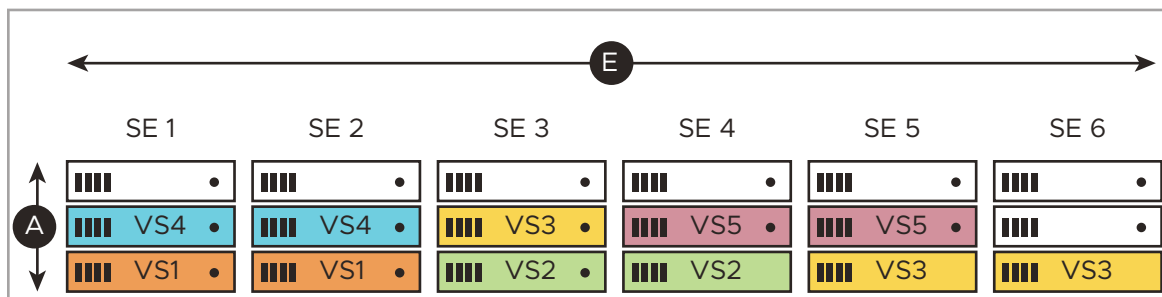
Cancel Save

Elastic HA Active/Active Example

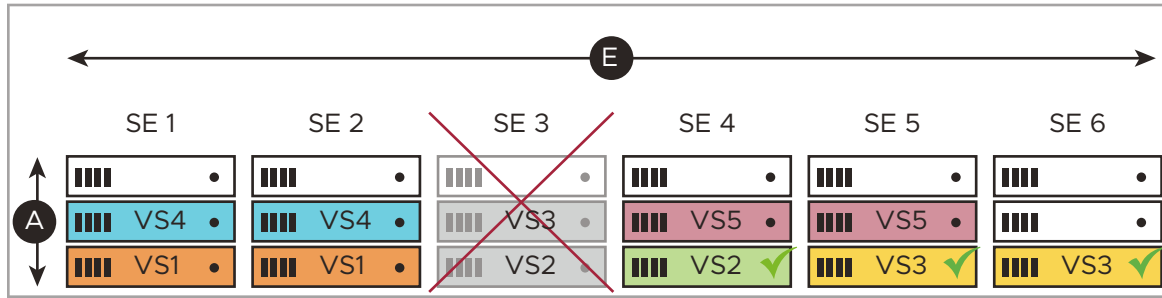
The following illustrations explain an SE failure and full recovery:

- Virtual Services per Service Engine
- Minimum Scale per Virtual Service
- Maximum Scale per Virtual Service
- Max Number of Services Engines

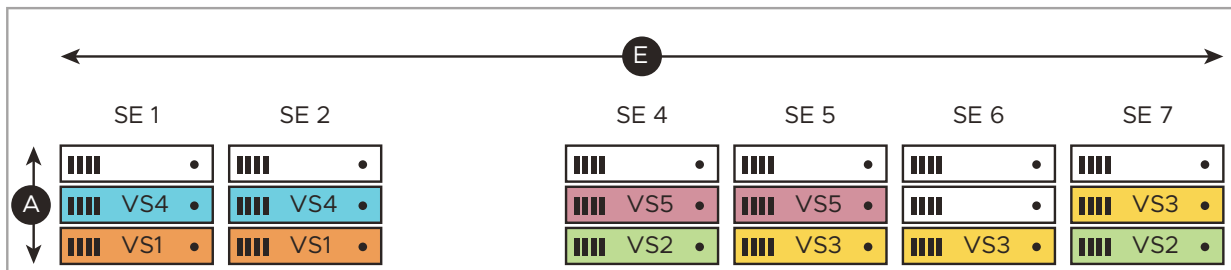
Over time, five virtual services (VS1-VS5) have been placed. One of them, VS3, has been scaled from its initial two placements to three, illustrating NSX Advanced Load Balancer's support for "N-way active" virtual services.



SE3 fails in the below image as a result, one of two VS2 instances fails and one of three VS3 instances fails. Three other virtual services (VS1, VS4, VS5) are unaffected. Neither VS2 nor VS3 are interrupted because of instances previously placed on SE4, SE5, and SE6. They continue, albeit with degraded performance.



As shown in the below image, the NSX Advanced Load Balancer Controller deploys SE7 as a replacement for SE3, and places VS2 and VS3 on it, bringing both virtual services up to their prior level of performance.



Virtual Service Placement Policy

This section explains the interaction between the elastic HA modes and NSX Advanced Load Balancer's VS placement policy for the SE group.

• Advanced HA & Placement •

Buffer Service Engines [?]

Scale per Virtual Service [?]
 Minimum Maximum

☐ CPU socket Affinity [?]
☐ Dedicated dispatcher CPU [?]

• High Availability & Placement Settings •

High Availability Mode [?]

Legacy HA
☐ Active/Standby

Elastic HA
☐ Active/Active
☒ N + M (buffer)

VS Placement across SEs [?]
☒ Compact
☐ Distributed

Compact Placement

This section discusses about Compact Placement.

When Compact Placement is ON (label G in figure 8b), the NSX Advanced Load Balancer uses the minimum number of SEs required. When distributed placement is ON, the NSX Advanced Load Balancer will use as many SEs as required, up to the limit allowed by "Max Number of Service Engines". By default, compact placement is ON for elastic HA N+M mode while distributed placement is ON for elastic HA active/active mode.

Compact Placement Example

Figure 9 shows the effect of compact placement on an elastic HA N+M SE group where the Max Number of Service Engines is 4.

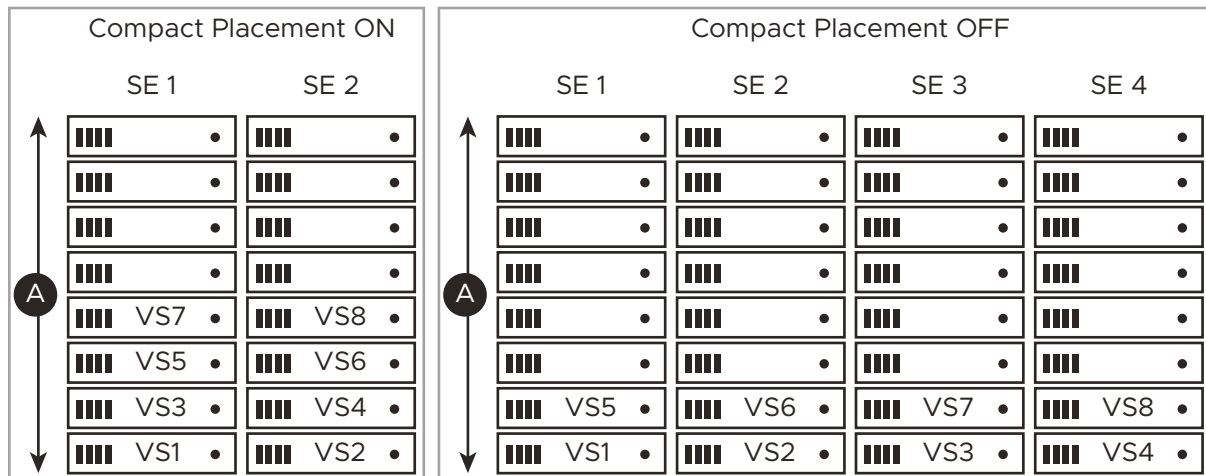
In both the compact and distributed placement examples:

- Eight virtual services are created in sequence.
- After VS1 is placed, SE2 is deployed because M=1 (handle one SE failure).
- When VS2 requires placement, the NSX Advanced Load Balancer assigns it to the otherwise idle SE2 to make best use of all running SEs.

At this point, placement behavior diverges.

With Compact Placement ON subsequent placements of VS3 through VS8 require no additional SEs to maintain HA (M=1 => one SE failure). The NSX Advanced Load Balancer prefers to place virtual services on existing SEs.

Distributed Placement ON Subsequent placements of VS3 and VS4 result in scaling the SE group out to its maximum, 4, illustrating NSX Advanced Load Balancer's preference for performance at the expense of resources. Having reached 4 deployed SEs, the maximum number of SEs for this group, the NSX Advanced Load Balancer places virtual services VS5 through VS8 on pre-existing, least-loaded SEs.



Interaction of Compact Placement with Elastic HA Modes

Compact placement interacts in a subtle way with the elastic HA modes in regards to timing.

Elastic HA N+M mode: In figure 2, because compact placement is ON by default in N+M mode, the NSX Advanced Load Balancer Controller deferred deployment of spare capacity, preferring instead to immediately pack virtual services densely onto existing SEs.

Elastic HA active/active mode: In figure 7, because distributed placement option is ON by default in active/active mode, the NSX Advanced Load Balancer Controller delays the placement of VS2 and VS3 until replacement SE7 spin ups. No additional burden is placed on the four surviving SEs (SE1, SE2, SE4, SE5). Instead, both virtual services are placed on a fresh SE, so that all virtual services may perform as they did before the failure.

Auto-Rebalance

This section discusses about Auto-Rebalance.

The auto-rebalance option applies only to the elastic HA modes, and is OFF by default. If auto-rebalance is left in its default OFF state, an event is logged instead of automatically performing migrations. Refer to [CLI Guide](#) to enable Auto-Rebalance.

If auto-rebalance is left in its default OFF state, an event is logged instead of automatically performing migrations.

Configuring Elastic HA

To configure elastic HA for an SE group:

Procedure

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Click the cloud name (for example, "Default-Cloud").
- 3 Click **Service Engine Group**.
- 4 Click the edit icon next to the SE group name, or click **Create** to create new one.
- 5 Fill out the requisite fields.
- 6 Click **Save**.

Notes and Recommendations

This section discusses about Notes and Recommendations.

Virtual services are non-disruptive during SE upgrade, with one exception: Elastic HA, N+M Buffer mode, for virtual services placed on just one SE (not scaled out). Read more about this subject in the [Upgrading NSX Advanced Load Balancer Software](#).

Elastic HA N + M mode (the default) is typically applied to applications wherein the following conditions apply:

- 1 The SE performance required by any application can be delivered by a fraction of one SE's capacity, hence each virtual service is typically placed on a single SE.
- 2 Applications can tolerate brief outages, albeit no longer than it takes to place a VS on an existing SE and plumb its network connections. This is usually no more than a few seconds.

The pre-existence of buffer SE capacity, coupled with the default setting of compact placement ON, speeds re-placement of virtual services affected by a failure. NSX Advanced Load Balancer does not wait for a substitute SE to be spun up; it immediately places affected virtual services on spare capacity.

Most applications need for HA is satisfied by $M=1$. However, in dev/test environments, M may be set to 0, since the dev/test users can wait for a new SE to be spun up before the VS is placed again.

Elastic HA active/active mode is typically applied to mission-critical applications where the virtual services must continue without interruption during the recovery period.

Additional Information

[Difference between HA_MODE_SHARED and HA_MODE_SHARED_PAIR options available on NSX Advanced Load Balancer CLI](#)

Preparing For Installation

2

This section explains the system requirements, software version compatibility, and the network considerations required for installing the cloud.

This chapter includes the following topics:

- [System Requirements](#)
- [Software Version Compatibility](#)
- [System Limits](#)
- [NSX Advanced Load Balancer IPAM and DNS](#)
- [Networking Considerations](#)
- [Changing NSX Advanced Load Balancer Controller Cluster Configuration](#)
- [NSX Advanced Load Balancer Controller Sizing](#)
- [Data Plane Development Kit \(DPDK\)](#)
- [Post-installation Configuration](#)

System Requirements

This section explains the system requirements of various clouds.

VMware Ecosystems

The following are the VMware vCenter versions supported by the NSX Advanced Load Balancer:

NSX Advanced Load Balancer ControllerVersion	VMware Ecosystems	Virtual Hardware Version 10
17.2.x	5.5, 6.0, 6.4, 6.5, or 6.7	10.0
18.1.x	5.5, 6.0, 6.4, 6.5, or 6.7	10.0
20.1.1 to 20.1.5	5.5, 6.0, 6.4, 6.5, 6.7, 7.0	10.0

NSX Advanced Load Balancer Controller Version	VMware Ecosystems	Virtual Hardware Version 10
20.1.6	6.0, 6.4, 6.5, 6.7, 7.0	11.0
21.1.x	6.0, 6.4, 6.5, 6.7, 7.0	11.0

Note When upgrading across releases having different virtual hardware versions, the existing SEs which are created with a previous virtual hardware versions continue to work. However, new SEs are spawned with the updated virtual hardware version.

NSX Advanced Load Balancer only works with the following switches in a vCenter Cloud:

- Standard Virtual Switch (vSwitch)
- Distributed Virtual Switch (vDS)

Also, the deployment of the Controller and SE OVA directly on an ESX host (for a no-access or other cloud-connector types) is not supported. They must be deployed using the vCenter UI, as the deployment requires Open Virtualization Format (OVF) properties to be configured.

- VMXNET3 network adapter is required for SE to operate in DPDK mode in VMware.

NSX-T - NSX Advanced Load Balancer Support Matrix

NSX-T Version	Minimum NSX Advanced Load Balancer Controller Version
2.5.1+	20.1.1
3.0.0+	20.1.1
3.1.0+	20.1.3

Jumbo-Drivers Supported

Supported Drivers	Unsupported Drivers
i40e, mlx5, bnxt, vmxnet3	ixgbe

- Jumbo frames are supported for non-DPDK mode and on VLAN interfaces for supported driver families. However, the jumbo frames are not supported on CSP.

Note `vmxnet3` interface will flap during MTU change.

- KNI MTU cannot exceed 1500 even when NIC MTU is configured.
- `se_mtu` vs `global_mtu`: `global_mtu` is an SE property that is used to configure the interface MTU. This can be used to accommodate any encapsulation overhead, that can enlarge the packet beyond the 1500 MTU.

- You can replace `global_mtu` with `se_mtu` as `se_mtu` supports jumbo frame. The `se_mtu` configuration parameter/ field always overrides `global_mtu`, if configured. For instance, if you configure `se_mtu` to 9000, then the system does not depend on `global_mtu` value.
-
- **Note** The `global_mtu` is retained only for backward compatibility, i.e., if you configure `global_mtu` in an earlier release and do an upgrade, then `global_mtu` should still take effect unless you configure `se_mtu` later.
-

OpenStack Version Support

NSX Advanced Load Balancer Controller Versions	Supported OpenStack Versions
20.1.1+	Queens, Rocky, Stein, Train
20.1.3+	Ussuri + All releases supported in 20.1.1
20.1.5+	Victoria + All releases supported in 20.1.3
21.1.2+	Wallaby + All releases supported in 20.1.5

Note NSX Advanced Load Balancer does not support Neutron DVR mode. It supports Keystone v3 in NSX Advanced Load Balancer Heat resources.

No-Access Mode

No-Orchestrator KVM (Red Hat/CentOS- 7.6 and Ubuntu 16.04) with SR-IOV NICs only.

Bare Metal (Linux Server Cloud)

NSX Advanced Load Balancer Controller Version	Bare Metal Hosts
18.1.2	<ul style="list-style-type: none"> ■ OEL 6.9, 7.2, 7.3, 7.4, 7.5 ■ RHEL 7.2, 7.3, 7.4, 7.5 ■ CentOS 7.2, 7.3, 7.4, 7.5
18.1.5	<ul style="list-style-type: none"> ■ OEL 7.6 ■ RHEL 7.6 ■ CentOS 7.6
18.2.6	<ul style="list-style-type: none"> ■ OEL 7.7 ■ RHEL 7.7 ■ CentOS 7.7
18.2.9, 20.1.1	<ul style="list-style-type: none"> ■ OEL 7.8 ■ RHEL 7.8 ■ CentOS 7.8
<ul style="list-style-type: none"> ■ In 18.2.x: version 18.2.12 onwards ■ In 20.1.x: version 20.1.3 onwards 	<ul style="list-style-type: none"> ■ OEL 7.9 ■ RHEL 7.9 ■ CentOS 7.9

NSX Advanced Load Balancer Controller Version	Bare Metal Hosts
21.1	<ul style="list-style-type: none"> ■ Ubuntu 18.04 ■ Ubuntu 20.04
All versions	Ubuntu 16.04

Note

- For OpenShift/Kubernetes Clouds, the host OS on OpenShift/Kubernetes nodes can have RHEL 7.9 from version 18.2.12 onwards.
- Rollback operation is not supported once host OS is upgraded to RHEL 8.

For more information on Kernel supported version in bare metal hosts, see [Kernel Supported Versions in Bare Metal - 20.1.x](#).

Bare-Metal NICs

- For non-DPDK mode, NSX Advanced Load Balancer supports any server NIC.
- For DPDK mode (recommended), NSX Advanced Load Balancer has qualified:

NICs Name	NSX Advanced Load Balancer Version
Intel NICs	<ul style="list-style-type: none"> ■ All 18.2.x and 20.1.x releases - 82599, X520, X540, X550, X552, X710, XL710, XXV710. ■ Starting from NSX Advanced Load Balancer version 18.2.2 and all 20.1.x releases - XXV710.
Mellanox NICs	<ul style="list-style-type: none"> ■ NICs Supported: <ul style="list-style-type: none"> ■ All 18.2.x and 20.1.x releases - ConnectX-4 25G and ConnectX-4 40G. ■ All 18.2.x and 20.1.x releases - MCX4121A-ACAT ConnectX-4 Lx EN 25G NICs. ■ Mellanox Technologies MT27800 Family [ConnectX-5] (introduced in 20.1.6 and 21.1.1 releases). ■ MLNX_OFED version (tested and qualified for NSX Advanced Load Balancer version 21.1.1 onwards) - MLNX_OFED_LINUX-5.1-2.5.8.0 (OFED-5.1-2.5.8).
Broadcom NICs	<ul style="list-style-type: none"> ■ BCM574XX NetXtreme-E family — Starting from NSX Advanced Load Balancer version 18.2.8 and all 20.1.x releases. ■ Firmware version — 219.0.111.0/pkg 21.90.13.50.

IPAM / DNS

The following IPAM DNS are supported:

- NSX Advanced Load Balancer DNS
- AWS Route 53
- Infoblox
- Microsoft Azure

Hardware Security Module (HSM)

The following HSM are supported:

- SafeNet Network HSM Client Software Release 5.4.1 for 64-bit Linux
- AWS CloudHSM v2

Kernel Supported Versions in Bare Metal - 21.1.x

This section lists the supported kernel versions in bare metal hosts for the NSX Advanced Load Balancer Controller.

Version 21.1.1

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 21.1.1 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64 ■ 3.10.0-1160.25.1.el7.x86_64 ■ 3.10.0-1160.31.1.0.1.el7.x86_64 ■ 3.10.0-1160.41.1.el7.x86_64 ■ 3.10.0-1160.42.2.el7.x86_64 ■ 3.10.0-1160.45.1.el7.x86_64 ■ 3.10.0-1160.53.1.0.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	<ul style="list-style-type: none"> ■ 4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-147.el8.x86_64 ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-193.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	<ul style="list-style-type: none"> ■ 5.4.0-26-generic

Version 21.1.2

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 21.1.2 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64 ■ 3.10.0-1160.25.1.el7.x86_64 ■ 3.10.0-1160.31.1.0.1.el7.x86_64 ■ 3.10.0-1160.36.2.el7.x86_64 ■ 3.10.0-1160.41.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	<ul style="list-style-type: none"> ■ 4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp ■ 4.15.0-20-generic
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-147.el8.x86_64 ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-193.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	<ul style="list-style-type: none"> ■ 5.4.0-26-generic

Version 21.1.3

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 21.1.3 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64 ■ 3.10.0-1160.21.1.el7.x86_64 ■ 3.10.0-1160.25.1.el7.x86_64 ■ 3.10.0-1160.31.1.0.1.el7.x86_64 ■ 3.10.0-1160.36.2.el7.x86_64 ■ 3.10.0-1160.41.1.el7.x86_64 ■ 3.10.0-1160.42.2.el7.x86_64 ■ 3.10.0-1160.45.1.el7.x86_64 ■ 3.10.0-1160.49.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64

Kernel Supported Versions in Bare Metal - 20.1.x

This section lists the supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.x.

Version 20.1.1

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.1 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp

Version 20.1.2

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.2 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp

Version 20.1.3

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.3 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64

Version 20.1.4

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.4 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64

Version 20.1.5

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.5 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp ■ 4.15.0-20-generic
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	5.4.0-26-generic

Version 20.1.6

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.6 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp ■ 4.15.0-20-generic
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	5.4.0-26-generic

Version 20.1.7

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.7 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	<ul style="list-style-type: none"> ■ 4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp ■ 4.15.0-20-generic
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	<ul style="list-style-type: none"> ■ 5.4.0-26-generic

Version 20.1.8

The supported kernel versions in bare metal hosts for NSX Advanced Load Balancer Controller version 20.1.8 are:

Supported Base Kernel Versions	Kernel Versions
3.8.13	<ul style="list-style-type: none"> ■ 3.8.13-35.3.1.el7uek.x86_64 ■ 3.8.13-118.10.2.el7uek.x86_64 ■ 3.8.13-16.2.1.el6uek.x86_64 ■ 3.8.13-98.7.1.el7uek.x86_64
3.10.0	<ul style="list-style-type: none"> ■ 3.10.0-327.28.2.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-514.16.1.el7.x86_64 ■ 3.10.0-514.6.1.el7.x86_64 ■ 3.10.0-693.el7.x86_64 ■ 3.10.0-693.1.1.el7.x86_64 ■ 3.10.0-693.2.2.el7.x86_64 ■ 3.10.0-862.3.2.el7.x86_64 ■ 3.10.0-862.11.6.el7.x86_64 ■ 3.10.0-862.9.1.el7.x86_64 ■ 3.10.0-862.6.3.el7.x86_64 ■ 3.10.0-862.3.3.el7.x86_64 ■ 3.10.0-862.2.3.el7.x86_64 ■ 3.10.0-862.el7.x86_64 ■ 3.10.0-862.14.4.el7.x86_64 ■ 3.10.0-862.29.1.el7.x86_64 ■ 3.10.0-957.el7.x86_64 ■ 3.10.0-957.1.3.el7.x86_64 ■ 3.10.0-957.5.1.el7.x86_64 ■ 3.10.0-957.10.1.el7.x86_64 ■ 3.10.0-957.10.1.0.1.el7.x86_64 ■ 3.10.0-957.12.1.el7.x86_64 ■ 3.10.0-957.12.2.0.1.el7.x86_64 ■ 3.10.0-957.21.2.el7.x86_64 ■ 3.10.0-957.21.3.el7.x86_64 ■ 3.10.0-1062.el7.x86_64 ■ 3.10.0-1062.9.1.el7.x86_64 ■ 3.10.0-1062.9.1.0.1.el7.x86_64 ■ 3.10.0-1062.12.1.el7.x86_64 ■ 3.10.0-1062.18.1.el7.x86_64 ■ 3.10.0-1062.26.1.el7.x86_64 ■ 3.10.0-1127.el7.x86_64 ■ 3.10.0-1127.0.0.0.1.el7.x86_64 ■ 3.10.0-1127.8.2.el7.x86_64 ■ 3.10.0-1127.8.2.0.1.el7.x86_64 ■ 3.10.0-1127.10.1.el7.x86_64 ■ 3.10.0-1127.13.1.0.1.el7.x86_64 ■ 3.10.0-1127.18.2.el7.x86_64 ■ 3.10.0-1127.19.1.el7.x86_64

Supported Base Kernel Versions	Kernel Versions
	<ul style="list-style-type: none"> ■ 3.10.0-1127.19.1.0.1.el7.x86_64 ■ 3.10.0-1127.19.1.0.2.el7.x86_64 ■ 3.10.0-1136.el7.x86_64 ■ 3.10.0-1160.el7.x86_64 ■ 3.10.0-1160.2.1.el7.x86_64 ■ 3.10.0-1160.2.2.el7.x86_64 ■ 3.10.0-1160.2.2.0.1.el7.x86_64 ■ 3.10.0-1160.6.1.el7.x86_64 ■ 3.10.0-1160.11.1.el7.x86_64 ■ 3.10.0-1160.15.2.el7.x86_64 ■ 3.10.0-1160.21.1.el7.x86_64 ■ 3.10.0-1160.25.1.el7.x86_64 ■ 3.10.0-1160.31.1.0.1.el7.x86_64 ■ 3.10.0-1160.36.2.el7.x86_64 ■ 3.10.0-1160.41.1.el7.x86_64 ■ 3.10.0-1160.42.2.el7.x86_64 ■ 3.10.0-1160.45.1.el7.x86_64 ■ 3.10.0-1160.49.1.el7.x86_64 ■ 3.10.0-1160.53.1.0.1.el7.x86_64
4.1.12	<ul style="list-style-type: none"> ■ 4.1.12-61.1.18.el7uek.x86_64 ■ 4.1.12-94.2.1.el7uek.x86_64
4.4.0	<ul style="list-style-type: none"> ■ 4.4.0-21-generic ■ 4.4.0-51-generic ■ 4.4.0-59-generic ■ 4.4.0-62-generic ■ 4.4.0-66-generic ■ 4.4.0-70-generic ■ 4.4.0-72-generic ■ 4.4.0-112-generic ■ 4.4.0-116-generic ■ 4.4.0-137-generic ■ 4.4.0-146-generic ■ 4.4.0-178-generic
4.14.35	<ul style="list-style-type: none"> ■ 4.14.35-1902.10.8.el7uek.x86_64
4.15.0	<ul style="list-style-type: none"> ■ 4.15.0-1017-gcp ■ 4.15.0-1027-gcp ■ 4.15.0-20-generic

Supported Base Kernel Versions	Kernel Versions
4.18.0	<ul style="list-style-type: none"> ■ 4.18.0-193.el8.x86_64 ■ 4.18.0-193.6.3.el8_2.x86_64 ■ 4.18.0-93.14.3.el8_2.x86_64 ■ 4.18.0-240.1.1.el8_3.x86_64
5.4.0	<ul style="list-style-type: none"> ■ 5.4.0-26-generic

Software Version Compatibility

This section gives details on software compatibility versions for various ecosystems.

Software Requirements for Linux Server Cloud

The following are the software requirements for installing NSX Advanced Load Balancer on a Linux server cloud:

Software	Version
NSX Advanced Load Balancer (distributed by VMware as a Docker image)	18.2 or greater
Docker (image management service that runs on Linux)	1.6.1 or greater
Operating systems (OS) and kernel versions to enable DPDK	See System Requirements section
Python to run scripts	Installable various ways; left to the customer's preference

Software Requirements for OpenStack Cloud

The following table lists the software requirements:

Software	Version
NSX Advanced Load Balancer Controller	20.1.x
OpenStack (and Neutron service)	<ul style="list-style-type: none"> ■ Ocata ■ Pike ■ Queens ■ Rocky ■ Stein ■ Train ■ Ussuri
Neutron extension for allowed-address-pair	

OpenStack Support Matrix

This section discusses the support matrix for OpenStack deployment in the NSX Advanced Load Balancer.

OpenStack Supported Releases

NSX Advanced Load Balancer Versions	Supported OpenStack Versions
20.1.1+	Queens, Rocky, Stein, Train
20.1.3+	Ussuri + All releases supported in 20.1.1
20.1.5+	Victoria + All releases supported in 20.1.3
21.1.2+	Wallaby + All releases supported in 20.1.5

Contrail

NSX Advanced Load Balancer Versions	Supported Contrail Versions
20.1.+	19.12, 5.1, 3.2

Supported Features

The following table lists the support matrix for OpenStack deployment in NSX Advanced Load Balancer for version 20.1.+:

Feature	Details
SDNs	<ul style="list-style-type: none"> ■ ML2/OVS ■ ML2/OVN ■ Contrail
DPDK Support	<ul style="list-style-type: none"> ■ 20.1.+
Hypervisors	<ul style="list-style-type: none"> ■ KVM
Scale out	<ul style="list-style-type: none"> ■ Layer 2 scale out ■ Layer 3 scale out using Contrail ■ ECMP scale out in Contrail environment
Image formats	<ul style="list-style-type: none"> ■ qcow2 and raw
Keystone	<ul style="list-style-type: none"> ■ Keystone v3
Connectivity	<ul style="list-style-type: none"> ■ Allowed Address Pair in ML2/OVS ■ Fixed IP address in Contrail

Note Unsupported installation methods for the NSX Advanced Load Balancer are PackStack, DevStack. For more information on NSX Advanced Load Balancer Heat Driver support matrix, see [NSX Advanced Load Balancer Heat Driver](#).

Deprecated Features

The following table lists the features deprecated on the NSX Advanced Load Balancer starting the mentioned version:

Feature	Details	NSX Advanced Load Balancer Version
BGP Scale out	BGP is not supported for OpenStack deployments	All versions
Keystone	Keystone version 2	18.2.8
Connectivity	Port security	18.2.8
IPAM	OpenStack IPAM for K8s/OpenShift and LSC	18.2.8
SDNs	<ul style="list-style-type: none"> ■ Nuage-as-the-SDN ■ ACI-as-the-SDN 	20.1.1
Modes of Provisioning	<p>LBaaSv2 support on NSX Advanced Load Balancer is deprecated starting from OpenStack 'Stein' release. Recommended to migrate to NSX Advanced Load Balancer managed mode.</p> <hr/> <p>Note</p> <ol style="list-style-type: none"> 1 LBaaSv1 is already deprecated as mentioned at LBaaS V2 Driver. 2 Octavia integration is not supported. 3 Recommended to use NSX Advanced Load Balancer managed mode. 	20.1.1
UI iframe	Horizon dashboard	20.1.1

System Limits

The NSX Advanced Load Balancer Controller includes a framework for tracking and enforcing various system-wide limits, based on the size of the Controller instances and other parameters.

Controller Classification Sizing

The Controllers are classified into the following categories:

Classification	vCPUs	Memory (GB)
Small	8-15	24-32
Medium	16-23	32-48
Large	>24	>48

For more information on CPU memory usage, see [NSX Advanced Load Balancer Controller Sizing](#).

Limits Enforced

For more information on the limit enforced for various entities, see [VMware Configuration Maximums](#).

NSX Advanced Load Balancer IPAM and DNS

This section discusses the native IPAM and DNS support for the various clouds.

Provider	NSX Advanced Load Balancer					
	Infoblox		Internal		Cloud-native	
Cloud Infrastructure	IPAM	DNS	IPAM	DNS	IPAM	DNS
VMware vCenter	Yes	Yes	Yes	Yes	NA	NA (Not Used)
No access cloud	Yes	Yes	Yes	Yes	Yes	No

- When creating virtual services in OpenStack or AWS cloud, a separate configuration for IPAM is not needed/allowed, since the cloud configuration has support for IPAM natively in NSX Advanced Load Balancer.
 - When you select 'Default', the cloud's IPAM/DNS support is accepted without additional action on the part of the administrator.
 - NSX Advanced Load Balancer supports Route 53 when AWS is the cloud provider configuration in NSX Advanced Load Balancer.
 - 'Not Used' means, although the cloud supports DNS, NSX Advanced Load Balancer does not use it.
- When creating a virtual service in the Linux Server cloud in AWS/GCP environment, you can use the cloud-native IPAM solution of AWS/GCP.
- NSX Advanced Load Balancer DNS service can be used with all these clouds.

IPAM and DNS Provider (Infoblox)

The Controller integrates with Infoblox's RESTful Web API (WAPI) for IPAM and DNS services.

These API calls are initiated by the Controller and directed to the Infoblox Grid Master IP address, or virtual IP address (VIP), if has been deployed in a HA pair. This integration enables the NSX Advanced Load Balancer to automate the allocation of IP addresses as well as the creation and deletion of host objects in DNS as new virtual services are created/deleted in the NSX Advanced Load Balancer environment.

It is assumed that all interested subnets and domain names (zones) have been configured in the Infoblox server for consumption by NSX Advanced Load Balancer. That said, when configuring Infoblox DNS and IPAM profiles, it is possible to be selective, as the next section will show.

Infoblox IPAM and DNS profiles should be independently defined and configured. For a given cloud, the permitted Infoblox combinations are:

IPAM Provider	DNS Provider
Infoblox IPAM	None, Infoblox DNS, NSX Advanced Load Balancer internal
Any	Infoblox DNS

Configuring NSX Advanced Load Balancer IPAM

This section discusses NSX Advanced Load Balancer's IPAM configuration that is supported on the following clouds:

- Linux Server Cloud (bare metal)
- VMware
- No Access

Configuring IPAM

NSX Advanced Load Balancer allocates IP addresses from a pool of IP addresses within the subnet configured as listed:

- 1 Navigate to **Infrastructure > Network**.
- 2 You can select cloud from **Select Cloud** drop-down menu and click **Create**.
- 3 Specify the network name.
- 4 Under **IP Address Management**, click the required option for **DHCP Enabled** and **IPv6 Auto Configuration**.
- 5 Add IPv4 and/or IPv6 networks for IP address allocation.
 - a Click **Add Subnet**.
 - b Specify the subnet address under **IP Subnet**.
 - c Enable **Add Static IP Address Pool** to specify the pool of IP addresses. Specify the range of the pool under **IP Address Pool**.
 - d Click **Save**.

e Repeat steps from 1 to 4 for each network used for IP address allocation.

6 Click **Save**.

Note

- Virtual service creation will fail if the static IP address pool is empty or exhausted.
- For east-west IPAM, create another network with the appropriate link-local subnet and a separate IPAM/DNS profile.
- Addition or deletion of VIP or changing the `vip_id` (use-case being multiple VIPs on a virtual service) is not supported on NSX Advanced Load Balancer IPAM.

Creating IPAM Networks using both IPv4 and IPv6 Subnets

The following is an instance of creating IPAM networks using both IPv4 and IPv6 subnets:

Navigate to **Templates > IPAM/DNS Profiles** and create a placeholder for IPAM.

You can assign one or more of the created networks to be the default usable network, if no specific network and/or subnet is provided in the virtual service configuration.

Edit IPAM/DNS Profile: IPv4v6_Avi_Vantage_IPAM

Name * ⓘ

IPv4v6_Avi_Vantage_IPAM

Type ⓘ

Avi Vantage IPAM

☐ Allocate IP in VRF ⓘ

Avi Vantage IPAM Configuration

Cloud for Usable Network ⓘ

Default-Cloud

Usable Network * ⓘ

Select Network

Search

IPv4v6_IPAM_Network - 10.140.116.0/24, fd00:0:0:116::/64

Save

VRF-aware IPAM

You can enable the **Allocate IP in the VRF** checkbox for NSX Advanced Load Balancer to allocate IPs from networks in the virtual service's VRF. This option is applicable only for the IPAM.

Selecting Network for IP Allocation

The selection of network for given allocated IP request is based on the following:

- 1 If a network and subnet are specified during virtual service creation, the system will allocate from that specific network/subnet. If that subnet does not have free static IPs, the API request will fail.
- 2 If no network/subnet is specified (only possible via CLI or API) during virtual service creation, the system will consider all networks in the **Usable Networks** of the IPAM/DNS profile and randomly select the one which has free IPs available.
 - a For v4 requests, the system will check for free IPs in networks with v4 subnets before considering networks with v4 and v6 subnets.
 - b For v6 requests, the system will check for free IPs in networks with v6 subnets before considering networks with v4 and v6 subnets.

Note Any change in the VIP's IPv4 or IPv6 address disrupt the virtual service. This can occur if the virtual service's auto allocate type is changed. For instance, if a virtual service's IPv4 address was allocated using a network with both v4 and v6 subnets, and its `auto_allocate_type` is changed from v4 to v4_v6 with a corresponding v6 subnet selected, the system will attempt to allocate an IPv6 address for that virtual service. If the allocation is successful, a virtual service disruption will occur.

IPAM Support for User Preferred IP Address

NSX Advanced Load Balancer IPAM supports virtual service creation with a user-preferred IP address and or IPv6 address with auto allocation. To use this feature,

- Set the `ip_address` or `ip6_address` field(s) of the `VsVip` object.
- Set the `auto_allocate_ip` field to `True` and the `auto_allocate_ip_type` field set correspondingly.

The Controller allocates that specific IP address for the virtual service. If the IP address is unavailable, the virtual service creation will fail. The specified IP address must exist in a static pool that is already configured on a network or the subnet.

This feature supports all three auto allocation types, namely, v4, v6, and v4_v6. When creating a virtual service IP address with v4_v6 allocation, both IPv4 and IPv6 addresses must be specified, or both should be left empty. Additionally, updating an existing auto-allocated IP address to a different preferred IP address of the same type (v4 or v6) is not allowed.

The following are the list of allowed operations:

- 1 Creating a VIP with a preferred static IP, supported for v4, v6, and v4_v6.

- 2 Changing an existing VIP's allocation type from v4 to v6 and specifying a preferred IPv6.
- 3 Changing an existing VIP's allocation type from v4 to v4_v6 and specifying a preferred IPv6.
 - a If the IPAM network and subnet are the same, the IPv4 address field must be either unset or kept the same (the existing IPv4 address will be preserved in both cases).
 - b If the IPAM network or subnet is different, the IPv4 address field must be unset.
- 4 Changing an existing VIP's allocation type from v6 to v4 and specifying a preferred IPv4.
- 5 Changing an existing VIP's allocation type from v6 to v4_v6, and specifying a preferred IPv4. If the IPAM network and subnet6 is the same, the **IPv6 Address** field must be either unset or kept the same (the existing IPv6 address will be preserved in both cases).
 - a If the IPAM network or subnet6 is different, the **IPv6 Address** field must be unset.

The **IPv6 Address** field must be kept the same to keep the IPAM network same, or left blank for changing the IPAM network.

The following operations are not supported in NSX Advanced Load Balancer 20.1.2:

- 1 Creating a VIP with v4_v6 allocation with only `ip_address` set or only `ip6_address` set.
 - a Both IP addresses must be set (preferred), or unset.
- 2 Updating an existing auto allocated IP address to a different preferred IP address of the same type (v4 or v6).
 - a An existing VIP with `IPv4-A` cannot be updated to a different preferred `IPv4-B`.
 - b If it is required to change the VIP's allocation network or subnet, the `ip_address/` `ip6_address` fields must be left blank (Controller will pick the IP address for the user).
 - c If a new preferred IP of the same type is needed, delete and recreate the VIP.

Configuring Virtual Service with Auto Allocate IP Address

Login to the CLI and use the `configure vsvip <name>` to set the auto allocate IP address.

```
[admin:10-79-108-162]: > show network network1
```

Field	Value
uuid	network-eea5aaa2-2225-40bd-b27d-60d7fe046d01
name	network1
vcenter_dvs	True
dhcp_enabled	True
exclude_discovered_subnets	False
configured_subnets[1]	
prefix	10.10.10.0/24
static_ranges[1]	
begin	10.10.10.100
end	10.10.10.150
vrf_context_ref	global
syncd_from_se	False
ip6_autocfg_enabled	True

```

| tenant_ref          | admin          |
| cloud_ref           | Default-Cloud  |
+-----+-----+
[admin:10-79-108-162]: > configure vsvip vsvip1

[admin:10-79-108-162]: vsvip> vip vip_id 1
New object being created
[admin:10-79-108-162]: vsvip:vip> auto_allocate_ip
[admin:10-79-108-162]: vsvip:vip> ip_address 10.10.10.120
[admin:10-79-108-162]: vsvip:vip> save
[admin:10-79-108-162]: vsvip> save
+-----+-----+
| Field              | Value          |
+-----+-----+
| uuid               | vsvip-54aa9247-d807-458d-b9e3-a8956bcb266a |
| name               | vsvip1         |
| vip[1]             |                |
|   vip_id           | 1              |
|   ip_address        | 10.10.10.120   |
|   enabled           | True           |
|   discovered_networks[1] |                |
|     network_ref     | network1       |
|     subnet[1]       | 10.10.10.0/24  |
|   auto_allocate_ip  | True           |
|   auto_allocate_floating_ip | False         |
|   avi_allocated_vip | False          |
|   avi_allocated_fip | False          |
|   ipam_network_subnet |                |
|     network_ref     | network1       |
|     subnet          | 10.10.10.0/24  |
|   auto_allocate_ip_type | V4_ONLY        |
|   prefix_length     | 32             |
| vrf_context_ref     | global         |
| east_west_placement | False          |
| tenant_ref          | admin          |
| cloud_ref           | Default-Cloud  |
+-----+-----+

```

Allocating different IPAM Ranges for SEs and Virtual IPs

You can specify whether a set of static IPs is used for SE vNIC only, or for VIP only or for both. For any given subnet, only the following configurations are supported:

- IP range(s) for VIP and/or IP range(s) for SE
- IP range(s) for both

The system will display an error message if a subnet contains an IP range for both and an IP range for either VIP or SE.

Using the UI

The following are the steps to allow separate IP range configurations for VIP and SE:

- 1 From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Networks**.

- 2 Click **Edit**.
- 3 In the **Edit Network Settings** screen, disable **Use Static IP Address for VIPs and SE** option. If you select this option, the IP ranges will be used for both VIPs and SE.
- 4 In the **Networks Overview** page, click an existing network to show the various configured static IP ranges. The combined free/total IP counts of all the `ip_range_runtimes` in the subnet are shown next to the subnet prefix.

Name	Discovered Subnets	Configured Subnets	Static IP Pools
vxw-dvs-34-virtualwire-33-sid-2140032-wdc-02-vc14-avi-dev026	100.64.34.0/24	100.64.34.0/24 [45/45]	5

IP Subnet (Configured) 100.64.34.0/24 [45/45]

Static IP Pools (Type) 100.64.34.100 - 100.64.34.110 (VIP)
 100.64.34.200 - 100.64.34.210 (VIP)
 100.64.34.140 - 100.64.34.150 (SE)
 100.64.34.240 - 100.64.34.250 (SE)
 100.64.34.195 - 100.64.34.195 (SE)

Via the CLI

Specify a static IP address range in the `static_ip_ranges` field and also how the IP ranges will be allocated.

The following is the `static_ip_ranges` configuration:

```
configure network vxw-dvs-34-virtualwire-33-sid-2140032-wdc-02-vc14-avi-dev026
[admin:1234]: network> configured_subnets prefix 100.64.34.0/24
[admin:1234]: network:configured_subnets> static_ip_ranges
New object being created
[admin:1234]: network:configured_subnets:static_ip_ranges> range begin 100.64.34.100
[admin:1234]: network:configured_subnets:static_ip_ranges:range> end 100.64.34.110
[admin:1234]: network:configured_subnets:static_ip_ranges:range> save
[admin:1234]: network:configured_subnets:static_ip_ranges> type static_ips_for_vip
[admin:1234]: network:configured_subnets:static_ip_ranges> save
[admin:1234]: network:configured_subnets> static_ip_ranges
[admin:1234]: network:configured_subnets:static_ip_ranges> range begin 100.64.34.140
[admin:1234]: network:configured_subnets:static_ip_ranges:range> end end 100.64.34.150
[admin:1234]: network:configured_subnets:static_ip_ranges:range> save
[admin:1234]: network:configured_subnets:static_ip_ranges> type static_ips_for_vip
[admin:1234]: network:configured_subnets:static_ip_ranges> save
[admin:1234]: network:configured_subnets:static_ip_ranges> save
[admin:1234]: network:configured_subnets> static_ip_ranges
[admin:1234]: network:configured_subnets:static_ip_ranges> range begin 100.64.34.240
[admin:1234]: network:configured_subnets:static_ip_ranges:range> end end 100.64.34.250
[admin:1234]: network:configured_subnets:static_ip_ranges:range> save
[admin:1234]: network:configured_subnets:static_ip_ranges> type static_ips_for_se
[admin:1234]: network:configured_subnets:static_ip_ranges> save
[admin:1234]: network:configured_subnets> static_ip_ranges
[admin:1234]: network:configured_subnets:static_ip_ranges> range begin 100.64.34.195
[admin:1234]: network:configured_subnets:static_ip_ranges:range> end end 100.64.34.195
[admin:1234]: network:configured_subnets:static_ip_ranges:range> save
[admin:1234]: network:configured_subnets:static_ip_ranges> type static_ips_for_se
[admin:1234]: network:configured_subnets:static_ip_ranges:range> save
```

```
[admin:1234]: network:configured_subnets:static_ip_ranges>save
[admin:1234]: network:configured_subnets>save
```

Note

- Within `ip_range_runtimes`, the allocated IPs are stored inside the `allocated_ips` field (previously named as `ip_allocated`).
- Inside an allocated IP, the `mac` field has been renamed to `obj_info` and the `se_ref` field has been renamed to `obj_ref`.
- By default, the option `STATIC_IPS_FOR_VIP_AND_SE` is configured as the type of allocation. When upgrading to NSX Advanced Load Balancer version 20.1.3, all existing `static_ips` and `static_ranges` will be converted to `static_ip_ranges` with type `STATIC_IPS_FOR_VIP_AND_SE`.

The following is the configured network:

Field	Value
<code>uuid</code>	<code>dvportgroup-233-cloud-4b5fd097-0a9a-444f-b328-1f016eb99987</code>
<code>name</code>	<code>vxw-dvs-34-virtualwire-33-sid-2140032-wdc-02-vc14-avi-dev026</code>
<code>vcenter_dvs</code>	<code>True</code>
<code>vimgrnw_ref</code>	<code>vxw-dvs-34-virtualwire-33-sid-2140032-wdc-02-vc14-avi-dev026</code>
<code>dhcp_enabled</code>	<code>True</code>
<code>exclude_discovered_subnets</code>	<code>False</code>
<code>configured_subnets[1]</code>	
<code>prefix</code>	<code>100.64.34.0/24</code>
<code>static_ip_ranges[1]</code>	
<code>range</code>	
<code>begin</code>	<code>100.64.34.100</code>
<code>end</code>	<code>100.64.34.110</code>
<code>type</code>	<code>STATIC_IPS_FOR_VIP</code>
<code>static_ip_ranges[2]</code>	
<code>range</code>	
<code>begin</code>	<code>100.64.34.200</code>
<code>end</code>	<code>100.64.34.210</code>
<code>type</code>	<code>STATIC_IPS_FOR_VIP</code>
<code>static_ip_ranges[3]</code>	
<code>range</code>	
<code>begin</code>	<code>100.64.34.140</code>
<code>end</code>	<code>100.64.34.150</code>
<code>type</code>	<code>STATIC_IPS_FOR_SE</code>
<code>static_ip_ranges[4]</code>	
<code>range</code>	
<code>begin</code>	<code>100.64.34.240</code>
<code>end</code>	<code>100.64.34.250</code>
<code>type</code>	<code>STATIC_IPS_FOR_SE</code>
<code>static_ip_ranges[5]</code>	
<code>range</code>	
<code>begin</code>	<code>100.64.34.195</code>
<code>end</code>	<code>100.64.34.195</code>
<code>type</code>	<code>STATIC_IPS_FOR_SE</code>
<code>vrf_context_ref</code>	<code>global</code>

synced_from_se	True	
ip6_autocfg_enabled	False	
tenant_ref	admin	
cloud_ref	Default-Cloud	
+-----+	+-----+	

Note The IP allocation and IP count information will be stored inside `ip_range_runtimes` field. The fields `ip_allocated`, `total_ip_count`, `used_ip_count`, and `free_ip_count` under `subnet_runtime` are deprecated. Each `ip_range_runtimes` entry will contain the combined IP allocation and count information for all static IP ranges of a particular type (SE, VIP, or both).

Internal IPAM for VIP Labels

You can use specific sets of networks from the IPAM profile for VIP allocation. Labels are added to both the usable networks in the IPAM profile and the `vsvip`.

Note This feature is currently supported only via the CLI/API.

The usable networks and vsvip are matched as shown below:

- A `vsvip` with label X can only use networks in the IPAM profile with label X
- A `vsvip` with no labels can use any network in the IPAM profile (with and without labels)

The labels for the networks in the IPAM profile is configured inside the profile's `usable_networks` field. The labels on the `vsvip` is configured inside the `ipam_selector` field.

Log in to the Controller and configure internal IPAM for VIP labels as shown below:

```
[admin:1234]: > configure vsvip vsvip1
[admin:1234]: vsvip> vip vip_id 1
New object being created
[admin:1234]: vsvip:vip> auto_allocate_ip
[admin:1234]: vsvip:vip> save
[admin:1234]: vsvip> ipam_selector
[admin:1234]: vsvip:ipam_selector> type selector_ipam
[admin:1234]: vsvip:ipam_selector> labels
New object being created
[admin:1234]: vsvip:ipam_selector:labels> key key2
[admin:1234]: vsvip:ipam_selector:labels> value value2
[admin:1234]: vsvip:ipam_selector:labels> save
[admin:1234]: vsvip:ipam_selector> save
[admin:1234]: vsvip> save
```

Note The `usable_networks_refs` field under `internal_profile` has been deprecated. To add networks, use the `usable_networks` field.

Changing an existing usable network's labels or vsvip's labels is allowed, and does not affect existing allocations. The new labels will be applicable for new allocations.

Only one label will be supported per usable network and per `vsvip`.

Configuring the IPAM for AWS

NSX Advanced Load Balancer integrates with Amazon Web Services (AWS) for providing IPAM services to applications running on instances in AWS.

If the NSX Advanced Load Balancer cloud type is AWS, there is support for IPAM by default, without the need for a separate IPAM configuration. A separate IPAM configuration (as described below) is required only for cases where AWS provides the infrastructure service for other clouds. For instance, Mesos Cluster running on AWS instances.

AWS IPAM is supported only for North-South IPAM Provider.

Configuring IPAM

To use AWS as the IPAM provider, one of the following types of credentials is required:

- Identity and Access Management (IAM) roles: Set of policies that define access to resources within AWS.
- AWS Customer Account Key: Unique authentication key associated with the AWS account.

Configuring the IPAM for GCP

NSX Advanced Load Balancer integrates with Google Cloud Platform (GCP) for providing Internet Protocol Address Management (IPAM) services to applications running on instances in GCP. This section explains how to create a GCP profile and configure GCP IPAM.

Note Starting with NSX Advanced Load Balancer version 20.1.3, Linux Server Cloud and GCP IPAM on GCP are not supported.

Prerequisites for GCP IPAM

- Set up the Controller and Service Engine instances on GCP.
- Install NSX Advanced Load Balancer in a Linux Cloud. To use GCP as the IPAM provider, all the Controller and Service Engine instances must run in Google Cloud. The cloud type selected within NSX Advanced Load Balancer should be Linux Server Cloud.
- GCP uses a /32-based interface IP configuration. To support this, perform the following after configuring the Linux Server Cloud.

The CLI command is as follows:

```
configure serviceengineproperties
se_runtime_properties
se_handle_interface_routes

service_ip_subnets subnet/mask
save
save
```

Reload all the Service Engines for the configuration to take effect.

The `service_ip_subnets` configuration optimizes the Docker routes on the Service Engines. If VIP allocation is done from multiple subnets, specify all of them.

Configuring GCP IPAM

Create the GCP IPAM in NSX Advanced Load Balancer, and configure it by following the steps given below.

You can create the IPAM profile as follows:

- 1 Navigate to **Templates > Profiles > IPAM/DNS profiles**.
- 2 Click **Create**.
- 3 Specify the **IPAM Profile Name**.
- 4 Select the **Type** as **Google Cloud Platform IPAM** from the drop-down menu.
- 5 Click **Save**.

You can configure GCP IPAM as follows:

- 1 Navigate to **Infrastructure > Cloud**.
- 2 Select **Default Cloud** and click the **Edit** icon.
- 3 The IPAM that was created appears in the **IPAM Profile** drop-down menu. Select the **IPAM Profile**.
- 4 Configure a Linux Server Cloud using the IP addresses for the Service Engine instances created.

GCP IPAM has been configured, and virtual services can now be provisioned.

Note On GCP only L3 scale out mode is supported for virtual services.

Some of the GCP zones may require the MTU to be lowered to 1400. To set the MTU on the Service Engines, you can use the following CLI command:

```
configure serviceengineproperties
se_runtime_properties
global_mtu 1400
save
save
```

The MTU value will take effect after the Service Engines are rebooted once.

Assigning Floating IP

You can assign floating IP using the following CLI command:

```
++Truncated Output++
+-----+
[admin:10-146-43-2]: virtualservice> vip index 1
[admin:10-146-43-2]: virtualservice:vip> auto_allocate_floating_ip
Overwriting the previously entered value for auto_allocate_floating_ip
```

```
[admin:10-146-43-2]: virtualservice:vip> save
[admin:10-146-43-2]: virtualservice> save
++ Truncated Output++
```

Note Assigning Floating IP is currently supported via the CLI only.

Auto-allocate Floating IP

All GCP resources will be created in the SEs project. Only auto-allocation of floating IP is supported.

```
++Truncated Output++
[2018-07-16 06:48:08,717] INFO [gcp_nlb.allocate_vip:63] old_vip_info: vip {
  addr: "55.55.55.100"
  type: V4
}
  alloc_fip: false
  || new_vip_info: vip {
    addr: "55.55.55.100"
    type: V4
  }
  alloc_fip: true
```

You can also use `Swagger` APIs to auto-allocate a Floating IP via a `PUT` request using the following CLI command as follows:

```
"vip" : [ {
  "availability_zone" : "aeiou",
  "ip6_address" : "",
  "subnet" : "",
  "port_uuid" : "aeiou",
  "subnet_uuid" : "aeiou",
  "avi_allocated_vip" : true,
  "vip_id" : "aeiou",
  "ip_address" : "",
  "auto_allocate_floating_ip" : true, *
  "enabled" : true,
  "floating_subnet6_uuid" : "aeiou",
  "auto_allocate_ip" : true,
  "subnet6" : "",
  "floating_ip" : "",
  "floating_subnet_uuid" : "aeiou",
  "avi_allocated_fip" : true,
  "subnet6_uuid" : "aeiou",
  "floating_ip6" : "",
  "ipam_network_subnet" : "",
  "network_ref" : "aeiou",
  "discovered_networks" : [ "" ]
```

NSX Advanced Load Balancer DNS

This section explains the configuration of the native DNS hosting solution.

The DNS solution is available for the following clouds:

- Containers (Docker UCP, Rancher)
- Linux server
- VMware
- No Access
- OpenStack
- Amazon Web Services
- Google Cloud Platform

Configuring DNS

You can configure DNS as follows:

- 1 Navigate to **Templates > IPAM/DNS Profile** and create a DNS profile.
- 2 Add one or more DNS Service Domain names. NSX Advanced Load Balancer will be the authoritative DNS server for these domains.
- 3 Configure a TTL value for all records for a particular domain, or leave the **Default Record TTL** field blank to accept the default TTL of 300 seconds.
- 4 Click **Save**.

New IPAM/DNS Profile: ns-dns

Name* ?
ns-dns

Type ?
Avi Vantage DNS

Vantage DNS Configuration

Default Record TTL ?
300 Sec

Domain Name* ?
testdomain.avi

Override TTL ?
300

Add DNS Service Domain

Save

Using NSX Advanced Load Balancer DNS

After configuring a DNS profile with a set of domains for which NSX Advanced Load Balancer DNS will be serving records, configure a DNS virtual service. The advantages of using a DNS virtual service are High Availability (HA) and Interoperability with other DNS providers.

NSX Advanced Load Balancer DNS Virtual Service

Create a DNS Pool with DNS server members. This will make NSX Advanced Load Balancer DNS handle all DNS requests for which it does not host domains natively. For instance, if the DNS profile contains authoritative domain 'test.avi,' then this DNS VS will host all the records for this domain and additionally with a pool configured as below, it can relay the requests to DNS server members for non-authoritative domains like 'google.com'.

New Pool:dns-vs-pool

Step 1: Settings

Step 2: Servers

Step 3: Advanced

Step 4: Review

Add Servers

Select Servers

☒ IP Address, Range, or DNS Name
 ☐ IP Group

Server IP Address

sub.corp.com, 1.2.3.4, 1.2.3.4 - 1.2.3.10, or 1.2.3.4:80

Add Server

Servers

Q

Displaying 1 item(s)

<input type="checkbox"/> ▾	Status	Server Name	IP Address	Port	Ratio	Network	Header Value	Rewrite Host H...
<input type="checkbox"/>	Enabled	10.10.0.100	10.10.0.100	53	1		Header Value	<input type="checkbox"/>

Cancel

Previous

Next

You can create DNS virtual service by providing a publicly-accessible virtual IP address and by selecting **System-DNS** in the **Application Profile** drop-down menu. You need to select the **Ignore network reachability constraints for the server pool** check box to ignore the network reachability.

New Virtual Service: dns-vs

Step 1: Settings Step 2: Policies Step 3: Analytics Step 4: Advanced

Name* ? dns-vs Enabled ? ☒

VIP Address

VIP Address* ? 10.160.160.100

Fully Qualified Domain Name ? dns-vs Select Domain

Service Port Switch to Advanced

Services ? 53 + Add Port

Profiles

Application Profile* ? System-DNS

TCP/UDP Profile* ? System-UDP-Per-Pkt

Pool

☒ Pool ☐ Pool Group

Pool ? dns-vs-pool

☒ Ignore network reachability constraints for the server pool ?

Cancel Next

If the corporate DNS server is on an external network (requires routing for SE data traffic to reach the DNS server), then add a static route to the external network as shown below (if a default route is not already configured):

- 1 Navigate to **Infrastructure > Cloud Resources > Routing > Create** and add a **Default-Gateway IP** address for the cluster. After this configuration and if the DNS virtual service was down, it should now be up.
- 2 Navigate to **Administration > Settings > DNS Service** and enable this DNS virtual service to start hosting application records. Multiple DNS virtual services can be configured for redundancy in the cluster.

Enable DNS service in your data center using one of the following two options:

- Add DNS VIPs ('10.160.160.100' as configured above) to the nameservers' list in `/etc/resolv.conf` on all nodes requiring service discovery. Create applications and verify resolution works for the application's FQDN by issuing `'dig app.testdomain.avi'` anywhere in the cluster.

- Add DNS VIP in the corporate DNS server as the nameserver for serving domain names configured in the DNS profile above. Any requests to `testdomain.avi` will be redirected to and serviced by the DNS service. Alternately, DNS server can be the main front-end DNS server and the corporation DNS servers can be configured as DNS pool members.

NSX Advanced Load Balancer DNS with AWS Cloud

You can choose NSX Advanced Load Balancer native DNS provider along with AWS Route 53 while creating an AWS cloud. The native DNS can be configured as explained in the above section (DNS Configuration) or can be created directly while creating the AWS cloud.

The following are the limitations and restrictions of using the DNS in AWS cloud:

- Both Route 53 and the native DNS cannot be configured/enabled at the same time on the AWS cloud.
- When the AWS cloud has a virtual service with FQDNs resolved by either DNS or Route 53.
 - Removal of the cloud is not allowed until the existing virtual service with FQDN is removed from the cloud.
 - Changing the DNS provider, i.e., either removal of existing or changing from existing DNS provider to Route 53 or vice versa is not allowed until the existing virtual service with FQDN is removed from the cloud.

Configuring IPAM and DNS Support

NSX Advanced Load Balancer can be configured to provide automatic IP address allocation for virtual services and to provide authoritative DNS resolution for their virtual IP addresses.

Starting with NSX Advanced Load Balancer version 20.1.3, GCP IPAM on GCP is not supported.

General Configuration Workflow

Initial configuration is common to both IPAM and DNS. This section lists the steps for configuring IPAM and DNS support. The configuration fields differ among the infrastructure types and the provider (NSX Advanced Load Balancer, Infoblox, AWS, GCP, and OpenStack).

- 1 Navigate to **Templates > Profiles**.
- 2 Click **IPAM/DNS Profile**.
- 3 Click **Create** and select the provider.
- 4 Specify the necessary details. Click **Save**. The profile appears in the list.
- 5 Navigate to **Infrastructure > Clouds**, and edit the cloud setting.
- 6 Select the **IPAM and DNS providers** from the drop-down menu. Either one or both need to be selected, based on the provider selected.
- 7 (Optional) For east-west virtual services in this cloud, select east-west IPAM and DNS providers from the drop-down menu. Either one or both can be selected.
- 8 Click **Save**.

Networking Considerations

NSX Advanced Load Balancer Service Engine data interfaces can be assigned to multiple VRFs (Virtual Routing and Forwarding Context). Virtual Routing Framework, or VRF, is a method of isolating traffic within a system. This is also referred to as a 'route domain' within the load balancer community.

NSX Advanced Load Balancer supports the assignment of SE data interfaces to multiple VRFs in the following clouds:

- No Access Cloud
- Linux server Cloud
- vCenter Cloud in provider mode

Note Multiple VRFs are only supported in Linux server clouds for SEs with DPDK enabled.

Types of Interface Supported

The VRF property for the following types of data interfaces can be modified, via the REST API, UI, or CLI.

- Physical interfaces
- Port-channel interfaces
- VLAN interfaces

The following types of data interfaces do not support modification of the VRF property. If you try to modify them, then the system will display an error message:

- Port-channel member interfaces
- Management interface

Dependency on In-band Management

Each deployed Service Engine has in-band management. When enabled, the management interface of the Service Engine (i.e., the interface used to communicate with the Controller cluster) is also used for data plane traffic.

If in-band management is enabled on an SE, that SE will not support multiple VRFs.

To enable multiple VRFs on an SE, it must be deployed with in-band management disabled. The caveat with disabling in-band management is that the management interface will not be used for data plane traffic. Hence, no virtual service will be placed on this interface, and this interface will not be used to communicate with back-end servers.

For more information on enabling or disabling in-band management, see [Configuring In-band Management for an NSX Advanced Load Balancer Service Engine](#).

Modifying SE Data Interface VRF using the UI

You can update SE physical port-channel and VLAN interface VRFs, if there are multiple VRFs configured in the tenant and cloud to which the SE belongs.

The screenshot shows a web form titled "Create VLAN Interface: eth1.100". It contains the following fields:

- Name:** A text input field containing "eth1.100".
- Parent Interface:** A dropdown menu showing "eth1".
- IP Address:** A text input field containing "10.10.100.0/24".
- VLAN:** A text input field containing "100".
- VRF:** A dropdown menu with options "test-vrf", "prod-vrf" (highlighted), and "global".

Modifying SE Data Interface VRF using the CLI

You can set VRF for physical and VLAN interfaces through CLI as follows:

```
[admin:10-10-24-89]: serviceengine>
[admin:10-10-24-89]: serviceengine> data_vnics if_name eth2
[admin:10-10-24-89]: serviceengine:data_vnics> vrf_ref prod-vrf
Overwriting the previously entered value for vrf_ref
[admin:10-10-24-89]: serviceengine:data_vnics> vlan_interfaces
New object being created
[admin:10-10-24-89]: serviceengine:data_vnics:vlan_interfaces> vlan_id 100
[admin:10-10-24-89]: serviceengine:data_vnics:vlan_interfaces> vrf_ref
global      management  prod-vrf    test-vrf
[admin:10-10-24-89]: serviceengine:data_vnics:vlan_interfaces> vrf_ref test-vrf
[admin:10-10-24-89]: serviceengine:data_vnics:vlan_interfaces> if_name eth1.100
[admin:10-10-24-89]: serviceengine:data_vnics:vlan_interfaces> sav
[admin:10-10-24-89]: serviceengine:data_vnics> sav
[admin:10-10-24-89]: serviceengine> sav
```

Ports and Protocols

This section explains the protocol ports used for management communication.

The various protocols and ports used are:

- Management communication
- Network services
- Cloud orchestrators

Ports for Management Communication

Ensure that the firewall allows traffic for the ports that the Controller and SEs use for management communication.

Traffic Source	Traffic Destination	Ports to Allow
NSX Advanced Load Balancer Controller	NSX Advanced Load Balancer Controller	TCP 22 (SSH) TCP 443 (HTTPS) TCP 8443 (HTTPS) TCP 5098 (SSH) (if Controller is a docker container, SSH is on port 5098)
	NSX Advanced Load Balancer SE	Not Required
NSX Advanced Load Balancer SE	NSX Advanced Load Balancer Controller	TCP 22 (SSH) TCP 8443 (HTTPS) UDP 123 (NTP) TCP 5098 (SSH) (if Controller is a docker container, SSH is on port 5098)

Note

- You do not have to open any firewall port from Controller to SE.
- The Controller IP is the source IP.
- The secure channel on port 22 (or 5098 in container environments) is used for communication between the NSX Advanced Load Balancer components for configuration sync, metrics and logs transfer, heartbeats, and other management processes.
- OpenStack mode does not support 5098 port on the container side.

Ports Used by Controller for Network Services

The Controller sends traffic to the following ports as part of network operation. The firewall must allow traffic from the Controller to these ports.

Traffic Source	Traffic Destination	Ports to Allow
External Network Services		TCP 22 (SSH) TCP 25 (SMTP) TCP 49 (TACACS+) UDP 53 (DNS) TCP 80 (HTTPS) (optional) UDP 123 (NTP) UDP 162 (SNMP traps) TCP or UDP 389 (LDAP) TCP 443 (HTTPS) UDP 514 (syslog) TCP or UDP 636 (LDAPS) TCP 5054 (CLI Server) (if using the optional CLI shell for remote management access) UDP 161 (SNMP agent listens to this port)

Protocols/Ports used by Cloud Orchestrators

Cloud Orchestrators	Protocols/ Ports Used
GCP	Port 443 is needed for the GCP cloud to connect to NSX Advanced Load Balancer
OpenStack	Some or all of the following ports may be required: <ul style="list-style-type: none"> ■ Keystone: TCP 5000, 35357 ■ Glance: TCP 9292 ■ Nova: TCP 8774 ■ Neutron: TCP 9696 ■ Heat (optional; used for autoscaling back-end members): TCP 8004
VMware vCenter	Controller-to-ESXi hosts: port 443
AWS	Port 443 for AWS cloud to connect to NSX Advanced Load Balancer
Azure	Port 443 for Azure cloud to connect to NSX Advanced Load Balancer

Service Engine Firewalls

The following protocols and ports are required for SE-SE management traffic:

Protocols	Ports
75	-
97	-
UDP	1550

To allow ingress traffic for SE to SE management traffic, see [Configuring Service Engine Ingress Rules](#).

To allow egress traffic for SE to SE management traffic, see [Configuring Controller Egress Rules](#).

Changing NSX Advanced Load Balancer Controller Cluster Configuration

This section describes how to change node membership or node IP address in an NSX Advanced Load Balancer Controller Cluster.

- 1 [Deploying an NSX Advanced Load Balancer Controller Cluster](#)
- 2 [Dismantling the Controller Cluster](#)
- 3 [Changing a Follower Node](#)
- 4 [Replacing the Leader Node](#)
- 5 [Replacing a Follower Node with a New Node](#)

Note If you change IP address of an NSX Advanced Load Balancer Controller node, you must run a script to update the configuration.

Deploying a Controller Cluster

This section shows the **web interface** pop-up and CLI commands for deploying a Controller cluster.

To deploy a Controller cluster, you must deploy a single Controller node (the leader), and then add the follower nodes to the leader. Ensure that after using the setup wizard to install NSX Advanced Load Balancer on the follower nodes, you do not make any other configuration changes on these nodes.

Web Interface

The following are the steps to deploy a cluster on the web interface of the Controller that will be the leader:

- 1 Navigate to **Administration > Controller > Nodes** and click **Edit**.
- 2 Specify Controller Cluster **Name** and **IP address**. For more information on Controller cluster IP address, see Controller Cluster IP address. This will be the shared management IP address of the cluster.
- 3 Specify the host IP addresses of each of the 3 nodes in the **Cluster Nodes** fields.
 - a Controller Node-1: Host IP address of the leader node.
 - b Controller Node-2: Host IP address of one of the follower nodes.
 - c Controller Node-3: Host IP address of one of the other follower nodes.
- 4 Click **Save**.

CLI

Login to the CLI (or CLI shell) and enter the commands shown in the following example:

Note Ensure that you specify the host IP addresses of your Controller nodes instead of the IP addresses shown in the example.

Configure the cluster with the Controller nodes at ['10.10.25.81', '10.10.25.82', '10.10.25.83'].

```
configure cluster
Updating an existing object. Currently, the object is:
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| uuid       | cluster-eb01bf05-7313-4a4f-91b6-21e46d3c237d |
| name       | cluster-0-1                               |
| nodes[1]   |                                             |
| name       | 10.10.25.81                               |
| ip         | 10.10.25.81                               |
| vm_ref     | EB01BF05-7313-4A4F-91B6-21E46D3C237D      |
| vm_mor     |                                             |
| vm_hostname | node1.controller.local                     |
| tenant_ref | admin                                       |
+-----+-----+

: cluster> nodes name 10.10.25.82 ip 10.10.25.82
New object being created
: cluster:nodes> save
: cluster> nodes name 10.10.25.83 ip 10.10.25.83
New object being created
: cluster:nodes> save
: cluster> save
```

If you add or remove nodes from the cluster, then you need to bring down this Controller and back it up with the new configuration.

```
Waiting for the cluster to be ready...
Controller is ready.
```

Dismantling the Controller Cluster

This section explains the steps to remove both followers from the Controller cluster.

Web Interface

- 1 Navigate to **Administration > Controller > Nodes**.
- 2 Click **Edit**.
- 3 Remove each of the follower IP addresses from the **Configuration** pop-up window.
- 4 Click **Save**.

CLI

Log in to the CLI (or CLI shell) and enter the following commands:

Note Ensure that you specify the host IP addresses of your Controller nodes instead of the IP addresses shown in the example.

```
configure cluster
Updating an existing object. Currently, the object is:
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| uuid       | cluster-eb01bf05-7313-4a4f-91b6-21e46d3c237d |
| name       | cluster-0-1                               |
| nodes[1]   |                                           |
| name       | node-1                                    |
| ip         | 10.10.25.81                               |
| vm_ref     | EB01BF05-7313-4A4F-91B6-21E46D3C237D      |
| vm_mor     |                                           |
| vm_hostname | node1.controller.local                   |
| nodes[2]   |                                           |
| name       | node-2                                    |
| ip         | 10.10.25.82                               |
| vm_ref     | EC123A05-7313-4A4F-91B6-21E46D3D46AF      |
| vm_mor     |                                           |
| vm_hostname | node2.controller.local                   |
| nodes[3]   |                                           |
| name       | 10.10.25.83                               |
| ip         | 10.10.25.83                               |
| vm_ref     | EA12C05-7313-4A4F-91B6-21E46D3E256EA      |
| vm_mor     |                                           |
| vm_hostname | node3.controller.local                   |
| tenant_ref | admin                                     |
+-----+-----+

: cluster> no nodes name node-2
Removed nodes with name=node-2
: cluster:nodes> save
: cluster> no nodes name node-3
Removed nodes with name=node-3
: cluster:nodes> save
: cluster> save
```

- If you add or remove nodes from the cluster, you need to restart the Controller and ensure the changes have been applied.
- The cluster is dismantled and recovered into a single node.
- Each follower is re-imaged into its default state with no configuration and no access to the leader.

- The leader holds the configuration. The SEs continue to connect to the leader.

```
Waiting for the cluster to be ready... Controller is ready.
```

Note During the transition from a cluster to a single node, the `REST` API will be unavailable for around 2 minutes.

Changing a Follower Node

This section explains how to remove one of the follower nodes and add another.

If the node is removed and replaced with a different node (different VM, container, or bare metal server), then see [Replacing a Follower Node with a New Node](#) section. The cluster has to be dismantled, and then recreated using the new node.

Web Interface

The following are the steps to change a follower mode in the web interface:

- 1 Navigate to **Administration > Controller > Nodes**.
- 2 Click **Edit**.
- 3 Edit the IP address for the follower node to be changed.
- 4 Click **Save**.

CLI

Login to the CLI (or CLI shell) and enter the commands shown in the following example:

Note Ensure that you enter the host IP addresses of your Controller nodes instead of the IP addresses shown in the example.

```
configure cluster
Updating an existing object. Currently, the object is:
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| uuid       | cluster-eb01bf05-7313-4a4f-91b6-21e46d3c237d |
| name       | cluster-0-1                             |
| nodes[1]   |                                           |
| name       | node-1                                   |
| ip         | 10.10.25.81                             |
| vm_ref     | EB01BF05-7313-4A4F-91B6-21E46D3C237D      |
| vm_mor     |                                           |
| vm_hostname | node1.controller.local                  |
| nodes[2]   |                                           |
| name       | node-2                                   |
| ip         | 10.10.25.82                             |
| vm_ref     | EC123A05-7313-4A4F-91B6-21E46D3D46AF      |
| vm_mor     |                                           |
| vm_hostname | node2.controller.local                  |
```

```

| nodes[3]      |      |
| name          | 10.10.25.83 |
| ip            | 10.10.25.83 |
| vm_ref        | EA12C05-7313-4A4F-91B6-21E46D3E256EA |
| vm_mor        |      |
| vm_hostname   | node3.controller.local |
| tenant_ref    | admin |
+-----+-----+
: cluster> no nodes name node-3
Removed nodes with name=node-3
: cluster:nodes> save
: cluster> nodes name node-4 ip 10.10.25.84
Removed nodes with name=node-4
: cluster:nodes> save
: cluster> save

```

Configuring the cluster with the Controller nodes at [10.10.25.81, 10.10.25.82, 10.10.25.84]. If you add or remove nodes from the cluster, you should bring down this Controller and then back up with the new configuration.

```

Waiting for the cluster to be ready...
Controller is ready.

```

After saving,

- The removed follower node is sent an API request to clear its own state.
- Since the old follower is not always expected to clear its own state, the leader will forcibly remove it if necessary.
- The new follower node is added.

Replacing the Leader Node

To replace the leader node, power it down to force one of the followers to take over leadership, then use the following steps to replace a follower.

- 1 Power down the leader node and leave it powered off for several minutes while one of the followers assumes leadership.
- 2 Use the steps for adding a new follower node.

Note The leader cannot be directly removed. Instead, it must be replaced with another leader.

Replacing a Follower Node with a New Node

These steps describe how to replace a follower node with a new node.

This procedure is applicable for replacing a follower node with a completely different node that is not currently in the cluster. For instance, these steps apply to replacing a follower node with a new VM or bare metal server.

The node cannot be replaced simply by inserting the node into the network and editing the cluster configuration information.

The following are the steps to replace a follower node:

- 1 Log in to the leader node, and use the steps to dismantle the cluster by removing both follower nodes.
- 2 On the new node, install NSX Advanced Load Balancer. This involves spinning up a new virtual machine (VM) or container from the Controller package (OVA, QCOW or Docker image) on the new node.
- 3 Follow the steps mentioned in Deploy a new cluster section.

NSX Advanced Load Balancer Controller Sizing

This section explains the system capacity of the Controller.

Specifying System Capacity of Controller

During deployment of a Controller, you can specify its system capacity based on allocations of its system resources like CPU, memory (RAM) and disk. The amount of these resources allocated have a direct impact on its performance.

The following table lists recommended allocations for each type of deployment:

Deployment Type	Node Count	Recommended Allocations - CPU	Recommended Allocations -Memory	Recommended Allocations - Disk
Demo / Customer Eval	1	8	24 GB	128 GB
Production	3	Refer to 'CPU/Memory Allocation' and 'Drive Allocation' section		

In demonstration and deployments, a single Controller is adequate and is used for all control-plane activities and workflows, as well as for analytics.

In a production deployment, a three-node cluster is recommended. In a three-node Controller cluster, one Controller is the leader that is used for control-plane activities and workflows while the other two are followers. The follower nodes are used for analytics. They also provide backup in case the leader fails.

The following sections provide specific allocation recommendations that are designed to fit most use cases but might not fit every conceivable deployment scenario.

Note The `avicontroller.service` file is not updated automatically whenever CPU and memory allocation on the host is increased. Manual update of the `avicontroller.service` file is required whenever these values are changed.

The parameter for changing CPU is `-cpu-quota`.

Allocating CPU/Memory

NSX Advanced Load Balancer uses the allocations of CPU and memory as follows:

CPU/Memory Allocation	8 CPUs / 24 GB	16 CPUs / 32 GB	24 CPUs / 48 GB
Base processes	15 GB	20 GB	24 GB
Log analytics	9 GB	13 GB	24 GB
Virtual Service Scale	0-200	200-500	500-5000
SE Scale	0-100	100-200	200-400

The base processes of the Controller includes dynamic processes, metrics collection and processing. The allocations shown here are based on assumptions of no more than ten percent disk swapping and twenty-five percent disk margin.

Allocating Disk Capacity

The amount of disk capacity allocated to the Controller is calculated based on the following parameters:

- The amount of disk capacity available on the Controller.
- The number of virtual services to support.

Note The default Controller OVA template should be increased to 128 GB. The Controllers in the same cluster should all have the same/similar disk capacity. Allocations of significantly different sizes should not be permitted for a prolonged time.

The following tables show recommended allocations based on each approach.

Allocating Disk Based on Available Disk Capacity

The disk space allocated to a Controller that is not used for base processes or analytics is used as follows:

- Logs: 70 percent of the disk that is not used for base processes or analytics.
- Metrics: the other 30 percent that is not used for base processes or analytics.

Disk Allocation based on Disk Space	128 GB	256 GB	512 GB	1 TB
Log analytics (70%)	56 GB	144 GB	328 GB	672 GB
Metrics (30%)	24 GB	64 GB	128 GB	288 GB
Base Processes	48 GB	48 GB	56 GB	64 GB

Disk drive quality impacts analytics performance and size:

- Traffic logs are deleted as the disk drive fills up.

- Metrics tables are deleted based on the archival scheme:
 - Realtime: deleted after 1 hour
 - 5-minute intervals: deleted after 1 day
 - 1-hour intervals: deleted after 1 week
 - 1-day intervals: deleted after 1 year

If the drive fills up, then current metrics tables are deleted to make room for new data.

Allocating Disk Based on Number of Virtual Services

Disk allocation based on VS count	Log analytics without full logs	Log analytics with full logs	Metrics	Base processes	Total (without full logs)
100 VS	16 GB	128 GB	16 GB	48 GB	80 GB
1,000 VS(100k transactions / year)	128 GB	1 TB	32 GB	56 GB	216 GB
5,000 VS	512 GB	Not Supported	160 GB	64 GB	736 GB

Assumptions and Sizing Data

The size recommendations shown in the table are based on the following operational assumptions:

- DDoS attacks are less than 1 percent of the traffic.
- Significant logs are no more than 10 percent of total logs. (This means 90 percent of the transactions are good and result only in non-significant logs.)
- Log analytics require about 10 KB disk space per log entry, i.e., 10 GB of disk space for 1 million log entries.
- Metrics and other analytics processing require about 32 MB per virtual service. Client insights require additional drive capacity.

Note A transaction is a single TCP or UDP connection (layer 4) or a single request-response exchange (layer 7). A traffic volume of 100,000 transactions per year is probably low for an e-commerce site but applies to most other types of applications.

Data Plane Development Kit (DPDK)

This section describes about DPDK libraries to provide high-packet performance.

DPDK supports the following clouds:

- Linux Server Cloud

- AWS
- OpenStack
- VMware
- GCP
- NSX-T

NSX Advanced Load Balancer Linux Server Cloud

You can configure a Linux server cloud on the NSX Advanced Load Balancer Controller and thereafter add Linux hosts as Service Engines (SEs) to it.

Running NSX Advanced Load Balancer directly on Linux servers leverages the raw horsepower of the underlying hardware without the overhead added by a virtualization layer. For instance, running NSX Advanced Load Balancer directly on Linux servers that support Data Plane Development Kit (DPDK) allows the feature's optimized packet processing to be leveraged for virtual service traffic. While adding a Linux host that is equipped with an OS and NIC on which DPDK is supported by NSX Advanced Load Balancer, ensure that DPDK option is enabled.

Post-installation Configuration

This section describes the post-installation process.

Configuring an Infoblox DNS Profile on the Controller

- Navigate to **Templates > IPAM/DNS Profiles** and click **Create**. Name the profile.
- Select **Infoblox DNS** option from the **Type** drop-down menu.
- Specify the IP address, DNS view and user credentials in **Infoblox Profile Configuration** section. The **Infoblox DNS Profile** editor behaves in a similar fashion, except that you can choose usable domains, as opposed to subnetworks.
- Specify the following details in **Settings** section:
 - WAPI Version — The WAPI version is independent of the version of the Infoblox appliance's operating system, known as NIOS. To determine the API version being used by Infoblox, access the following URI on the **Infoblox Grid Master**: <https://wapidoc/>.
 - Usable Domain — Select all or a subset of the domains configured in Infoblox to be used for DNS purposes from the drop-down menu. If none is specified, all domains are available during virtual service creation.
- After specifying the necessary details, click **Save**.

Configuring an Infoblox IPAM Profile on NSX Advanced Load Balancer Controller

- Navigate to **Templates > IPAM/DNS Profiles** and click **Create** button. Name the profile. Select **Infoblox IPAM** from the **Type** drop-down menu.
- Specify the IP address and user credentials in **Infoblox Profile Configuration** section. Also, specify network view as configured in Infoblox (the default network view is named 'default').
- Specify the following details in **Settings** section:
 - **WAPI Version** — The WAPI version is independent of the version of the Infoblox appliance's operating system, known as NIOS. To determine the API version being used by Infoblox, access the following URI on the **Infoblox Grid Master**: <https://wapidoc/>.
 - **Usable Subnet** — Select the usable subnet from the drop-down menu to pick all or a subset of the networks configured in Infoblox to be used for IPAM purposes. If none is specified, all networks are available during virtual service creation.
 - You can add IPv4 and IPv6 subnet details by clicking **Add Usable Subnet** option.
 - You can select either a IPv4, IPv6 or both for each row. If both IPv4 and IPv6 subnets are populated on a given row, they are paired up for VIP allocation. For instance, if a VIP needs both v4 and v6, then you need to specify both IPv4 and IPv6 details.

Note Both IPv4 and IPv6 should be a part of the same underlying port-group/VLAN for virtual service traffic to not fail.

If you do not specify any value, then all networks will be available during virtual service creation.

You can send extensible attributes in the data while requesting an IP from Infoblox in the **Extensible Attributes** section. You can input these attributes as key-value pairs in the Infoblox profile.

After specifying the necessary details, click **Save**.

Credential Verification and Infoblox Network/Domain Selection

When configuring/editing Infoblox DNS or IPAM profiles, the NSX Advanced Load Balancer first verifies credentials.

Note This verification is only applied to Infoblox and Azure profiles.

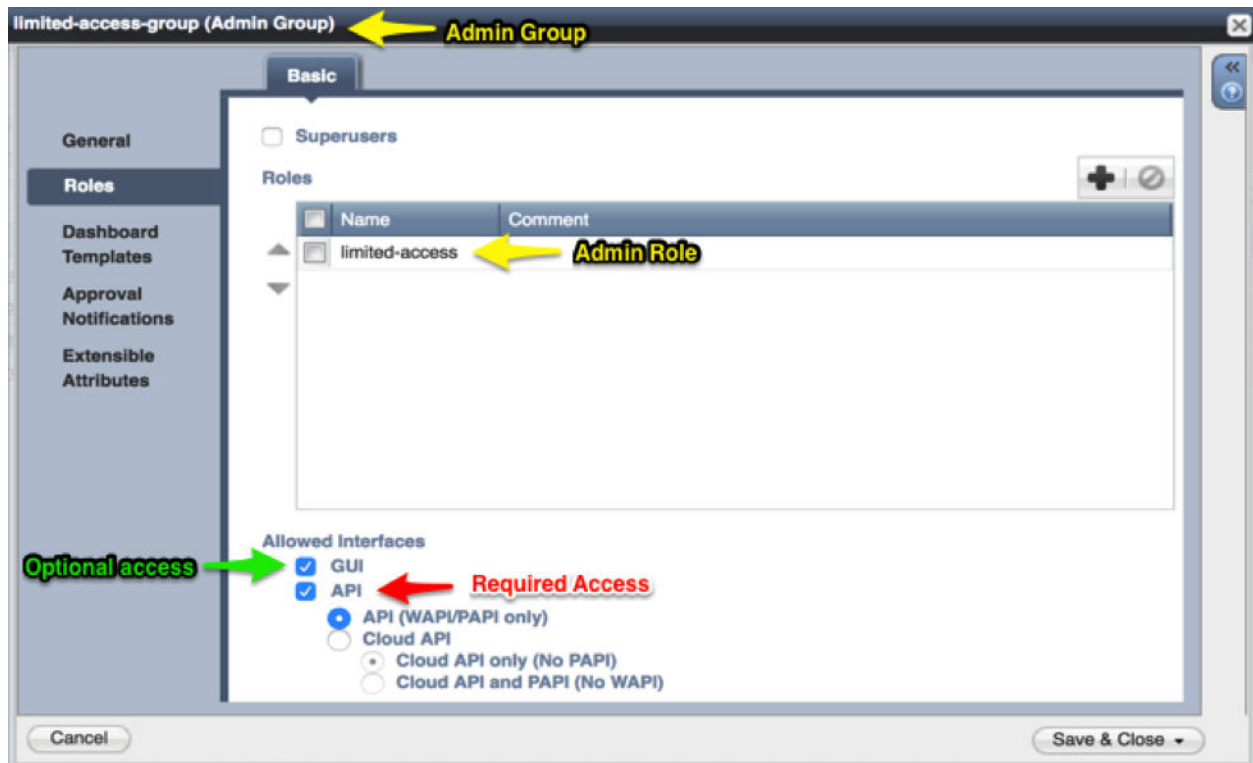
Configuring an Infoblox IPAM Profile

If you have entered invalid credentials and clicked **Connect** button, the system displays an error message. If the credentials are correct, the new screen is displayed, confirming that the entered credentials are correct. The **Connect** button will change to a **Change Credentials** button, enabling you to change the credentials.

User Permissions Required in Infoblox

For the NSX Advanced Load Balancer Controller to properly select the next available IP address from available subnets and register host objects in the correct DNS zones, the user-defined in the Infoblox IPAM/DNS profile must have Read/Write WAPI access to Infoblox. In real production environments, it is recommended to create a new user account that will have the minimum required access to Infoblox.

Granular access control can be defined using object-level permissions within the Infoblox permissions model for the specific DNS zones and IPAM networks that NSX Advanced Load Balancer will be modifying via the Infoblox WAPI. In addition, one can set the 'API Only' bit as an allowed interface for configuring Infoblox so that you cannot log into the admin UI but is instead restricted solely to API access. In the sample screenshot below, a new user group called 'limited-access-group' and a new role called 'limited-access' has been created. Object-level permissions are then applied to the 'limited-access' role and inherited by users that are added to the 'limited-access-group'.



Note Although API access is all that is required for NSX Advanced Load Balancer-to-Infoblox integration to function correctly, it is recommended to enable the NSX Advanced Load Balancer UI access while testing so that the results of the granular, object-level permissions can be visually verified. After the desired results have been achieved, you can safely disable UI access for the user defined in IPAM or DNS profiles.

Creating Virtual Services in a VRF

This task explains how to create virtual services in a VRF.

The following are the steps to create virtual services in a VRF:

Procedure

- 1 Navigate to **Applications > Dashboard**.
- 2 Click **Create Virtual Service**.
- 3 Select **Basic Setup** option.
- 4 Select the cloud name.
- 5 Click **Next**.
- 6 Select the VRF context from the list and click **Next**.
- 7 Specify a name for the **virtual service**, **virtual IP address (VIP)** and other properties of the virtual service.
- 8 Click **Save**.

Results

You can create a virtual service in a VRF from the admin tenant or another tenant.

Creating VRF Contexts

This task explains how to create VRF contexts.

The following are the steps to create VRF contexts:

Procedure

- 1 Navigate to **Infrastructure > Cloud Resources > Routing**.
- 2 Select the cloud from the **Cloud Name** drop-down menu. The cloud name, by default, will display as **Default-Cloud** if the VMware vCenter cloud is the only one configured.
- 3 In the **VRF Context** tab, click **Create**.
- 4 Specify the name of the VRF context.
- 5 Click **Save**.

Installing NSX Advanced Load Balancer in Google Cloud Platform

3

NSX Advanced Load Balancer supports integration with Google Cloud Platform (GCP). It offers elastic application services that extend beyond load balancing to deliver real-time application and security insights, simplify troubleshooting, auto-scale predictively, and enable developer self-service and automation, for applications hosted on Google Cloud Platform.

Note

- Starting with NSX Advanced Load Balancer version 20.1.3, Linux Server Cloud and GCP IPAM on GCP are not supported.
-

Features

NSX Advanced Load Balancer for GCP provides the following functionalities:

- In NSX Advanced Load Balancer, full automation on the GCP cloud is available. You can create a virtual service and the SEs get automatically spun up in GCP.
- A role is a group of permissions that can be assigned to members. You can set up custom roles in GCP projects. These roles will be assigned to the service account for NSX Advanced Load Balancer to create resources in GCP.
- Google Cloud Platform (GCP) firewall rules let you allow or deny traffic to and from your virtual machine (VM) instances based on a configuration you specify. By creating a firewall rule, you specify a Virtual Private Cloud (VPC) network and a set of components that define what the rule does.
- NSX Advanced Load Balancer run with a single Controller (single-node deployment) or with a three-node Controller Cluster. In a deployment that uses a single Controller, the Controller performs the administrative functions as well as all analytics, data gathering, and processing.
- The Customer Managed Encryption Key (CMEK) supports encryption of SE disks for the GCP cloud.
- In NSX Advanced Load Balancer, VIP as GCP Internal Load Balancer (ILB) is supported. With this, the VIP reachability is through ILB, where VIP will be allocated from a GCP subnet and the VIP will be frontend IP of the ILB. The ILB backend will have all the SEs on which the virtual service is placed.
- It allows GCP cloud network configuration.

This chapter includes the following topics:

- [Overview](#)
- [Networking Considerations](#)
- [NSX Advanced Load Balancer Controller Installation](#)
- [Configuring the NSX Advanced Load Balancer Controller Cloud Connector](#)

Overview

NSX Advanced Load Balancer serves as an application delivery controller for application workloads running on Google Cloud Platform (GCP). The purpose of this document is to describe the process of provisioning and configuring this solution.

About NSX Advanced Load Balancer

The Platform provides enterprise-grade distributed ADC solutions for on-premises as well as public cloud infrastructure. It also offers built-in analytics to diagnose and improve the end-user application experience, while making operationalizing easier for network administrators.

NSX Advanced Load Balancer is a complete software solution that runs on commodity x86 servers or as a virtual machine and is entirely accessible through REST API calls.

Note NSX Advanced Load Balancer supports cloud operations with the non-admin tenant.

Networking Considerations

Firewall Target Tags are applied on the SE VMs to allow the ingress and egress traffic to the Controller, other SEs and virtual services. The firewall tags need to be created in the VPC in the network project.

Configuring Firewall Rules in GCP

Google Cloud Platform (GCP) firewall rules let you allow or deny traffic to and from your virtual machine (VM) instances based on a configuration you specify.

By creating a firewall rule, you specify a Virtual Private Cloud (VPC) network and a set of components that define what the rule does. For more information refer to [GCP Firewall Rules](#).

Firewall rules need to be configured to allow ingress and egress traffic for the Controller, SE, and the application servers.

Note By default, egress is allowed in GCP for all protocols and ports but if egress is denied by some firewall rules, then the specific destination protocol and port have to be allowed.

Skip the egress rule configuration if egress traffic is allowed.

Configuring firewall rules allow the following communication:

- 1 Management Traffic
 - a The Controller - Service Engines
 - b Network services used by the Controller
 - c Service Engine - Service Engine
- 2 Data Traffic
 - a Virtual service traffic on Service Engines
 - b Service Engine - Application servers

Create the following firewall rules using the steps below:

Note Make a note of the Target tags that will be created below since the target tags will be applied on the Controller and Service Engine virtual machines.

Management Traffic

For the list of protocols and ports required for ingress and egress management traffic, refer [Ports for Management Communication](#)

Controller Firewall Rules

To configure a firewall rule to allow ingress traffic for the NSX Advanced Load Balancer Controller, refer to [Configuring Controller Ingress Rules](#)

To configure firewall rules to allow outgoing traffic from a Controller, refer to [Configuring Controller Egress Rules](#)

Data Traffic

To allow ingress for data traffic, refer to [Configuring Service Engine Ingress Rules for Virtual Service Ports](#)

To allow egress for data traffic, refer to [Configuring Service Engine Egress Rules for Back end Server Ports](#)

Configuring Controller Ingress Rules

This section discusses configuration of firewall rule for permitting ingress traffic.

To configure a firewall rule to allow ingress traffic for the Controller:

Procedure

- 1 From the **GCP console**, navigate to **VPC network**.
- 2 In the **Create a firewall rule** screen, select **Ingress** as the **Direction of traffic** and **Allow** as the **Action on match**.

- 3 Select **Specified target tags** in the **Targets** drop-down menu, to apply the firewall rules only to the selected instances within the virtual network.
- 4 Enter the **Target tags**.
- 5 Select the option **Specified protocol and ports** and enter the **tcp**, **udp** ports to which the firewall rules are applicable.

Targets ?

Specified target tags ▼

Target tags

controller ×

Source filter ?

IP ranges ▼

Source IP ranges ?

0.0.0.0/0 ×

Second source filter ?

None ▼

Protocols and ports ?

☐ Allow all

☒ Specified protocols and ports

☒ tcp : 22, 443, 8443, 5054, 5098

☒ udp : 123

☐ Other protocols
protocols, comma separated, e.g. ah, sctp

⌵ Disable rule

Create Cancel

- 6 Click the **Create** button.

Results

Firewall rule to allow ingress traffic for the Controller is now configured.

Configuring Controller Egress Rules

This section describes the steps required to configure Egress rules to allow traffic from the Controller, for network services and SE communication.

To configure firewall rules to allow outgoing traffic from a Controller:

Procedure

- 1 In the **Create a firewall rule** screen, select **Egress** as the **Direction of traffic** and **Allow** as the **Action on match**.
- 2 Enter the **Target tags**.
- 3 Select the option **Specified protocol and ports**.

- 4 Refer to the Protocol Ports article to identify the ports to which the Controller sends traffic as a part of the network operation and add the ports.

Targets ?

Specified target tags ▼

Target tags

controller ✕

Destination filter ?

IP ranges ▼

Destination IP ranges ?

0.0.0.0/0 ✕

Protocols and ports ?

☐ Allow all

☒ Specified protocols and ports

☒ tcp : 22,443,8443,5098

☐ udp : all

☐ Other protocols

protocols, comma separated, e.g. ah, sctp

⌵ Disable rule

Create Cancel

- 5 Click the **Create** button.

Configuring Service Engine Ingress Rules

This section discusses creation of rule required to allow Ingress traffic from one SE to another.

To allow ingress traffic for SE to SE management traffic:

Procedure

- 1 In the **Create a firewall rule** screen, select **Ingress** as the **Direction of traffic** and **Allow** as the **Action on match**.

- 2 Enter the **Target tags**.
- 3 Select the option **Specified protocol and ports**.

Targets ?

Specified target tags ▼

Target tags

service-engine-mgmt ✕

Source filter ?

IP ranges ▼

Source IP ranges ?

0.0.0.0/0 ✕

Second source filter ?

None ▼

Protocols and ports ?

☐ Allow all
☒ Specified protocols and ports

☐ tcp :

22

☒ udp :

1550

☒ Other protocols

75, 97

[⌵ Disable rule](#)

Create

Cancel

- 4 Click the **Create** button.

Configuring Service Engine Egress Rules

This rule is required to control egress traffic from SE to the Controller and SE to SE.

To allow egress traffic for SE to SE management traffic:

Procedure

- 1 In the **Create a firewall rule** screen, select **Ingress** as the **Direction of traffic** and **Allow** as the **Action on match**.
- 2 Enter the **Target tags**.
- 3 Select the option **Specified protocol and ports**.

Targets ?

Specified target tags ▼

Target tags

service-engine-mgmt ✕

Destination filter ?

IP ranges ▼

Destination IP ranges ?

0.0.0.0/0 ✕

Protocols and ports ?
☐ Allow all
☒ Specified protocols and ports

☒ tcp :

22, 8443, 5098

☒ udp :

1550, 123

☒ Other protocols

75, 97

⌵ Disable rule

Create

Cancel

- 4 Click the **Create** button.

Configuring Service Engine Ingress Rules for Virtual Service Ports

This rule is required to allow ingress for data traffic for all virtual service ports.

To allow ingress for data traffic (for ports 4000-5000):

Procedure

- 1 In the **Create a firewall rule** screen, select **Ingress** as the **Direction of traffic** and **Allow** as the **Action on match**.
- 2 Enter the **Target tags**.
- 3 Select the option **Specified protocol and ports** and enter a range.

Targets ?

Specified target tags ▼

Target tags

service-engine-data ✕

Source filter ?

IP ranges ▼

Source IP ranges ?

0.0.0.0/0 ✕

Second source filter ?

None ▼

Protocols and ports ?

☐ Allow all

☒ Specified protocols and ports

☒ tcp : 4000-5000

☐ udp : all

☐ Other protocols

protocols, comma separated, e.g. ah, sctp

⌵ Disable rule

Create Cancel

- 4 Click the **Create** button.

Configuring Service Engine Egress Rules for Back end Server Ports

This rule is required to allow sending traffic to the back end server ports.

To allow egress for data traffic:

Procedure

- 1 In the **Create a firewall rule** screen, select **Egress** as the **Direction of traffic** and **Allow** as the **Action on match**.
- 2 Enter the **Target tags**.
- 3 Select the option **Specified protocol and ports** and enter a value.

Targets ?

Specified target tags

Target tags

service-engine-data

Destination filter ?

IP ranges

Destination IP ranges ?

0.0.0.0/0

Protocols and ports ?

☐ Allow all

☒ Specified protocols and ports

☒ tcp : 80

☐ udp : all

☐ Other protocols

protocols, comma separated, e.g. ah, sctp

⌵ Disable rule

Create Cancel

- 4 Click the **Create** button.

Deployment Prerequisites

This section discusses the deployment prerequisites for installing the NSX Advanced Load Balancer in GCP.

To deploy the NSX Advanced Load Balancer in GCP, follow the below steps:

- 1 [Authentication](#)
- 2 [System Requirements](#)

Authentication

This task explains how to use a service account to authenticate and authorize the NSX Advanced Load Balancer.

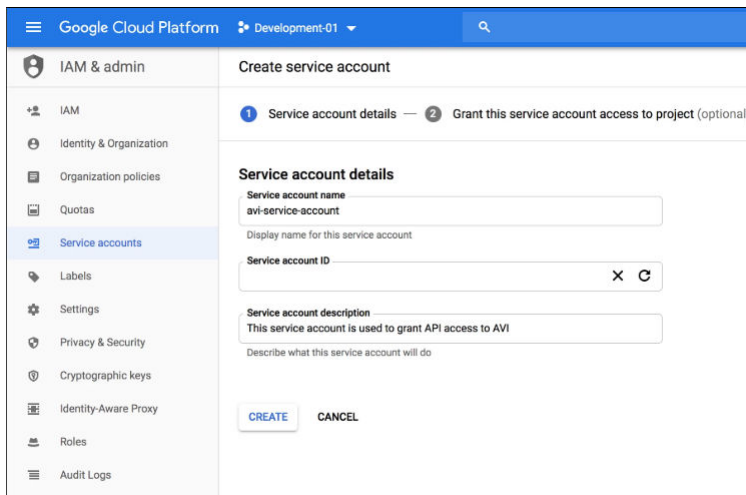
To create a service account:

Prerequisites

NSX Advanced Load Balancer needs a [GCP service account](#) to authenticate and authorize access to GCP APIs. The service account can be created in any GCP Project.

Procedure

- 1 From the [Google Cloud Console](#), select the required project.
- 2 Navigate to **IAM & admin > Service Accounts**.
- 3 From the **Create Service Accounts** screen, click **+Create Service Account** and enter the details.



The screenshot shows the Google Cloud Platform console interface for creating a service account. The left sidebar displays the 'IAM & admin' menu with 'Service accounts' selected. The main panel is titled 'Create service account' and shows two steps: '1 Service account details' and '2 Grant this service account access to project (optional)'. Under 'Service account details', there are three input fields: 'Service account name' (containing 'avi-service-account'), 'Display name for this service account' (empty), and 'Service account ID' (empty with a copy icon). Below these is a 'Service account description' field containing the text 'This service account is used to grant API access to AVI'. At the bottom are 'CREATE' and 'CANCEL' buttons.

- 4 The service account details can be provided to the Controller using either one of the following ways:
 - a Providing just the service account email ID.

Note

- In this case, the service account has to be attached to the Controller VM in GCP. This is done while creating the Controller.
 - This option works only when the Controller VM is running inside GCP.
-

- b The service account JSON key is added to the NSX Advanced Load Balancer cloud.

Note

- This option can be used irrespective of where the Controller is running (inside GCP or outside).
 - The service account JSON key has to be specified in NSX Advanced Load Balancer while creating GCP Cloud in the Controller.
-

To create the JSON key, see [JSON Key](#).

The private key is created and downloaded to your computer.

- 5 The same service account has to be added in all the GCP projects as mentioned in [GCP Project Selection](#).

Add this service account as a member with the required GCP role in the required project. For instance, add this service account as a member in the network project and grant it the NSX Advanced Load Balancer Role.

Refer to [Roles and Permissions \(GCP Full Access\)](#) to know how to create roles with the required permissions in projects as per the deployment topology.

JSON Key

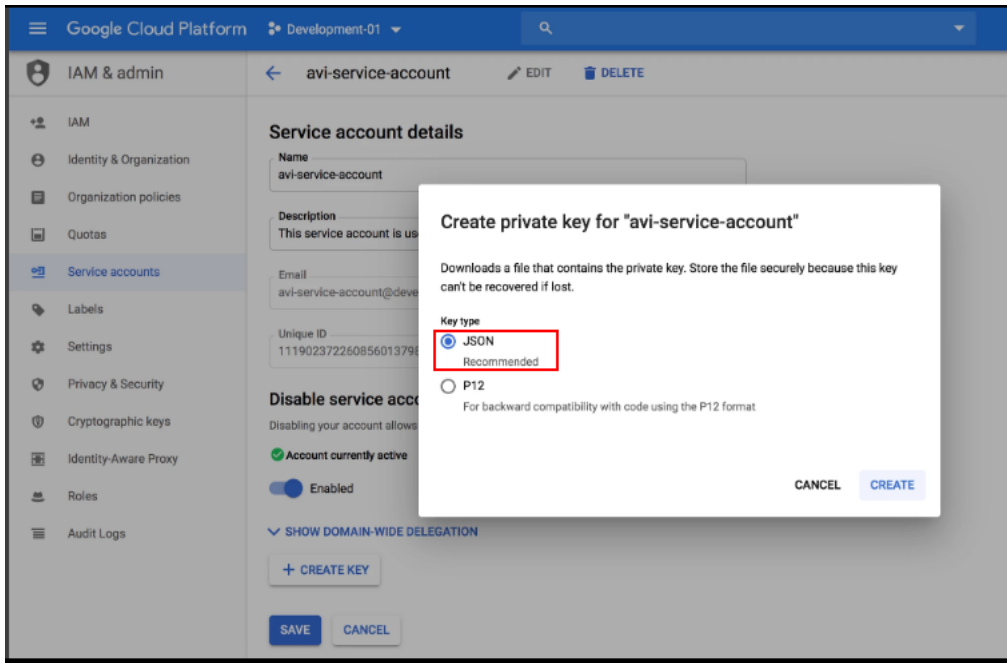
This task explains how to create the JSON key.

To create the JSON key:

Procedure

- 1 Navigate to **IAM & admin > Service Accounts**.
- 2 Click the service account that was created in Step 4a in the previous section.
- 3 Click the edit icon.
- 4 Click the **Create Key**.

- 5 Click **JSON** as the **Key type** to download the private key.



- 6 Click **Create**.

Roles and Permissions (GCP Full Access)

A role comprises of a group of permissions that can be assigned to members. This section explains how to set up custom roles in GCP projects. These roles are assigned to the service account for NSX Advanced Load Balancer to create resources in GCP.

Permissions

Definitions for various roles of the NSX Advanced Load Balancer are detailed in this section.

Network Project

The Role Definition (list of permissions included for a role) for the network project role, the service engine project role, and the storage project role are tabulated here:

Permissions	Role Definition Files
compute.networks.get compute.networks.list compute.networks.updatePolicy compute.regions.get compute.routes.create compute.routes.delete compute.routes.list compute.subnetworks.get compute.subnetworks.list compute.subnetworks.use	network_project_role.yaml

Service Engine Project

Permissions	Role Definition Files
compute.disks.create compute.forwardingRules.get compute.forwardingRules.create compute.forwardingRules.delete compute.forwardingRules.list compute.globalOperations.get compute.images.create compute.images.delete compute.images.get compute.images.list compute.images.setLabels compute.images.useReadOnly compute.instances.create compute.instances.delete compute.instances.get compute.instances.list compute.instances.setLabels compute.instances.setMetadata compute.instances.setTags compute.instances.use compute.machineTypes.get compute.regionOperations.get compute.regions.get compute.regions.list compute.targetPools.addInstance compute.targetPools.create compute.targetPools.delete compute.targetPools.get compute.targetPools.list compute.targetPools.removeInstance compute.targetPools.use compute.zoneOperations.get compute.zones.list	service_engine_project_role.yaml

GCP Instance Group Autoscaling Service Engine Project

Permissions	Role Definition Files
pubsub.subscriptions.consume pubsub.subscriptions.create pubsub.subscriptions.delete pubsub.subscriptions.get pubsub.subscriptions.list pubsub.topics.attachSubscription pubsub.topics.create pubsub.topics.delete pubsub.topics.get pubsub.topics.getIamPolicy pubsub.topics.list pubsub.topics.setIamPolicy	autoscaling_service_engine_project_role.yaml

ILB, BYOIP Service Engine Project

Permissions	Role Definition Files
compute.addresses.create compute.addresses.createInternal compute.addresses.delete compute.addresses.deleteInternal compute.addresses.get compute.addresses.list compute.addresses.setLabels compute.addresses.use compute.addresses.useInternal compute.healthChecks.create compute.healthChecks.delete compute.healthChecks.get compute.healthChecks.list compute.healthChecks.update compute.healthChecks.use compute.healthChecks.useReadOnly compute.instanceGroups.create compute.instanceGroups.delete compute.instanceGroups.get compute.instanceGroups.list compute.instanceGroups.update compute.instanceGroups.use compute.regionBackendServices.create compute.regionBackendServices.delete compute.regionBackendServices.get compute.regionBackendServices.list compute.regionBackendServices.setSecurityPolicy compute.regionBackendServices.update compute.regionBackendServices.use	ilb_service_engine_project_role.yaml

Storage Project

Permissions	Role Definition Files
storage.buckets.create storage.buckets.delete storage.objects.create storage.objects.delete storage.objects.list	storage_project_role.yaml

GCP Instance Group Autoscaling Server Project

Permissions	Role Definition Files
compute.instanceGroupManagers.list compute.instanceGroups.get compute.instanceGroups.list compute.instances.get compute.instances.list compute.projects.get logging.sinks.create logging.sinks.delete logging.sinks.get logging.sinks.list logging.sinks.update	server_project_role.yaml

Cluster IP

Permissions	Role Definition Files
compute.instances.get compute.instances.list compute.instances.updateNetworkInterface	cluster_vip_role.yaml

Service Account Project

Permissions	Role Definition Files
compute.instances.setServiceAccount iam.serviceAccountUser	Pre-created in GCP

Creating Roles in GCP

You can create custom roles either by using the gcloud command-line tool or the GCP console.

Creating Roles Using the GCloud Command Line Tool

This section discusses the usage of Command Line Tool for role creation.

To create roles using the gcloud command-line tool:

- Download the role definition YAML files.
- Run the following commands for each of the project.

If there is only one project where you have to create all the network, storage, and Service Engine objects, then create all the roles in the same project.

Commands for Service Engine Project Role

```
$ gcloud iam roles create avi.se --project se-project --file service_engine_project_role.yaml
Created role [avi.se].
description: Access to resources required for operations on Service Engines and Virtual
  Services
etag: B*****k=
includedPermissions:
- compute.addresses.create
- compute.addresses.delete
- compute.addresses.get
- compute.addresses.list
- compute.addresses.use
- compute.disks.create
- compute.forwardingRules.create
- compute.forwardingRules.delete
- compute.forwardingRules.list
- compute.globalOperations.get
- compute.images.create
- compute.images.delete
- compute.images.list
- compute.images.useReadOnly
- compute.instances.create
- compute.instances.delete
- compute.instances.get
- compute.instances.list
- compute.instances.setLabels
- compute.instances.setMetadata
```

```

- compute.instances.setTags
- compute.instances.use
- compute.instances.updateNetworkInterface
- compute.regionOperations.get
- compute.regions.get
- compute.regions.list
- compute.targetPools.addInstance
- compute.targetPools.create
- compute.targetPools.delete
- compute.targetPools.get
- compute.targetPools.list
- compute.targetPools.removeInstance
- compute.targetPools.use
- compute.zoneOperations.get
- compute.zones.list
name: projects/se-project/roles/avi.se
stage: ALPHA
title: AVI Service Engine Project Role

```

Commands for Network Project Role

```
$ gcloud iam roles create avi.network --project network-project --file
network_project_role.yaml
```

Note: permissions [compute.subnetworks.get, compute.subnetworks.list] are in 'TESTING' stage which means the functionality is not mature and they can go away in the future. This can break your workflows, so do not use them in production systems!

```
Are you sure you want to make this change? (Y/n)? y
```

```

Created role [avi.network].
description: Access to resources required for operations in Network Project
etag: B*****k4=
includedPermissions:
- compute.networks.get
- compute.networks.list
- compute.networks.updatePolicy
- compute.regions.get
- compute.routes.create
- compute.routes.delete
- compute.routes.list
- compute.subnetworks.get
- compute.subnetworks.list
- compute.subnetworks.use
name: projects/network-project/roles/avi.network
stage: ALPHA
title: AVI Network Project Role

```

Commands for Storage Project Role

```
$ gcloud iam roles create avi.storage --project storage-project --file
storage_project_role.yaml
```

```
Created role avi.storage.
```

```
description: Access to resources required for operations on GCS Buckets and Objects
etag: B*****g=
includedPermissions:

storage.buckets.create
storage.buckets.delete
storage.objects.create
storage.objects.delete
storage.objects.list
name: projects/storage-project/roles/avi.storage
stage: ALPHA
title: AVI Storage Project Role
```

Creating Roles Using the GCP Console

This section details the usage of GCP console for role creation.

To create the custom role:

Procedure

- 1 Navigate to **Roles** page.
- 2 Click **Create Role** in the **IAM & admin** page.
- 3 Specify a **Title**, **Description**, and **ID** for the role in the **Create Role** screen.
- 4 Click **Add Permissions** and include the required permissions. The **Create Role** screen for each role.
 - Creating Service Engine Project Role

IAM & admin

- IAM
- Identity & Organization
- Organization policies
- Quotas
- Service accounts
- Labels
- Settings
- Privacy & Security
- Cryptographic keys
- Identity-Aware Proxy
- Roles**
- Audit Logs

← Create Role

Custom roles let you group permissions and assign them to members of your project or organization. You can manually select permissions or import permissions from another role. [Learn more](#)

Title *

Custom AVI Service Engine Project Role

38 / 100

Description

Access to resources required for operations on Service Engines and Virtual Services

83 / 256

ID *

aviSe

Role launch stage

Alpha

+ ADD PERMISSIONS

34 assigned permissions

Filter table

Permission ↑	Status
<input checked="" type="checkbox"/> compute.addresses.create	Supported
<input checked="" type="checkbox"/> compute.addresses.delete	Supported
<input checked="" type="checkbox"/> compute.addresses.get	Supported
<input checked="" type="checkbox"/> compute.addresses.list	Supported
<input checked="" type="checkbox"/> compute.addresses.use	Supported
<input checked="" type="checkbox"/> compute.disks.create	Supported
<input checked="" type="checkbox"/> compute.forwardingRules.create	Supported
<input checked="" type="checkbox"/> compute.forwardingRules.delete	Supported
<input checked="" type="checkbox"/> compute.forwardingRules.list	Supported
<input checked="" type="checkbox"/> compute.globalOperations.get	Supported

1 – 10 of 34 < >

CREATE

CANCEL

■ Creating Network Project Role

IAM & admin

IAM
Identity & Organization
Organization policies
Quotas
Service accounts
Labels
Settings
Privacy & Security
Cryptographic keys
Identity-Aware Proxy
Roles
Audit Logs

Create Role

Custom roles let you group permissions and assign them to members of your project or organization. You can manually select permissions or import permissions from another role. [Learn more](#)

Title *

AVI Network Project Role

24 / 100

Description

Access to resources required for operations in Network Project

62 / 256

ID *

aviNetwork

Role launch stage

Alpha

+ ADD PERMISSIONS

10 assigned permissions

Filter table

Permission ↑	Status
<input checked="" type="checkbox"/> compute.networks.get	Supported
<input checked="" type="checkbox"/> compute.networks.list	Supported
<input checked="" type="checkbox"/> compute.networks.updatePolicy	Supported
<input checked="" type="checkbox"/> compute.regions.get	Supported
<input checked="" type="checkbox"/> compute.routes.create	Supported
<input checked="" type="checkbox"/> compute.routes.delete	Supported
<input checked="" type="checkbox"/> compute.routes.list	Supported
<input checked="" type="checkbox"/> compute.subnetworks.get	Testing ⓘ
<input checked="" type="checkbox"/> compute.subnetworks.list	Testing ⓘ
<input checked="" type="checkbox"/> compute.subnetworks.use	Supported

SHOW ADDED AND REMOVED PERMISSIONS

CREATE

CANCEL

■ Creating Storage Project Role

IAM & admin

- IAM
- Identity & Organization
- Organization policies
- Quotas
- Service accounts
- Labels
- Settings
- Privacy & Security
- Cryptographic keys
- Identity-Aware Proxy
- Roles**
- Audit Logs

Create Role

Custom roles let you group permissions and assign them to members of your project or organization. You can manually select permissions or import permissions from another role. [Learn more](#)

Title *
AVI Storage Project Role

Description
Access to resources required for operations on GCS Buckets and Objects

ID *
avi.storage

Role launch stage
Alpha

[+ ADD PERMISSIONS](#)

5 assigned permissions

Filter table

Permission	Status
storage.buckets.create	Supported
storage.buckets.delete	Supported
storage.objects.create	Supported
storage.objects.delete	Supported
storage.objects.list	Supported

[SHOW ADDED AND REMOVED PERMISSIONS](#)

CREATE **CANCEL**

5 Click **Save**.

Assigning Roles to Service Account

Created roles are assigned to the service account in the respective project.

If the network, storage, and Service Engine resources are to be created in the same project, assign all the roles to the service account in the same project.

If the network, storage, and Service Engine resources are to be created in different projects, assign resource specific roles to the service account in the resource project.

You can assign roles to the service account either using the GCP command-line tool or the GCP Console.

Assigning Roles using the GCloud Command-line Tool

The following are the commands used to assign roles to the service account using the GCP command-line tool.

Commands for Service Engine Project

```
$ gcloud projects add-iam-policy-binding se-project --member serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com --role projects/se-project/roles/avi.se
Updated IAM policy for project [se-project].
bindings:
- members:
  - serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com
    role: projects/se-project/roles/avi.se
etag: B*****2=
version: 1
```

Commands for Network Project

```
$ gcloud projects add-iam-policy-binding network-project --member serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com --role projects/network-project/roles/avi.network
Updated IAM policy for project [network-project].
bindings:
- members:
  - serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com
    role: projects/network-project/roles/avi.network
etag: B*****Q=
version: 1
```

Commands for Storage Project

```
$ gcloud projects add-iam-policy-binding storage-project --member serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com --role projects/storage-project/roles/avi.storage
Updated IAM policy for project [storage-project].
bindings:
- members:
  - serviceAccount:avi-service-account@any-project.iam.gserviceaccount.com
    role: projects/storage-project/roles/avi.storage
etag: B*****=
version: 1
```

Assigning Roles Using the GCP Console

This section details assignment of roles in the GCP console.

To assign roles using the GCP console:

Procedure

- 1 Navigate to the **IAM & admin** page.
- 2 Click **Add**.
- 3 Specify the service account email address in the field **New Members**.
- 4 Select the required role to assign to the service account.
- 5 Click **Save**.

Results

The **Add Members to** sub-screen for each service account with respective roles selected is displayed:

- Service Engine Project

Add members to "Development-01"

Add members, roles to "Development-01" project

Enter one or more members below. Then select a role for these members to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New members

a ✕ ?

Role

AVI Service Engine Projec... ▼

Access to resources required for operations on Service Engines and Virtual Services

[+ ADD ANOTHER ROLE](#)

SAVE

CANCEL

- Network Project

Add members to "Development-02"

Add members, roles to "Development-02" project

Enter one or more members below. Then select a role for these members to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New members

a

Role

AVI Network Project Role

Access to resources required for operations in Network Project

+ ADD ANOTHER ROLE

SAVE

CANCEL

- Storage Project

Add members to "Jenkins-Prod-01"

Add members, roles to "Jenkins-Prod-01" project

Enter one or more members below. Then select a role for these members to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New members

a

Role
AVI Storage Project Role

Access to resources required for operations on GCS Buckets and Objects

+ ADD ANOTHER ROLE

SAVE

CANCEL

System Requirements

The minimum configuration required for the NSX Advanced Load Balancer Controller VM is as below:

Memory	24 GB
vCPUs	8
Disk	128 GB

The recommended minimum GCP machine-type is n1-standard-8 (8 vCPUs, 30GB Memory).

Note The machine-type depends on the NSX Advanced Load Balancer Controller scale requirement.

For more information, see [NSX Advanced Load Balancer Controller Sizing](#).

For more information on creating a cluster, see [Controller Cluster IP in GCP](#).

Controller Cluster IP in GCP

NSX Advanced Load Balancer can run with a single Controller (single-node deployment) or a three-node Controller Cluster. In a deployment that uses a single Controller, the Controller performs administrative functions as well as analytics, data gathering, and processing.

Adding two nodes to create a three-node cluster provides node-level redundancy for the Controller and also maximizes the performance for CPU-intensive analytics functions. In this case, one node is the leader that performs the administrative functions. The other two nodes are the followers that perform data collection for analytics, in addition to being on standby as backup for the leader.

This article explains the cluster virtual IP (VIP) configuration for the Controller in the GCP environment. It applies to docker-based as well as GCP full-access -based controller deployments.

Prerequisites

- Ensure that the default service account associated with the Controller virtual machines have appropriate permissions to configure the Controller Cluster IP on its interfaces. Use the `cluster_vip_role.yaml` file to create a role with permissions required for configuring the Controller Cluster IP.
- Choose a free IP in the same subnet for the cluster IP.
- Set up firewall rules to allow access to all the Controller ports mentioned here using the cluster IP explicitly.

Configuring Controller Cluster IP

This document details the steps to configure the Controller Cluster IP.

To configure the Controller cluster IP:

Procedure

- 1 Navigate to **Administration > Controller > Edit**.
- 2 Enter the virtual IP address in the field **Controller Cluster IP**.
- 3 Under **Cluster Nodes**, enter the **Hostname/IP, Name, Password**, and the **Public IP** address or hostname of the Controller VM.

The **Edit Controller Configuration** screen appears.

Note In case of an existing two-node or three-node cluster configuration, to change the Controller cluster IP from the cluster IP configuration screen, delete the existing cluster IP and add a new one.

Results

The Controller Cluster IP gets programmed as a sub-interface on the leader node's primary NIC, and should be in the same subnet as the controller nodes. Whenever there is a failover, this IP is reprogrammed on the new leader's primary NIC.

Example

10.146.3.2 is the Controller cluster IP programmed on a leader in a three-node cluster.

```

admin@demo-gcp-fa-cluster-node-1:~$ ip a
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group default qlen 1000
link/ether 42:01:0a:92:03:1c brd ff:ff:ff:ff:ff:ff
inet 10.146.3.28/32 brd 10.146.3.28 scope global eth0
valid_lft forever preferred_lft forever
inet 10.146.3.2/32 scope global eth0:1
valid_lft forever preferred_lft forever
inet6 fe80::4001:aff:fe92:31c/64 scope link
valid_lft forever preferred_lft forever

```

NSX Advanced Load Balancer Controller Installation

This section lists all the steps required to set up the Controller VM in GCP.

Deployment Topology

This section lists down the various GCP projects to be considered before deploying the NSX Advanced Load Balancer.

GCP Project Selection

The first step to provision NSX Advanced Load Balancer on Google Cloud is to decide on the GCP project(s) where the Controller and SEs will be created.

GCP Projects are classified as:

1 Controller Project

- The Controller project will be used to create the Controller
- This project is required only if the Controller is in GCP

2 Service Engine Project

The service Engines exist in this project. The following resources are created in this project:

- The Service Engine image
- The Service Engine VMs
- The network load balancer, if the public Virtual IP (VIP) is configured

3 Network Project

The network project is where the GCP Virtual Private Cloud (VPC) exists and is used to

- Allocate IP for management and data NICs of the Service Engine virtual machines (VM)
- Create routes for VIP
- Create firewall rules

4 Storage Project

- This project is required to create Google Cloud Storage (GCS) Buckets and Objects.
- The Service Engine image will be uploaded to the buckets in this project.
- Once the image is created in GCP, the buckets will be deleted.

Note The listed projects can be in the same GCP project or in different projects depending on the deployment topology.

Creating the NSX Advanced Load Balancer Controller Image

This section explains the creation of NSX Advanced Load Balancer Controller image.

To create the Controller image:

Procedure

- 1 Download the Controller image for GCP from the [NSX Advanced Load Balancer Networks customer portal](#).
- 2 Follow the steps below to create a new bucket in the GCP storage project for uploading the SE images:

To re-use an existing bucket, skip the steps below and proceed to step 3.

Note This bucket will be deleted when the Controller image is successfully created in the Google Compute Engine.

- a From the **GCP Console**, navigate to **Storage > Browser**.
- b Click **Create Bucket**.
- c Specify the name of the bucket.

- d Click **Choose where to store your data**.
- e Under **Location Type**, select **Region**.
- f Select the **Location** where the Controller VM has to be created.
- g Click **Create**.

- 3 Upload the Controller image `gcp_controller.tar.gz` into the Google Cloud Storage (GCS) bucket created. This will become the source for the Controller Google Compute Engine image. This can be done using the GCP Console or the CLI.

To upload the image using GCP Console:

- a From the **GCP console**, navigate to **Storage > Browser**.
- b Select the bucket created in GCS.
- c Click **Upload files** and select the `gcp_controller.tar.gz` file to upload it to this bucket.

To upload the image using the GCP CLI:

- a Download and install the `gsutil` tool.
- b Upload the `gcp_controller.tar.gz` to the bucket using the commands shown below:

```
$ gsutil cp gcp_controller.tar.gz gs://avi-demo-bucket Copying file://
gcp_controller.tar.gz [Content-Type=application/x-tar].. - [1 files][ 2.7 GiB/ 2.7
GiB] 29.3 MiB/s Operation completed over 1 objects/2.7 GiB.
```

- c Verify the MD5 hash of the file uploaded in the bucket with the `gcp_controller.tar.gz.md5` file available in the [NSX Advanced Load Balancer Networks customer portal](#) as shown below:

```
$ cat gcp_controller.tar.gz.md5 5c4a4f35ee1a0bd97409fbfe6d90188 $ gsutil hash -mh
gs://avi-demo-bucket/gcp_controller.tar.gz Hashes [hex] for gcp_controller.tar.gz:
Hash (md5): <value>
```

- 4 Create the image in the Google Compute Engine using the image uploaded in the bucket.
 - a Create the image in any GCP project where the NSX Advanced Load Balancer Controller will be installed or in any other project that can share images with the Controller Project. This can be done using the GCP console or the gcloud tool.

To create the image using the GCP Console:

- 1 From the **GCP Console**, navigate to **Compute Engine**Images.
- 2 Click **Create Image**.
- 3 Enter the **Name**.
- 4 Select **Cloud Storage File** as the **Source**.
- 5 Select the `gcp_controller.tar.gz` file in the Bucket which was uploaded.

Figure 3-1.

The screenshot shows the 'Create an image' interface in the Google Cloud Platform console. The left sidebar lists various Compute Engine resources, with 'Images' selected. The main panel contains the following fields and options:

- Name:** avi-controller
- Source:** Cloud Storage file
- Cloud Storage file:** A red box highlights this section, which includes a checked checkbox and the file path `avi-demo-bucket/gcp_controller.tar.gz`. A 'Browse' button is also present.
- Location:** Regional (selected), us-central1 (Iowa) (default)
- Family (Optional):** (Empty field)
- Description (Optional):** Avi Controller Image
- Labels (Optional):** + Add label
- Encryption:** Data is encrypted automatically. Select an encryption key management solution.
 - ☒ Google-managed key (No configuration required)
 - ☐ Customer-managed key (Manage via Google Cloud Key Management Service)
 - ☐ Customer-supplied key (Manage outside of Google Cloud)

At the bottom, there is a note: 'You will be billed for this image. [Compute Engine pricing](#)'. Below this are 'Create' and 'Cancel' buttons.

- 6 Click **Create**.

To create the image using the gcloud tool:

```
$ gcloud compute images create avi-controller --project=controller-project
--description="Avi Controller Image" --source-uri=https://storage.googleapis.com/
avi-demo-bucket/gcp_controller.tar.gz Created [https://www.googleapis.com/compute/v1/
projects/controller-project/global/images/avi-controller]. NAME PROJECT FAMILY
DEPRECATED STATUS avi-controller controller-project READY
```

- b Delete the object and bucket that was created in GCS since they are no longer required.

Configuring the NSX Advanced Load Balancer Controller Cloud Connector

This article discusses how to configure the NSX Advanced Load Balancer cloud connector.

Creating Controller VM from GCP Console

This task explains how to create controller VM from GCP console.

To create controller VM from GCP console:

Procedure

- 1 From the **GCP Console**, navigate to **Compute Engine > VM instances**.
- 2 Click **Create Instance**.
- 3 Enter the **Name**.
- 4 Select the **Region** and **Zone** where the VM has to be created.
- 5 Select the **Machine type**. This depends on the scale required. A minimum of **n1-standard-8** is recommended.
- 6 Under **Boot disk**, click on **Change** to configure the boot disk details.
- 7 Click the **Custom Images** tab and select the image that was created in the section [Creating the NSX Advanced Load Balancer Controller Image](#).

- 8 Select **SSD persistent disk** as the **Boot disk type** with a minimum of 128 GB disk space.

Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk

OS images Application images **Custom images** Snapshots Existing disks

Show images from
Development-01

☒ **avi-controller**
Avi Controller Image
Created from Development-01 on Sep 9, 2019, 8:07:49 AM

Can't find what you're looking for? Explore hundreds of VM solutions in [Marketplace](#)

Boot disk type ? Size (GB) ?
SSD persistent disk 128

Select Cancel

- 9 Click **Select** to save the configuration.
- 10 Under **Identity and API Access**, select a **Service account** to be attached to the VM, using which the [Creating the NSX Advanced Load Balancer Controller Image](#) authenticates GCP for API access.

Note If you don't want to add the service account to the VM then service account's JSON key can be added to the NSX Advanced Load Balancer Cloud later, refer to [Authentication](#).

Identity and API access ?

Service account ?
avi-service-account

Access scopes ?
Use IAM roles with service accounts to control VM access [Learn more](#)

- 11 Add the GCP Firewall target tags created to allow traffic from the SE to the Controller and the clients accessing the Controller.
- 12 Click the **Networking** tab.
- 13 Click **Network Interfaces** and select the VPC network and subnet in which the Controller should be created.

Note The VPC can be in another Project and is shared with this Controller project.

14 Click **Create**.

Results

The Controller VM will be up in a few minutes.

Configuring NSX Advanced Load Balancer

Once the Controller is up and running, open the NSX Advanced Load Balancer from the browser.

Procedure

- 1 Configure the admin detail.
- 2 Configure DNS and NTP settings. This can be edited later if required.

- 3 Enter the email address to be used for alerts from the controller. This can be set up later.
- 4 Select **No Orchestrator**.

- 5 Select **No** for **Support multiple Tenants**. Multi-tenancy can be enabled later if required.

The setup is completed.

Configuring GCP Cloud in the NSX Advanced Load Balancer Controller

In this section, the steps to create NSX Advanced Load Balancer cloud of type Google Cloud Platform (GCP) are explained. It spawns the Service Engines in the configured GCP Project, zone, and VPC. Service Engines start load balancing the workloads in GCP.

To create the NSX Advanced Load Balancer Cloud of type GCP:

Procedure

- 1 Log in to the Controller by entering your **Username** and **Password**.
 - 2 From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Clouds**
 - 3 Click **Create** to add a new cloud.
 - 4 Under **Step 1: Select Cloud**, complete the following steps:
 - a Enter a **Name** for the cloud.
 - b Select **Google Cloud Platform** as the **Type**.
 - c Click **Next**.
 - 5 Under the tab **Step 2: General** in the **New Cloud** screen, enter information related to the GCP Service Engine project, service account, and IPAM configuration for allocation of VIPs.
- 6 Enter the **Service Engine Project ID**. This is the **GCP project ID** where the Service Engines will be created.
 - 7 Configure GCP API Access.

To create an IPAM provider which will be required to allocate VIP from the NSX Advanced Load Balancer internal network, refer to [Creating an IPAM Provider](#)

To configure GCP API access, refer to [Configuring GCP API Access](#)

The **New Cloud** screen is as shown below:

New Cloud: GCP-Cloud

Step 1: Select Cloud Step 2: General Step 3: Location/Network

☐ Pay-as-you-go ☐ Bring your own license

License Type: Cores License Tier: Enterprise 18

• IPAM/DNS •

IPAM Provider: avi-internal-ipam

DNS: ☒ None ☐ Other

• Google Cloud User Credentials •

Service Engine Project ID: service-engine-project-id

☐ Use Controller Virtual Machine Service Account

Google Cloud Credentials: avi-service-account

Cancel Previous Next

8 Click **Next**.

9 Under the tab **Step 3: Location/Network**, configure the region and zones where Service Engines must be deployed. Configure the Service Engine image and the network settings as shown below:

- a Select the GCP **Service Engine Region** where the Service Engines will be deployed.
- b Select the **Zones** in the selected region. The Service Engines will be distributed among the selected zones.

Note It is recommended to have more than one zone for Service Engine High Availability.

- c Enter the GCP project ID where the VPC is present, in the **VPC Project ID** field. By default, the Service Engine Project ID is used.
- d Select the VPC network in the network project.

- e Select the subnet (from which the management IP of Service Engines will be allocated) in the VPC network.

New Cloud: GCP-Cloud

Step 1: Select Cloud Step 2: General Step 3: Location/Network

• Region & Zones •

Service Engine Region * ⓘ
us-central1

Zones * ⓘ
us-central1-a ✕ us-central1-b ✕ us-central1-c ✕

• Network •

Network Mode * ⓘ
☒ Inband Management ⓘ

VPC Project ID ⓘ
network-project-id

VPC Network Name * ⓘ
dev-vnet-1

VPC Subnet Name * ⓘ
subnet-1

- f Firewall Target Tags will be applied on the Service Engine VMs to allow the ingress and egress traffic to the Controller, other Service Engines, and virtual services.

The firewall tags need to be created in the VPC in the network project. To know more, refer to [Configuring Firewall Rules in GCP](#).

- g Enter the **Cloud Storage Project ID** in which NSX Advanced Load Balancer will create the bucket to upload the SE image in GCP.

Note By default, the Service Engine Project ID is used.

- h Enter the **Cloud Storage Bucket Name**.

Note

- **Cloud Storage Bucket Name** is required only in cases where the service account does not have the permissions to create a bucket in the Google Storage Project.
- If the service account has permissions to create the bucket in the Google storage project as described in the [Authentication](#) section, the NSX Advanced Load Balancer creates the bucket while creating the SE image in GCP and deletes the bucket once the image is created.

The screenshot shows the 'New Cloud: GCP-Cloud' configuration window, specifically Step 3: Location/Network. The window has a dark header with the title and a close button. Below the header, there are three tabs: 'Step 1: Select Cloud', 'Step 2: General', and 'Step 3: Location/Network'. The 'Step 3: Location/Network' tab is active. The form contains several input fields and sections:

- VPC Project ID**: A text input field containing 'network-project-id'.
- VPC Network Name**: A dropdown menu with 'dev-vnet-1' selected.
- VPC Subnet Name**: A dropdown menu with 'subnet-1' selected.
- Firewall**: A section with a header '• Firewall •'. It contains two text input fields for 'Firewall Target Tags': 'avi-management' and 'virtual-service'. There are trash icons next to each field. Below these fields is a link '+ Add Firewall Target Tag'.
- Storage**: A section with a header '• Storage •'. It contains two text input fields: 'Cloud Storage Project ID' with 'storage-project-id' and 'Cloud Storage Bucket Name' with 'Cloud Bucket Name'.

At the bottom of the window, there are two buttons: 'Cancel' and 'Complete'. The 'Complete' button is highlighted in green.

10 Click Complete.

Results

The Controller validates the configuration and starts creating the Service Engine image in the SE project. After the image is successfully uploaded, the NSX Advanced Load Balancer cloud becomes ready for virtual service creation.

Creating an IPAM Provider

The creation of IPAM provider, that is required to allocate VIP from the NSX Advanced Load Balancer internal network, is discussed in this section.

To create an IPAM provider:

Procedure

- 1 Click the **IPAM Provider** field and click **Create IPAM/DNS Profile**.
- 2 Enter the **Name** and select NSX Advanced Load Balancer IPAM for the **Type** field.

The screenshot shows a configuration window titled "New IPAM/DNS Profile: avi-internal-ipam". It contains the following fields and options:

- Name:** A text input field containing "avi-internal-ipam".
- Type:** A dropdown menu currently showing "Avi Vantage IPAM".
- Allocate IP in VRF:** An unchecked checkbox.
- Avi Vantage IPAM Configuration:** A section header with a green link "+ Add Usable Network" below it.
- Save:** A green button at the bottom right.

- 3 Click **Save**.

Configuring GCP API Access

NSX Advanced Load Balancer needs a GCP service account to authenticate access to GCP APIs.

To configure GCP API access:

- If the Controller is running in GCP and a service account is attached to the Controller VM, select the option **Use Controller Virtual Machine Service Account**.
- If the Controller is outside GCP (in some other public/private cloud) or there is no GCP service account attached to the Controller VM, add a user with GCP service account JSON key in NSX Advanced Load Balancer.

Procedure

- 1 Click the **Google Cloud Credentials** field.
- 2 If the credential object is already created, select it or click **Create Credentials** to create a new one.

- Enter the **User** name and the service account JSON Key which was downloaded (shown in the Authentication section).

New GCP Credentials: avi-service-account

User*

avi-service-account

GCP Credentials

Import Service Account Key File Data* ? ☒ Enter Text ☐ Upload File

```
{
  "type": "service_account",
  "project_id": "service-account-project",
  "private_key_id": "c087daedfawekjshafkajhsdfasdfsdf",
  "private_key": "-----BEGIN PRIVATE KEY-----
  \nMIIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQCQEC/VUTuS6rHj\ning/NWI12cVlv6KGWi3H9kWNFNC4Y
  OfcdTP8OhJRe3VXqg4laBh0DtGV+wQmZVTkUdULYRmFfAja8eu6cFOY73wURLVfPaZNCDCcFOpw9Lq48p7fw7yYWy\inkUyY
  Ya+HS/XsleNDNraq9c=\n-----END PRIVATE KEY-----\n",
  "client_email": "avi-service-account@service-account-project.iam.gserviceaccount.com",
  "client_id": "1044567929374873775",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/avi-service-account%service-account-
  project.iam.gserviceaccount.com"
}
```

Import

- Click **Import**.

Configuring Service Engine Group Properties

The GCP machine type can be configured using SE group properties, from which the SE virtual machines are created in GCP. If the instance flavor in a SE group is not defined, the Service Engine virtual machines are created with the memory and vCPUs defined in the SE group.

To configure GCP machine type:

Procedure

- From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Service Engine Group**
- Select the **GCP Cloud**.
- By default, a service engine group with the name Default-Group is created for the cloud. Either edit this or create a new service engine group.

- 4 Select the GCP machine type in the **Instance Flavor** field.

- 5 Click **Save**.

High Availability Options

The following High Availability (HA) modes are available for the GCP cloud:

- Legacy HA Active/Standby mode
- Elastic HA N+M mode
- Elastic HA Active/Active mode

For more information about the modes, see the following section.

Service Engine Group

A SE Group is a collection of SEs with information on how the SEs should be sized, placed, and made highly available.

The options within an SE group may vary based on the type of cloud within which they exist and its settings, such as no access versus write access mode. SEs may only exist within one group. Each group acts as an isolation domain. SE resources within an SE group may be moved around to accommodate virtual services, but SE resources are never shared between SE groups.

Based on its nature, a change made to an SE group

- might be applied immediately,
- only applied to SEs created after the changes are made, or
- might require existing SEs to be disabled before the changes can take effect.

Multiple SE groups can exist within a cloud. A newly created virtual service is placed in the default SE group. This can be changed using the **VS > Advanced** page while creating a virtual service using the **Advanced** wizard. To move an existing virtual service from one SE group to another, the VS must first be disabled, moved, and then re-enabled. SE groups provide data plane isolation; therefore moving a VS from one SE group to another is disruptive to existing connections through the virtual service.

Note In some clouds, some of the settings described below are not available.

The following is the CLI used to decide the range of port numbers. These port numbers are used to open backend server connections.

```
configure serviceenginegroupproperties
```

```
configure serviceenginegroup Default-Group ephemeral_portrange_start 5000 configure
serviceenginegroup [name] ephemeral_portrange_start 4096 configure serviceenginegroup [name]
ephemeral_portrange_end 61440
```

Note Default port for start is 4096, default port for end is 61440.

SE Group Basic Settings Tab

For more information, see [SE Group Basic Settings Tab](#).

SE Group Advanced Tab

The screenshot shows the 'Advanced' tab for an SE Group configuration. The main title is 'Advanced HA & Placement'. Below this, there are four main sections:

- Buffer Service Engines**: A single input field with the value '1'.
- Scale per Virtual Service**: Two input fields, 'Minimum' with value '1' and 'Maximum' with value '4'.
- Security**: A dropdown menu for 'HSM Group' with the text 'Select HSM Group'.
- Log Collection and Streaming Settings**: Four input fields:
 - Significant Log Throttle**: '100' with unit 'Logs/Second'.
 - UDF Log Throttle**: '100' with unit 'Logs/Second'.
 - Non-significant Log Throttle**: '100' with unit 'Logs/Second'.
 - Number of Streaming Threads**: '1' with unit 'Threads'.

At the bottom of the form, there are 'Cancel' and 'Save' buttons.

The advanced tab in the **Service Engine group** popup supports the configuration of optional functionality for SE groups. This tab only exists for clouds configured with write access mode. The appearance of some fields is contingent upon selections made.

Service Engine Name Prefix

Enter the prefix to use when naming the SEs within the SE group. This name will be seen both within NSX Advanced Load Balancer, and as the name of the virtual machine within the virtualization orchestrator.

Delete Unused Service Engines After

Enter the number of minutes to wait before the Controller deletes an unused SE. Traffic patterns can change quickly, and a virtual service may therefore need to scale across additional SEs with little notice. Setting this field to a high value ensures that NSX Advanced Load Balancer keeps unused SEs around in the event of a sudden spike in traffic. A shorter value means the Controller may need to recreate a new SE to handle a burst of traffic, which may take a couple of minutes.

SE Group Basic Settings Tab

To access the Service Engine group editor:

- 1 Navigate to **Infrastructure > Service Engine Group**
- 2 Click the **pencil** icon to edit a pre-existing SE group, or click the **blue** button to create a new one.
- 3 Start your specification for the SE group by giving it a name (or accept the default name, which is Default-Group).

Real-Time Metrics

At the top right of the **Basic Settings** tab you can turn on real-time metrics, which causes SEs in the group to upload SE-related metrics to the Controller once every 5 seconds, as opposed to once per five minutes or longer. After clicking the box, select the duration in minutes for real-time updating to last. A value of 0 is interpreted to mean “forever.”

For more information on metrics, see metrics-upload intervals.

High Availability & Placement Settings

The high availability mode of the SE group controls the behavior of the SE group in the event of an SE failure. It also controls how the load is scaled across SEs. Selecting a particular HA mode will change the settings and options that are exposed in the UI. These modes span a spectrum, from the use of the fewest virtual machine resources on one end to providing the best high availability on the other.

Legacy HA Active/Standby Mode

This mode is primarily intended to mimic a legacy appliance load balancer for easy migration to NSX Advanced Load Balancer. Only two Service Engines might be created. For every active virtual service configured on one, a standby service is configured on the other to readily take over, in the event of a failure of the active SE. There is no Service Engine scale out in this HA mode.

Elastic HA N + M Mode

This default mode permits up to N active SEs to deliver virtual services, with the capacity equivalent of M SEs within the group ready to absorb SE failure(s).

Elastic HA Active/Active Mode

This HA mode distributes virtual services across a minimum of two SEs.

For additional considerations for SE high availability, including VS placement, see [Overview of NSX Advanced Load Balancer High Availability](#).

VS Placement across SEs

When the placement is compact (previously referred to as “Compactor”), NSX Advanced Load Balancer prefers to spin up and fill up the minimum number of SEs; it tries to place virtual services on SEs which are already running. When the placement is distributed, NSX Advanced Load Balancer maximizes VS performance by avoiding placements on existing SEs. Instead, it places virtual services on newly spun-up SEs, up to the maximum number of Service Engines. By default, placement is compact for elastic HA N+M mode and legacy HA active/standby mode. By default, placement is distributed for elastic HA active/active mode.

Virtual Services per Service Engine

This parameter establishes the maximum number of virtual services the Controller cluster can place on any one of the SEs in the group.

Per-app SE mode

Select this option to deploy dedicated load balancers per application, i.e., per virtual service. In this mode, each SE is limited to a maximum of 2 virtual services. vCPUs in per-app SEs count towards licensing at a 25% rate.

SE Self-Election

Checking this option enables SEs in the group to elect a primary SE amongst themselves in the absence of connectivity to a Controller. This ensures SE high availability in handling client traffic even in headless mode. For more information, refer to the Service Engine Self Election article.

Service Engine Capacity and Limit Settings

• Service Engine Capacity and Limit Settings •

Max Number of Service Engines ?

10 Maximum

• Memory Allocation •

☐ Host Geolocation Profile ?

Memory for Caching * ? 0 %

Available Memory for Connections and Buffers ? 100 %

Connections and Buffers Memory Distribution (slide the bar left or right) ?

Connections: 50% Buffers: 50%

Max Number of Service Engines

[Default = 10, range = 0-1000] Defines the maximum SEs that may be created within an SE group. This number, combined with the virtual services per SE setting, dictates the maximum number of virtual services that can be created within an SE group. If this limit is reached, it is possible new virtual services may not be able to be deployed and will show a gray, un-deployed status. This setting can be useful to prevent NSX Advanced Load Balancer from consuming too many virtual machines.

Memory per Service Engine

[Default = 2 GB, min = 1 GB] Enter the amount of RAM, in multiples of 1024 MB, to allocate to all new SEs. Changes to this field will only affect newly-created SEs. Allocating more memory to an SE will allow larger HTTP cache sizes, more concurrent TCP connections, better protection against certain DDoS attacks, and increased storage of un-indexed logs. This option is only applicable in write access mode deployments.

Memory Reserve

[Default is ON] Reserving memory ensures an SE will not have contention issues with over-provisioned host hardware. Reserving memory makes that memory unavailable for use by another virtual machine, even when the virtual machine that reserved those resources is powered down. NSX Advanced Load Balancer strongly recommends reserving memory, as memory contention may randomly overwrite part of the SE memory, destabilizing the system. This option is applicable only for deployments in write access mode. For deployments in read access mode deployments or no access mode, memory reservation for the SE VM must be configured on the virtualization orchestrator.

vCPU per Service Engine

[Default = 1, range = 1-64] Enter the number of virtual CPU cores to allocate to new SEs. Changes to this setting do not affect existing SEs. This option is only applicable in write access mode. Adding CPU capacity will help with computationally expensive tasks, such as SSL processing or HTTP compression.

CPU Reserve

[Default is OFF] Reserving CPU capacity with a virtualization orchestrator ensures an SE will not have issues with over-provisioned host hardware. Reserving CPU cores makes those cores unavailable for use by another virtual machine, even when the virtual machine that reserved those resources is powered down. This option is only applicable in write access mode deployments.

Disk per Service Engine

[min = 10 GB] Specify an integral number of GB of the disk to allocate to all new SEs. This option is only applicable in write access mode deployments. The value appearing in the window is either:

- 10 GB (the absolute minimum allowed), or
- a value auto-calculated by the UI as follows: 5 GB + 2 x memory-per-SE, or
- a number explicitly keyed in by the user (values less than 5 GB + 2 x memory-per-SE will be rejected)

Host Geo Profile

[Default is OFF] Enabling this provides extra configuration memory to support a large geo DB configuration.

Connection Memory Percentage

The percentage of memory reserved to maintain connection state. It comes at the expense of memory used for HTTP in-memory cache. Sliding the bar causes the percentage devoted to the connection state to range between its limits, 10% minimum and 90% maximum.

For information on hyper-threading modes, refer to [Hyper-Threading Modes](#)

Datapath Heartbeat and IPC Encap Configuration

In NSX Advanced Load Balancer version 20.1.3, the following datapath heartbeat and IPC encap config knobs are moved to the segroup:

- dp_hb_frequency
- dp_hb_timeout_count
- dp_aggressive_hb_frequency
- dp_aggressive_hb_timeout_count
- se_ip_encap_ipc
- se_l3_encap_ipc

The seproperties based APIs for these config knobs will only work for the NSX Advanced Load Balancer version prior to 20.1.3 and they will not take any effect from 20.1.3 onwards. Likewise, SE group based APIs for these config knobs will take effect starting from NSX Advanced Load Balancer version 20.1.3.

However, upgrade from pre-20.1.3 seproperties based configuration to 20.1.3 will automatically migrate the config to segroup as a part of the upgrade migration routine.

License

License Tier

Specifies the license tier to be used by new SE groups. By default, this field inherits the value from the system configuration.

License Type

If no license type is specified, NSX Advanced Load Balancer applies default license enforcement for the cloud type. The default mappings are maxed SEs for a container cloud, cores for OpenStack and VMware, and sockets for Linux.

Instance Flavor

Instance type is an AWS term. In a cloud deployment, this parameter identifies one of a set of AWS EC2 instance types. The flavor is the analogous OpenStack term. Other clouds (especially public clouds) may have their terminology for essentially the same thing.

Hyper-Threading Modes

In NSX Advanced Load Balancer, two knobs are introduced to control the use of hyper-threaded cores and the distribution (placement) of se_dps on the hyper-threaded CPUs.

These two knobs are part of the Service Engine group.

The following are the two knobs -

You can enable hyper-threading on the SE using:

```
use_hyperthreaded_cores -
True [default] | False

enable or disable se_dps to use hyper-threaded cores
```

You can control the placement of se_dps on the hyper-threaded CPUs using:

```
se_hyperthreaded_mode -
SE_CPU_HT_AUTO[default]
SE_CPU_HT_SPARSE_DISPATCHER_PRIORITY
SE_CPU_HT_SPARSE_PROXY_PRIORITY
SE_CPU_HT_PACKED_CORES
```

controls the distribution of se_dps on hyper-threads

Note The processor needs to support hyper-threading and use of hyper-threading should be enabled at BIOS to utilize these knobs.

use_hyperthreaded_cores — You can use this knob to enable or disable the use of hyper-threaded cores for se_dps. This knob can be configured using both CLI and the NSX Advanced Load Balancer UI.

The following are the CLI commands:

```
[admin:vpr-ctrl]: serviceenginegroup> use_hyperthreaded_cores
[admin:vpr-ctrl]: serviceenginegroup> se_hyperthreaded_mode SE_CPU_HT_AUTO
[admin:vpr-ctrl]: serviceenginegroup> save
```

se_hyperthreaded_mode — You can use this knob to influence the distribution of se_dp on the hyper-threaded CPUs when the number of datapath processes is less than the number of hyper-threaded CPUs online. The knob can be configured only using the CLI.

Note You should set use_hyperthreaded_cores to True for the mode configured using se_hyperthreaded_mode to take effect.

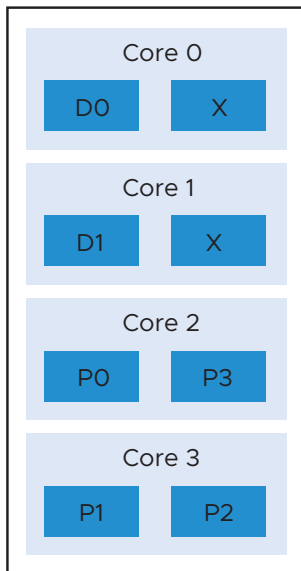
The following are the supported values for se_hyperthreaded_mode:

SE_CPU_HT_AUTO

This is the default mode. The SE automatically determines the best placement. This mode preserves the existing behavior in accordance with CPU hyper-threading topology. If the number of data path processes is less than the number of CPUs, this is equivalent to SE_CPU_HT_SPARSE_PROXY_PRIORITY mode.

SE_CPU_HT_SPARSE_DISPATCHER_PRIORITY

This mode prioritizes the dispatcher instances by attempting to place only one data-path process in the physical CPU. This mode exhausts the physical cores first and then hyper-threads in numerically descending order of CPU number.

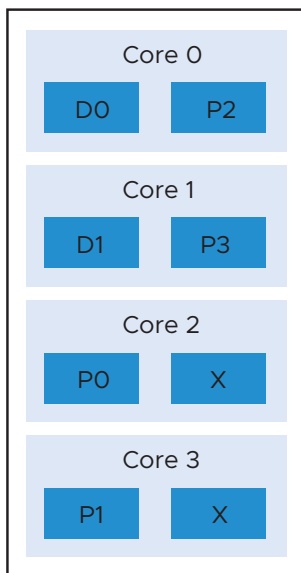


Legend:

SE_CPU_HT_SPARSE_DISPATCHER_PRIORITY on 4 core machine with HT enabled.
 Number of datapath processes is 6.
 D0,D1 are dispatchers. P0,P1...P3 are proxy cores.
 X is unused CPU / HT sibling.

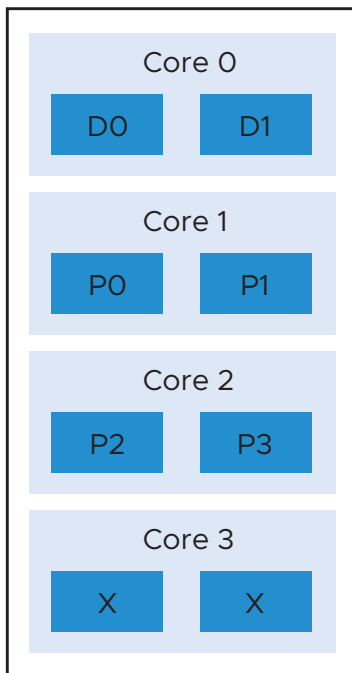
SE_CPU_HT_SPARSE_PROXY_PRIORITY

This mode prioritizes the proxy (non-dispatcher) instances by attempting to place only one data-path process in the physical CPU. This is useful when the number of data path processes is less than the number of CPUs. This mode exhausts the physical cores and then hyper-threads in numerically ascending order of CPU number.

**Legend:**

SE_CPU_HT_SPARSE_PROXY_PRIORITY on 4 core machine with HT enabled.
Number of datapath processes is 6.
D0,D1 are dispatchers. P0,P1...P3 are proxy cores.
X is unused CPU / HT sibling.

This mode places the data path processes on the same physical core. Each core can have two dispatchers or two non-dispatcher (proxy) instances being adjacent to each other. This mode is useful when the number of data path processes is less than the number of CPUs. This mode exhausts the hyper-threads serially on each core before moving on to the next physical core.



Legend:

SE_CPU_HT_PACKED_CORE on 4 core machine with HT enabled.
 Number of datapath processes is 6.
 D0,D1 are dispatchers. P0,P1...P3 are proxy cores.
 X is unused CPU / HT sibling.

Security

Hardware security modules (HSM) is an external security appliance used for secure storage of SSL certificates and keys.

HSM may be configured within the **Templates > Security > HSM Groups**. An HSM group dictates how Service Engines can reach and authenticate with the HSM.

For more information on physical security for SSL Keys, refer to Physical Security for SSL Keys.

Log Collection & Streaming Settings

Significant Log Throttle

This limits the number of significant log entries generated per second per core on an SE. Set this parameter to zero to disable throttling of the UDF log.

UDF Log Throttle

This limits the number of user-defined (UDF) log entries generated per second per core on an SE. UDF log entries are generated due to the configured client log filters or the rules with logging enabled. Default is 100 log entries per second. Set this parameter to zero to disable throttling of the UDF log.

Non-Significant Log Throttle

This limits the number of non-significant log entries generated per second per core on an SE. Default is 100 log entries per second. Set this parameter to zero to disable throttling of the non-significant log.

Number of Streaming Threads

Number of threads to use for log streaming, ranging from 1 to 100.

Other Settings

By default, the Controller creates and manages a single security group (SG) for an NSX Advanced Load Balancer SE. This SG manages the ingress/egress rules for the SE's Control Plane and Data Plane traffic. In certain customer environments, it may be required to provide custom SGs associated with the NSX Advanced Load Balancer SE's management and/or Data Plane vNICs.

For more information about SGs in OpenStack and AWS clouds, read these articles:

- Custom Security Groups in OpenStack article
- Security Group Options for AWS Deployment with NSX Advanced Load Balancer

NSX Advanced Load Balancer Managed Security Group

Supported only for AWS clouds, when this option is enabled, the NSX Advanced Load Balancer will create and manage security groups along with the custom security groups provided by the user. If disabled, the NSX Advanced Load Balancer will only make use of custom security groups provided by the user.

Management vNIC Custom Security Groups

Custom security groups to be associated with management vNICs for SE instances in OpenStack and AWS clouds

Data vNIC Custom Security Groups

Custom security groups to be associated with data vNICs for SE instances in OpenStack and AWS clouds.

Add Custom Tag

In NSX Advanced Load Balancer, custom tags are supported for Azure and AWS clouds. These tags are useful in grouping and managing resources for easier management. To reveal the UI window, click the **Add Custom Tag** hyperlink. For more information on CLI interface, refer to [How to Add Custom Tags Using CLI for Azure and AWS](#).

- Azure tags enable key: value pairs to be created and assigned to resources in Azure. For more information on Azure tags, refer to [Azure Tags](#).
- AWS tags help manage instances, images, and other Amazon EC2 resources, you can optionally assign your metadata to each resource in the form of tags. For more information on AWS tags, refer to [AWS Tags and Configuring a Tag for Auto-created SEs in AWS](#).

VIP Autoscale

Display FIP subnets only

Only display FIP subnets in the drop-down list.

VIP Autoscale Subnet

UUID of the subnet for the new IP address allocation.

NSX Advanced Load Balancer and Internal IPAM Configuration

The VIP of a virtual service can either be a static IP that the user manages, or it can be allocated from the NSX Advanced Load Balancer internal IPAM.

Note The subnet from which the VIP is allocated, should not overlap with the existing subnets in GCP.

If the VIPs for a virtual service need to be managed by NSX Advanced Load Balancer, then the NSX Advanced Load Balancer virtual network needs to be created and then added to the IPAM created.

Creating a Network

This task explains how to create a network.

To create a network:

Procedure

- 1 From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Networks**.
- 2 Select the cloud for which the network must be created.
- 3 Click **Create**.
- 4 Enter the **Name** for the network.
- 5 Click the **Add Subnet** button.
- 6 Enter the subnet range for the VIPs.

Note The subnets should not overlap with any of the existing subnets in the VPC.

- 7 Select the **Add Static IP Address Pool**.

VIPs will be allocated from this pool.

New Network Settings: VIP Network

Name *

VIP Network

• IP Address Management •

☒ DHCP Enabled ? ☒ IPv6 Auto Configuration ?

• Add/Modify Static IP Subnet •

IP Subnet * ?

192.168.32.0/24

☒ Add Static IP Address Pool ?

IP Address Pool *

192.168.32.2-192.168.32.100

• Network IP Subnets •

Q

Displaying 0 items

IP Subnet	Type	IP Address Pool
-----------	------	-----------------

Cancel Save

- 8 Click **Save**.

Updating NSX Advanced Load Balancer Internal IPAM with VIP Networks

This section details how to update the internal IPAM with the created VIP Network.

To update the internal IPAM:

Procedure

- 1 From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Clouds**.
- 2 Select the **GCP cloud** and click the **edit icon**.
- 3 Click the **edit icon** for **IPAM Provider**.
- 4 Click **Add Usable Network**.
- 5 Add the network that was created.

Edit IPAM/DNS Profile: avi-internal-ipam

Name * ⓘ
avi-internal-ipam

Type ⓘ
Avi Vantage IPAM

☐ Allocate IP in VRF ⓘ

Avi Vantage IPAM Configuration

Cloud for Usable Network ⓘ
GCP.Cloud

Usable Network * ⓘ
VIP Network - 192.168.32.0/24

+ Add Usable Network

Save

- 6 Click **Save**.
- 7 Click **Next** in the **Edit Cloud** screen and then click **Save** to complete configuring usable networks.

Virtual Service Configuration

Prior to creating a virtual service, a pool with GCP Servers is required and the NSX Advanced Load Balancer SEs will load balance traffic among the servers in the pool.

Creating Server Pool

This section explains how to create a Server Pool.

To create a pool:

Procedure

- 1 From the NSX Advanced Load Balancer UI, navigate to **Applications > Pools**.
- 2 Click **Create Pool**.
- 3 Select the **GCP cloud**.
- 4 Enter the pool **Name**.

The screenshot shows the 'Step 1: Settings' configuration page for a new pool. The page has a blue header with four steps: Step 1: Settings (active), Step 2: Servers, Step 3: Advanced, and Step 4: Review. The main content area contains the following fields and controls:

- Name:** A text input field containing 'server-pool-1'.
- Enabled:** A toggle switch that is turned on (green).
- Default Server Port:** A text input field containing '80'.
- Graceful Disable Timeout:** A text input field containing '1' with a 'Minutes' unit selector.
- Load Balance:** A dropdown menu set to 'Least Connections'.
- Persistence:** A dropdown menu set to 'None'.
- AutoScale Policy:** A dropdown menu set to 'None'.
- AutoScale Launch Config:** A dropdown menu set to 'None'.
- Analytics Profile:** A dropdown menu set to 'System-Analytics-Profile' with a close (x) and edit (pencil) icon.
- Health Monitors:** A section with a checked 'Passive Health Monitor' checkbox and an '+ Add Active Monitor' button. Below this, it states 'No active health monitors have been added to this pool.'
- Logins:** A checkbox labeled 'Logins: Secure by Name' which is currently unchecked.

At the bottom of the form, there are 'Cancel' and 'Next >' buttons.

- 5 Click **Next**.
- 6 Enter the **Server IP Address** (one or more application (back end) servers in this pool).
The servers can be in any GCP Project or VPC but should be reachable from the SEs.

New Pool: server-pool-1

Step 1: Settings Step 2: Servers Step 3: Advanced Step 4: Review

+ Add Servers +

Select Servers ⓘ

IP Address, Range, or DNS Name IP Group Auto Scaling groups

Server IP Address

10.152.132.68

Add Server

+ Servers +

Q

Displaying 2 items

<input type="checkbox"/>	Status	Server Name	Resolve by DNS	IP Address	Port	Ratio	Network	Header Value	Rewrite Host H...
<input type="checkbox"/>	Enabled		<input type="checkbox"/>	10.152.132.66	Inherit	1		Header Value	<input type="checkbox"/>
<input type="checkbox"/>	Enabled		<input type="checkbox"/>	10.152.132.67	Inherit	1		Header Value	<input type="checkbox"/>

Cancel Previous Next

7 Click **Next**. Navigate to **Step 4: Review**.

8 Click **Save**.

Creating Virtual Service

This section explains the creation of virtual service.

To create a virtual service:

Procedure

- 1 Navigate to **Applications > Virtual Services**.
- 2 Click **Create Virtual Services**.
- 3 Select **Advanced Setup**.
- 4 Select the **GCP cloud**.
- 5 Click **Next**.
- 6 Enter the **Name**.
- 7 In the field **Network for VIP Address Allocation**, select the VIP that was created.
- 8 If the virtual service needs to be accessible via the Internet, select the option **Assign Floating IP for External Client Access**.

The Public IP will be allocated from GCP and the virtual service will be configured with that IP.

- 9 Enter the service port under **Services**.

10 Select the pool, which has the application servers.

The screenshot displays the 'New Virtual Service: gcp-vs' configuration interface. The interface is divided into several sections:

- Step 1: Settings:** Includes fields for Name (gcp-vs), Enabled (checked), Traffic Enabled (checked), and Virtual Hosting VS (unchecked).
- VIP Address:** Includes Auto Allocate (checked), IPv4 VIP (192.168.32.0/24), Assign Floating IP for External Client Access (checked), Floating IP (Auto Allocate), Network for VIP Address Allocation (VIP Network - 192.168.32.0/24), Allocation IP Type (Only IPv4), and IPv4 Subnet (192.168.32.0/24).
- Profiles:** Includes Application Profile (System-HTTP), TCP/UDP Profile (System-TCP-Proxy), WAF Policy (Select WAF Policy), and Error Page Profile (Select Error Page Profile).
- Service Port:** Includes Services (80) and SSL (unchecked).
- Pool:** Includes Pool (selected) and Pool Group (unchecked). The Pool dropdown is set to 'server-pool-1'.
- Other Settings:** Includes a Description field.

The 'Next' button is visible at the bottom right of the screen.

11 Click **Next** in the **New Virtual Service:** screen and navigate to **Step 4: Advanced**.

12 Click **Save**.

Results

Wait for the virtual service status to turn green. On the creation of the first virtual service for a cloud, the service engine VMs will be spawned and configured in GCP.

Installing NSX Advanced Load Balancer in Linux Server Cloud

4

This article describes the installation process of NSX Advanced Load Balancer in a Linux Server Cloud (LSC).

This chapter includes the following topics:

- [Overview](#)
- [Considerations for Deploying NSX Advanced Load Balancer in Linux Server Cloud](#)
- [Installing NSX Advanced Load Balancer in a Linux Server Cloud](#)
- [Configuring the NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Deployment Options](#)

Overview

NSX Advanced Load Balancer is a software-based solution that provides real-time analytics and elastic application delivery services, including user-to-application timing, SSL termination, and load balancing. Installing it directly onto Linux servers leverages the raw horsepower of the underlying hardware without the overhead added by a virtualization layer.

VMware recommends that you disable hyper threading (HT) in the BIOS of the Linux servers upon which NSX Advanced Load Balancer runs before installing it. It does not get changed often, but RHEL, OEL and CentOS may map physical and hyperthreaded cores differently. Rather than basing its decision on the behaviour or characteristics of a core, the load balancer has a predictive map of the host OS via which it skips or ignores hyperthreaded cores. When an OS gets upgraded, this map might change, which means you might be utilizing an HT core instead of a physical core, which will impact the performance.

Docker Container

The NSX Advanced Load Balancer Linux server cloud solution uses containerization provided by the Docker for support across operating systems and easy installation.

Considerations for Deploying NSX Advanced Load Balancer in Linux Server Cloud

This section explains about deployment topologies and prerequisite details.

Deployment Topologies

NSX Advanced Load Balancer can be deployed onto a Linux Server Cloud in the following topologies. The minimum number of Linux servers required for deployment depends on the deployment topology. A three-controller cluster is strongly recommended for production environments.

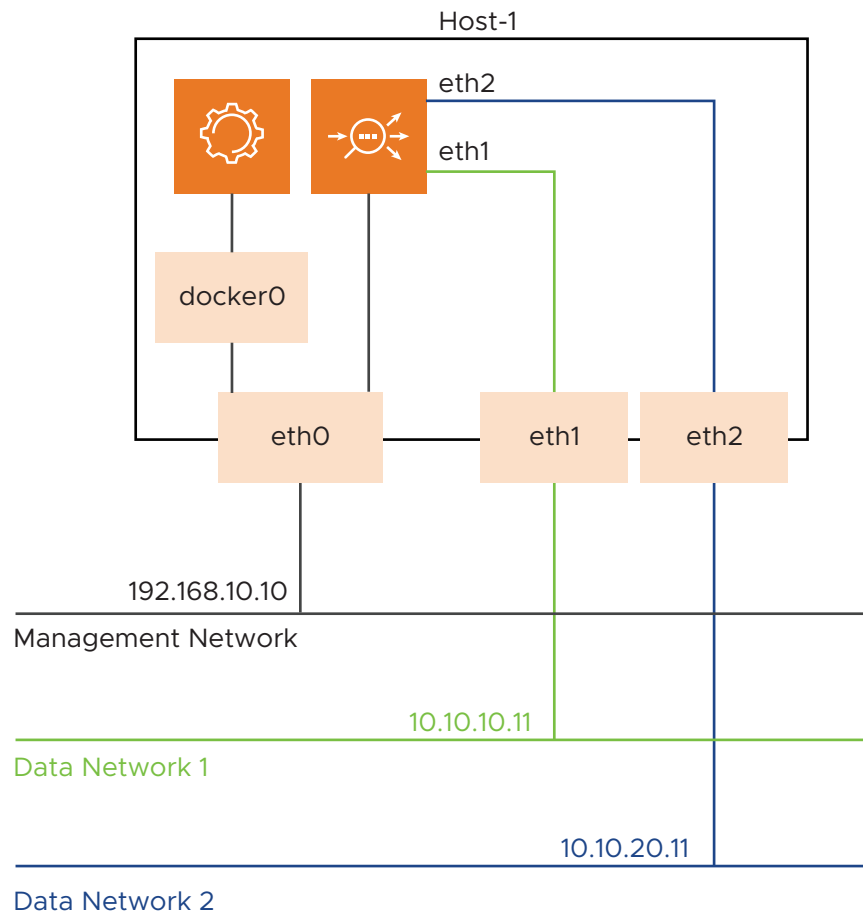
Deployment Topology	Minimum Linux Servers Required	Description
Single host	1	Both the Controller and the SE run on a single host.
Separate hosts	2	The Controller and SE run on separate hosts. The Controller is deployed on one of the hosts. The SE is deployed on the other host.
3-host cluster	3	Provides high availability for the Controller.

A single instance of the Controller is deployed on each host. At any given time, one of the Controllers is the leader and the other two are followers.

Single-host Deployment

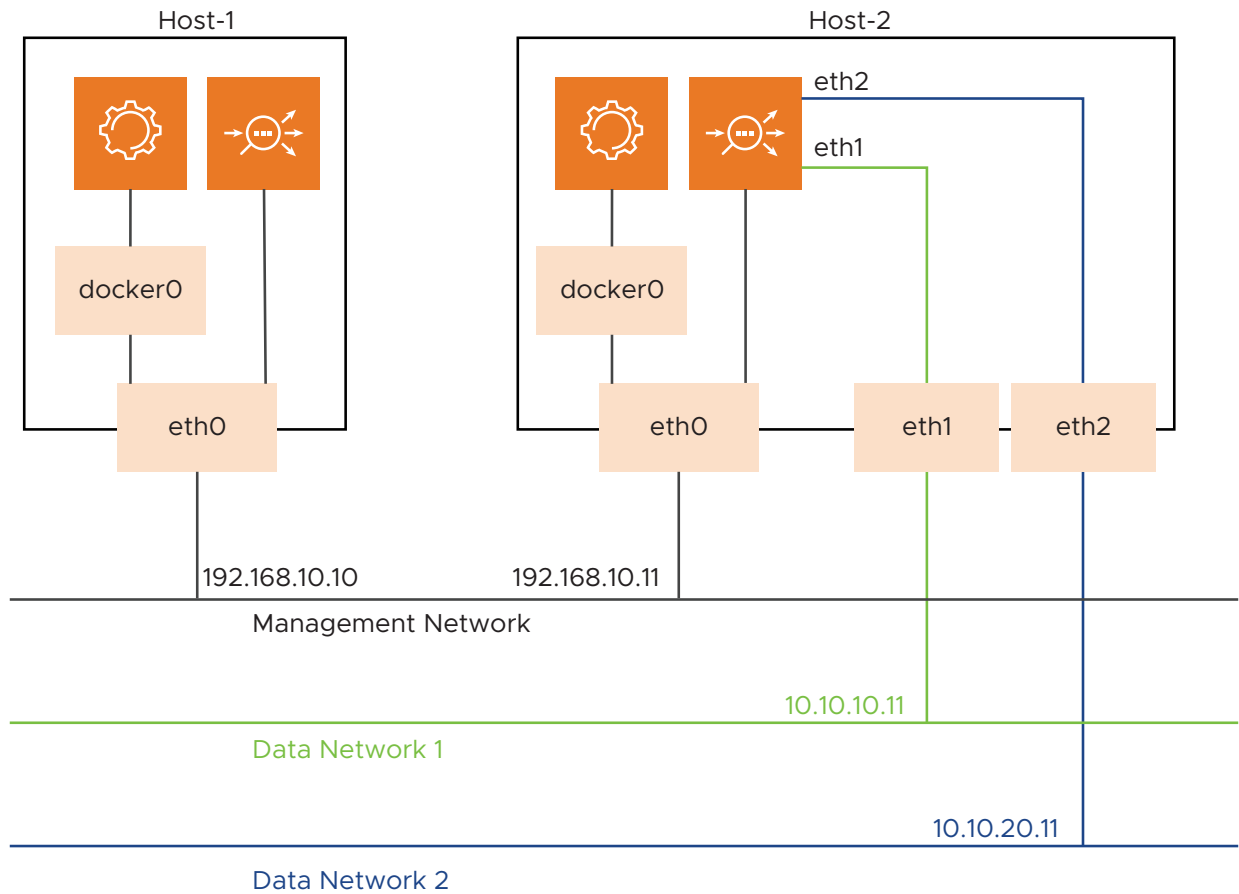
Single-host deployment runs the Controller and the SE on the same Linux server. This is the simplest topology to deploy. However, this topology does not provide high availability for either the Controller or the SE.

Note In-band management is not supported in the single host mode.



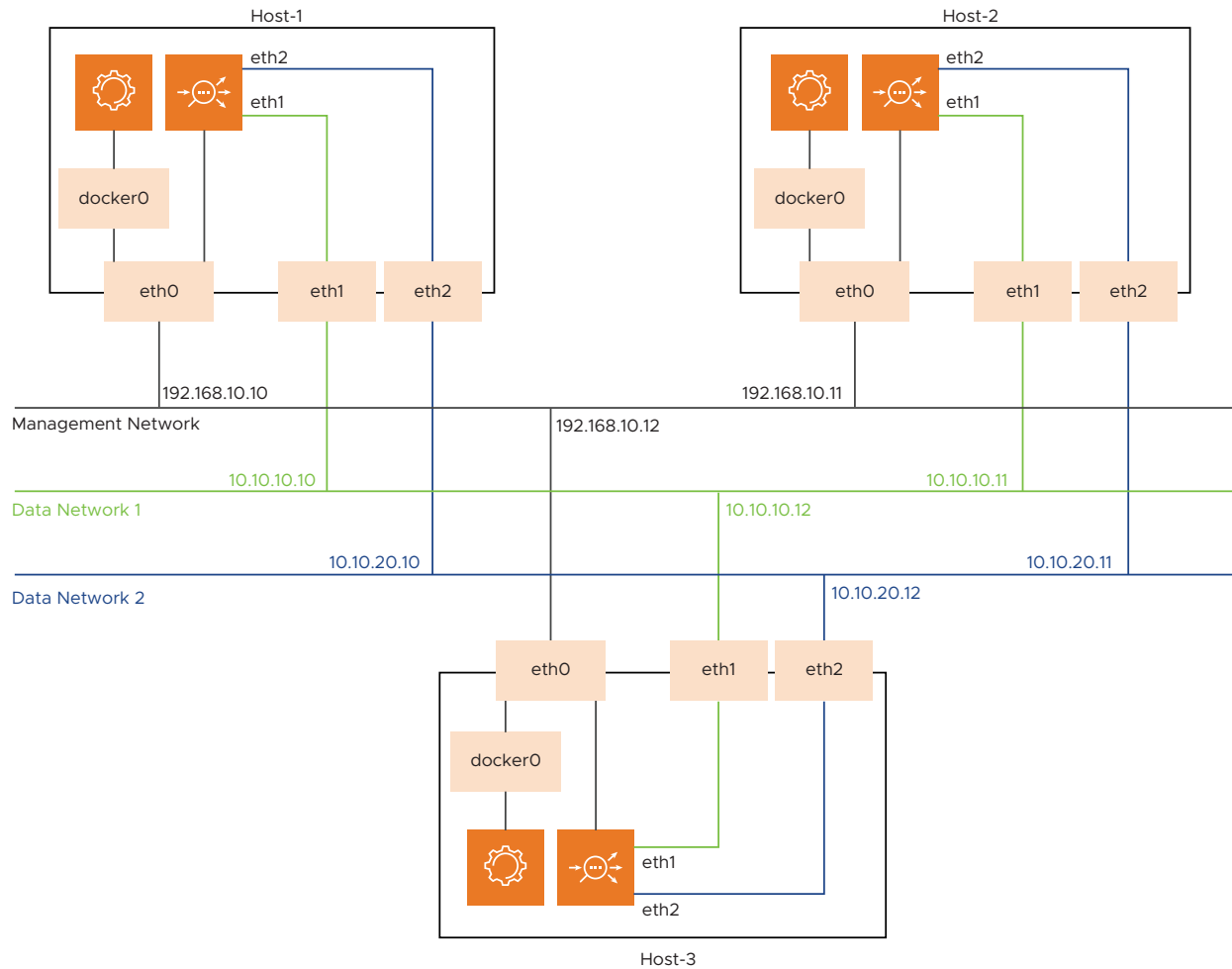
Two-host Deployment

Two-host deployment runs the Controller on one Linux server and the SE on another.



Three-host Cluster Deployment

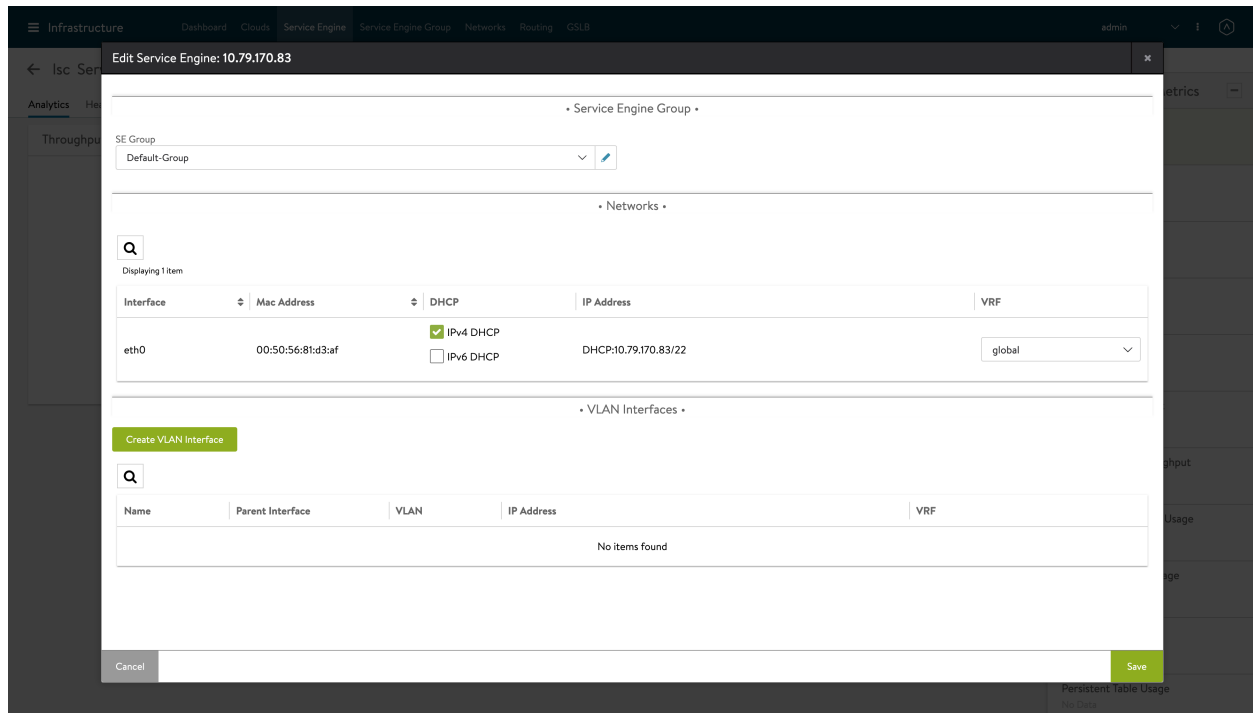
Three-host deployment requires a separate instance of the Controller on each of the three Linux servers.



In a three-host cluster deployment, one of the Controller instances is the leader. The other two are followers. If the leader goes down, one of the followers takes over so that the control-plane functionality for users is continued.

DHCP on Datapath Interfaces

The DHCP mode is supported on datapath interfaces (regular interfaces / bond) in baremetal / LSC Cloud and can be enabled from the Controller GUI.



You can also enable DHCP from the Controller using the `configure serviceengine <serviceengine-name>` command. You can check the desired `data_vnics index (i)` using the following command:

```
data_vnics index <i>
dhcp_enabled
save
save
```

This enables DHCP on the desired interface.

To disable DHCP on a particular `data_vnic`, you can replace `dhcp_enabled` by `no dhcp_enabled` in the above command sequence.

If DHCP is disabled on a datapath interface, a static IP address should be configured on it. Otherwise the connectivity through the NIC would be lost.

Note It is recommended to enable DHCP only on interfaces that are actively used. It is advised to leave DHCP disabled on unmanaged/unused interfaces.

Prerequisites

This section lists the minimum installation requirements.

Hardware Requirements

Each Linux server to be managed by the NSX Advanced Load Balancer must meet at least the following physical requirements:

Docker local storage (default `/var/lib/docker`) should be at least 18 GB to run NSX Advanced Load Balancer containers. If the Load Balancer SE is instantiated through the cloud UI, add 5 GB to run it.

By default, the Docker `devicemapper` storage driver is configured in `loop-lvm` storage mode, which is only recommended for testing. NSX Advanced Load Balancer recommends configuration of the driver according to Docker Best Practices for Controller hosts which will be used for production.

Note See [Docker storage drivers](#) for recommendations on choosing a storage driver.

You need to turn off IOMMU if SE has to be used in DPDK mode.

The following are the minimum hardware requirement details of the Controller:

Component	Minimum Requirement
CPU	8 cores
Memory	24 GB RAM
Disk	64 GB
Network Interface Controller (NIC)	

Note IOMMU can be turned off through BIOS or by specifying `intel_iommu=off` or `amd_iommu=off` based on the architecture type in Kernel command line of the host.

Software Requirements

For more information, see [Software Requirements for Linux Server Cloud](#).

You can place the Controller and the SE containers on the same host starting only from RHEL version 7.4. If co-located on the same host, restarting either container will fail for RHEL versions prior to 7.4. For more information on supported versions of RHEL, see System Requirements section. It is mandatory to disable SELinux when deploying the Controller as a Podman instance (RHEL 8.0). By default, SELinux is enabled in CentOS/RHEL.

The following are the steps to disable SELinux:

- 1 Log in to the host machine as root.
- 2 Invoke the `getenforce` command to determine whether SELinux is enforcing, permissive, or disabled.
- 3 If the output of `getenforce` is either permissive or disabled, nothing more need be done. If the output is enforcing, perform all subsequent steps.
- 4 Open the `/etc/selinux/config` file (in some systems, open `/etc/sysconfig/selinux` instead).
- 5 Change the line `SELINUX=enforcing` to `SELINUX=permissive`.
- 6 Save and close the file.

7 Reboot the system.

The following are the default port assignments. If the below mentioned ports are in use, then choose alternative ports for the purpose listed.

Ports	Purpose
5098	SSH (CNTRL_SSH_PORT)
8443	Secure bootstrap communication between SE and Controller (SYSINT_PORT)
80, 443	Web server ports (HTTP_PORT, HTTPS_PORT)
161	SNP MIB walkthrough
5054	Shell CLI

Preparing the Server

This section describes the types of packages to run an application.

Installing a Docker

A Docker image is an executable package that includes everything needed to run an application. NSX Advanced Load Balancer for Linux server cloud is distributed as a Docker image. Therefore, having Docker installed is mandatory for Linux server cloud deployments. If deploying onto multiple hosts, repeat the applicable installation procedure on each host.

To ensure proper Controller core dumps in a container environment (running either host or bridge mode), take care that the dump path is properly set. If not, the core will not get generated. The core-pattern on the container's host machine must be initialized prior to bringing up the Controller. The recommendation is to set it to `/var/crash/core_dump-%E-%p-%t`, where:

- %E = pathname of executable with slashes('/') replaced by exclamation marks ('!')
- %p = PID of dumped process, as seen in the PID namespace in which the process resides
- %t = time of dump, expressed as seconds since the Epoch, 1970-01-01 00:00:00 + 0000 (UTC)

Docker Editions

The following are the Docker editions:

- Community Edition (CE) – This edition is suitable for individual developers and small teams exploring container-based applications.
- Docker Enterprise Edition (EE) – This edition is suitable for enterprise development and IT teams who build, ship, and run critical business-critical applications in production at scale. The following are the different types of Docker Enterprise Edition:
 - ■ Enterprise Edition Basic

- Enterprise Edition Standard
- Enterprise Edition Advanced

To understand which edition of Docker is applicable to you, see [Overview of Docker Editions](#).

Installing Docker EE

This section explains the installation process of Docker EE for [Ubuntu](#), [RHEL](#), [CentOS](#), and [Oracle Linux](#). It is recommended to ensure the respective prerequisites are met before installing Docker EE.

Red Hat Packet Manager (RPM)-based distributions such as RHEL, use a tool called YUM that work with your repositories to manage dependencies and provide automatic updates.

Selecting a Storage Driver for a Docker

Docker supports several storage drivers using a pluggable architecture. The storage driver controls how images and containers are stored and managed on your Docker host. Docker advises a storage driver for production environments and documents their operation and merits in the [Storage Drivers](#) link.

For detailed information, see [Supported Storage Drivers per Linux Distribution](#).

Device Mapper

Device Mapper is a kernel-based framework that underpins many advanced volume management technologies on Linux. A `devicemapper` storage driver is mandatory if you use Docker EE on RHEL, CentOS, or Oracle Linux. It is also supported on Docker CE running on CentOS, Fedora, Ubuntu, or Debian. Docker recommends that a device mapper be configured in `direct-lvm-mode` for production.

For detailed information, see [Device Mapper Driver](#).

AUFS Overlay

For Docker CE, AUFS is supported on Ubuntu, and it is a default storage driver for Ubuntu. If your Linux kernel is version 4.7 or higher, and you use Docker CE, you need to use `overlay2`.

AUFS cannot use `aufs`, `btrfs`, or `ecryptfs` as a backing filesystems. This means that the filesystem which contains `/var/lib/docker/aufs` cannot be one of these file system types.

`devicemapper` is not recommended for Ubuntu. You can use `overlay2`, if supported or AUFS.

For detailed information on **AUFS Overlay**, see:

- [Use the AUFS Storage Driver](#)
- [Use the OverlayFS Storage Driver](#)

Verifying Docker Installation and Version

Specify the `docker version` command to confirm the appropriate version is installed and running, for instance, if the version is 1.8.1, the following is the CLI command to be used:

```
docker version
Client:
Version: 1.8.1
API version: 1.20
Go version: go1.4.2
Git commit: d12ea79
Built: Thu Aug 13 02:35:49 UTC 2015
OS/Arch: linux/amd64
Server:
Version: 1.8.1
API version: 1.20
Go version: go1.4.2
Git commit: d12ea79
Built: Thu Aug 13 02:35:49 UTC 2015
OS/Arch: linux/amd64
```

Note Docker local storage can be given at docker daemon start. If you do not specify storage at docker daemon start, docker picks up the default local storage based on underlying storage driver. For `devicemapper - /var/lib/docker..etc.`, NSX Advanced Load Balancer adds the requirement of 18 GB for running NSX Advanced Load Balancer containers.

Network Interface on the Server

The network interface on the server supports the VLAN configuration and port channeling on Linux server hosts.

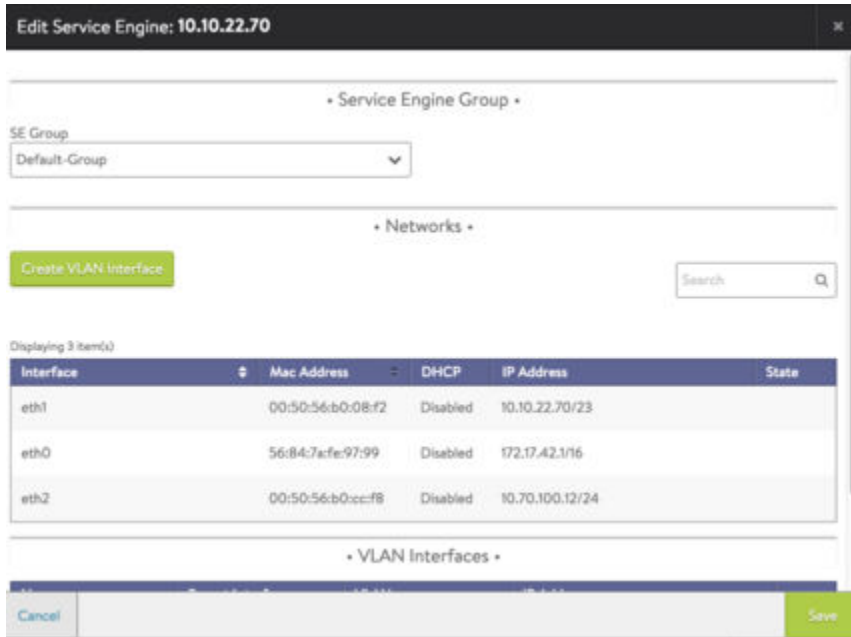
Configuring VLAN on a Bare Metal

NSX Advanced Load Balancer supports VLAN trunking on baremetal servers. If the Controller is deployed on a bare metal server, the individual physical links of the server can be configured to support 802.1q-tagged VLANs. Each VLAN interface has its own IP address. Multiple VLAN interfaces per physical link are supported.

NSX Advanced Load Balancer supports VLAN interface configuration on Linux server cloud as well. The steps to configure a VLAN interface on a bare metal server are:

Procedure

- 1 Navigate to **Infrastructure**, and click **Service Engine (SE)** where the VLAN interface is to be created. Click **Edit** button to edit the details.



- 2 Click **Create VLAN Interface**.
- 3 Select the parent interface from the drop-down menu, and fill in the **VLAN and IP Address** fields, click **Save**. The VLAN interface appears on the **Service Engine Edit** page.
- 4 Click **Save** option to commit the change.

Configuring Port Channel on Linux Server Hosts

NSX Advanced Load Balancer on Linux server (bare metal) cloud supports port channel (bond) interfaces. A port channel interface groups multiple physical interfaces into a single logical interface and provides fault tolerance, bandwidth aggregation, and traffic load balancing. A port channel interface can be configured with an IP address and VLAN trunking. Up to eight physical links can be grouped into a single port channel interface.

Note

- 1 Port channeling is also referred to as port bonding, port trunking, and link aggregation.
- 2 This feature is supported for IPv6 in NSX Advanced Load Balancer.

Link Interface Load Balancing

Based on the source and destination IP address and the Layer 4 protocol ports of the outgoing traffic, a hash is generated. The hash determines the transmitting link for this traffic to achieve load balancing.

Link Failure Recovery

Traffic directed on a failing link is automatically redirected to other links within the port channel interface to achieve fault tolerance.

Configuring Port Channel

This section discusses a set of sample port channel configuration files. In the Linux interface configuration files, a bond interface consists of a bond interface and one or more member (secondary) interfaces.

Note You can configure port channeling using the Linux server's interface configuration files. This configuration is not supported on NSX Advanced Load Balancer.

- The `mode=4` bonding option stands for Link Aggregation Control Protocol (LACP).
- `ens1f0` and `ens1f1` are the two member interfaces of `bond0` interface. The following options are configured for these interfaces as they are the secondary members of a logical interface with `bond0` as its primary member:
 - `MASTER=bond0`
 - `SECONDARY=yes`
- `bond0.652` interface is the VLAN interface under `bond0`, configured with `VLAN=yes` option.

Interface `bond0`'s Configuration file: `/etc/sysconfig/network-scripts/ifcfg-bond0`

```
DEVICE=bond0
IPADDR=10.124.251.101
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
NM_CONTROLLED=no
BONDING_OPTS="mode=4 miimon=100 xmit_hash_policy=layer3+4 use_carrier=1"
```

Interface `ens1f0`'s Configuration file: `/etc/sysconfig/network-scripts/ifcfg-ens1f0`

```
DEVICE=ens1f0
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
NM_CONTROLLED=no
```

Interface `ens1f1`'s Configuration file: `/etc/sysconfig/network-scripts/ifcfg-ens1f1`

```
DEVICE=ens1f1
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```



```
USERCTL=no
NM_CONTROLLED=no
```

Configuring vLAN on NSX Advanced Load Balancer

You can configure VLANs and port channel interfaces with IPv6 addresses as well.

The following are the steps to configure a VLAN on a logical interface eth0 in NSX Advanced Load Balancer:

1. Navigate to **Infrastructure**, and select the Service Engine (SE) where the VLAN interface is to be created and click on **Edit** icon.

SE Group: Default-Group

• Networks •

Create VLAN Interface

Displaying 3 item(s)

Interface	Mac Address	DHCP	IP Address	State	VRF
eth0	00:50:56:bd:16:7e	Disabled		Unknown	global
eth1	00:50:56:bde5:a4	Disabled	STATIC IPv4: 10.140.117.28/24 STATIC IPv6: fd00:0:0:117::505/64 DHCP: 10.140.117.16/24, fe80::250:56ff:febd:e5a4/64	Unknown	global
eth2	00:50:56:bd:6b:6e	Disabled	STATIC IPv4: 10.140.116.29/24 STATIC IPv6: fd00:0:0:116::506/64 DHCP: 10.140.116.26/24, fe80::250:56ff:febd:6b6e/64	Unknown	global

• VLAN Interfaces •

Name	Parent Interface	VLAN	IP Address	VRF
------	------------------	------	------------	-----

Cancel Save

2. Click on **Create VLAN Interface**.
3. Select the parent interface from the drop-down menu, and specify the **VLAN and IP Address** fields. Choose **Global** from the drop-down menu for VRF. Specify the IP prefix of the Service Engine's data vNIC in the **Static IP Prefix** field. Click on **Save** option.
4. The configured vLAN interface will be displayed in the **VLAN Interfaces** section of the **Service Engine** edit page.
5. Click on **Save** to commit the change.

Public Key Management on SE Hosts

This section describes how to set up SSH on the Controller and each SE host so that the Controller can log onto the SEs in a Linux server cloud. A part of this process takes place on the Controller while the other part takes place on each SE hosts.

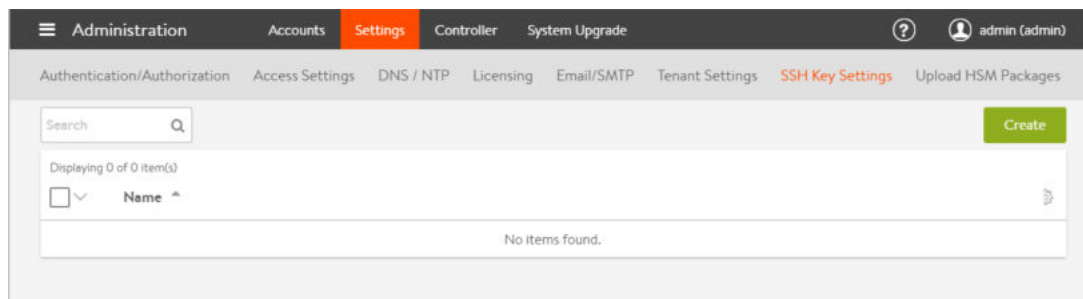
While installing NSX Advanced Load Balancer for a Linux server cloud, part of the deployment process for a new SE is to add an SSH user to the Controller, then add the same user and its public key to the SE host. The SSH user and key are used by the Controller to log onto the SE host, transfer the Docker container for the SE onto the host, and start the SE within the Docker container.

Adding SSH User to the NSX Advanced Load Balancer Controller

On the Controller, add the SSH user and the user's public-private key pair. You can create an SSH account on the Controller (the easy way), or an existing account can be used by adding its username and importing its keys.

Use this section even if the SSH user has already been added. You can copy the user's public key so it can be pasted into a command line on each of the SE hosts.

Navigate to **Administration > Settings**, and click **SSH Key Settings**. If any SSH users have already been added to the Controller, they will be listed here.



If there are more than one account, you can use the same account for all the SE hosts. A unique account is not required for each SE host (The account serves a similar purpose to the well-known secret in a routing protocol topology).

Creating New SSH User

The following are the steps to create new SSH user:

- 1 Click **Create SSH User**.
- 2 Specify the user name.
- 3 Select **Generate SSH Key Value Pair** radio button and click **Generate SSH Key Pair**.
- 4 Click **Copy to Clipboard**.
- 5 Click **Save** option. The SSH user appears in the list.

Preparing SE Hosts

To prepare a host where SEs are launched, login to the host as a user that has sudo privileges and run the following command:

```
curl -ks https://[controller-ip]/api/linux_host_install?username=[username] | sudo bash
```

This command invokes an API to download a script that has the public key credentials of the user associated with the cloud and the necessary steps to set up the user in this host. The output of the script is piped to `sudo bash`.

```
curl -ks https://10.10.25.46/api/linux_host_install?username=newuser | sudo bash
Updating the authorized keys under /etc/ssh/authorized_keys_newuser
Checking settings for key-based login...
PubKeyAuthentication based login is already set up.
Finished configuration
```

Verifying if SE Hosts are Setup

You can verify if the host has been setup with the SSH credentials correctly from the NSX Advanced Load Balancer Controller.

This verification can be done as a part of adding a server in the Linux cloud by clicking **?** icon.

If a host is not set up correctly, the system will display an error message with the instructions to setup the host.

Alternatively, you can also verify that the SE hosts have been setup as a part of the SSH user configuration.

Installing NSX Advanced Load Balancer in a Linux Server Cloud

You can install NSX Advanced Load Balancer in a Linux server cloud.

The following are the steps to install NSX Advanced Load Balancer in a Linux Server cloud:

Prerequisites

You need to install `libselinux-python` only if all the following conditions are met:

- The host runs with OEL 6.9 or prior
- The host uses `selinux` for security

While installing NSX Advanced Load Balancer, the installation wizard for Controller must be run on the Linux server that will host it. If deploying a 3-host cluster of Controllers, run the wizard only on the host that will be the cluster leader (The cluster can be configured at any time after installation is complete).

Procedure

- 1 Install the Docker platform, if you have not installed it yet. On Ubuntu, you can use `apt-get install docker.io` command.
- 2 Install NTP server using `sudo yum install ntp` command on the host operating system.

3 Install the Controller image onto a Linux server.

- a Use SCP to copy the `.tgz` package onto the Linux server that will host NSX Advanced Load Balancer using `scp docker_install.tar.gz root@Host-IP:/tmp/` command.
- b Use SSH to log into the host using `ssh root@Host-IP` command.
- c Change to the `/tmp` directory: `cd /tmp/` using `cd /tmp/` command.
- d Unzip the `.tgz` package using `sudo tar -xvf docker_install.tar.gz`.
- e Run the `setup.py` script. It can be run in interactive mode or as a single command string.
 - If you run the script as a command string, then the script sets the options that are included in the command string to the specified values, and leaves the other values set to their defaults. Next go to Step f.
 - If you run the script as interactive mode, the script displays a prompt for configuring each option. Next go to Step g.

Note To ensure proper operation of the `avi_baremetal_setup.py` script in either steps 'f' or 'g', the locale must be set to English. You can use the `LANG=en_US.UTF-8` command.

- f Run the setup script as a single command using `./avi_baremetal_setup.py -c -cc 8 -cm 24 -i 10.120.0.39` command string. The CLI command is as follows:

```
avi_baremetal_setup.py [-h] [-d] [-s] [-sc SE_CORES] [-sm SE_MEMORY_MB] [-c] [-cc
CON_CORES] [-cm CON_MEMORY_GB] -i CONTROLLER_IP -m MASTER_CTL_IP-h, --help show this
help message and exit
-d, --dpdk_mode Run SE in DPDK Mode. Default is False
-s, --run_se Run SE locally. Default is False
-sc SE_CORES, --se_cores SE_CORES
Cores to be used for AVI SE. Default is 1
-sm SE_MEMORY_MB, --se_memory_mb SE_MEMORY_MB
Memory to be used for AVI SE. Default is 2048
-c, --run_controller Run Controller locally. Default is No
-cc CON_CORES, --con_cores CON_CORES
Cores to be used for AVI Controller. Default is 4
-cm CON_MEMORY_GB, --con_memory_gb CON_MEMORY_GB
Memory to be used for AVI Controller. Default is 12
-i CONTROLLER_IP, --controller_ip CONTROLLER_IP
Controller IP Address
-m MASTER_CTL_IP, --master_ctl_ip MASTER_CTL_IP
Master controller IP Address
```

- g To run in an interactive mode, specify `avi_baremetal_setup.py`

```
./avi_baremetal_setup.pyWelcome to AVI Initialization ScriptDPDK Mode:
Pre-requisites(DPDK): This script assumes the below utilities are installed:
docker (yum -y install docker)
Supported Nics(DPDK): Intel 82599/82598 Series of Ethernet Controllers
Supported Vers(DPDK): OEL/CentOS/RHEL - 7.0,7.1,7.2Non-DPDK Mode:
Pre-requisites: This script assumes the below utilities are installed:
docker (yum -y install docker)
Supported Vers: OEL/CentOS/RHEL - 7.0,7.1,7.2Caution : This script deletes existing
AVI docker containers & images.Do you want to proceed in DPDK Mode [y/n] y
Do you want to run AVI Controller on this Host [y/n] y
Do you want to run AVI SE on this Host [n] n
Enter The Number Of Cores For AVI Controller. Range [4, 39] 8
Please Enter Memory (in GB) for AVI Controller. Range [12, 125] 24
Please Enter directory path for Avi Controller Config (Default [/opt/avi/controller/
data/])
Please Enter disk (in GB) for Avi Controller config (Default [30G])
Do you have separate partition for Avi Controller Metrics? If yes, please enter
directory path, else leave it blank
Do you have separate partition for Avi Controller Client Log? If yes, please enter
directory path, else leave it blank
Please Enter Controller IP 10.120.0.39
Run SE : No
Run Controller : Yes
Controller Cores : 8
Memory(mb) : 24
Controller IP : 10.120.0.39Disabling AVI Services...
Loading AVI CONTROLLER Image. Please Wait..
kernel.core_pattern = /var/crash/%e.%p.%t.coreInstallation Successful. Starting
Services...
```

- h Start NSX Advanced Load Balancer on the host to complete installation using `sudo systemctl start avicontroller` command.
- i If you are deploying a 3-host cluster, repeat the steps above on the hosts for each of the other 2 Controllers.

Configuring the NSX Advanced Load Balancer Cloud Connector

This section explains configuring linux cloud using CLI as well as using the UI method.

Configuring Linux Cloud using CLI

This section provides an example of using the CLI to create a Linux server cloud.

The following are the steps to configure Linux cloud using the CLI method:

Procedure**1 Create your cloud as follows:**

```
[admin:10-50-0-2]: > configure cloud gcp
```

2 Attach an IPAM profile as follows:

```
[admin:10-50-0-2]: cloud> ipam_provider_ref gcp
```

3 Create Linux server configuration as follows:

```
[admin:10-50-0-2]: cloud> linuxserver_configuration
[admin:10-50-0-2]: cloud:linuxserver_configuration> hosts host_ip 10.50.0.5
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts> host_attr attr_key CPU attr_val 2
New object being created
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts:host_attr> save
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts> host_attr attr_key MEMORY
attr_val 4
New object being created
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts:host_attr> save
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts> host_attr attr_key DPDK attr_val
No
New object being created
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts:host_attr> save
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts> host_attr attr_key
SE_INBAND_MGMT attr_val True
New object being created
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts:host_attr> save
[admin:10-50-0-2]: cloud:linuxserver_configuration:hosts> save
```

Repeat the above host steps to incorporate additional Service Engines.

4 Save your config as follows:

```
[admin:10-50-0-2]: cloud:linuxserver_configuration> save
[admin:10-50-0-2]: cloud> save
```

5 After specifying the Linux server configuration, you can go to edit mode using the following command:

```
[admin:10-50-0-2]: cloud:linuxserver_configuration> new
```

6 Copy in the host data and change the SE resource variables and `ssh_user_ref` accordingly as follows:

```
hosts:
- host_attr:
- attr_key: CPU
attr_val: '2'
- attr_key: MEMORY
attr_val: '4'
- attr_key: DPDK
```

```

attr_val: 'No'
- attr_key: SE_INBAND_MGMT
attr_val: 'True'
host_ip:
addr: 10.50.0.4
type: V4
- host_attr:
- attr_key: CPU
attr_val: '2'
- attr_key: MEMORY
attr_val: '4'
- attr_key: DPDK
attr_val: 'No'
- attr_key: SE_INBAND_MGMT
attr_val: 'True'
host_ip:
addr: 10.50.0.6
type: V4
se_inband_mgmt: false
se_log_disk_size_GB: 5
se_sys_disk_size_GB: 10
ssh_user_ref: root

```

Configuring Linux Cloud using UI

This section describes how to add a new SE to an existing Linux server cloud.

The following are the steps to add servers to a Linux server cloud:

Prerequisites

- The Controller should be already installed.
- The Linux server cloud configuration should exist on the Controller.

Procedure

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Click **Edit** icon to open the configuration pages for the cloud. The cloud name by default is named as **Default-Cloud**, unless another cloud has already been configured.

3 Click **Linux Server Configuration** tab in **Default Cloud** window.

The screenshot shows the 'Edit Cloud: Default-Cloud' window in the NSX Advanced Load Balancer interface. The 'Linux Server Configuration' tab is active. The 'SSH User' is set to 'root'. The 'License Type' is 'Cores'. The 'IPAM Profile' is 'GCP_ipamdnsprofile'. The 'DNS Profile' is 'Select DNS Profile'. The 'Use Inband Management' checkbox is checked. The 'Path for System Logs' is set to '10 GB' and the 'Path for Client Logs' is set to '5 GB'. The 'Server List' table at the bottom shows one server with the following details:

Host IP Address	Number of Cores	Memory (GB)	DPDK	Inband Management
10.146.1.5	1	2	No	Yes

- 4 The **SSH User** field shows the SSH user name the Controller will use to log onto each of the SE hosts. A key generated by the Controller for that SSH user account is required, and must be present in the authorized keys store of each SE host. For managing keys for SE hosts, see [Public Key Management on SE Hosts](#) section.

Note NSX Advanced Load Balancer supports access to the host via a password instead of an SSH key, with the user which has the password-less sudoer privileges.

- 5 A default path and disk space are allocated to the system and client logs. You can customize the directory path and disk space by selecting separate paths for system and client logs and entering relevant information in the following fields:

- Path for System Logs
- Size: Disk space for System Logs
- Path for Client Logs
- Size: Disk Space for Client Logs

For each Linux server that will host an SE, specify the following information:

- Host IP address: Must already be configured on the host
- Number of Cores (CPUs)
- Memory DPDK: Yes, if installing directly onto bare metal with an applicable Ethernet NIC; No, if installing onto a virtual machine (VM) or a NIC not supported for DPDK
- In-band Management

- 6 You can configure a Service Engine by selecting the options from **Template Service Engine Group** field in the **Service Engine** section.
- 7 Click **Save** option.

Verifying NSX Advanced Load Balancer SE Creation

This section explains the steps to verify the SE information.

The following are the steps to verify if the SE is created or not:

- 1 In the Controller UI, navigate to **Infrastructure > Clouds**.
- 2 Click the row for **Default-Cloud**. (This is the name of the cloud created by the setup wizard.) Information for each of the SE hosts in the cloud is displayed.

Click the row for the cloud to toggle display of the SE information.

For instance, if the state information for the host is **SSH Failed**, which may be the most common issue encountered during Linux server cloud installation. This indicates that the Controller tried to log into the SE host with the SSH user account and corresponding key specified during setup, but could not successfully negotiate the handshake.

If the Controller is able to log onto the SE host, the following messages should appear:

- Image in Progress
- Image Complete
- Start in Progress
- Started
- In Progress
- Placement Ready

After the **Placement Ready** status appears for the SE, virtual services are created and placed on it.

Additional Deployment Options

This section gives information on the additional deployment options suggested in Linux server cloud installation guide.

Configuring In-band Management for an NSX Advanced Load Balancer Service Engine

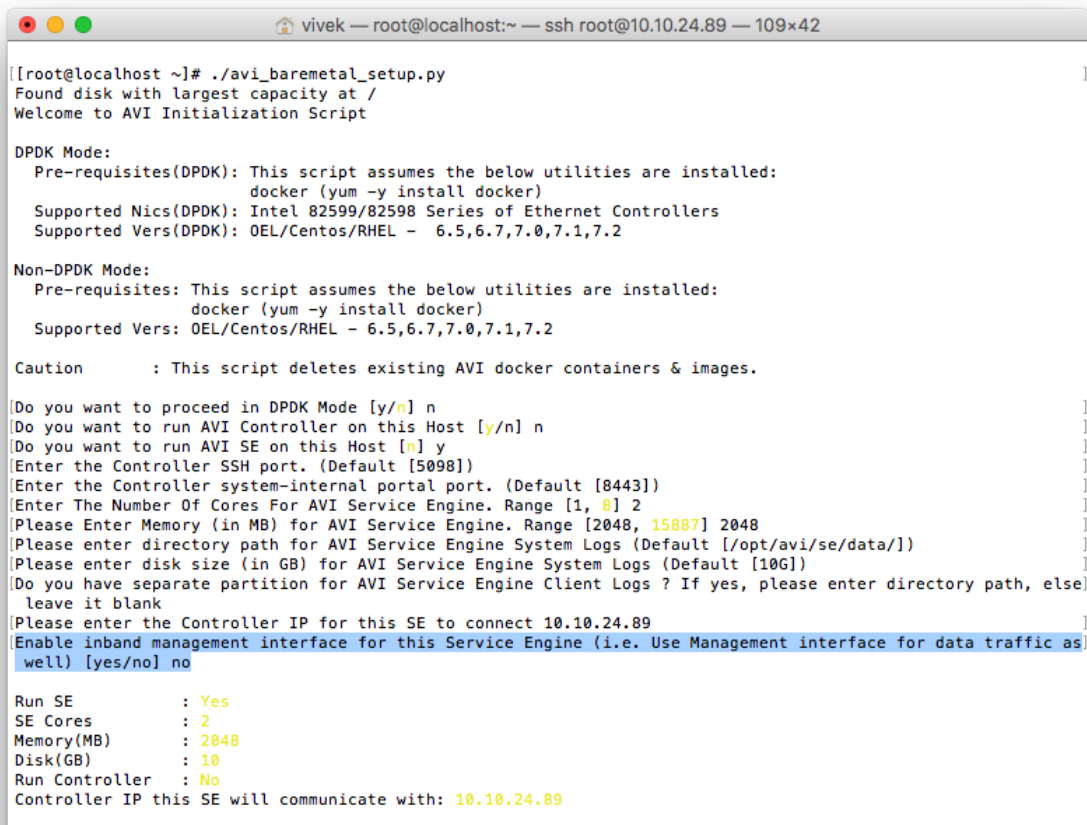
SE interface communicates with the Controller cluster used for its data plane traffic. This section explains how to enable/disable the in-band management attribute on an SE.

Note

- If in-band management is enabled on an SE, that SE will not support multiple VRFs.
- To enable multiple VRFs on an SE, it must be deployed with in-band management disabled. The caveat with disabling in-band management is that the management interface will not be used for data plane traffic. Hence no virtual service will be placed on this interface and this interface will not be used to communicate with the back end servers.

SE In-band Management Configuration Using Bare-Metal Script

This section details deployment of a Service Engine with in-band management enabled or disabled using the `avi_baremetal_setup.py` script.



```

vivek — root@localhost:~ — ssh root@10.10.24.89 — 109x42

[[root@localhost ~]# ./avi_baremetal_setup.py
Found disk with largest capacity at /
Welcome to AVI Initialization Script

DPDK Mode:
  Pre-requisites(DPDK): This script assumes the below utilities are installed:
                        docker (yum -y install docker)
  Supported Nics(DPDK): Intel 82599/82598 Series of Ethernet Controllers
  Supported Vers(DPDK): OEL/Centos/RHEL - 6.5,6.7,7.0,7.1,7.2

Non-DPDK Mode:
  Pre-requisites: This script assumes the below utilities are installed:
                  docker (yum -y install docker)
  Supported Vers: OEL/Centos/RHEL - 6.5,6.7,7.0,7.1,7.2

Caution      : This script deletes existing AVI docker containers & images.

[Do you want to proceed in DPDK Mode [y/n] n
[Do you want to run AVI Controller on this Host [y/n] n
[Do you want to run AVI SE on this Host [n] y
[Enter the Controller SSH port. (Default [5098])
[Enter the Controller system-internal portal port. (Default [8443])
[Enter The Number Of Cores For AVI Service Engine. Range [1, 8] 2
[Please Enter Memory (in MB) for AVI Service Engine. Range [2048, 15887] 2048
[Please enter directory path for AVI Service Engine System Logs (Default [/opt/avi/se/data/])
[Please enter disk size (in GB) for AVI Service Engine System Logs (Default [10G])
[Do you have separate partition for AVI Service Engine Client Logs ? If yes, please enter directory path, else
  leave it blank
[Please enter the Controller IP for this SE to connect 10.10.24.89
[Enable inband management interface for this Service Engine (i.e. Use Management interface for data traffic as
well) [yes/no] no

Run SE           : Yes
SE Cores         : 2
Memory(MB)      : 2048
Disk(GB)        : 10
Run Controller   : No
Controller IP this SE will communicate with: 10.10.24.89

```

SE In-band Management Configuration Using Linux Server Cloud

The section details deployment of an SE with in-band management enabled or disabled using the Linux Server Cloud configuration on the Controller UI.

A new server added to the Linux Server configuration will inherit the cloud-level in-band management property. This property can be explicitly modified per host to override the cloud-level value.

Edit Cloud: Default-Cloud

Select Cloud | Linux Server Configuration

SSH User * ⓘ
Select SSH User

License Type ⓘ
Cores

☐ Use Static Route for VIP Placement ⓘ ☒ Use Inband Management ⓘ

IPAM Profile ⓘ
Select IPAM Profile

DNS Profile ⓘ
Select DNS Profile

Directory Path for SE Usage
☒ Auto-select path ☐ Single path for all logs ☐ Separate paths for System and Client logs

• Server List •

Host IP Address *	DPDK *	Number of Cores *	Memory (GB) *
10.10.24.89	Yes	All	All

☒ Inband Management

[Add new server](#)

Flag to notify the SE's in this cloud have an inband management interface, this can be overridden at SE host level by setting host_attr attr_key as SE_INBAND_MGMT with appropriate value

Installing NSX Advanced Load Balancer in OpenStack

5

NSX Advanced Load Balancer OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. NSX Advanced Load Balancer OpenStack allows you to deploy virtual machines and other instances that handle different tasks for managing a cloud environment instantly. It makes horizontal scaling easy, which means that tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning up more instances. OpenStack provides Infrastructure as a Service (IaaS).

This chapter includes the following topics:

- [Overview](#)
- [Considerations for Deploying NSX Advanced Load Balancer with OpenStack](#)
- [Installing NSX Advanced Load Balancer in OpenStack](#)
- [Configuring the NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Deployment Options](#)

Overview

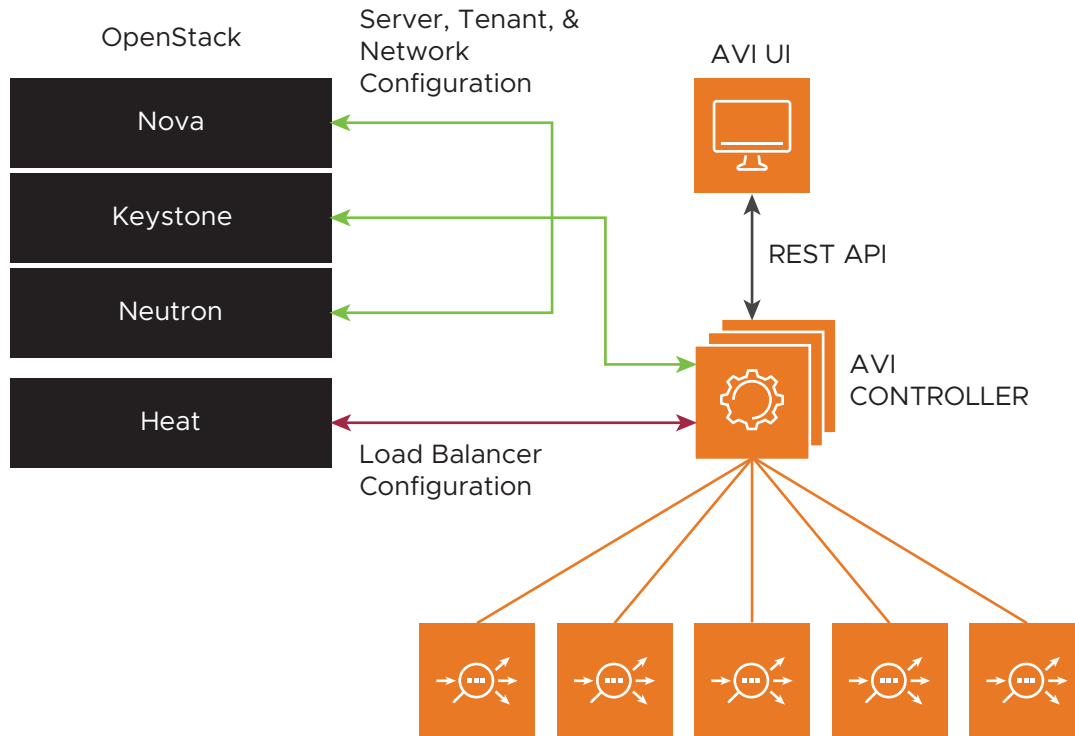
NSX Advanced Load Balancer integrates with OpenStack infrastructure components to provide centralized automation, monitoring, and management of application discovery and delivery.

NSX Advanced Load Balancer integrates with the following OpenStack services:

- **Keystone** - The NSX Advanced Load Balancer uses Keystone API to authenticate any user accessing its API. Also, when an OpenStack user logs in, the Controller can automatically import tenant / project and role information from Keystone to provide appropriate privileges on Controller.
- **Glance** - The NSX Advanced Load Balancer uses Glance for storing Service Engine image.
- **Nova** - The NSX Advanced Load Balancer uses Nova API to automatically create and destroy application delivery Service Engine as needed to support High Availability and guarantee performance.
- **Neutron** - The NSX Advanced Load Balancer uses Neutron API to plug Service Engines into right Neutron networks for receiving and sending application traffic.

- Heat - OpenStack administrators can optionally install NSX Advanced Load Balancer Heat package on the Heat Engine servers to expose all Controller API resource types for users to use in their heat templates. In contrast to LBaaS (v1 or v2) resource types, NSX Advanced Load Balancer Heat resource types expose significantly advanced features.

The following is the flowchart for NSX Advanced Load Balancer integration with OpenStack:



Considerations for Deploying NSX Advanced Load Balancer with OpenStack

This section explains deployment modes, prerequisites, protocols, and ports details.

Deployment Modes

NSX Advanced Load Balancer can be deployed into an OpenStack cloud in one of the following modes. These modes differ depending on whether the NSX Advanced Load Balancer Controller and SEs are placed in the same OpenStack tenant, and whether Neutron LBaaS API or NSX Advanced Load Balancer API is used to create the load balancers.

- Single-tenant mode
- NSX Advanced Load Balancer-managed mode

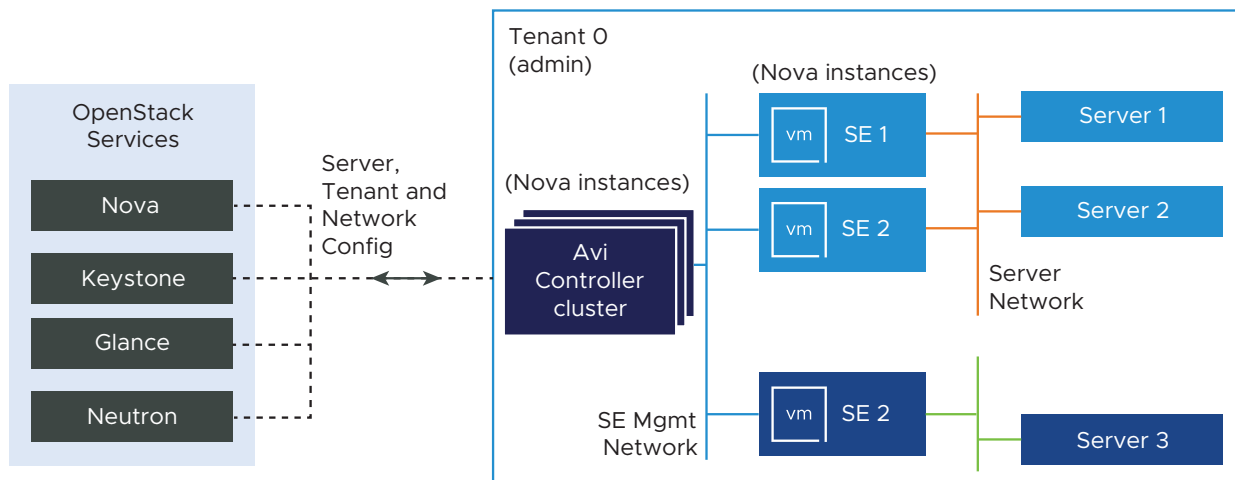
The NSX Advanced Load Balancer-managed option is recommended for its ease of use and advanced feature accessibility.

The following table compares each deployment mode:

	Single-tenant Mode	NSX Advanced Load Balancer-managed Mode
Require administrator privileges for cloud?	No	Yes
Managed by tenant user	No	Yes
Automated tenant creation	NA	Yes
Advanced load-balancing features available	Yes	Yes
Analytics service	Yes	Yes

Deploying Single Tenant Mode

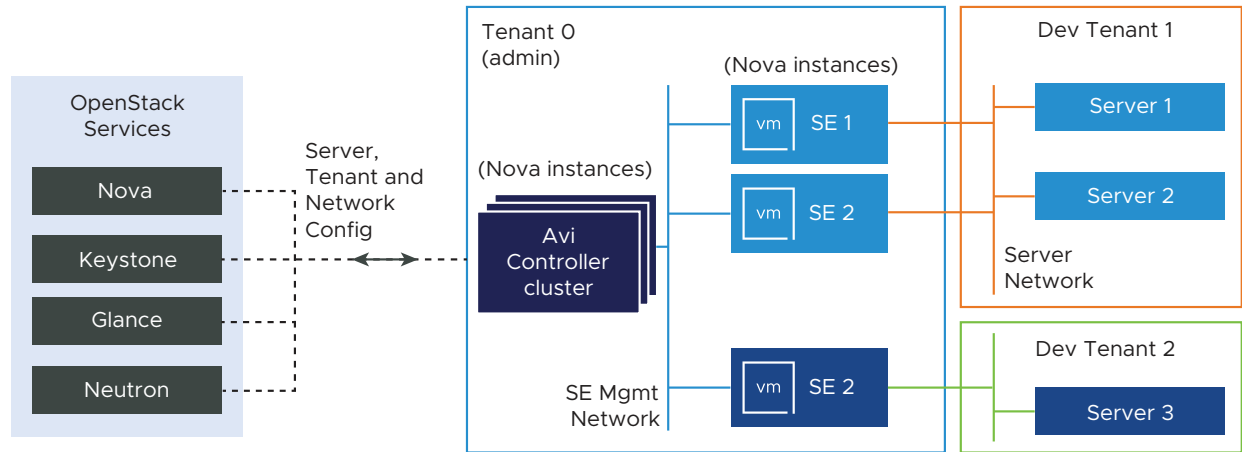
The NSX Advanced Load Balancer Controller and the SEs are deployed together in the same tenant. The Controller has administrator privileges within the tenant. Tenant users with administrator privileges within the tenant can install and manage the NSX Advanced Load Balancer. Use this deployment mode if you do not have administrator privileges for the cloud.



In single-tenant mode, the Controller and SEs are installed in the same tenant, and have member privileges for that tenant. The member privilege grants the Controller full access to the tenant so that it can automatically spin-up and spin-down an SE. Each tenant is responsible for installing and operating the NSX Advanced Load Balancer.

Deploying NSX Advanced Load Balancer-managed Mode

The Controller and SEs are installed in separate tenants. The Controller has administrator privileges for the cloud and can manage SEs that are in different tenants. A tenant administrator can log onto the Controller to manage the infrastructure resources within the administrator's own tenant but cannot access the resources within other tenants. The tenant administrator can configure and manage load balancing services through the Controller web interface or through the NSX Advanced Load Balancer REST API.



NSX Advanced Load Balancer-managed mode provides tenant users with its advantages, without the need to deploy or maintain it. The cloud administrator takes up this work. The Controller and SEs in the administrative tenant are shared by other tenants. Users of those tenants are able to secure and optimize their applications using the NSX Advanced Load Balancer resources that reside in the administrative tenant.

Although you can use an existing tenant instead of creating a new one, it is recommended to create a new tenant for easy maintenance.

Deployment Prerequisites

The physical and software requirements differ depending on the deployment mode.

Software Requirements

For details on supported software version, refer to [Software Requirements for OpenStack Cloud](#) section.

The NSX Advanced Load Balancer image is available in `qcow2` (QEMU Copy ON Write) format or raw image of the Controller and SEs. The SE software is embedded in the Controller image and does not require separate installation. In case of OpenStack generic cloud (with NSX Advanced Load Balancer Cloud Connector), the Controller pushes `qcow2` image for SE towards OpenStack Glance. In case of a no-access cloud, you need to download the `qcow2` image for SE and then manually upload to OpenStack Glance.

Note In KVM/OpenStack, the Service Engines have a limitation of 24. In order to support 24 vNICs, Service Engines should be configured with at least 3GB of memory.

OpenStack Network Configuration for NSX Advanced Load Balancer Controller Cluster

This section explains how to configure a cluster in NSX Advanced Load Balancer for an OpenStack cloud. To provide Controller High availability (HA), add two additional Controller nodes to create a three-node Controller cluster.

Prerequisites for Cluster Deployment

The following are the prerequisites for OpenStack network configuration:

- A Neutron port is created and is available for cluster VIP
- A floating IP is available for Neutron port

Deploying an NSX Advanced Load Balancer Controller Cluster

The following steps are for creating OpenStack floating IP and binding that with the cluster IP:

- Write Mode
- No-Access Mode

For more details on deploying a cluster, see [Deploying an NSX Advanced Load Balancer Controller Cluster](#) section.

Deploying an NSX Advanced Load Balancer Controller Cluster for Write Mode

You can deploy the Controller Cluster for Write mode as follows:

Access OpenStack Horizon CLI and list the OpenStack network list. This indicates the configured requisite networks.

```
root@openstack-mitaka:/root# openstack network list
+-----+
+-----+
| id | name |
subnets |
+-----+
+-----+
| 10a514a3-d843-499d-80fd-28274d4a4912 | webserver-net |
3ebfb2ef-9b47-44f7-9da5-5245e1d0ed53 192.168.10.0/24 |
| 5dd0b1cb-ebba-4ff9-84fd-74dcf13c7f86 | client-net |
a9a00d61-6ee8-4fac-80df-4e0bb8c8b4f3 192.168.11.0/24 |
| c1c045f5-2d0f-43e3-ab43-55f990cde9b7 | provider1 |
1b65c0da-38c7-4c85-88a9-30c52c6a4558 10.130.128.0/18 |
| dd9dab27-9228-4765-96f2-d56194136ba0 | avimgmt | 5785c1cf-
a222-4b0a-9343-003153f37a65 172.16.0.0/24 |
+-----+
+-----+
```

Create a floating IP, using the command `openstack floating ip create provider1`, where `provider1` is the network used.

```
root@openstack-mitaka:/root# openstack floating ip create provider1
```

Get the `port-id` for cluster IP as follows:

```
openstack port list -c ID -c 'Fixed IP Addresses'|grep 172.16.0.65
95665123-64a4-453a-abde-70fdb3d2ae2a| ip_address='172.16.0.65', subnet_id='5785c1cf-
a222-4b0a-9343-003153f37a65'
```


Associate the cluster IP with the floating IP. Using the port-id from the command above (95665123-64a4-453a-abde-70fdb3d2ae2a in this case), associate it with the floating IP created above.

```
root@openstack-mitaka:/root# openstack floating ip set --port 95665123-64a4-453a-abde-70fdb3d2ae2a 4ec57a12-7357-461a-80f6-d87ae7536335
```

```
+-----+
| Field          | Value                                     |
+-----+
| description    |                                         |
| fixed_ip_address | 172.16.0.65                             |
| floating_ip_address | 10.130.170.86                           |
| floating_network_id | c1c045f5-2d0f-43e3-ab43-55f990cde9b7 |
| id             | 4ec57a12-7357-461a-80f6-d87ae7536335 |
| port_id        | 95665123-64a4-453a-abde-70fdb3d2ae2a |
| router_id      | 2d3b93a2-7804-4841-90c4-be15b148d099 |
| status         | ACTIVE                                  |
| tenant_id      | 904fb201a92f443297bffca3b354d52d    |
+-----+
```

Add the cluster IP and the secondary IP for the cluster leader. For instance, 172.16.0.65 is the cluster IP which is added as the secondary IP for the cluster leader.

```
root@172-16-0-66:~# ip a
eth0: (BROADCAST,MULTICAST,UP,LOWER_UP) mtu 1500 qdisc mq state UP group default qlen 1000
link/ether 00:50:56:bd:5a:0f brd ff:ff:ff:ff:ff:ff
inet 172.16.0.66/24 brd 172.16.0.255 scope global eth0
valid_lft forever preferred_lft forever
inet 172.16.0.65/32 scope global eth0:1 Cluster IP
```

Deploying an NSX Advanced Load Balancer Controller Cluster for No-Access Mode

For OpenStack No-Access cloud type, you need to manually configure the AAP entries using the following command.

```
root@openstack-mitaka:/root# openstack port set --allowed-address ip-address=172.16.0.133 Controller_Port
```

For instance,

```
root@openstack-mitaka:/root# openstack port set --allowed-address ip-address=172.16.0.133 d0bf0bda-02e2-46bf-abd2-0d05cc4654df
root@openstack-mitaka:/root# openstack port show d0bf0bda-02e2-46bf-abd2-0d05cc4654df
+-----+
+-----+
| Field          | Value                                     |
+-----+
| admin_state_up |                                         |
+-----+
```

```

True
| allowed_address_pairs      | {"ip_address": "172.16.0.131", "mac_address":
"fa:16:3e:47:6b:70"}
| binding:host_id            | openstack-
mitaka
| binding:profile            |
{}
| binding:vif_details        | {"port_filter":
true}
| binding:vif_type           |
bridge
| binding:vnic_type           |
normal
| created_at                 |
2018-01-12T13:58:02
| description                 |
|
| device_id                  | 2adedfc3-75d6-4296-ad18-
bfc38873485c
| device_owner                |
compute:nova
| extra_dhcp_opts             |
|
| fixed_ips                  | {"subnet_id": "5785c1cf-a222-4b0a-9343-003153f37a65",
"ip_address": "172.16.0.133"}
| id                          | d0bf0bda-02e2-46bf-
abd2-0d05cc4654df
| mac_address                 |
fa:16:3e:47:6b:70
| name                       |
|
| network_id                 | dd9dab27-9228-4765-96f2-
d56194136ba0
| port_security_enabled       |
True
| security_groups             | 3cc1092e-538c-4ff7-b4ac-
eeff84731f75
| status                     |
ACTIVE
| tenant_id                  |
904fb201a92f443297bffca3b354d52d
| updated_at                 |
2018-01-12T14:19:06
+-----+
+-----+

```

Create the neutron port for the VIP by using the following command:

```

openstack port create --network "neutron_network_name" --
allowed-address mac-address="fa:16:3e:52:81:03",ip-address="172.16.0.63" --allowed-
address mac-address="fa:16:3e:52:81:04",ip-address="172.16.0.64" --allowed-address mac-
address="fa:16:3e:52:81:06",ip-address="172.16.0.66" --fixed-ip ip-address="172.16.0.65" --
project "904fb201a92f443297bffca3b354d52d"

```

For instance,

```
openstack port create --network "neutron_network_name" --
allowed-address mac-address="controller_mac1",ip-address="controller_ip1" --allowed-
address mac-address="controller_mac2",ip-address="controller_ip2" --allowed-address mac-
address="controller_mac3",ip-address="controller_ip3" --fixed-ip ip-address="cluster_ip" --
project "project-id"
```

Note When the leader Controller fails (or reboots), a follower Controller will take over the cluster IP (in this case 172.16.0.65), and the mapping between floating IP (10.130.170.86) and cluster IP (172.16.0.65) will not change. Therefore, without intervention, the floating IP and cluster IP association will work as expected.

Installing NSX Advanced Load Balancer in OpenStack

You can install NSX Advanced Load Balancer in Openstack.

Installing Single Tenant and NSX Advanced Load Balancer-Managed Mode

This section explains the initial setup and installation process for single tenant mode and NSX Advanced Load Balancer-managed mode.

Installing Process

This section explains the installation process for single-tenant mode. You can create a tenant for the Controller and the SEs.

To create a tenant for the Controller and SEs:

- 1 Log in to the **OpenStack Horizon** dashboard with an account that has cloud administrator privileges and navigate to **Identity > Projects**.
- 2 Click **New Project** and follow the wizard's instructions.
- 3 To deploy NSX Advanced Load Balancer , specify a project name, (NSX Advanced Load Balancer-tenant, for instance) and click the **Project Members** tab. You can add a user account to **Project Members** and assign the admin role to the account. Click the **Quota** tab and modify the maximum resources. These settings allow for three Controllers (for redundancy), up to 1000 SEs and some other managerial instances, if required, as shown below.

Figure 5-1. Quota Tab

Project Information *	Project Members	Quota *
Metadata Items *	<input type="text" value="2400"/>	
VCPUs *	<input type="text" value="440"/>	
Instances *	<input type="text" value="210"/>	
Injected Files *	<input type="text" value="5"/>	
Injected File Content (Bytes) *	<input type="text" value="10240"/>	
RAM (MB) *	<input type="text" value="409600"/>	
Security Groups *	<input type="text" value="220"/>	
Security Group Rules *	<input type="text" value="2200"/>	
Floating IPs *	<input type="text" value="50"/>	
Networks *	<input type="text" value="10"/>	
Ports *	<input type="text" value="1920"/>	
Routers *	<input type="text" value="1"/>	
Subnets *	<input type="text" value="10"/>	

Creating Multiple Flavors of Controller Image

You can create multiple flavors of NSX Advanced Load Balancer as follows:

- 1 In the **Horizon** dashboard, navigate to **Admin > system > Flavors** and click **Create Flavor**.
- 2 Create an appropriate flavor for the Service Engine.
- 3 Create appropriate flavor for the Controller.

You can manually configure the flavor if you want to use flavors other than the recommended ones using CLI. Also ensure that you check minimum and recommended resources required for Service Engine.

Note The OpenStack flavor name should be specified and not the flavor ID or UUID.

To install single tenant and NSX Advanced Load Balancer-managed mode:

Uploading the Controller Image

You can upload the Controller image by copying the Controller `qcow2` image onto your hard drive. Then navigate to **Project > Images** in the **Horizon** dashboard. Click **Create Image** and fill out the form. Use at least these resource allocations.

Creating Management Network

A management network is required for communication between the Controller and the SEs. An existing network can be used but a dedicated management network is recommended.

You can create management network by navigating to **Network > Networks** on the **Horizon** dashboard. Click **Create Network** and follow the wizard's instructions.

- 1 Connect the network to your Neutron router by navigating to **Network > Routers**. Click the router in the **Name** column to add an interface to the network.
- 2 Click the **Interfaces** tab and click **Add Interface** to add the interface details.

Creating Security Group

A security group is required to allow the Controller and SEs to exchange management traffic. The group specifies the protocol ports for which traffic will be allowed. For ingress traffic, the group must allow these ports. For egress traffic, the group can allow all ports.

The NSX Advanced Load Balancer Controller automatically creates a security group for the SEs.

You can create a security group (for instance, `Avi-mgmt-sg`) and allow management traffic by navigating to **Project > Access & Security** on the **Horizon** dashboard. Click **Create Security Groups** and add rules, for instance, `192.168.10.0/24` is the management network.

Deploying Controller and Assigning it a Floating IP

You can assign a floating IP address to the Controller by navigating to **Project > Compute > Access & Security** on the **Horizon** dashboard. If no floating IP address is available, click **Allocate IP to Project**. If a floating IP address is already available, you can associate it with the NSX Advanced Load Balancer Controller instance.

Performing Initial Setup

This section explains the steps to perform initial configuration of the Controller using its deployment wizard. You can change or customize settings following initial deployment using the Controller's web interface.

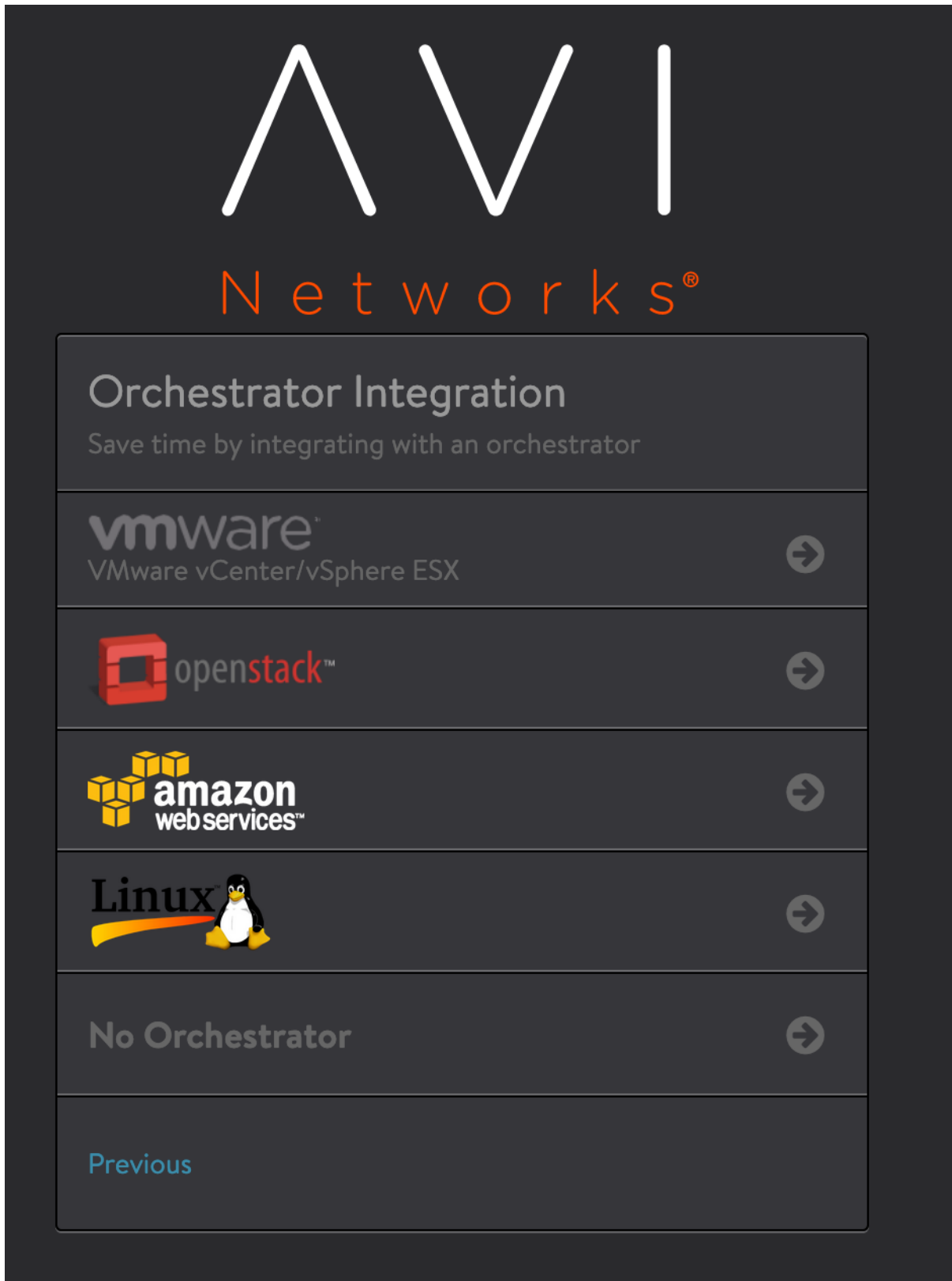
While the system is booting up, a blank web page or 503 status code may appear. In this case, wait for 5 to 10 minutes; then follow the instructions for the setup wizard.

Procedure

- 1 Configure basic system settings, such as administrator account details, DNS and NTP server information along with email and SMTP information.

- 2 Set the **Infrastructure Type** to **OpenStack** and specify the OpenStack settings.

Figure 5-2. Infrastructure



- Specify the tenant user credentials. If you are using Keystone V3 and want to provide a user in the non-default domain, then use the notation **user@domain-name** in the **Username** field.
- If you create a username test as a Keystone v3 user in a domain named default, then you should explicitly specify **test@testdomain** while logging into the Controller. If the domain name is not specified, Keystone looks for a domain with UUID testdomain and not the name testdomain. Since no domain with a UUID of testdomain exists, Keystone fails, thereby returning the error as invalid user/password.
- Use the full value in the **Keystone Auth URL** field. NSX Advanced Load Balancer determines the Keystone API version automatically. When the auth URL is a secure URL (HTTPS), the system will display an option to either allow or disallow self-signed certificates. You can disable the checkbox in a production environment, since OpenStack services should use proper, trusted certificates.
- Enable the **Keystone Auth** option and click **Next** button.

Figure 5-3. OpenStack Login

AVI
Networks®

OpenStack Login
Please provide credentials

Username*
admin

Password*
.....

Keystone Auth URL*
https://10.10.16.82:5000/v3

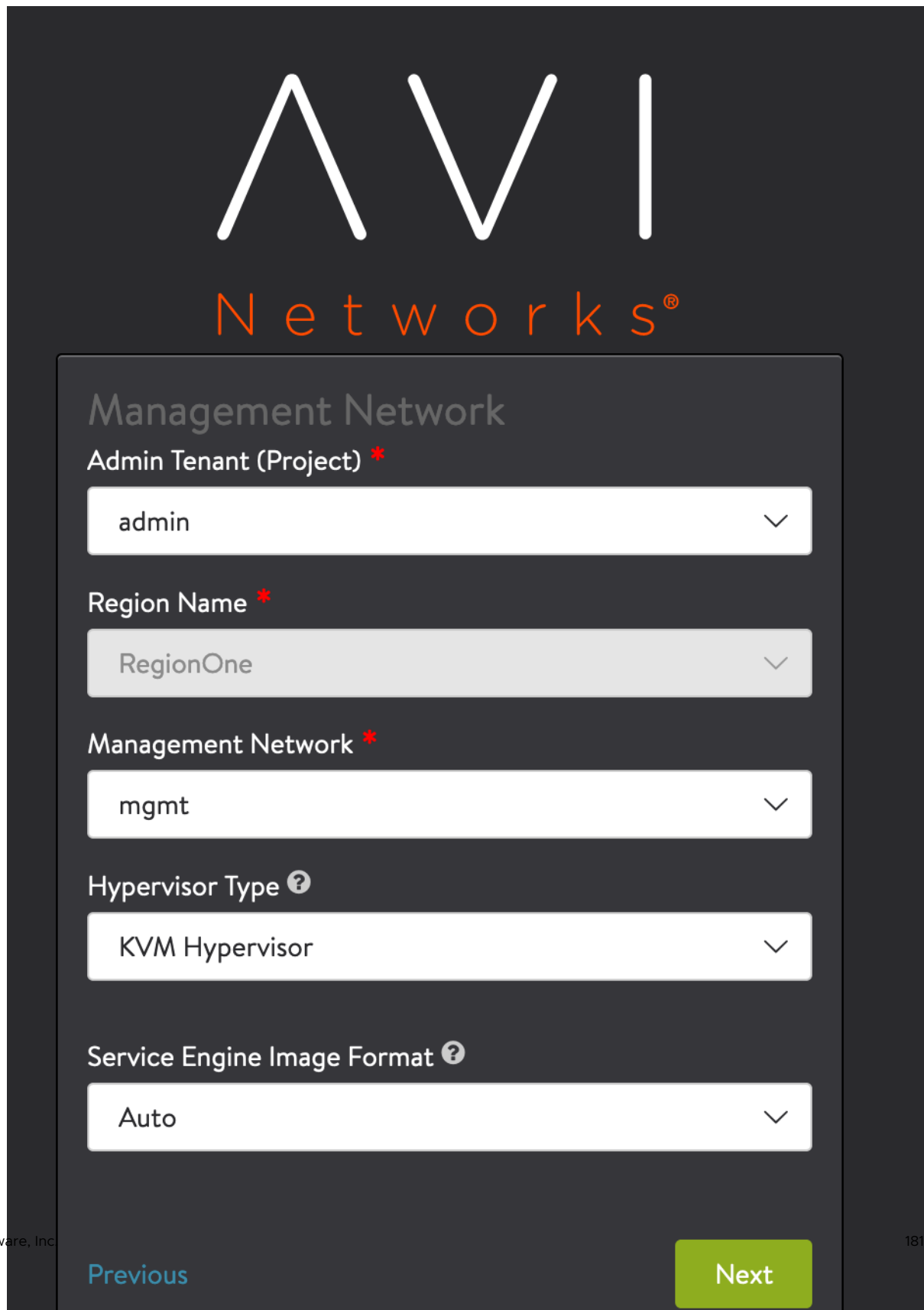
☒ Allow Self-signed Certificates ?

☒ Use Keystone Auth ?

[Previous](#) [Next](#)

- 3 In **Management Network** window, select a tenant. In this deployment, it should be the same tenant into which the Controller is deployed. Choose the management network created previously. Also specify the **Hypervisor Type** and **Service Engine Image Format**. Click **Next** button.

Figure 5-4. Controller Setup Management



AVI
Networks®

Management Network

Admin Tenant (Project) *

admin

Region Name *

RegionOne

Management Network *

mgmt

Hypervisor Type ?

KVM Hypervisor

Service Engine Image Format ?

Auto

Previous Next

- 4 In **Keystone Role Mapping** window, select an NSX Advanced Load Balancer user role as the default user role. If a NSX Advanced Load Balancer user logs in with valid Keystone credentials, but with a role that does not have the same name as any of the user roles defined on the Controller, the default role is assigned to the user. You can skip this option to disallow access for any user who does not have a role that is defined on the Controller. Click **Next** button.

Figure 5-5. Keystone Mapping

AVI
Networks®

Keystone Role Mapping

Keystone Role Mapping to Avi Roles

Keystone Role ? Avi Role ?

* Tenant-Adm... ▼

+ Add Role Mapping

[Previous](#) **Next**

- 5 You can configure tenant settings by navigating to **Administration > Settings > Tenant Settings**. Click edit icon. The **Tenant Settings Config** window is displayed.

Figure 5-6. Tenant Settings Config

Tenant Settings Config

IP Route Domain ⓘ

☐ Per tenant IP route domain. ☒ Share IP route domain across tenants.

Service Engines Context ⓘ

☐ Service Engines are managed within the tenant context, not shared across tenants.

☒ Service Engines are managed within the provider context, shared across tenants.

☒ Tenant has Read Access to Service Engines

☐ Tenant has No Access to Service Engines

Cancel Save

Specify the following details:

IP Route Domain — This options allows you to select tenant’s IP route domain.

- **Per tenant IP route domain** — If you select this option each tenant gets its own routing domain that is not shared with any other tenant.
- **Share IP route domain across tenants** — If you select this option all tenants share the same routing domain.

Service Engines Context — This option controls the ownership of Service Engines. Service Engines can either be exclusively owned by each tenant or owned by the administrator and shared by all tenants. When Service Engines are owned by the administrator, each tenant can have either read access or no access to their Service Engines. You can select one of the options.

After specifying the necessary details, click on **Save** option.

Integrating Neutron SDN Plugin

NSX Advanced Load Balancer integrates with the following Neutron SDN plugins to provide VIP placement and floating IP (FIP) association to VIP.

Contrail SDN Using NSX Advanced Load Balancer UI

During cloud configuration, select the **Integration with Contrail** checkbox and provide the endpoint URL of Contrail VNC API-server. The Keystone credentials from the OpenStack configuration will be used to authenticate with the API-server service.

Note Contrail-Interface-IP is handled gracefully by NSX Advanced Load Balancer. Creating and editing the cloud should be left intact while integrating Contrail SDN under **Network Settings**.

Contrail SDN Using NSX Advanced Load Balancer CLI

The following is the CLI command for contrail SDN:

```
: > configure cloud oscontrail
: cloud> vtype cloud_openstack
: cloud> openstack_configuration
: cloud:openstack_configuration>
: cloud:openstack_configuration> privilege write_access
: cloud:openstack_configuration> username admin
: cloud:openstack_configuration> password xxxyyyyzzz
: cloud:openstack_configuration> admin_tenant admin
: cloud:openstack_configuration> mgmt_network_name avi-mgmt
: cloud:openstack_configuration> region RegionOne
: cloud:openstack_configuration> use_keystone_auth
: cloud:openstack_configuration> import_keystone_tenants
: cloud:openstack_configuration> no use_admin_url
: cloud:openstack_configuration> auth_url http://172.16.11.50:5000/v2.0
: cloud:openstack_configuration> no neutron_rbac
: cloud:openstack_configuration> contrail_endpoint http://10.10.10.100:8082
: cloud:openstack_configuration> role_mapping os_role * avi_role Tenant-Admin
New object being created
: cloud:openstack_configuration:role_mapping> save
: cloud:openstack_configuration> save
: cloud> save
```

Installing NSX Advanced Load Balancer OpenStack in No-Access OpenStack Cloud

This section lists the steps need to install NSX Advanced Load Balancer into an OpenStack cloud for the case in which NSX Advanced Load Balancer has no access to OpenStack, the orchestrator.

In No-Access mode, NSX Advanced Load Balancer has no access to OpenStack as an orchestrator. Adding, removing, or modifying properties of a Service Engine requires an administrator to manually perform the changes. Servers and networks cannot be auto-discovered by NSX Advanced Load Balancer; they must be manually configured.

To install NSX Advanced Load Balancer in No-Access OpenStack Cloud:

Prerequisites

NSX Advanced Load Balancer should be instantiated in the No-Orchestration mode.

Procedure

- 1 Create an OpenStack No-Access cloud.

Figure 5-7. OpenStack No Access Mode

Q			Create
<input type="checkbox"/> Name ^	Type	Status	
<input type="checkbox"/> AWS-Cloud	AWS	●	✎ ⌕ ⚙ +
<input type="checkbox"/> Default-Cloud	VMware vCenter/vSphere ESX	●	✎ ⌕ ⚙ ⚙ +
<input type="checkbox"/> GCP-Cloud	Linux Server	●	✎ ⌕ ⚙ +
<input type="checkbox"/> Openstack-Cloud	OpenStack	●	✎ ⌕ ⚙ ⚙ +
<input type="checkbox"/> Openstack-No-Access-Cloud	None	●	✎ ⌕ ⚙ -
IP Address Management DHCP VIP Static Routes Disabled Route Placement Preference Interface Routes Static Routes			

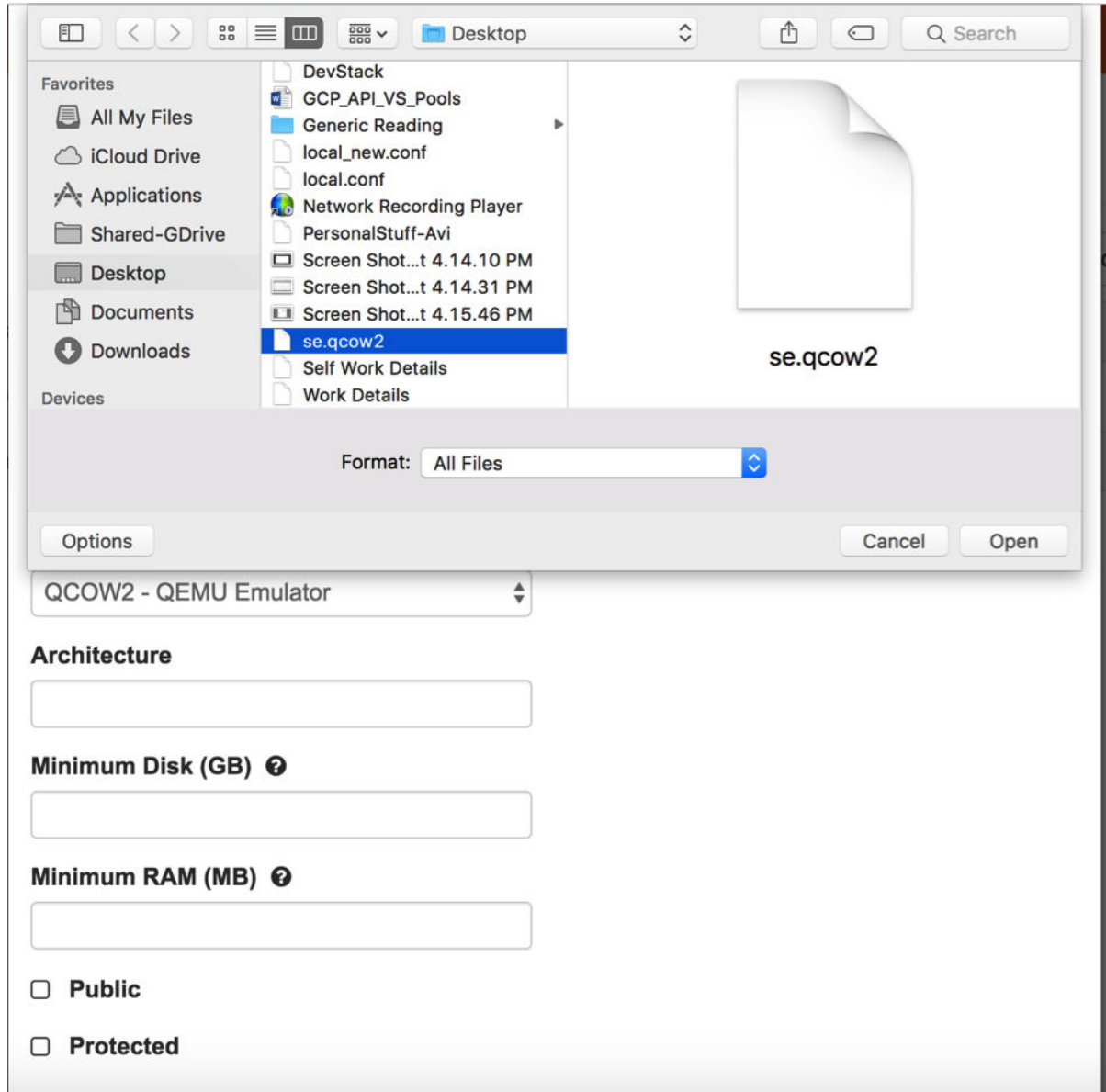
- 2 Select the IP address management as **Use DHCP** in **DHCP Settings** tab.
- 3 Download **SE qcow2** image, as this will be pushed to Glance.

Figure 5-8. Download SE QCOW2

Infrastructure				Dashboard	Clouds	Service Engine	Service Engine Group	Networks	Routing	GSLB	?	admin(admin)
Q												Create
<input type="checkbox"/> Name ^	Type	Status										
<input type="checkbox"/> AWS-Cloud	AWS	●	✎ ⌕ ⚙ +									
<input type="checkbox"/> Default-Cloud	VMware vCenter/vSphere ESX	●	✎ ⌕ ⚙ ⚙ +									
<input type="checkbox"/> Openstack-Cloud	OpenStack	●	✎ ⌕ ⚙ ⚙ +									
<input type="checkbox"/> Openstack-No-Access-Cloud	None	●	✎ ⌕ ⚙ +									
			Ova Qcow2									

- 4 Log in to the OpenStack instance under the respective tenant ('admin' in this case) and click **Create** . Select the format as **QCOW2** and provide the image file for the SE **QCOW2** that was downloaded.

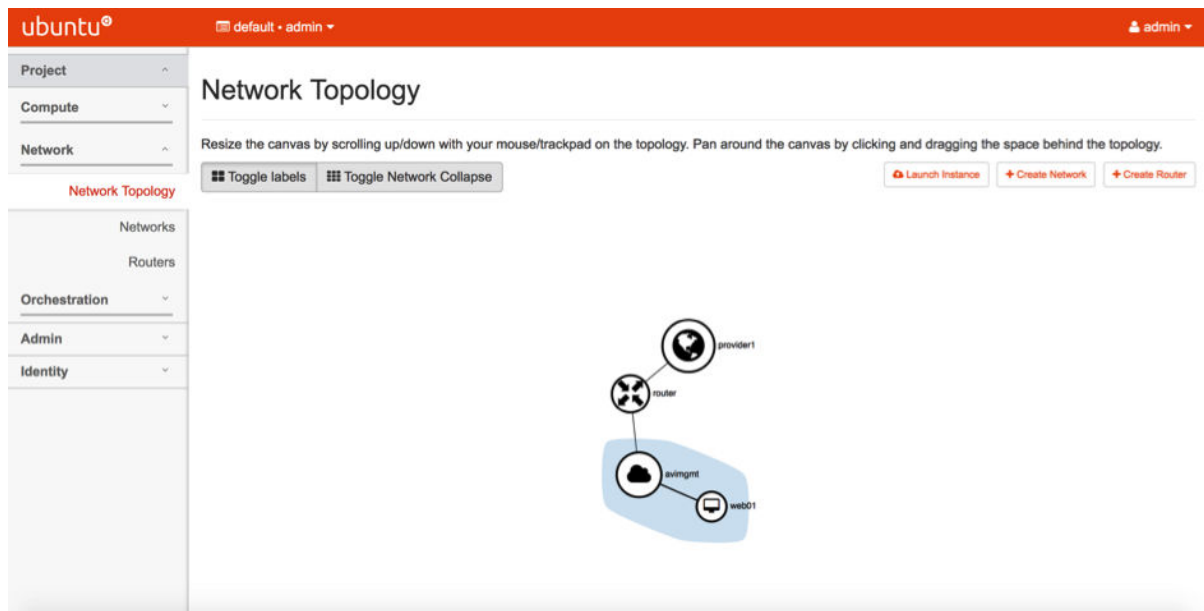
Figure 5-9. QCOW2 Image



- 5 Upload the `se.qcow2` image to Glance.
 - This is needed only if there is no existing network that can be used as the NSX Advanced Load Balancer network.
 - This network will be used by SEs to communicate with the Controller. Therefore, either create a new network or use an existing network and ensure that VMs created on that network can reach the Controller.

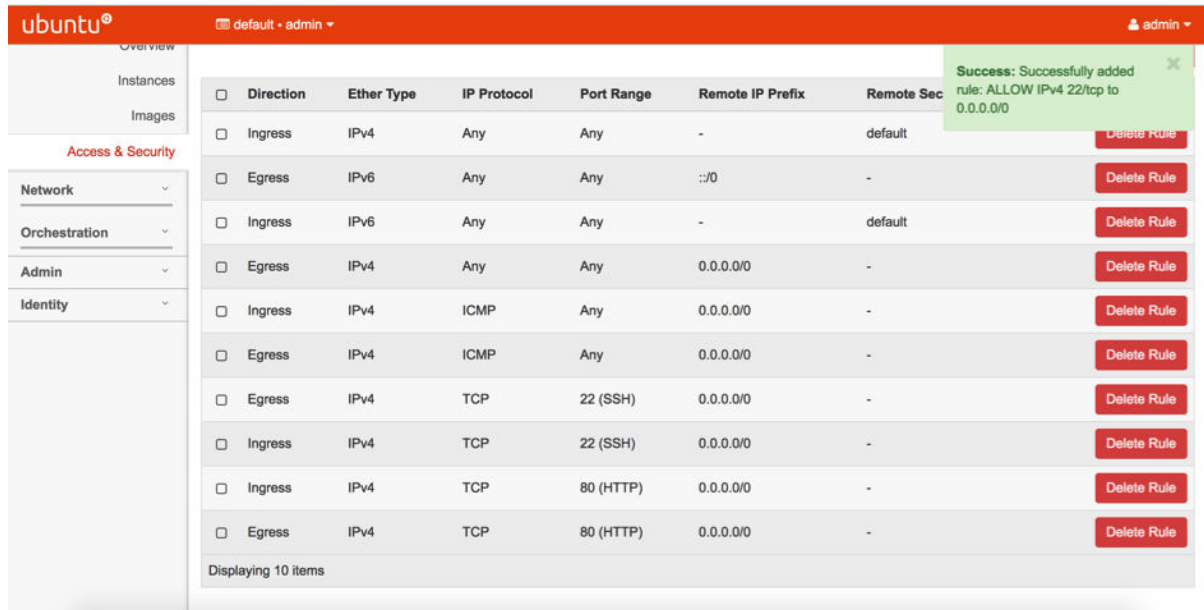
- 6 Specify the network name in **Network** tab. Also provide appropriate subnet to the network in **Subnet** tab. Check **Enable DHCP** box in **Subnet Details** tab and create the network.
 - This step is required only if a new external network needs to be created. Create the network that will be the outbound network and will provide floating IP access. For instance, you can provide the **Network Name** as **provider1** in **Network** tab and provide the other details in the respective tabs.
 - This step is required only if a new router needs to be created for an external connectivity. You can create a router by providing the router name, admin state and external network details.
- 7 Additionally, you can deploy a web server in the avimgmt network to do tests. This could be a server of an OS type and the network topology would look something like this:

Figure 5-10. Network Topology



- 8 Create a security group as below and associate it with the Service Engine to make sure that ICMP traffic and SSH and HTTP traffic is allowed.

Figure 5-11. Security Group associated with SE



- 9 Create an NSX Advanced Load Balancer SE Instance. SEs can be created using heat-templates as well. For more information on this, see [Creating Service Engine using Heat-Templates in No-Access OpenStack Cloud](#).

Figure 5-12. launch-instance

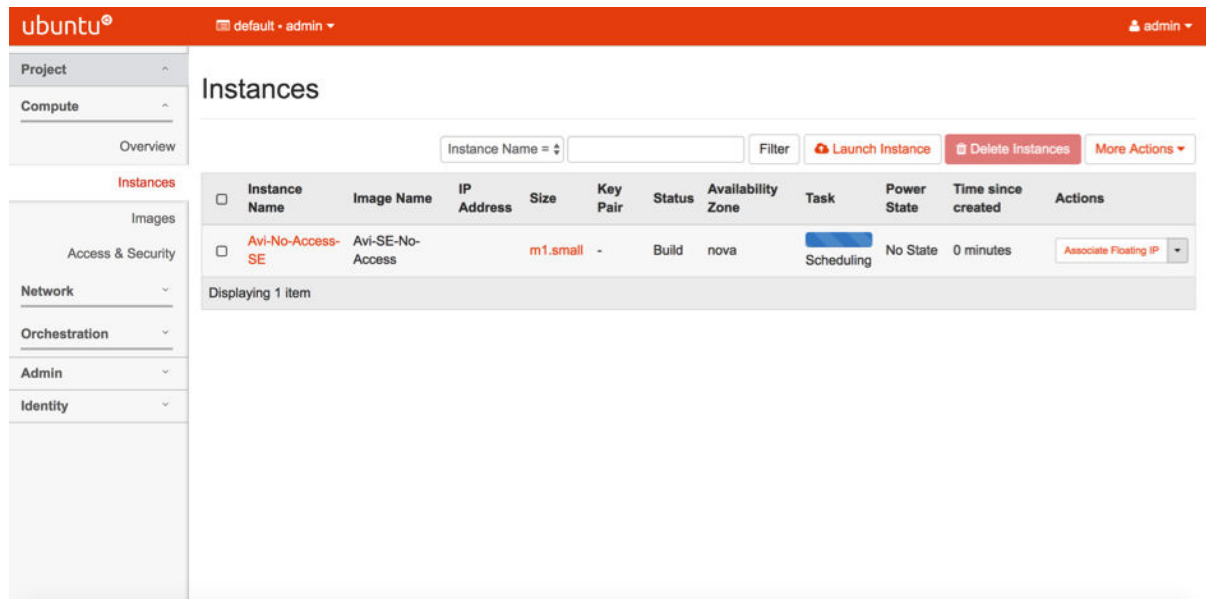
The screenshot shows the "Launch Instance" dialog box. The left sidebar contains the following tabs: Details, Source, Flavor, Networks, Network Ports, Security Groups, Key Pair, Configuration, and Metadata. The main area contains the following fields and information:

- Instance Name ***: Avi-No-Access-SE
- Availability Zone**: nova
- Count ***: 1
- Total Instances (10 Max)**: A progress indicator showing 10% usage.
- Usage Legend**:
 - 0 Current Usage
 - 1 Added
 - 9 Remaining

At the bottom, there are buttons for "Cancel", "< Back", "Next >", and "Launch Instance".

- 10 Select the appropriate `qcw2` image for the SE that needs to be instantiated from the **Source** tab.
- 11 Select the respective flavor for the SE from the **Flavor** tab. In this case it would be **m1.small**.
- 12 Select the **avimgmt** network from the **Networks** tab for instantiating the SE.
- 13 The SE gets spawned as below:

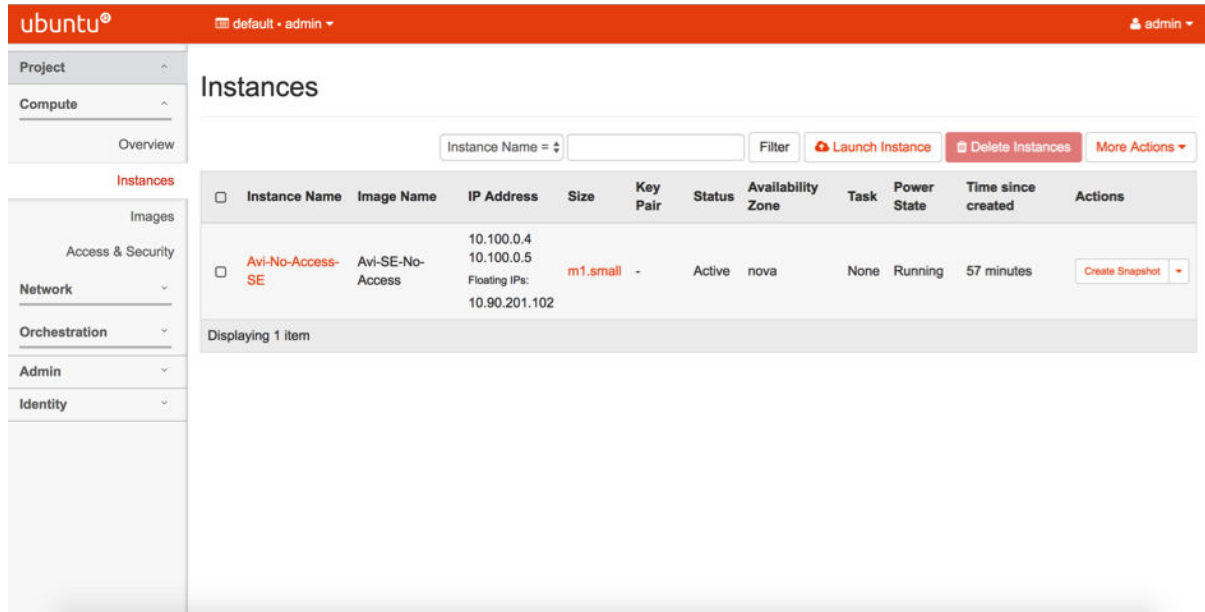
Figure 5-13. Spawned SE



- 14 Associate a floating IP to the instance. This step is required only where SEs are not reachable directly.
- 15 Attach another interface to the SE. This would be the data vNIC.

- 16 The SE gets created with one management vNIC and one data vNIC, the latter associated with a floating IP.

Figure 5-14. SE with vNIC and Floating IP



- For the SE to connect to the Controller, copy the token for the SE from NSX Advanced Load Balancer UI (outlined in Installing NSX Advanced Load Balancer for VMware vCenter) for the respective cloud and run the script at `/opt/avi/scripts/init_system.py` on the SE, which would then ask for the Controller IP and the token (the token expires in 60 minutes and is for a single SE). You need root access privileges to run this script.

- ```
root@Avi-Service-Engine:/opt/avi/scripts# ./init_system.py -h
usage: init_system.py [-h] -c CONTROLLER [-d] [-i MGMT_IP] [-m MGMT_MASK]
[-g GATEWAY] [-t TOKEN] [-r]
optional arguments:
-h, --help show this help message and exit
-c CONTROLLER, --controller CONTROLLER
Controller IP address.
-d, --dhcp DHCP
-i MGMT_IP, --mgmt-ip MGMT_IP
IP address for Management Interface (eg.
192.168.10.10)
-m MGMT_MASK, --mgmt-mask MGMT_MASK
Subnet mask for Management interface (eg. 24 or
255.255.255.0)
-g GATEWAY, --gateway GATEWAY
Default gateway
-t TOKEN, --token TOKEN
```

```
Auth token generated in the Controller for this SE
-r, --restart Restart SE for changes to take effect
root@Avi-Service-Engine:/opt/avi/scripts# ./init_system.py -c 172.16.0.10 -d -i
172.16.0.7 -m 255.255.255.0 -g 172.16.0.1 -t c708a2cd-69e2-4057-923d-a09de94914f6 -r
```

Reboot the SE for it to connect to the Controller.

- 17 Wait for the NSX Advanced Load Balancer SEs to show up in the UI's **Infrastructure > Service Engine** list under the respective cloud.
- 18 Edit each SE and enable DHCP for each data network.
- 19 Create a virtual service and choose an IP address from the data network.

Since this is a no-Access cloud, you cannot configure a 'floating VIP' in the virtual service configuration. For the Controller to communicate with OpenStack Nova to assign an allocated floating IP to virtual IP address, you need to create a binding association as shown below through the CLI for the Neutron port with the VIP.

If you need a floating IP for the VIP address, create a port in the network where the VIP address lies.

```
$> neutron port-create --fixed-ip subnet_id=subnet ID of the network in which VIP is
placed,ip_address=VIP IP --name anyname network ID in which the VIP is being placed
```

An example for the above syntax is as follows:

```
$> neutron port-create --fixed-ip subnet_id=55daee6b-32b7-4f9c-945e-
bcd2acb7272f,ip_address=172.16.0.231 --name test200vip f14eb427-4087-4dce-8477-479e10804ba1
```

Create a floating IP and associate it with that VIP address.

```
$> neutron floatingip-associate bf7c870e-6608-4512-b73d-faab5b18af04 ff67ae44-9874-43e6-
a194-f336b9b1d7b5
```

- 20 Create a pool of server(s) to be associated with the virtual service created above. In this case, this would be the server created using the Horizon UI in step 14.

You cannot use the select-servers-by-network feature, as you do not have access to the infra manager. Therefore, specify the IP addresses manually.

- 21 The virtual service should be up and running.
- 22 Check the respective Service Engine to verify that the VIP is associated with it.
  - The `allowed-address-pairs` Neutron extension allows traffic with specific CIDRs to egress from a port. NSX Advanced Load Balancer uses this extension to place VIPs on SE data ports, thereby allowing VIP traffic to egress these data ports.

- Add `allowed-address-pairs` on the SE ports so that security groups do not drop the packets. For the ML2/OVS plugin, you can add `allowed-address-pairs` with '0.0.0.0/0' once for each of the SE ports or specific VIP IP address.

```
neutron port-update da0e1e9a-312d-41c2-b15f-f10ac344ef03 --allowed-address-pairs
type=dict list=true ip_address=192.168.1.222/32
```

- If True, the `allowed-address-pairs` extension will be used. If the underlying network plugin does not support this feature, then VIP traffic will not work unless there are other means to achieve the same effect. This option can be turned off if the underlying network supports turning off security/firewall/spoof filter rules on ports.
- In cases where port-security is available, you can disable port-security on the SE's data vNIC Neutron port. This is another alternative for above.

**23** Ensure that you can SSH into one of the instances (Service Engines)

## Creating Service Engine using Heat-Templates in No-Access OpenStack Cloud

This section discusses creating SEs using heat templates in a no-access OpenStack cloud.

Ensure the following:

- Heat stack is installed on the OpenStack Controller node.
- Cloud is on no-orchestrator type.
- SE is downloaded as `qcow2` and has been pushed into Glance. For more details, refer to [Installing NSX Advanced Load Balancer OpenStack in No-Access OpenStack Cloud](#) section.
- Heat template files are present on the OpenStack Controller node. You can download the files from [this](#) location.

## Deploying an NSX Advanced Load Balancer Service Engine

You can deploy the SE as follows:

- Edit the `se-no-orc-env.yaml` file and alter the following parameters as explained:

```
se_networks: avimgmt, client-network, webserver-network, provider1.
```

These are the networks that the SE can be part of.

- `num_se_networks`: 4. This is the number of networks that the SE is part of.
- `allowed_address_cidr`: 12.10.0.0/24. AAP entry.
- `avi_se_name`: OS-SE- No-Orch. An arbitrary name for the SE.
- `auth_token`: 4c7f5489-df57- 4f4d-a5b6- c6fa5fb88ea5. Needs to be fetched from the NSX Advanced Load Balancer UI.
- `controller_ip`: 10.140.4.46. NSX Advanced Load Balancer Controller IP.
- `avi_se_flavor`: m1.small - The flavor needed to spin up the SE. Ensure that this flavor exists in the OpenStack Controller node.

- `avi_se_image`: Avi-SE- OS-No- Orch - The name of the SE image which is created from Horizon or using CLI.

### Note

- If multi-tenancy has to be used, ensure that the token is used or fetched from the respective tenant in NSX Advanced Load Balancer UI and is populated in the yaml file.
- In case more than one SE has to be spinned up, the YAML file has to run with a different token as NSX Advanced Load Balancer uses only one token per SE per tenant.

## Creating Service Engine using Heat-Templates

You can create SE using heat-templates as follows:

```
heat stack-create -f se-no-orc.yaml -e ./se-no-orc-env.yaml ocata
```

where, ocata is the name of the heat-stack that we are creating.

Check the stack creation using the following command:

```
root@openstack-ocata:/root# openstack stack list
```

```
+-----+-----+-----+-----+
+-----+
| ID | Stack Name | Stack Status | Creation Time
| Updated Time |
+-----+-----+-----+-----+
+-----+
| 45c44c59-7d24-4d41-ac93-0d6f4200d5b5 | ocata | CREATE_COMPLETE | 2017-12-06T08:41:48Z
| None |
+-----+-----+-----+-----+
+-----+
+-----+
```

Check the server list using the following command:

```
root@openstack-ocata:/root# openstack server list
```

```
+-----+-----+-----+
+-----+
+-----+
+-----+
| ID | Name | Status |
Networks
| Image Name |
+-----+-----+-----+
+-----+
+-----+
+-----+
| 5ad18760-2475-41d6-ad59-855e1a8bcd5a | OS-SE-No-Orcgh | ACTIVE | client-
network=192.168.10.4; avimgmt=172.16.0.6; provider1=10.134.7.207; webserver-
network=192.168.11.12 | Avi-SE-OS-No-Orch |
| baef34e9-2526-4965-9638-955616e8bd16 | Avi-test-server-2 | ACTIVE | webserver-
network=192.168.11.6,
10.134.7.199 | perf-client-server
```

```

|
| 853b0591-2102-4d87-b713-ca726bd16e43 | Avi-test-server-1 | ACTIVE | webserver-
network=192.168.11.4,
10.134.7.193 | perf-client-server
|
| 2023fcf5-ddd6-4c3d-afd0-826c5fab65c7 | Avi-test-client-1 | ACTIVE | client-
network=192.168.10.8,
10.134.7.200 | perf-client-
server |
| e38e7992-305e-47c0-bb07-3bf6d58767a7 | Avi-Controller-Ocata | ACTIVE | avimgmt=172.16.0.11,
10.134.7.196 |
AviController |
+-----+-----+-----+
+-----+-----+-----+
-----+-----+

```

## Installing Heat Plugins in NSX Advanced Load Balancer OpenStack using Kolla Ansible

This section explains the steps to be followed to install NSX Advanced Load Balancer OpenStack plugins in Kolla-Ansible based OpenStack deployments.

The following are the environment details:

- OpenStack version: Queens and Rocky
- Deployment: Kolla-Ansible based deployment
- Base image used for building images: CentOS
- Ansible Inventory: all-in-one
- NSX Advanced Load Balancer Heat plugin version to be installed: 17.2.7

## Deploying NSX Advanced Load Balancer Plugins in Kolla-Ansible

Kolla-Ansible provides a way to deploy OpenStack services and customize them. It uses ansible and docker to run the services on hosts and uses Jinja templates for customization. Kolla-Ansible is usually installed on an ansible control host, and Kolla-Ansible commands are executed from there. Kolla builds are also carried out on the same host using Kolla-build command. Kolla-build builds the OpenStack service container images and provides options to customize the image to include third-party plugins.

Customizations are done using combination of Jinja templates and config files. Using templates, you can install additional packages like third-party plugins in image. Additional configuration is provided to the plugin/driver using the custom config files.

Customization to existing templates are given in a Jinja-based template which extends the parent template. Parent templates are the templates from upstream for OpenStack services and it defines the contents of image.

Custom configurations are provided from `node_custom_config` directory usually located in `/etc/kolla/config` directory. You can change this in Kolla-Ansible's `globals.yml` file. For instance, if you need to provide extra config to heat service, create a file in `/etc/kolla/config/heat.conf` directory and provide additional configuration in it. Kolla-build will merge this config file with the upstream config file and provide it in heat container images.

## NSX Advanced Load Balancer Heat Plugin

NSX Advanced Load Balancer Heat Plugin is a heat resource module for a heat-engine. This plugin is loaded by heat-engine. The Heat Plugin provides NSX Advanced Load Balancer resources and handlers for them. On a Kolla-based deployment, you need to install NSX Advanced Load Balancer Heat Plugin in heat-engine image.

The following are the steps to deploy the Heat Plugin:

Step 1: On Kolla build host, create a Jinja template as follows to add NSX Advanced Load Balancer heat plugin to heat-engine image, `template-override.j2`.

```
{ extends parent_template }
{ block heat_engine_footer }
RUN yum install -y epel-release && yum install -y python-pip \
 && git clone https://github.com/avinetworks/avi-heat.git \
 && cd avi-heat && git checkout -b 17.2.7 origin/17.2.7 \
 && pip --no-cache-dir install . && cd -
{ endblock }
```

---

**Note** This format will be maintained for both source-build types and binary-build types.

---

You can use the following template to directly use the RPM/DEB packages, if available:

```
{ extends parent_template }
{ block heat_engine_footer }
RUN yum install -y wget \
 && wget https://github.com/avinetworks/avi-heat/releases/download/v17.2.7/
 aviheat-17.2.7b201812211855-1.noarch.rpm \
 && yum install -y aviheat-17.2.7b201812211855-1.noarch.rpm
{ endblock }
```

The link to RPM/DEB packages for a particular release of NSX Advanced Load Balancer Heat Plugin is available in <https://github.com/avinetworks/avi-heat/releases>.

Step 2: Run the following command to build heat-engine image:

```
python tools/build.py --template-override template-overrides.j2 --tag 'queens' heat-engine
```

OR,

if you are using Kolla-build, run the following command:

```
kolla-build --template-override template-overrides.j2 --tag 'queens' heat-engine
```

This will build a heat-engine image as follows with NSX Advanced Load Balancer plugin installed in it:

```
kolla/centos-binary-heat-engine queens e1a090136e5a About an hour ago 940.5 MB
```

**Note** The image name can change based on Linux distribution being used.

Step 3: Add the NSX Advanced Load Balancer Heat Plugin specific configuration to Kolla custom config. Usually Kolla custom config is located in `/etc/kolla/config` directory:

```
[root@kolla ~]# cat /etc/kolla/config/heat.conf
[DEFAULT]
plugin_dirs=/usr/lib/python2.7/site-packages/avi/heat
avi_controller=Avi Controller VIP
```

For Ubuntu based distro, the `plugin_dirs` changes to:

```
/usr/local/lib/python2.7/dist-packages/avi/heat
```

**Note** The custom config directory is configurable and you need to check it in `/etc/kolla/globals.yml` directory, with variable name as `node_custom_config`.

Step 4: Once you have the custom-image and config in place, you need to reconfigure the services. Kolla-Ansible should pick this newly built custom image, for which, you need to add the below mentioned code in `/etc/kolla/globals.yml` directory:

```
Custom image of heat heat_engine_image: centos-binary-heat-engine
```

However, you can also manually tag the image and give that image name here.

```
docker tag img-id kolla/heat-avi-plugin:queens
```

Step 5: Run Kolla-Ansible reconfigure as follows:

```
kolla-ansible -i INVENTORY reconfigure
```

This will deploy the newly built heat engine container.

Step 6: Verify if the Heat Plugin is installed by using `heat resource-type-list` command. NSX Advanced Load Balancer Heat Resources should be available in output.

```
(avi-dev-venv) ~ $> heat resource-type-list | grep Avi
WARNING (shell) "heat resource-type-list" is deprecated, please use "openstack orchestration
resource type list" instead
| Avi::LBaaS::ActionGroupConfig |
| Avi::LBaaS::AlertConfig |
| Avi::LBaaS::AlertEmailConfig |
| Avi::LBaaS::AlertObjectList |
| Avi::LBaaS::AlertScriptConfig |
| Avi::LBaaS::AlertSyslogConfig |
```



|                                           |  |
|-------------------------------------------|--|
| Avi::LBaaS::AnalyticsProfile              |  |
| Avi::LBaaS::Application                   |  |
| Avi::LBaaS::ApplicationPersistenceProfile |  |
| Avi::LBaaS::ApplicationProfile            |  |
| Avi::LBaaS::AuthProfile                   |  |
| Avi::LBaaS::AutoScaleLaunchConfig         |  |
| Avi::LBaaS::BackupConfiguration           |  |
| Avi::LBaaS::CertificateManagementProfile  |  |
| Avi::LBaaS::Cloud                         |  |
| Avi::LBaaS::CloudConnectorUser            |  |
| Avi::LBaaS::CloudProperties               |  |
| Avi::LBaaS::Cluster                       |  |
| Avi::LBaaS::ClusterCloudDetails           |  |
| Avi::LBaaS::ControllerLicense             |  |
| Avi::LBaaS::ControllerProperties          |  |
| Avi::LBaaS::CustomIpamDnsProfile          |  |
| Avi::LBaaS::DebugController               |  |
| Avi::LBaaS::DebugServiceEngine            |  |
| Avi::LBaaS::DebugVirtualService           |  |
| Avi::LBaaS::DnsPolicy                     |  |
| Avi::LBaaS::ErrorPageBody                 |  |
| Avi::LBaaS::ErrorPageProfile              |  |
| Avi::LBaaS::Gslb                          |  |
| Avi::LBaaS::GslbGeoDbProfile              |  |
| Avi::LBaaS::GslbService                   |  |
| Avi::LBaaS::GslbSite                      |  |
| Avi::LBaaS::GslbThirdPartySite            |  |
| Avi::LBaaS::HTTTPolicySet                 |  |
| Avi::LBaaS::HardwareSecurityModuleGroup   |  |
| Avi::LBaaS::HealthMonitor                 |  |
| Avi::LBaaS::IpAddrGroup                   |  |
| Avi::LBaaS::IpAddrGroup::Addr             |  |
| Avi::LBaaS::IpamDnsProviderProfile        |  |
| Avi::LBaaS::L4PolicySet                   |  |
| Avi::LBaaS::MicroService                  |  |
| Avi::LBaaS::MicroServiceGroup             |  |
| Avi::LBaaS::Network                       |  |
| Avi::LBaaS::NetworkProfile                |  |
| Avi::LBaaS::NetworkSecurityPolicy         |  |
| Avi::LBaaS::PKIProfile                    |  |
| Avi::LBaaS::Pool                          |  |
| Avi::LBaaS::Pool::Server                  |  |
| Avi::LBaaS::PoolGroup                     |  |
| Avi::LBaaS::PoolGroupDeploymentPolicy     |  |
| Avi::LBaaS::PriorityLabels                |  |
| Avi::LBaaS::Role                          |  |
| Avi::LBaaS::SSLKeyAndCertificate          |  |
| Avi::LBaaS::SSLProfile                    |  |
| Avi::LBaaS::Scheduler                     |  |
| Avi::LBaaS::SeProperties                  |  |
| Avi::LBaaS::ServerAutoScalePolicy         |  |
| Avi::LBaaS::ServiceEngine                 |  |
| Avi::LBaaS::ServiceEngineGroup            |  |
| Avi::LBaaS::SnmpTrapProfile               |  |
| Avi::LBaaS::StringGroup                   |  |

```
| Avi::LBaaS::SystemConfiguration |
| Avi::LBaaS::TrafficCloneProfile |
| Avi::LBaaS::UserAccountProfile |
| Avi::LBaaS::VIMgrHostRuntime |
| Avi::LBaaS::VIMgrVcenterRuntime |
| Avi::LBaaS::VSDataScriptSet |
| Avi::LBaaS::VirtualService |
| Avi::LBaaS::VrfContext |
| Avi::LBaaS::VsApicExtension |
| Avi::LBaaS::VsVip |
| Avi::LBaaS::WafCRS |
| Avi::LBaaS::WafPolicy |
| Avi::LBaaS::WafProfile |
| Avi::LBaaS::Webhook |
(avi-dev-venv) ~ $>
```

## Configuring the NSX Advanced Load Balancer Cloud Connector

This section explains configuration of OpenStack using NSX Advanced Load Balancer cloud connector.

### Importing User Accounts from Keystone

Using the NSX Advanced Load Balancer REST API, you can export user roles from Keystone into the Controller and directly map to role names in the Controller. You need not recreate the accounts on the Controller. For instance,

```
"openstack_configuration":
{

 "role_mapping": [
 {"os_role": "admin",
 "avi_role": "Tenant-Admin"},
 {"os_role": "_member_",
 "avi_role": "Tenant-Admin"},
 {"os_role": "*",
 "avi_role": "Application-Operator"}
],

}
```

The `role_mapping` parameter is an ordered list, where each item specifies how a Keystone role (`os_role`) maps to a role in the Controller (`avi_role`). You can define a default mapping for any Keystone role by specifying the “/\*” wildcard for the `os_role` field. In the above example, roles administrator and member from Keystone are mapped to the **Tenant-Admin** role in the Controller. Further, any other role from Keystone is mapped to **Application-Operator** role on the Controller.

In the following example, only the users with role `lbaas_project_admin` are allowed to access the Controller:

```
"openstack_configuration":
{

 "role_mapping": [
 {"os_role": "lbaas_project_admin",
 "avi_role": "Tenant-Admin"}
],

}
```

## Metadata instead of config\_drive for NSX Advanced Load Balancer SEs

In some OpenStack environments, `config_drive` support is either absent or not installed properly. Also, under certain conditions, you may not allow NSX Advanced Load Balancer SEs to use `config_drive`, as VM can prevent SE migration while configuring. The NSX Advanced Load Balancer OpenStack configuration option uses metadata instead of `config_drive` for SE VMs. You can enable NSX Advanced Load Balancer to use metadata by disabling `config_drive`.

The following is the CLI command to disable `config_drive`:

```
: > configure cloud Default-Cloud
: cloud> openstack_configuration
: cloud:openstack_configuration> no config_drive
: cloud:openstack_configuration> save
: cloud> save
```

## NSX Advanced Load Balancer OpenStack External Networks

NSX Advanced Load Balancer supports using OpenStack external networks, i.e., those marked as “external=True” as VIP and / or pool networks. If these external networks have been correctly provisioned / configured, the VIP is directly reachable from outside OpenStack without a need for a floating-IP association. This feature is disabled by default.

### Enabling External Networks Using CLI

To enable external networks Using CLI command:

```
configure cloud Default-Cloud
cloud> openstack_configuration
cloud:openstack_configuration> external_networks
cloud:openstack_configuration> save
cloud> save
```

## Enabling External Networks Using UI

You can enable external networks using UI as follows:

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Navigate to **Step 3:Network** tab in **New Cloud** or **Edit Cloud** window.
- 3 Select the check box **External Networks** to enable external networks.

## Additional Deployment Options

This section gives information on the additional deployment options suggested in OpenStack installation guide.

### NSX Advanced Load Balancer OpenStack Full Access Integration using Non-admin Privileges

This section explains the integration process of OpenStack full access using non-admin privileges.

You require admin privileges to deploy NSX Advanced Load Balancer solution in Provider mode on OpenStack. Admin privileges provides system-wide access to OpenStack resources like Tenants (Projects), Users, Roles, Networks, Routers, Ports etc., thereby enabling the NSX Advanced Load Balancer - OpenStack integration to work seamlessly. The cloud admin or the operator does not have to do any additional configuration in OpenStack.

However, for some deployments, admin privileges will be disabled as per business or IT policies. In such cases, you will not be able to deploy and use NSX Advanced Load Balancer Vantage with Provider mode. This guide outlines the OpenStack services policy changes, and constraints on OpenStack network resources in order to deploy and use NSX Advanced Load Balancer Vantage in Provider mode without giving it admin privileges.

### Creating Admin User without Admin Privileges

For OpenStack to work without admin privileges, OpenStack admin needs to create NSX Advanced Load Balancer admin user, an NSX Advanced Load Balancer admin role, and a tenant in OpenStack to be used in NSX Advanced Load Balancer cloud configuration. Existing tenant can also be used.

You can use admin user configured in NSX Advanced Load Balancer OpenStack cloud (called NSX Advanced Load Balancer Admin user) for establishing initial connection, authentication with keystone, and accessing projects, deployed regions and networks for initial setup. Once initial setup is complete, you can use this to import tenants from OpenStack, and access resources in various tenants. Ensure that NSX Advanced Load Balancer admin user has enough privileges to read the Tenants (Projects), Users, Roles, and list the role assignments for users in tenants.

Admin tenant in NSX Advanced Load Balancer OpenStack cloud (called NSX Advanced Load Balancer Admin tenant) is the tenant in which NSX Advanced Load Balancer will spin up the SEs in Provider mode. Ensure that the admin user has enough permissions to upload the SE images, list the flavors, and spin up the VMs in networks created or shared in the NSX Advanced Load Balancer Admin tenant.

You can create NSX Advanced Load Balancer admin user in OpenStack as follows:

- 1 Create `aviadmin` role in OpenStack using admin credentials. You can use the following command:

```
source admin-openrc.sh
```

- 2 Create user with name `aviuser` in OpenStack in OpenStack Cloud. You can use the following command:

```
config
$ openstack user create --password avil23 --description 'Avi Admin user' aviuser
```

- 3 Create `aviadmin` role using the following command:

```
$ openstack role create aviadmin
```

- 4 Create project with name `avilbaas` project to be used as NSX Advanced Load Balancer Admin Tenant in OpenStack. The following is the CLI command:

```
Cloud config
$ openstack project create --description 'Avi LBaaS Project' avilbaas
```

- 5 Assign `aviuser` user, `aviadmin` role in `avilbaas` tenant using the following command:

```
$ openstack role add --project avilbaas --user aviuser aviadmin
```

- 6 Add `aviuser` user, `aviadmin` role to other projects where NSX Advanced Load Balancer Load balancer service is required. The following is the CLI command:

```
$ openstack role add --project demo --user aviuser aviadmin
```

- 7 Verify the roles assigned to `aviuser` in all tenants using the following command:

```
$ openstack role assignment list --user aviuser --names
```

- 8 For non-default domain, use the desired domain while creating the NSX Advanced Load Balancer Admin user and NSX Advanced Load Balancer Admin tenant.

## OpenStack Policy Changes

NSX Advanced Load Balancer Controller interacts with Keystone, Nova, Neutron, and Glance OpenStack service. Following is a list of policy changes required per service:

## Keystone Policy Changes

The following are the steps to interact with Keystone policy change:

- 1 Identify policy file from `keystone.conf` file, for instance,

```
[oslo_policy]
policy_file = keystone.policy.yaml
```

- 2 Define `aviadmin` role in keystone policy file, for instance, `keystone.policy.yaml`,

```
"aviadmin_role": "role:aviadmin"
```

- 3 Give access to the following APIs for `aviadmin_role`, for instance, in `keystone.policy.yaml`,

```
Show domain details
GET /v3/domains/{domain_id}
Intended scope(s): system
"identity:get_domain": "rule:admin_required or token.project.domain.id:%
(target.domain.id)s or rule:aviadmin_role"

List domains
GET /v3/domains
Intended scope(s): system
"identity:list_domains": "rule:admin_required or rule:aviadmin_role"

List endpoints
GET /v3/endpoints
Intended scope(s): system
"identity:list_endpoints": "rule:admin_required or rule:aviadmin_role"

List projects
GET /v3/projects
Intended scope(s): system
"identity:list_projects": "rule:admin_required or rule:aviadmin_role"

Show role details
GET /v3/roles/{role_id}
HEAD /v3/roles/{role_id}
Intended scope(s): system
"identity:get_role": "rule:admin_required or rule:aviadmin_role"

List roles
GET /v3/roles
HEAD /v3/roles
Intended scope(s): system
"identity:list_roles": "rule:admin_required or rule:aviadmin_role"

List role assignments
GET /v3/role_assignments
HEAD /v3/role_assignments
Intended scope(s): system
"identity:list_role_assignments": "rule:admin_required or rule:aviadmin_role"
```

```

List services
GET /v3/services
Intended scope(s): system
"identity:list_services": "rule:admin_required or rule:aviadmin_role"

Show user details
GET /v3/users/{user_id}
HEAD /v3/users/{user_id}
"identity:get_user": "rule:admin_or_owner or rule:aviadmin_role"

List users
GET /v3/users
HEAD /v3/users
Intended scope(s): system
"identity:list_users": "rule:admin_required or rule:aviadmin_role"

```

- Restart Keystone service using the following command:

```
apache2ctl restart
```

## Nova Policy Changes

The following are the steps to interact with Nova policy change:

- Identify policy file from `nova.conf` file:

```
[oslo_policy] policy_file = nova.policy.yaml
```

- Define `aviadmin` role in nova policy file:

```
"aviadmin_role": "role:aviadmin"
```

- Give access to following APIs for `aviadmin_role` in nova policy file:

```

List all aggregates
GET /os-aggregates
"os_compute_api:os-aggregates:index": "rule:admin_api or rule:aviadmin_role"

Show details for an aggregate
GET /os-aggregates/{aggregate_id}
"os_compute_api:os-aggregates:show": "rule:admin_api or rule:aviadmin_role"

List availability zone information without host information
GET /os-availability-zone
"os_compute_api:os-availability-zone:list": "rule:admin_or_owner or rule:aviadmin_role"

List detailed availability zone information with host information
GET /os-availability-zone/detail
"os_compute_api:os-availability-zone:detail": "rule:admin_api or rule:aviadmin_role"

List available extensions and show information for an extension by
alias
GET /extensions
GET /extensions/{alias}

```

```
"os_compute_api:extensions": "rule:admin_or_owner or rule:aviadmin_role"
```

It also allows access to the full list of tenants that have access to a flavor via an os-flavor-access API.

```
GET /flavors/{flavor_id}/os-flavor-access
```

```
GET /flavors/detail
```

```
GET /flavors/{flavor_id}
```

```
POST /flavors
```

```
PUT /flavors/{flavor_id}
```

```
"os_compute_api:os-flavor-access": "rule:admin_or_owner or rule:aviadmin_role"
```

List all servers

```
GET /servers
```

```
"os_compute_api:servers:index": "rule:admin_or_owner or rule:aviadmin_role"
```

List all servers with detailed information

```
GET /servers/detail
```

```
"os_compute_api:servers:detail": "rule:admin_or_owner or rule:aviadmin_role"
```

List all servers for all projects

```
GET /servers
```

```
"os_compute_api:servers:index:get_all_tenants": "rule:admin_api or rule:aviadmin_role"
```

List all servers with detailed information for all projects

```
GET /servers/detail
```

```
"os_compute_api:servers:detail:get_all_tenants": "rule:admin_api or rule:aviadmin_role"
```

#### 4 Restart nova-apIService:

```
service nova-api restart
```

## Neutron Policy Changes

The following are the steps to interact with Neutron policy change:

#### 1 Identify policy file from neutron.conf file:

```
[oslo_policy]
policy_file = neutron.policy.json
```

#### 2 Define aviadmin role in neutron.policy file (neutron.policy.json):

```
"aviadmin_role": "role:aviadmin",
```

#### 3 Give access to the following APIs for aviadmin\_role in neutron.policy.json:

```
"create_port:device_owner": "not rule:network_device or rule:context_is_advsvc or
rule:admin_or_network_owner or rule:aviadmin_role",

"create_port:fixed_ips": "rule:context_is_advsvc or rule:admin_or_network_owner or
rule:aviadmin_role",
"create_port:fixed_ips:ip_address": "rule:context_is_advsvc or
rule:admin_or_network_owner or rule:aviadmin_role",
"create_port:fixed_ips:subnet_id": "rule:context_is_advsvc or rule:admin_or_network_owner
```



```

or rule:shared or rule:aviadmin_role",
 "create_port:allowed_address_pairs": "rule:admin_or_network_owner or rule:aviadmin_role",
 "get_port": "rule:context_is_advsvc or rule:admin_owner_or_network_owner or
rule:aviadmin_role",
 "update_port:fixed_ips": "rule:context_is_advsvc or rule:admin_or_network_owner or
rule:aviadmin_role",
 "update_port:fixed_ips:ip_address": "rule:context_is_advsvc or
rule:admin_or_network_owner or rule:aviadmin_role",
 "update_port:fixed_ips:subnet_id": "rule:context_is_advsvc or rule:admin_or_network_owner
or rule:shared or rule:aviadmin_role",
 "update_port:allowed_address_pairs": "rule:admin_or_network_owner or rule:aviadmin_role ",
 "get_agent": "rule:admin_only or rule:aviadmin_role",
 "get_l3-agents": "rule:admin_only or rule:aviadmin_role",
 "get_loadbalancer-agent": "rule:admin_only or rule:aviadmin_role",
 "get_loadbalancer-pools": "rule:admin_only or rule:aviadmin_role",
 "get_agent-loadbalancers": "rule:admin_only or rule:aviadmin_role",
 "get_loadbalancer-hosting-agent": "rule:admin_only or rule:aviadmin_role",
 "create_floatingip:floating_ip_address": "rule:admin_only or rule:aviadmin_role",
 "get_floatingip": "rule:admin_or_owner or rule:aviadmin_role",

```

#### 4 Restart neutron-server service using the command:

```
service neutron-server restart
```

## Configuring Cloud

On NSX Advanced Load Balancer Controller, configure the cloud with `aviuser` credentials, and `avilbaas` as admin tenant. In OpenStack role mapping, map OpenStack `aviadmin` role to NSX Advanced Load Balancer System-Admin or Tenant-Admin role.

You can set `map_admin_to_cloud_admin` to `True`. This will map `avilbaas` tenant to admin tenant in NSX Advanced Load Balancer, and any action taken in admin tenant in NSX Advanced Load Balancer will reflect in `avilbaas` tenant.

## Example: Configuration Example

In this example, all OpenStack roles are given Tenant-Admin role.

```
[admin:avi-controller]: show cloud Default-Cloud
```

| Field                   | Value                                      |
|-------------------------|--------------------------------------------|
| uuid                    | cloud-4db84437-f236-41cb-996f-11e450976744 |
| name                    | Default-Cloud                              |
| vtype                   | CLOUD_OPENSTACK                            |
| openstack_configuration |                                            |
| username                | aviuser                                    |
| password                | [sensitive]                                |
| admin_tenant            | avilbaas                                   |
| mgmt_network_name       | mgmt                                       |
| privilege               | WRITE_ACCESS                               |
| use_keystone_auth       | True                                       |
| region                  | RegionOne                                  |
| hypervisor              | KVM                                        |

|  |                                 |  |                             |  |
|--|---------------------------------|--|-----------------------------|--|
|  | tenant_se                       |  | False                       |  |
|  | import_keystone_tenants         |  | True                        |  |
|  | anti_affinity                   |  | True                        |  |
|  | port_security                   |  | False                       |  |
|  | security_groups                 |  | False                       |  |
|  | allowed_address_pairs           |  | True                        |  |
|  | free_floatingips                |  | False                       |  |
|  | img_format                      |  | OS_IMG_FMT_AUTO             |  |
|  | use_admin_url                   |  | True                        |  |
|  | role_mapping[1]                 |  |                             |  |
|  | os_role                         |  | *                           |  |
|  | avi_role                        |  | Tenant-Admin                |  |
|  | use_internal_endpoints          |  | False                       |  |
|  | config_drive                    |  | True                        |  |
|  | auth_url                        |  | http://10.10.32.213:5000/v3 |  |
|  | insecure                        |  | False                       |  |
|  | external_networks               |  | False                       |  |
|  | neutron_rbac                    |  | True                        |  |
|  | map_admin_to_cloudadmin         |  | True                        |  |
|  | nuage_port                      |  | 8443                        |  |
|  | contrail_plugin                 |  | False                       |  |
|  | name_owner                      |  | True                        |  |
|  | use_nuagevip                    |  | False                       |  |
|  | nuage_virtualip                 |  | False                       |  |
|  | contrail_disable_policy         |  | False                       |  |
|  | apic_mode                       |  | False                       |  |
|  | dhcp_enabled                    |  | True                        |  |
|  | mtu                             |  | 1500 bytes                  |  |
|  | prefer_static_routes            |  | False                       |  |
|  | enable_vip_static_routes        |  | False                       |  |
|  | license_type                    |  | LIC_CORES                   |  |
|  | state_based_dns_registration    |  | True                        |  |
|  | ip6_autocfg_enabled             |  | True                        |  |
|  | tenant_ref                      |  | admin                       |  |
|  | license_tier                    |  | ENTERPRISE_18               |  |
|  | autoscale_polling_interval      |  | 60 seconds                  |  |
|  | +-----+-----+-----+-----+-----+ |  |                             |  |

## Requirements for OpenStack Keystone/Nova/Neutron Resources

The following are the requirements for OpenStack resources:

- 1 NSX Advanced Load Balancer Controller should be in `avilbaas` tenant for cluster VIP to work. Without admin privileges, the Controller will not be able to look into other tenants for Controller and it expects the Controllers and SE to be in same tenant (`avilbaas` tenant).
- 2 Flavors - The Controller will be able to use only public flavors or the flavors accessible to `avilbaas` tenant. Share the flavor to `avilbaas` tenant in order to use the flavor for SE. After sharing the flavor, set the `instance_flavor` option in SE Group.

- 3 Add `aviadmin` role in all tenants - You need to have an admin role for integration to work.
  - a a. NSX Advanced Load Balancer admin user cannot create ports in tenant networks, without having an admin role. This cannot be changed via Neutron policy. To fix this issue, the CC agent will use the tenant scoped client to create virtual service VIP port.
  - b b. Without having an admin role or without being network owner, NSX Advanced Load Balancer admin user will not be able to add `allowed-address-pair` entry to the port created in tenant network. This will cause virtual service placement to fail. This can be fixed using policy changes. To fix this, you can add `aviadmin` role to NSX Advanced Load Balancer admin user in the tenants, and make policy changes as mentioned above in Neutron Policy Changes section.
- 4 Tenant Networks
  - a When deploying in provider mode, tenant networks must be shared with NSX Advanced Load Balancer admin tenant, for instance, `avilbaas` tenant. Without this the Controller will not be able to connect SE to the networks. For instance,
 

```
neutron rbac-create --target-tenant [avilbaas-tenant-uuid] --action access_as_shared
--type network [network-uuid]
```
  - b When deploying in non-provider mode (dedicated SE mode, SE in tenant context), management network should be shared in all tenants that you want to deploy NSX Advanced Load Balancer . Tenant networks used for creating Virtual Service VIP or backend pool members need not be shared with `avilbaas` tenant.
- 5 FIP allocation — The routers connecting the tenant networks to provider networks must be created in the tenant. If the routers are created in admin tenant (or any other tenant), the Controller will not be able to access them and identify which provider networks to use for FIP allocation.

---

**Note** For using floating IP feature, having router in different tenant will not work even if you manually pass the floating network/subnet UUID in VS VIP request. This will not work because the access is limited to tenant in which VS is being created, and if the Controller cannot see the VIP network connected to a router, it will not be able to see the FIP network as well. FIP networks are derived from the tenant router's interfaces to provider networks, and requires the routers to be the tenant hosting the VIP networks.

---

## Upgrade Feature

The existing deployments use admin credentials with admin role. You can re-configure the cloud to use new credentials with `aviadmin` role.

Case 1: OpenStack admin tenant is used as NSX Advanced Load Balancer admin tenant. You need to add `aviadmin` role to the admin user in admin tenant, and share all the tenant networks with admin tenant. Upgrading to use this feature will not be disruptive.

Case 2: OpenStack non-admin tenant is used as NSX Advanced Load Balancer admin tenant. In this case, `aviadmin` role is added to admin user in this tenant. This tenant is configured as NSX Advanced Load Balancer admin tenant in NSX Advanced Load Balancer Cloud config. All the tenant networks need to be shared with this tenant. Upgrading to use this feature will not be disruptive.

Case 3: Changing the NSX Advanced Load Balancer admin tenant in cloud. This will be disruptive for some instances, since the admin tenant will be used to spin up the SEs. Configuring different admin tenant would mean using the other tenant to host all the SEs. This is disruptive, and all the virtual service need to be disabled/enabled in a maintenance window.

# Installing NSX Advanced Load Balancer in VMware NSX-T Environments

## 6

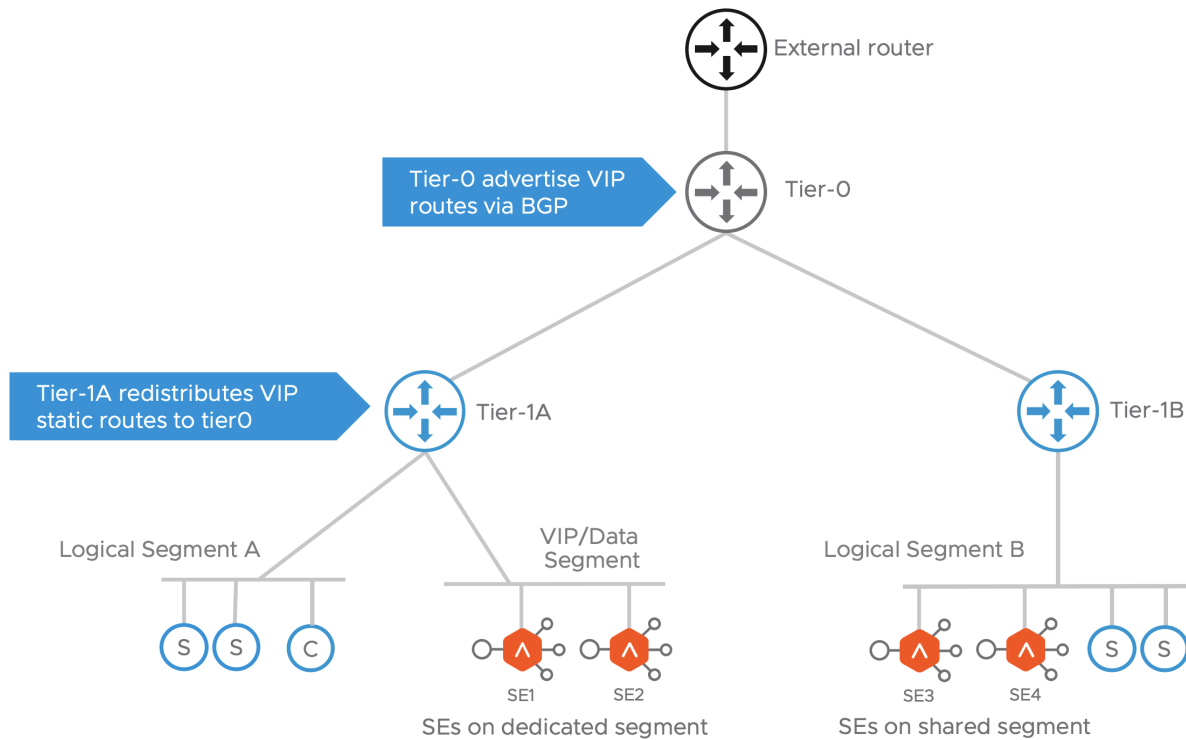
This section provides information about how to deploy NSX Advanced Load Balancer in VMware vSphere with NSX-T managed networking and security.

This chapter includes the following topics:

- [Overview of NSX-T Integration with NSX Advanced Load Balancer](#)
- [Installing the Controller](#)
- [Configuring the Cloud Connector](#)

## Overview of NSX-T Integration with NSX Advanced Load Balancer

VMware NSX-T provides an agile software-defined infrastructure to build cloud-native application environments. It includes networking, security, automation, and operational simplicity for emerging application frameworks and architectures with heterogeneous endpoint environments and technology stacks. NSX-T supports cloud-native applications, bare metal workloads, multi-hypervisor environments, public clouds, and multiple clouds.



For more information about VMware NSX-T, see [VMware NSX-T documentation](#).

## Installing the Controller

Follow these steps to install the Controller:

- 1 Ensure the prerequisites are met.
- 2 Deploy the Controller OVA.
- 3 Set up the Controller.

## Prerequisites to Install the Controller

Ensure the following prerequisites are met before installing the Controller:

- Configuring NSX-T Role Requirements
- Configuring vCenter Role Requirements
- Creating a Content Library

## Configuring Roles and Permissions for vCenter and NSX-T Users

The NSX-T cloud connector interacts with vCenter for the SE lifecycle management, and with the NSX-T manager to sync and create objects for networking and security.

The admin creates vCenter and NSX-T user credentials which have the necessary permissions for the NSX Advanced Load Balancer to perform these operations.

## Creating NSX-T User Credentials

The NSX-T cloud is configured with the admin credentials for the NSX-T manager.

### Procedure

- 1 From the NSX-T manager UI, navigate to **Administration >User Credentials**.
- 2 Enter the **Name**.
- 3 Select **NSX-T** as the **Credentials Type**.
- 4 Enter the **Username** and **Password**.
- 5 Click **Save**.

---

**Caution** In case the password expired, the NSX Advanced Load Balancer tries to reconnect using the expired password. After five consecutive failed login attempts, the administrator account is locked for 15 minutes. For more information, see [Account Lockout](#).

---

## Configuring vCenter User Credentials

Create the username and password to talk to the vCenter server as explained below:

### Procedure

- 1 Enter the **Name**.
- 2 Select vCenter as the **Credentials Type**.
- 3 Enter the **Username** and **Password**.
- 4 Click **Save**.

## Creating a vCenter Role

This section shows the creation of AviRole - Global. This role applies global permissions. It allows the user to upload the SE OVF to the content library allocate space on datastore to create a virtual machine and assign networks to it. Similarly, other vCenter roles can be created with relevant permissions.

To know how to create various vCenter roles, click [here](#).

The AviRole - Global requires the following permissions:

- Content Library
  - Add library items
  - Delete library items
  - Update files
  - Update library items
- Datastore
  - Allocate space

- Remove space
- Network
  - Assign network
  - Remove network
- vApp
  - Import
- Virtual Machine
  - Change configuration
    - Add new disk

#### Procedure

- 1 Log in to the vCenter UI as admin
- 2 Navigate to **Administration > Roles**
- 3 Click the **+** sign to create a new role
- 4 Click on **Content Library** and the following permissions:
  - a **Add library item**
  - b **Delete library item**
  - c **Update files**
  - d **Update library item**
- 5 Click on **Datastore** and select the following permissions:
  - a **Allocate space**
  - b **Remove file**
- 6 Click **Network** and select the following permissions
  - a **Assign network**
  - b **Remove**
- 7 Click **Virtual Machine** and select **Add new disk**
- 8 Click **vApp** and select **Import**
- 9 Click **Next**
- 10 Enter the **Role name** as AviRole-Global and enter a **Description**, if required
- 11 Click **Finish**.

#### Assigning a vCenter Role

This section shows how to assign a vCenter role.



**Procedure**

- 1 Log in to the vCenter UI and navigate to **Global Permissions**
- 2 Click the + sign to add a new permission
- 3 Select the **Domain**
- 4 Select the required username
- 5 Select **Propagate to children**

The **Add Permission** screen is shown below:

**Add Permission** | Global Permission Root ✕

Domain vsphere.local ▼

User/Group 🔍 aviuser

Role AviRole-Global ▼

☒ Propagate to children

CANCEL OK

- 6 Click **OK**

**Configuring NSX-T Roles**

This section discusses the roles required to be assigned to the NSX-T user. Local user creation is not allowed on NSX-T. The admin can select a VMware Identity Manager (VIDM) or an LDAP user and assign the required roles to it.

Customized role creation is not supported in NSX-T 3.0. So, the user has to be assigned an existing role that has all the permissions required by the NSX Advanced Load Balancer - NSX-T cloud. Consider an example in which the role is assigned to a VIDM user.

### Procedure

- 1 Log in to the NSX-T manager UI as an admin user.
- 2 Navigate to **System > User and Roles > USERS**
- 3 Click **Add** and select **Role Assignment for VIDM**
- 4 Select the **Network Engineer** role.
- 5 Click **Save**.

## Creating a Content Library

The NSX Advanced Load Balancer Controller uploads the Service Engine image to the content library on the vCenter server and uses this to create new virtual machine (VM) every time a new Service Engine is required. The content library must be created on vCenter before configuring the NSX-T cloud.

### Procedure

- 1 In the vCenter vSphere client, navigate to **Content Libraries**.
- 2 Click **Create**. The **New Content Library** wizard opens.
- 3 In the **Name and location** page, enter the **Name** and select a **vCenter Server instance** for the content library.
- 4 Click **Next**.
- 5 In the **Configure content library** page, select **Local content library**.
- 6 Click **Next**.
- 7 In the **Add storage** page, select datastore as a storage location for the library contents.
- 8 Click **Next**.

- 9 In the **Ready to complete** page, review the details.

The screenshot shows the 'New Content Library' wizard in the vSphere Client. The wizard has four steps: 1 Name and location, 2 Configure content library, 3 Add storage, and 4 Ready to complete. Step 4 is the current step, indicated by a blue button and a checkmark. The 'Ready to complete' section shows the following settings:

|                 |                         |
|-----------------|-------------------------|
| Name:           | Avi Content Library     |
| vCenter Server: | vc70.avidemo.vmware.com |
| Type:           | Local Content Library   |
| Publishing:     | Disabled                |
| Storage:        | datastore01             |

At the bottom right, there are three buttons: CANCEL, BACK, and FINISH. The FINISH button is highlighted in blue.

- 10 Click **Finish**.

## Deploying the Controller OVA

The Controller cluster virtual machines are deployed using the OVA, connected to the same management port group as the NSX-T Manager.

### Procedure

- 1 Log in to the vCenter server through a vCenter client.
- 2 From the **File** menu, select **Deploy OVF Template**.
- 3 Select the `controller.ova` file from your local machine.
- 4 In the **Deploy OVF Template** wizard,
  - a Select the **Virtual machine name** and the location to deploy.
  - b Select the compute resource.
  - c Review the details and click **Next**.
  - d Select the storage and set the disk format to **Thick Provision Lazy Zeroed**.
  - e In the Select networks section, choose a management network for the Controller.

- f Enter the management IP address, subnet mask and default gateway. In the case of DHCP, leave this field empty.

**Deploy OVF Template**

✓ 1 Select an OVF template  
 ✓ 2 Select a name and folder  
 ✓ 3 Select a compute resource  
 ✓ 4 Review details  
 ✓ 5 Select storage  
 ✓ 6 Select networks  
**7 Customize template**  
 8 Ready to complete

**Customize template**  
 Customize the deployment properties of this software solution.

✓ All properties have valid values

Application 4 settings

Management Interface IP Address  
 IP address for the Management Interface. Leave blank if using DHCP. Example: 192.168.10.4  
 10.10.1.100

Management Interface Subnet Mask  
 Subnet mask for the Management Interface. Leave blank if using DHCP. Example : 24 or 255.255.255.0  
 24

Default Gateway  
 Optional default gateway for the Management Network. Leave blank if using DHCP.  
 10.10.1.1

CANCEL BACK NEXT

**Note** Using static IP address is recommended for production setups.

The Controller OVA supports additional OVF properties. The following properties have been added to facilitate automated deployment of the Controller by the NSX Manager in a future release:

- NSX-T Node ID
- NSX-T IP Address
- Authentication token of NSX-T
- NSX-T thumbprint
- Hostname of NSX Advanced Load Balancer Controller

These fields should be left blank in case of a direct deployment of the Controller.

- g Review the setting and click **Finish**.

- 5 Power on the virtual machine.

## Setting up the NSX Advanced Load Balancer Controller

This section shows the steps to perform initial configuration of the NSX Advanced Load Balancer Controller using its deployment wizard. You can change or customize settings following initial deployment using the NSX Advanced Load Balancer Controllers web interface.

---

**Note** The NSX-T cloud is not a part of the deployment wizard. Therefore, select **No Orchestrator** as the integration option.

---

### Procedure

- 1 Navigate to the NSX Advanced Load Balancer Controller IP on your browser.
- 2 Enter the **User Name** and **Password**.
- 3 Enter the DNS and NTP server information.
- 4 Configure the **Email/SMTP** information.
- 5 Under **Orchestrator Integration**, select **No Orchestrator**.
- 6 Select **No** under **Support Multiple Tenants**.

## Configuring the Cloud Connector

The SE management interface has to be connected to an overlay logical segment. It also needs a tier-1 gateway to provide external connectivity to be able to reach the Controller management IP.

## Prerequisites for Creating the NSX-T Cloud

It is recommended to have a dedicated tier-1 gateway and segment for the SE management.

### Creating a Tier-1 Gateway

A tier-1 gateway is created in an NSX-T manager a tier-1 gateway has northbound connections to tier-0 gateways and southbound connections to segments.

### Procedure

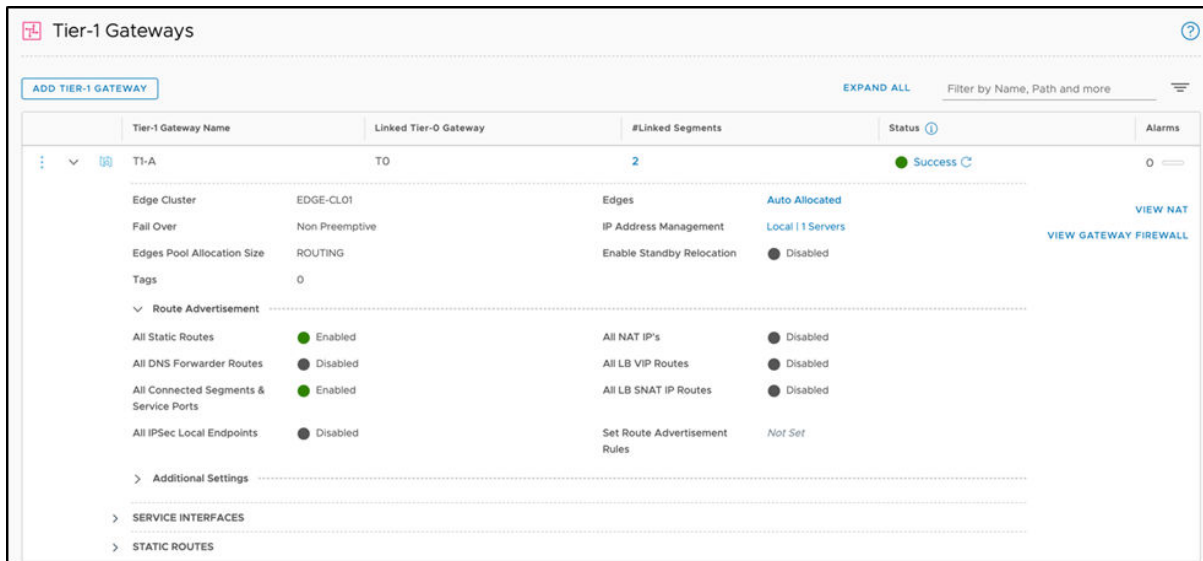
- 1 From the NSX-T manager, navigate to **Networking > Tier-1 Gateways**.
- 2 Click **Add Tier-1 Gateway**.
- 3 Enter the **Name** and select a tier-0 gateway to connect to this tier-1 gateway.
- 4 (Optional) Select an NSX **Edge Cluster** if you want this tier-1 gateway to host stateful services such as NAT, load balancer, or firewall.
- 5 (Optional) Next to **IP Address Management**, click **No Dynamic IP Allocation**.

- 6 (Optional) In the **Type** drop-down menu, select **DHCP Server** and select a DHCP profile to attach to this gateway.

**Note** Enabling DHCP on the tier-1 gateway is optional. SEs can also be configured to have static IPs configured for its interfaces by configuring IP address pool on the corresponding network object on the NSX Advanced Load Balancer Controller.

- 7 Click **Save**.
- 8 Under **Route Advertisement**, enable the options **All Static Routes**, and **All Connected Segments & Service Ports**.

The Tier-1 gateway is as shown below:



- 9 Click **Save**.

## Creating a Segment

This section shows how to create a Segment in an NSX-T manager

### Procedure

- 1 In the NSX-T manager, navigate to **Networking>Segments**.
- 2 Click **Add Segment**.
- 3 Enter a **Name** for the segment.
- 4 Under **Connectivity**, select the [Creating a Tier-1 Gateway](#) that has to be connected.
- 5 Select the Overlay **Transport Zone**.
- 6 Enter the **Subnets**.
- 7 (Optional) To configure DHCP on the segment, click on **Set DHCP Config**.

## 8 (Optional) Enable **DHCP Config** and enter the **DHCP Ranges**.

**Set DHCP Config**

IPv4 Server | IPv6 Server

Settings | Options

DHCP Config ☒ Enabled ⓘ

DHCP Server Address 192.168.100.10/24

DHCP Ranges 99 Maximum | Format 172.16.14.10-172.16.14.100 or 172.16.14.0/24

10.10.0.10-10.10.0.100 X  
Enter DHCP Ranges

Lease Time (seconds) Default value is 86400

DNS Servers Enter IP Addresses  
e.g. 10.10.10.10

CANCEL APPLY

9 Click **Apply**.

10 Click **Save**.

## Creating an NSX-T Cloud

The NSX-T cloud is created from the NSX Advanced Load Balancer Controller.

### Procedure

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Click **Create** and Select **NSX-T Cloud**.
- 3 Enter the **Name** of the NSX-T cloud.

**Note** NSX-T Cloud is selected as the **Cloud Type** by default.

- 4 Check the DHCP option if SE management segment has DHCP enabled.

**Important** The prefix string must only have letters, numbers and underscore. This field cannot be changed once the cloud is configured.

- 5 Enter the NSX-T manager hostname or IP address as the **NSX-T Manager Address** and select the **NSX-T Manager Credentials**.
- 6 Click **Connect** to to authenticate with the NSX-T manager.
- 7 Select the **Transport Zone** required from the drop-down (Only the Overlay type is supported).

- 8 In the **Management Network Segment**, select the Tier-1 Logical Router ID and Segment ID.

---

**Note** Currently, only **Manual** is supported as the **Logical Segments Config Mode**. Hence the option is greyed out. This requires the segment to be pre-created in the NSX manager.

---

- 9 Select the tier-1 gateway and logical switch for VIP placement.
- 10 Click **Add** to select one more tier-1 router and a connected logical segment for VIP placement.
- 11 Under **vCenter Servers**, click **Add**.
- 12 Enter the vCenter Server Name, and configure the credentials.
- 13 Click **Connect**.
- 14 Select the **Content Library** and click **Done**.
- 15 Select the IPAM/DNS Profile, as required.
- 16 Click **Save**.

## Multiple NSX-T Clouds

Starting with NSX Advanced Load Balancer version 20.1.3, multiple NSX-T clouds (a maximum of 5) can be created.

Each NSX-T manager can be either created for the same NSX-T manager or different NSX-T manager. If different NSX-T managers are pointing to the same vCenter, then only one SE image per vCenter will be created.

If there are multiple NSX-T managers pointing to respective different vCenters then the SE image will be created in the respective content libraries.

---

**Note** The cleanup of the SE image happens only after the last NSX-T cloud pointing to the SE image is removed.

---

The NSX-T cloud(s) thus created can be used to create a virtual service.



# Installing NSX Advanced Load Balancer on Linux KVM (No-Access mode)

## 7

This section explains the installation procedure for the Controller and SEs on the KVM hypervisor.

NSX Advanced Load Balancer is a software-based solution that provides real time analytics, application visibility, and elastic application delivery services, such as SSL termination, load balancing, and content acceleration.

NSX Advanced Load Balancer includes two components, the Controller and SEs. The Controller stores and manages all policies related to application delivery services and acts as a single point of control while providing GUI, REST APIs, and real time analytics services. The SEs are micro load balancers, which handle network traffic and provide application delivery services.

This chapter includes the following topics:

- [Considerations for Deploying NSX Advanced Load Balancer in KVM based environments](#)
- [Preparing the server](#)
- [Installing the Controller](#)
- [Additional Deployment Options](#)
- [Disabling NSX Advanced Load Balancer Controller and Service Engines](#)

## Considerations for Deploying NSX Advanced Load Balancer in KVM based environments

This section explains considerations for Deploying NSX Advanced Load Balancer in KVM based environments.

Before starting the installation, ensure the following minimum requirements are met under Role Requirement.

### Role Requirement

This section discusses Role Requirement for the following:

- Role Variable for Service Engine
- Role Variable for Controller

## Role Variable for Service Engine

This section details the Role Variables for SE.

### Role Variables

| Variable             | Required | Default | Comments                                                                                                    |
|----------------------|----------|---------|-------------------------------------------------------------------------------------------------------------|
| kvm_vm_hostname      | Yes      |         | Name for VM                                                                                                 |
| kvm_vm_base_img      | No       |         | se.qcow2 file                                                                                               |
| kvm_vm_vcpus         | No       | 2       | How many cpus the service engine will use.                                                                  |
| kvm_vm_ram           | No       | 2048    | How much memory the service engine will use.                                                                |
| kvm_vm_os_disk_size  | No       | 20      | How much disk size the service engine will use.                                                             |
| kvm_host_mgmt_intf   | Yes      |         | host management interface name                                                                              |
| se_kvm_ctrl_ip       | Yes      |         | The IP address of the controller.                                                                           |
| se_kvm_ctrl_username | Yes      |         | The username to login into the controller.                                                                  |
| se_kvm_ctrl_password | Yes      |         | The password to login into the controller.                                                                  |
| se_kvm_ctrl_version  | Yes      |         | The controller version.                                                                                     |
| state                | No       | create  | If create then create service engine and for delete it will delete the service engine.                      |
| se_auth_token        | No       |         | If defined it will be the token used to register the service engine to the controller                       |
| kvm_force_mode       | No       | TRUE    | If VM with the given name exist then it will overwrite to that VM and for false it will stop the execution. |
| se_bond_seq          | Yes      |         | Bonding sequence                                                                                            |
| se_kvm_mgmt_ip       | Yes      |         | Management Ip for the service engine                                                                        |
| se_kvm_mgmt_mask     | Yes      |         | Subnet mask                                                                                                 |
| se_kvm_default_gw    | Yes      |         | Default gateway for service engine                                                                          |
| kvm_pinning          | Yes      |         | If you want to enable pinning CPU for the VM                                                                |

| Variable           | Required | Default | Comments                                         |
|--------------------|----------|---------|--------------------------------------------------|
| kvm_total_num_vfs  | Yes      |         | Numbers VFs will be pass-through to VM           |
| kvm_virt_intf_name | Yes      |         | Virtual Function name will be pass-through to VM |

## Standard Example

Kvm host (inventory) file

```
[kvm]
10.170.5.51
[kvm:vars]
ansible_ssh_user=root
ansible_ssh_pass=<password>
```

```
- hosts: kvm
 vars:
 state: create
 kvm_vm_hostname: "se1"
 kvm_vm_vcpus: "2"
 kvm_vm_ram: "2048"
 kvm_host_mgmt_intf: eno1.100
 se_kvm_ctrl_ip: "10.170.5.21"
 se_kvm_ctrl_username: "admin"
 se_kvm_ctrl_password: "<controller password>"
 se_kvm_ctrl_version: "18.2.2"
 se_bond_seq: "1,2,3,4"
 se_kvm_mgmt_ip: "10.170.5.15"
 se_kvm_mgmt_mask: "255.255.255.0"
 se_kvm_default_gw: "10.170.5.1"
 kvm_pinning: true
 kvm_total_num_vfs: 4
 kvm_virt_intf_name:
 - enp24s17f1
 - enp24s17f3
 - enp24s17f5
 - enp24s17f7
 tasks:
 - name: Avi SE | KVM | Create SE VM on KVM
 include_role:
 name: avinetworks.avise_kvm
```

Command to run the playbook

```
ansible-playbook kvm.yml -i <inventory file> -vv
```

## Role Variable for Controller

This section tabulates and explains the Role Variables for Controller.

## Role Variables

| Variable                | Required | Default  | Comments                                                                                                    |
|-------------------------|----------|----------|-------------------------------------------------------------------------------------------------------------|
| kvm_vm_hostname         | Yes      |          | Name for VM                                                                                                 |
| kvm_vm_base_img         | Yes      |          | controller.qcow2 file                                                                                       |
| kvm_vm_vcpus            | No       | 8        | How many cpus the controller will use.                                                                      |
| kvm_vm_ram              | No       | 24576 MB | How much memory the controller will use.                                                                    |
| kvm_network_type        | No       | direct   | Which network type you have to use.                                                                         |
| kvm_network_source_mode | No       | bridge   | Which network bridge you have to use.                                                                       |
| kvm_network_model       | No       | virtio   | Which network model you have to use.                                                                        |
| kvm_vm_os_disk_size     | No       | 100      | How much disk size the controller will use.                                                                 |
| kvm_host_mgmt_intf      | Yes      |          | host management interface name                                                                              |
| ctrl_mgmt_ip            | Yes      |          | Management Ip for the controller                                                                            |
| ctrl_mgmt_mask          | Yes      |          | Subnet mast for the controller                                                                              |
| ctrl_default_gw         | Yes      |          | Default gateway for controller                                                                              |
| state                   | No       | create   | If create then it will create controller and for delete it will delete the controller.                      |
| kvm_pinning             | Yes      |          | If you want to enable pinning CPU for the VM                                                                |
| kvm_force_mode          | No       | TRUE     | If VM with the given name exist then it will overwrite to that VM and for false it will stop the execution. |

## Standard Example

Kvm host (inventory) file:

```
[kvm]
10.170.5.51
[kvm:vars]
```

```
ansible_ssh_user=root
ansible_ssh_pass=<password>
```

```
- hosts: kvm
 vars:
 kvm_vm_hostname: "ctrl1"
 kvm_host_mgmt_intf: eno1.100
 ctrl_mgmt_ip: 10.130.5.12
 ctrl_mgmt_mask: 255.255.255.0
 ctrl_default_gw: 10.130.5.1
 kvm_pinning: true
 tasks:
 - name: Create KVM VM
 include_role:
 name: avinetworks.avicontroller_kvm
```

Command to run the playbook:

```
ansible-playbook kvm.yml -i <inventory file> -vv
```

## Hardware

This section discusses about Hardware requirements for the Controller.

### Procedure

- 1 Ensure Intel Virtualisation Technology [Intel VT-d] in CPU configuration in the BIOS is enabled. This can be turned on in the BIOS settings during bootup time. Navigate to **BIOS Settings > CPU configuration**. Enable Intel Virtualisation Technology [Intel VT-d] option Once the system has booted up, run `lscpu` or `grep` command to see if your virtualization support is enabled:

```
lscpu | egrep Virtualization
Virtualization: VT-x
```

This ensures VT-d is enabled

```
$ egrep -o '(vmx|svm)' /proc/cpuinfo | sort | uniq
vmx
```

- 2 Ensure SRIOV-support for PCI in the BIOS is enabled. This can also be turned on in the BIOS settings during bootup time.
- 3 In grub config [/etc/default/grub], add `intel_iommu=on` in `GRUB_CMDLINE_LINUX` statement. Rebuild grub config using `sudo update-grub` for Ubuntu distros OR via `grub2-mkconfig -o /boot/grub2/grub.cfg` on others. Ensure the same is reflected in `cat /proc/cmdline` output.

## Software

This section explains about the software requirements for the Controller.

## Ubuntu Distro

| DISTRIB_ID          | Ubuntu                           |
|---------------------|----------------------------------|
| DISTRIB_RELEASE     | 16.04                            |
| DISTRIB_CODENAME    | xenial                           |
| DISTRIB_DESCRIPTION | Ubuntu 16.04.6 LTS               |
| OS kernel version   | 4.4.0-131-generic                |
| python              | version 2.7.12 and above         |
| virsh               | version 1.3.1 and above          |
| libvirt-bin         | version 1.3.1 and above          |
| ansible             | version 2.0.0.2 and above        |
| virt-manager        | version 1.3.2 and above          |
| qemu-kvm            | version 2.5.0 and above          |
| genisoimage         | version 1.1.11 (Linux) and above |

## RHEL/CentOS distro

| CentOS Linux release | 7.6.1810 (Core)                  |
|----------------------|----------------------------------|
| OS kernel version    | 3.10.0-957.5.1.el7.x86_64        |
| python               | version 2.7.5 and above          |
| vrish                | version 4.5.0 and above          |
| libvirt-bin          | version 4.5.0 and above          |
| ansible              | version 2.4.2.0 and above        |
| virt-manager         | version 1.5.0 and above          |
| qemu-kvm             | version 1.5.3                    |
| genisoimage          | version 1.1.11 (Linux) and above |

## NIC Information

The following table explains about the driver and NIC information for KVM installation in SR-IOV mode.

| Explanation                                         | Comments                                                                                                                                                                                     | Drivers and Supported NICs                                                                                                                                                                  |
|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Virtual network functions created from physical NIC | Allows physical NICs to be shared amongst VMs without sacrificing performance, since packets are switched in the hardware.<br>Maximum 32 virtual functions (VFs) can be configured per pNIC. | ixgbe-vf driver supports these NICs (and bonding): 10Gb NICs 82599, X520, X540, X550, X552.<br>i40e-vf driver supports these NICs (bonding not supported): 10Gb NICs X710, 40Gb NICs XL710. |

## Preparing the server

Before starting the process, the following requirements must be completed:

- Ansible Playbook
- Virtual Functions (VF) from PF
- Post Host Reboot

### Ansible Playbook

This section discusses about Ansible Playbook.

Use the following commands to install Ansible and NSX Advanced Load Balancer Python SDK:

NSX Advanced Load Balancer Python SDK - `pip install avisdk`

---

**Note** Starting with Ansible version 2.9 you need not install Avi SDK.

---

Install `avinetworks.avisdk` Ansible role using the following command:

```
ansible-galaxy install -f avinetworks.avisdk
```

Use the following commands to download the required Ansible roles:

- `ansible-galaxy install avinetworks.avicontroller_kvm`
- `ansible-galaxy install avinetworks.avise_kvm`

### Example

Below is an example playbook:

```
- hosts: localhost
 connection: local
 vars:
 controller: 10.10.28.4
 username: admin
 password: avi123$%
 api_version: "17.2.8"
 roles:
 - avinetworks.avisdk
 tasks:
 - avi_pool:
```

```

controller: ''
api_version: ''
username: ''
password: ''
name: testpool2
state: present
health_monitor_refs:
 - '/api/healthmonitor?name=System-HTTP'
servers:
 - ip:
 addr: 10.90.130.8
 type: V4
 - ip:
 addr: 10.90.130.7
 type: V4
- avi_virtualservice:
 controller: ''
 api_version: ''
 username: ''
 password: ''
 name: newtestvs
 state: present
 performance_limits:
 max_concurrent_connections: 1000
 ssl_profile_ref: '/api/sslprofile?name=System-Standard'
 application_profile_ref: '/api/applicationprofile?name=System-Secure-HTTP'
 ssl_key_and_certificate_refs:
 - '/api/sslkeyandcertificate?name=System-Default-Cert'
 vip:
 - ip_address:
 addr: 10.90.131.103
 type: V4
 vip_id: 1
 services:
 - port: 443
 enable_ssl: true
 - port: 80
 pool_ref: '/api/pool?name=testpool2'

```

NSX Advanced Load Balancer supports API versioning for backward compatibility with automation scripts written for an object model older than the current one. Such scripts need not be updated to keep up with object model changes. Accordingly, playbook authors should set `api_version` to the oldest version of NSX Advanced Load Balancer in which their code can work.

The support for Ansible collection is available for NSX Advanced Load Balancer deployments. The Ansible collections are supported for Ansible version 2.9.10 or the later versions. The detailed information about installation and usage is available at [Advanced Load Balancer Ansible Collection](#).



## Virtual Functions (VF) from PF

Ensure the requisite number of VF's are created from the parent PF using the parent PF's BDF value.

**Note** This has to be redone afresh every time the host machine is rebooted unless the config is made persistent, explained in the following section:

```
echo vf-num /sys/bus/pci/devices/parent PF bdf/sriov_numvfs
```

Using the `lspci` command, `grep` to search for Virtual Functions carved out from PF.

## CSP NIC Mode

The following table explains NIC mapping options on CSP and the corresponding performance implications:

| Mode        | Explanation                                         | Comments                                                                                                                                                                   |
|-------------|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SR-IOV mode | Virtual Network Functions created from physical NIC | Allows pNICs to be shared amongst VMs without sacrificing performance, since packets are switched in HW.<br>Maximum 32 virtual functions (VFs) can be configured per pNIC. |

## Making the Virtual Functions Persistent

To make the Virtual Functions persistent across reboots, use the editor of your choice to create an udev rule similar to the following, where you specify the intended number of VFs (in this example, 2), up to the limit supported by the network interface card.

```
vim /etc/udev/rules.d/enp14s0f0.rules
ACTION=="add", SUBSYSTEM=="net", ENV{ID_NET_DRIVER}=="ixgbe",
ATTR{device/sriov_numvfs}="2"
```

In the following example, replace `enp14s0f0` with the PF network device name(s) and adjust the value of `ENV{ID_NET_DRIVER}` to match the driver in use.

To find the driver in use, run the following command:

```
ethtool -i | grep driver
```

This will ensure the feature is enabled at boot-time.

## Post Host Reboot

This section discusses the actions to be followed post every reboot.

## Procedure

- 1 Post reboot all VMs will automatically be in the `STOP` state. All the VM names can be checked using `virsh list` output. Bring up all the VMs using the `virsh start <VM-name>` command.
- 2 If the VF config was not made persistent as described above, repeat the steps mentioned under Virtual Functions from PF.

VFs should be configured with unique MAC addresses through PF using the following command:

```
ip link set <PF_interface_name> vf <VF_index> mac <unique_mac_addr>
```

## Installing the Controller

This section explains about installing the NSX Advanced Load Balancer Controller.

Edit the KVM host file in `avinetworks.avicontroller_kvm` (inventory) according to your Controller configuration needs and run the playbook as follows:

```
ansible-playbook kvm.yml -i inventory-file -vv
```

## Additional Deployment Options

This section discusses about additional deployment options.

### Configuring MAC VLAN

MAC VLAN need to be configured before using the script to create the VM and pass through the interfaces. This is configured for the data path NICs by using Transparent VLAN or Tagged VLAN.

### Transparent VLAN

Transparent VLAN is used by the NIC to isolate VF traffic. From the perspective of VF driver or guest OS, the NIC is in access mode and it should not see any VLAN header. Transparent VLAN is to be specified by the system admin at the time of creation of VF. In the TX path, the NIC inserts the VLAN header in the packet before putting it on wire. In the RX path, the NIC strips the VLAN header before handing over the packet to VF driver.

For instance, if VF number 0 of the parent PF `eno2` has to be configured in transparent VLAN 109, the same can be configured via the following configuration.

The following is the code snippet for the configuration:

```
ifconfig eno2 down
ip link set dev eno2 vf 0 trust on
ip link set dev eno2 vf 0 spoofchk off
ip link set eno2 vf 0 vlan 0
ip link set eno2 vf 0 vlan 109
```

```
[Vlan num of the transparent vlan which needs to be configured]
ifconfig eno2 up
```

In this case, once the Service Engines are up on the UI, navigate to **Infrastructure > Service Engine** and select the relevant SE. Use the **Edit** button to go to the respective interface and configure the IP address/mask for either static or DHCP.

## Tagged VLAN

If in case the goal is to create a VLAN-tagged interface instead, the following steps can be used:

For instance, if vf number 0 of the parent PF eno2 has to be configured in tagged VLAN 109, the same can be configured via the following configuration.

```
ifconfig eno2 down
ip link set dev eno2 vf 0 trust on
ip link set dev eno2 vf 0 spoofchk off
ip link set eno2 vf 0 vlan 0
ifconfig eno2 up
```

Once the Service Engines are up on the UI, navigate to **Infrastructure > Service Engine** and select the relevant SE. Use the **Edit** button to go to **Create VLAN interface** and configure tagged VLAN interface in VLAN 109.

## Bond Interface

Bond Interfaces can be specified via the `SE Ansible yaml` file input. The name of the interfaces are taken as input and the bond-sequence can be specified as follows:

- Bond-if sequence: 1,2 3,4  
Implies interface 1,2 are in bond and interface 3,4 are in bond (Note the space between 1,2 and 3,4).
- Bond-if sequence: 1,2,3,4  
Implies interface 1,2,3,4 are in bond.

---

**Note** See the template example described in README section of the `avinetworks.avise_kvm` Ansible role for more details on SE yaml.

---

## Disabling NSX Advanced Load Balancer Controller and Service Engines

This section discusses about disabling the Controller and SEs.

VMs and their corresponding images can be cleared using the following commands:

```
virsh undefine vm-name
virsh destroy vm-name
```

At the root directory:

```
rm -rf /var/lib/libvirt/images/vm-name.qcow2
rm -rf vm-name
```

The `rm -rf vm-name` command will delete the local folder created in root directory for the relevant VM.

---

#### Note

- 1 If at anytime if one sees the disk space of the host getting exhausted it might be due to the N number of qcow2's being used in `/var/lib/libvirt/images/` as part of the VM's creation. Clear up any unused ones by deleting the respective VM and deleting the respective qcow2 image alongwith.
  - 2 Please ensure to manually cleanup (force-delete) the stale SE entries present in the Controller GUI post destroying SE VMs.
-

# Installing NSX Advanced Load Balancer in Microsoft Azure



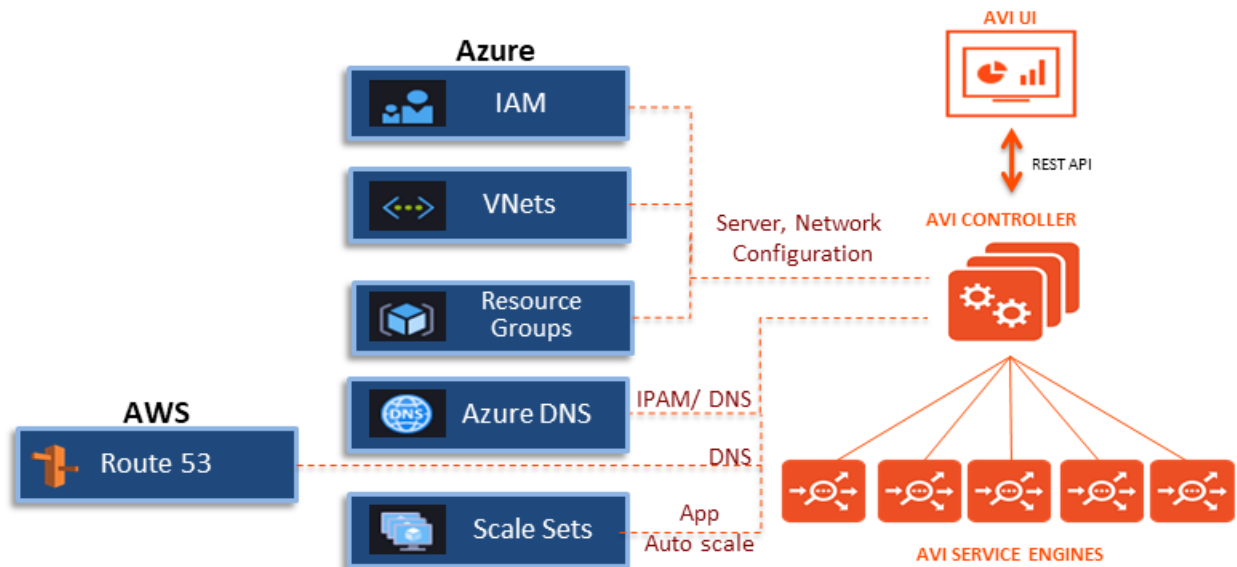
NSX Advanced Load Balancer supports Microsoft Azure integration. It offers elastic application services that extend beyond load balancing to deliver real-time application and security insights, simplify troubleshooting, auto scale predictively, and enable developer self-service and automation in Microsoft Azure.

The NSX Advanced Load Balancer deployment for Azure offers the following advantages:

- A single Controller cluster manages services for applications across multiple virtual networks (VNETs).
- Auto virtual IP address allocation and auto DNS registration allows full automation for application load balancing.
- The Controller automatically detects membership change in Azure scale set and updates pool membership for application autoscaling. NSX Advanced Load Balancer offers a flexible, yet comprehensive solution that is infrastructure-independent, agile, and elastic at a reduced Total Cost of Ownership (TCO). The following figure depicts a sample Azure deployment.

## NSX Advanced Load Balancer - Microsoft Azure

Figure 8-1.



The NSX Advanced Load Balancer Controller is available in Azure Marketplace and can be deployed directly from the marketplace.

After the Controller is deployed, Azure account details and credentials are provided. The Controller then connects to the Azure infrastructure and automatically provisions SEs as required.

This chapter includes the following topics:

- [Deployment Considerations](#)
- [Installing NSX Advanced Load Balancer Controller](#)
- [Configuring the NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Deployment Options](#)

## Deployment Considerations

This section explains the deployment considerations required for the installation process for NSX Advanced Load Balancer .

### Minimum Requirements

Before you deploy NSX Advanced Load Balancer in Microsoft Azure, your system must comply with all hardware, software, networking, and storage requirements.

For more information on system requirements, see [System Requirements](#).

## Ports and Protocols

The Service Engine subnet should allow incoming TCP connections on port 7 from the IP address 168.63.129.16. This is used by Azure to probe the Service Engine health.

For more information, see [Ports and Protocols](#) used by NSX Advanced Load Balancer for Management communication.

## Azure Roles and Permissions

This article provides access control requirements for deploying NSX Advanced Load Balancer solution in Microsoft Azure.

Azure Role-Based Access Control (RBAC) provides granular access to its resources. This enables configuring various users with IAM roles.

By default, Microsoft Azure has various built-in roles which define the level of access to the resources associated with it. For instance, the reader role has read-only access, where as, the owner role provides complete access.

In addition to built-in roles, Azure allows creating custom roles. Along with access permissions, custom role provides finer control over specific resources. The NSX Advanced Load Balancer solution interacts with a multitude of objects in the user's Azure subscription. This article provides guidelines on the permissions required for various objects to maximize security posture.

### Role Requirements for Deploying in Azure

The role requirements are defined for the following two stages:

The role requirements are defined for the following two stages:

- Controller Deployment
- Microsoft Azure Cloud Configuration

#### NSX Advanced Load Balancer Controller Deployment

The NSX Advanced Load Balancer Controller cluster needs to be deployed in a resource group where the controller admin has a role of **contributor** or higher.

#### Microsoft Azure Cloud Configuration

In Azure, the NSX Advanced Load Balancer Controller interacts with various resources and manages their lifecycles.

These operations require specific permissions. The contributor role provides sufficient level of permissions when attached to the required resource groups. However, NSX Advanced Load Balancer solution requires permissions that is a subset of those granted by the contributor role.

Hence, it is recommended to use a custom role that provides appropriate level of access that are limited to the required resources.

- 1 The NSX Advanced Load Balancer Controller is configured to deploy Service Engines in a specific resource group that is tied to the user's subscription. This user should have the contributor role for this resource group.
- 2 The deployed cloud can provide load balancing services to a VNet present in a different resource group from the one mentioned above. In addition, the NSX Advanced Load Balancer Controller uses the following resources in this resource group:
  - a If enabled during cloud configuration in the Controller, DNS zones to be used for Azure DNS
  - b Scale sets and Azure VMs used as back-end servers.

Resource groups provide an easy way to manage access to a group of resources in Azure. It is recommended to provision NSX Advanced Load Balancer Controller cluster in a new resource group of its own, for better isolation. Service Engine VM instances and all other Azure resources that are dynamically created by NSX Advanced Load Balancer Controller can reside in the same resource group (for small deployments), or can exist in a resource group of their own.

The Controller and Service Engines can be attached to an existing VNet for connectivity, independent of which resource group they reside in.

---

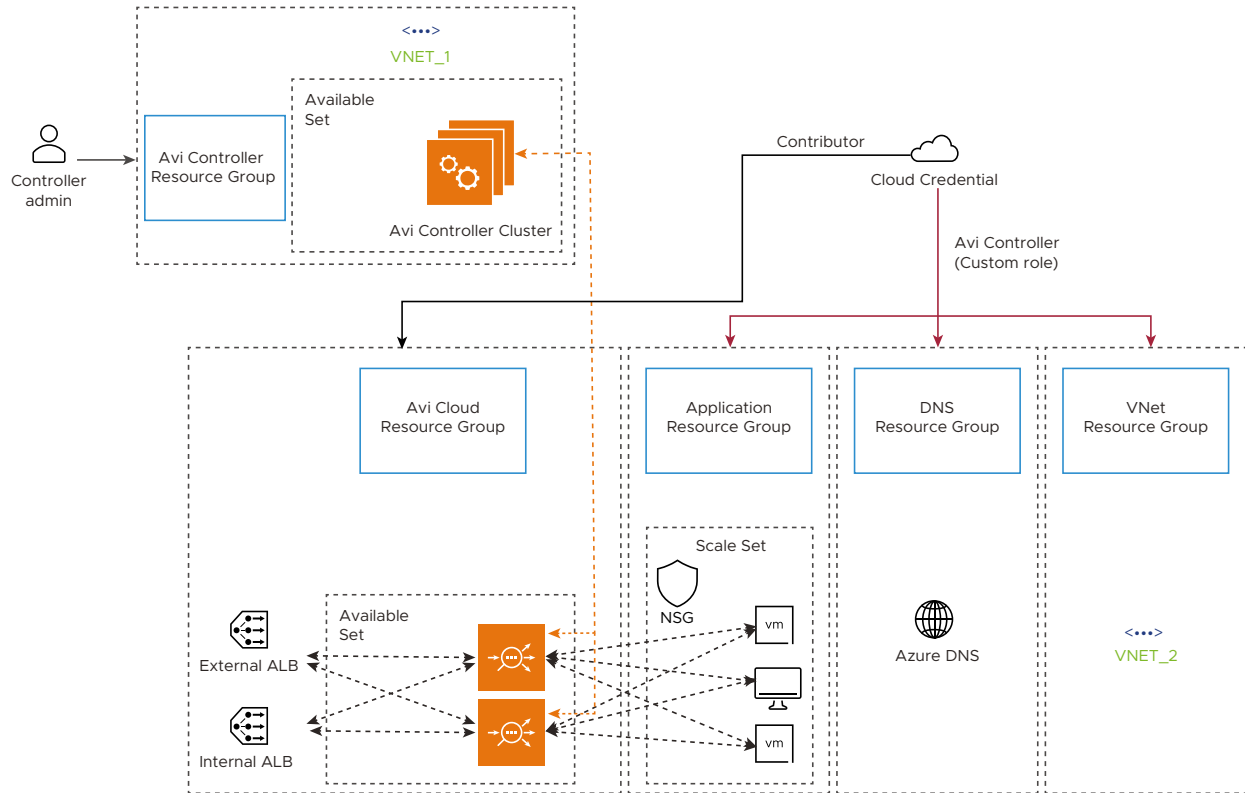
**Note**

- For resource group where the Controller is spawned, a role of contributor or higher is required.
  - For virtual network where the Service Engine instances are to be deployed, a role of NSX Advanced Load Balancer Controller or higher is required.
-



## Deployment Scenario

Figure 8-2. Figure 1. Role definition in NSX Advanced Load Balancer deployment for Azure



In Figure 1, Cloud Credential is a credential asset which could either be a service principle object, as in case of an application or an username/password credential set, as in the case of an user.

The NSX Advanced Load Balancer Controller belongs to NSX Advanced Load Balancer Controller Resource Group. The Controller admin exercises his privileges to deploy the Controller in this resource group.

The Controller creates the required resources in the NSX Advanced Load Balancer Cloud Resource Group. The credential asset needs contributor or a role of higher access to the NSX Advanced Load Balancer Cloud Resource Group.

The credential asset also needs custom role access to other resources, such as, VNet, DNS zones, and scale sets. This custom role helps define access to specific resources. Details on configuring custom role is provided in the following section.

The NSX Advanced Load Balancer cloud and VNet resource groups are configured as a part of the credential asset.

## Authentication Modes

This section discusses the authentication modes available for NSX Advanced Load Balancer integration with Azure.

## Azure Application ID

This article explains how to generate a unique application ID for an Azure Active Directory application using the Azure portal

Each Azure AD application is assigned a unique application ID using the App registration feature available on the Azure portal. For more information on Azure Application ID, refer to [How to Use an Azure AD application](#).

This section covers the followings:

- 1 Generating a Unique ID for Azure Active Directory Application
- 2 Associating Application to a Secret Key
- 3 Associating Application with IAM roles

### Generating a Unique ID for Azure Active Directory Application

Follow the steps mention below to generating a Unique ID for Azure Active Directory Application.

Log in to your Azure account using the Azure portal. Navigate to **Azure Active Directory (Azure AD)**. For the detailed steps to generate an application ID, see [How to: Use the portal to create an Azure AD application and service principal that can access resources](#).

#### Procedure

- 1 Log in to your Azure account using the Azure portal. Navigate to **Azure Active Directory (Azure AD)**. For the detailed steps to generate an application ID, see [How to: Use the portal to create an Azure AD application and service principal that can access resources](#)
- 2 Once the application is registered in Azure AD, a corresponding application ID is created.
- 3 Associate the application to a secret Key. For more information, see [How to: Use the portal to create an Azure AD application and service principal that can access resources](#).

### Associating Application with Azure roles

Follow the steps below to associate an application with Azure roles:

#### Procedure

- 1 To access resources in your subscription, assign the application created in the previous section to the desired role. The Application ID created in the previous steps can be used to provide additional permission to a role.
- 2 Assign a role (for example, the Contributor role) on a given subscription, resource group, vnet, or resource by selecting the respective resource and custome roles.

## MSI

This article explains how to configure MSI authentication on NSX Advanced Load Balancer for Microsoft Azure.

Managed services identity-based authentication for Microsoft Azure provides an automatically managed identity in Azure AD. You can use the identity to authenticate to any service that supports Azure AD authentication, including Key Vault, without any credentials in your code.

Microsoft Azure supports the following two types of managed identity service-based authentication:

- System-assigned managed identity
- User-assigned managed identity

### **System-assigned Managed Identity**

This feature is enabled directly on an Azure service instance. When the identity is enabled, Azure creates an identity for the instance in the Azure AD tenant that is trusted by the subscription of the instance. After the identity is created, the credentials are provisioned onto the instance. The lifecycle of a system-assigned identity is directly tied to the Azure service instance that it is enabled on. If the instance is deleted, Azure automatically cleans up the credentials and the identity in Azure AD.

### **User-assigned Managed Identity**

This feature is created as a standalone Azure resource. Through the create process, Azure creates an identity in the Azure AD tenant that is trusted by the subscription in use. After the identity is created, the identity can be assigned to one or more Azure service instances. The lifecycle of a user-assigned identity is managed separately from the lifecycle of the Azure service instances to which it is assigned. NSX Advanced Load Balancer release 18.1.4 supports managed services identity (MSI) authentication for Microsoft Azure. NSX Advanced Load Balancer only supports system-assigned managed identity. This article explains how to configure MSI authentication on NSX Advanced Load Balancer for Microsoft Azure.

The configuration of MSI for NSX Advanced Load Balancer consists the following sections:

- Configuring Microsoft Azure for MSI authentication
- Configuring NSX Advanced Load Balancer to support MSI authentication

### **Configuring Microsoft Azure for MSI Authentication**

This section explains about the MSI feature which is used during the deployment of NSX Advanced Load Balancer Controller.

This section covers the following:

- Enabling MSI Authentication
- Enabling MSI on NSX Advanced Load Balancer Controller VM
- Assigning Role to NSX Advanced Load Balancer Resource Group
- Assigning Role to VNet Resource Group

## Prerequisites

- For a resource group where the Controller is spawned, the role of a Contributor or higher is required.
- For the virtual network where the Service Engine instances are to be deployed, the role of the Controller or higher is required.

## Procedure

- 1 To enable MSI Authentication, see [Managed identities for Azure resources](#).
- 2 Assigning Role to NSX Advanced Load Balancer Controller Resource Group
  - a Navigate to the Cloud resource group and select Access Control (IAM). The Controller will create all its resources in this resource group.
  - b Add a new role assignment of Contributor or higher for the controller VM.
  - c Save the above configuration.
- 3 Assigning Role to VNet Resource Group.
  - a Navigate to the VNet resource group.
  - b Add a new role assignment of NSX Advanced Load Balancer Controller for the Controller VM. The custom role can be configured using Azure CLI, PowerShell, or REST API.
  - c Save the above configuration.
  - d Repeat the above steps for the DNS Application Group and Application Resource Group.

## Configuring NSX Advanced Load Balancer for MSI authentication

This section explains the steps to configure NSX Advanced Load Balancer for MSI authentication.

## Procedure

- 1 Follow the steps mentioned below to enable MSI authentication using NSX Advanced Load Balancer UI.
  - a Navigate to **Infrastructure > Cloud**. Create a new cloud of type Microsoft Azure.
  - b Enable the checkbox for Use **Azure Managed Service Identity (MSI) Authentication**. Continue the next steps to select the Vnet.

Figure 8-3. Enabling MSI

- c For Vnet, select the virtual network with which Virtual IPs are associated. Select the desired Resource Group. The Controller creates all its resources in this resource group.
  - d Save and navigate back to **Infrastructure > Cloud** to check the cloud created.
- 2 Enabling MSI Authentication using the NSX Advanced Load Balancer CLI. Login to the NSX Advanced Load Balancer shell prompt and use the configure cloud <cloud name> to enable MSI authentication for the cloud.
  - a Login to the shell prompt and use the configure cloud <cloud name> to enable MSI authentication for the cloud.

## Service Engine Disc Encryption

This section explains the steps to enable the Disc Encryption for the Service Engines.

Azure disk encryption is used to secure data hosted on or access through Azure virtual machines. Azure supports the following disk encryption types:

- Azure Disk Encryption
- Server-side Managed Disk Encryption
  - Platform-managed keys

- Customer-managed keys in customer-controlled hardware
- Customer-managed keys

By default, Microsoft-managed keys secure the data stored in a storage account on Azure VM. The customer-managed key provides additional control over the encryption method to the user.

Starting with NSX Advanced Load Balancer release 20.1, the use of the customer-managed key is supported for server-side disk encryption. A RSA key is imported to the Key Vault on Azure, or a new RSA key is generated to use the customer-managed key for the server-side encryption.

Azure-managed disks use envelope encryption to encrypt and decrypt the data. It encrypts data using an AES 256-based data encryption key (DEK). DEK is protected using customer keys, which is called key encryption key (KEK).

## NSX Advanced Load Balancer for Microsoft Azure - License Models

This section discusses the licensing option available for NSX Advanced Load Balancer deployment with Microsoft Azure

NSX Advanced Load Balancer supports the following two licenses which must be chosen during the cloud creation:

- Bring your own license
- Pay-as-you-go license

### Bring Your Own License (BYOL)

Please obtain the license from your NSX Advanced Load Balancer representative, and upload to the NSX Advanced Load Balancer Controller. The following license units are available:

- Cores (vCPUs)
- Service Engine Bandwidth (25Mbps, 200Mbps, and 1Gbps)

---

**Note** The NSX Advanced Load Balancer Controller includes a 30-day, full-featured BYOL trial license.

---

### Azure Pay-as-you-go (Azure PAYG)

When the Azure PAYG license is selected, a license file is not required. The licensing and usage is calculated based on the SEs instantiated in Azure. Usage and billing for the Azure PAYG license are calculated based on the time the SE is in the powered-on state in Azure, in an hourly increment.

The following bandwidth licenses are available:

- 25Mb (default)
- 200Mb1Gb

## NSX Advanced Load Balancer Controller's SKUs in Azure Marketplace

An SKU is the commercial name for a virtual machine image in Azure. NSX Advanced Load Balancer has two Controller SKUs in Azure Marketplace:

- **avi-vantage-adc-byol:** NSX Advanced Load Balancer Controller Version 17.2.x - BYOL
  - Use the 17.2.x SKU (currently, version 17.2.12) for production environments. The SKU version 17.2.x is currently the preferred, long-term release.
- **avi-vantage-adc-1801:** NSX Advanced Load Balancer Controller Version 18.1.x - BYOL and PAYG
  - Use the 18.1.x SKU for evaluating NSX Advanced Load Balancer's Azure PAYG licensing. This version is currently available as a Beta support only. It supports both the BYOL and the Azure PAYG license models. The license model should be selected when the Azure cloud is configured.

---

**Note** Both the SKUs are tagged as BYOL in the Azure marketplace as shown below, as there is no software cost associated with the NSX Advanced Load Balancer Controller.

---

For more information on these SKUs, see [VMware NSX Advanced Load Balancer \(Avi Networks\)](#).

## Service Engine SKUs

The followings new, hidden SE SKUs have been published in the Azure marketplace:

- **avi-vantage-se-1801-byol:** This SKU will be used to spin up an SE when the BYOL license model is selected. This SKU does not have a software cost associated with it.
- **avi-vantage-se-1801-payg-bw-25m:** This SE SKU will be used to spin up an SE when the PAYG, 25M bandwidth license is selected. This license model is the default for the PAYG licensing. This SKU has a software cost associated with it.
- **avi-vantage-se-1801-payg-bw-200m:** This SE SKU will be used to spin up an SE when the PAYG, 200M bandwidth license is selected. This SKU has a software cost associated with it.

When the BYOL license is selected,

- The **avi-vantage-se-1801-byol** license is used to create a SE. As this is a BYOL SKU, no software charge will be incurred.
- When Azure PAYG license is selected, the **avi-vantage-se-1801-payg-bw** SKUs will be used to create a Service Engine. As these SKUs have an hourly cost associated with them, the respective software charge will be incurred and billed directly by Azure.

## Microsoft Azure Resource Limits

This section discusses about the various resource limits to consider while integrating Microsoft Azure with NSX Advanced Load Balancer.

Microsoft Azure objects have predefined limits to the number of instances that can be instantiated.

These limits are based on the location of a given subscription. For instance, the total number of cores that can be used by the subscription in a particular location defines these limits.

The following limits must be increased appropriately, to allow scaling NSX Advanced Load Balancer virtual service and object creation in Microsoft Azure:

- Networking Limits
- Load Balancer Limits

## Networking Limits

Public IP addresses - Static. The default value is 20. This value should be increased if the deployment is expected to have more 20 public IPs.

## Load Balancer Limit

### 1 Frontend IP configuration - Basic

The default value is 10. It is recommended to set this to a higher value. Each virtual service IP and port combination consumes one frontend IP configuration.

### 2 Rules per resource - Basic

The default value is 150. It is recommended to increase this to a higher value. Each virtual service IP and port combination consumes one rule.

### 3 Load Balancers

The default value is 100. This limit should be raised as required, if more than 100 Service Engine groups are expected.

---

**Note** The above limits can be increased by submitting a request to Microsoft Azure using a support case. For more information, see [Azure subscription and service limits, quotas, and constraints](#).

---

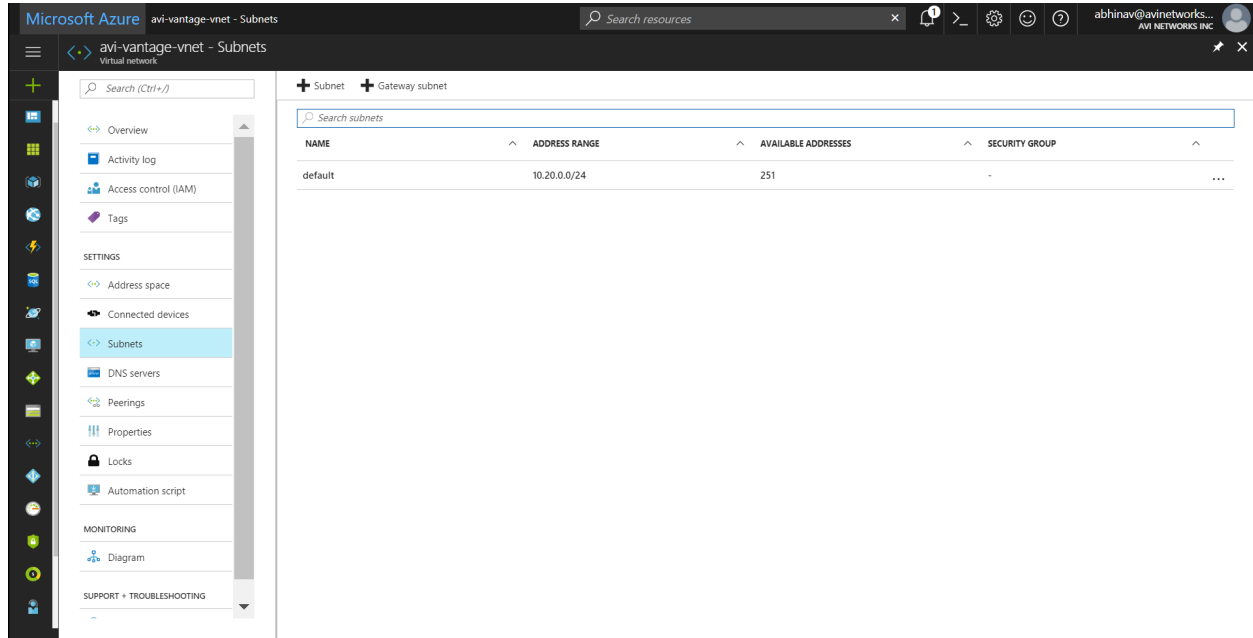
## Azure Virtual Network

This section explains the Azure Virtual Network option which is used while deploying the NSX Advanced Load Balancer.

The resource group must have an Azure Virtual Network (VNet) configured with a subnet. For the purpose of this document, the resource group avi-vantage will be used to deploy the NSX Advanced Load Balancer solution. As displayed in the screenshot below, this group has avi-vantage-vnet VNet, with an available address space of 10.20.0.0/16 and a subnet of 10.20.0.0/24.



Figure 8-4. Azure VNet



## Installing NSX Advanced Load Balancer Controller

This section covers the steps to deploy a NSX Advanced Load Balancer Controller in Microsoft Azure.

NSX Advanced Load Balancer is available in Azure Marketplace as a Bring Your Own License (BYOL) offering.

- 1 Access Azure Marketplace at <https://azuremarketplace.microsoft.com/> and log in using your Azure credentials. Alternately, you can log in using your Azure credentials at <https://portal.azure.com>. Marketplace Link:
- 2 Navigate to the NSX Advanced Load Balancer page on Azure Marketplace.
- 3 Click **Get it Now** to start the deployment process. If the deployment is via the Azure portal then create a new VM and search for NSX Advanced Load Balancer Networks. The NSX Advanced Load Balancer VM will show up in the search results.

Follow the given steps to initiate the deployment:

- 1 Provide the information requested under the Basics tab. Click on **OK** to continue to the next tab. Based on deployment scale considerations, choose an appropriate VM size.
- 2 In the **Settings** tab, select the following options:
  - a **Availability set:** It is recommended to use an availability set for the Controllers.
  - b **Storage:** Select Yes for managed disks.
  - c **Virtual network:** Create a new VNet, or use an existing VNet.

- d **Subnet:** Select a subnet for the NSX Advanced Load Balancer Controller management IP address to be allocated from.
  - e (Optional) Public IP address: Allocate an existing or new public IP address to the controller VM.
  - f **Network security group (firewall):** Apply an existing or new network security group to restrict traffic to the controller.
- 3 Click **OK** followed by Purchase to run final validations and initiate the deployment.

To ensure complete redundancy create a NSX Advanced Load Balancer Controller cluster. Creating the Controller cluster is an optional step.

Two additional Controller nodes can be added to create a 3-node NSX Advanced Load Balancer Controller cluster.

To create a Controller cluster,

- If deploying from the Microsoft Azure Marketplace, use the JSON template found here.
- If deploying from a downloaded version of VHD, use the JSON template found here.

Once the Controller is up, it can be configured via a web browser. The FQDN will be mentioned as an output of the template execution, as in this case *avicontrollerpubip.westus.cloudapp.azure.com*.

Follow the given steps to complete the initial configuration.

- 1 Provide credentials for the administrator account (Username: admin).
- 2 Provide DNS and NTP Settings (Can be edited later).
- 3 Provide an email address to be used for alerts from the controller (Can be set up at a later stage). Select **No Orchestrator** to complete the initial configuration.
- 4 Continue by clicking **No** for Support multiple Tenants (Multi-tenancy can be enabled later).
- 5 Once the setup is completed, the browser will automatically refresh to the NSX Advanced Load Balancer Controller dashboard. The next step is to configure Microsoft Azure cloud on NSX Advanced Load Balancer.

## Configuring the NSX Advanced Load Balancer Cloud Connector

This section discusses the steps required to create a cloud configuration of type Azure.

Use NSX Advanced Load Balancer UI to create a cloud configuration of type Azure, so that NSX Advanced Load Balancer can spin up Service Engines in the Azure VNet, and the load balance workloads present there.

Follow the given steps to complete the cloud configuration.

- 1 Log in to the NSX Advanced Load Balancer UI using the Controller IP address, navigate to **Infrastructure > Clouds**.
- 2 Click the **Create** button to add a new cloud. Provide a name, and select **Microsoft Azure** as the Cloud infrastructure type.
- 3 On the next tab, provide information related to the Azure account.

Figure 8-5.

The screenshot shows the 'New Cloud: Avi-Azure' configuration page in the NSX Advanced Load Balancer UI. The page is divided into three steps: Step 1: Select Cloud, Step 2: Credentials, and Step 3: Location/Network. Step 2 is currently active. The form includes the following fields and options:

- Name:** A text field containing 'Avi-Azure'.
- Azure User Credentials:** A section containing:
  - Azure Credentials:** A dropdown menu with 'Select Credentials'.
  - Subscription ID:** A text field with 'Subscription ID'.
- License Model:** Two radio buttons: 'Pay-as-you-go' (unselected) and 'Bring your own license' (selected).
- License Type:** A dropdown menu with 'Cores'.
- License Tier:** A dropdown menu with 'Enterprise 18'.

At the bottom of the form, there are 'Cancel', 'Previous', and 'Next' buttons.

- 4 NSX Advanced Load Balancer UI has the option to select the desired License Model.
- 5 Start by clicking **Create Credentials** tab and provide Azure credentials. You can either choose an Azure account username/password, or an Application ID.
- 6 In the screenshot below, the username method is used.

Figure 8-6. Azure Credentials

**New Azure Credentials: user**

User\*

user

Azure Credentials

Authentication

☒ Username/Password ☐ Application ID

Username\* ?

user

Password\* ?

....

Active Directory ?

Active Directory

Save

- 7 Select the license model that you want to use. You can either select the **Pay-as-you-go**, or the **Bring your own license** option. The below screenshot exhibits the option for the Pay-as-you-go license model. For the PAYG licence model, the license type is set to SE Bandwidth automatically.

Figure 8-7. Azure License

Infrastructure Dashboard Clouds Service Engine Service Engine Group Networks Routing GSLB User Credentials ? admin(admin)

**New Cloud: Avi-Azure**

Step 1: Select Cloud Step 2: Credentials Step 3: Location/Network

Name\*

Avi-Azure

Azure User Credentials

Azure Credentials\* ?

Select Credentials

Subscription ID\* ?

Subscription ID

License Model

☒ Pay-as-you-go ☐ Bring your own license

License Type ?

SE Bandwidth

Cancel Previous Next

- a For the **Bring your own license** model, you can use the drop-down menu to use the following licence types: Cores **SE Bandwidth**.

- 8 Save and select these newly created credentials and provide the Azure subscription ID.
- 9 Click **Next**. Provide the Azure location details. These details are associated with the location of resource group, the resource group and the VNet that can be used, and the subnet for the Service Engine management network.
- 10 Optionally, a DNS provider can be selected as well. Instead of Azure DNS, AWS Route 53 can also be used by selecting Other.
- 11 Click **Complete**, to provision the Azure cloud.
- 12 At this time, the Controller will upload the Service Engine VHD into an Azure storage account, so that SEs can be deployed as required by the applications.
- 13 Save the settings.

The system is now ready for virtual service creation.

## Additional Deployment Options

This section explains the additional deployment available for the integration of Microsoft Azure with NSX Advanced Load Balancer

### Configuring Service Engine Disk Encryption

This section explains the steps to enable the Disk Encryption option for the SEs.

Limitations:

- Only soft and hard RSA keys of size 2048 are supported. For more information, refer to Azure Key Management Table.
- All resources related to the customer-managed keys (Azure Key Vaults, disk encryption sets, VMs, disks, and snapshots) must be in the same subscription and region.
- Disks, snapshots, and images encrypted with customer-managed keys cannot move to another subscription.

### Configuring Microsoft Azure

To configure the key vault and Setting up your Azure Key Vault and DiskEncryptionSet, follow the steps from step 1 to 4 of [Server-side encryption of Azure Disk Storage](#).

### Configuring NSX Advanced Load Balancer for Disk Encryption

The Disk Encryption Set option is available on the NSX Advanced Load Balancer UI to select DES ID.

Navigate to **Infrastructure > Cloud > Location/Network** and use the drop-down menu to select the DES ID as shown below:

Figure 8-8. Service Engine Disk Encryption

The screenshot shows the 'Edit Cloud: azure-01' configuration window in the NSX Advanced Load Balancer GUI. The 'Location/Network' tab is selected, showing the 'Service Engine Network' dropdown set to 'jenkins-subnet-76 - 10.130.76.0/24'. Below this, there is a checkbox for 'Use Dedicated Management Interface' which is unchecked. The 'Service Engine' section shows a dropdown for 'Template Service Engine Group' set to 'None'. The 'Disk Encryption' section shows a dropdown for 'Disk Encryption Set' set to 'desjenkins - JENKINS-RESOURCE-GROUP-2'. The 'Other Settings' section is empty. At the bottom, there is a '+ Add Custom Tag' link and 'Cancel' and 'Save' buttons.

## Configuring Disk Encryption using the CLI

Server-side managed disk encryption with customer-managed keys allows the Controller to create encrypted SE Image and create SEs with encryption on OS and Azure Managed disk only.

Starting with NSX Advanced Load Balancer release 20.1.1, a new field `des_id` is introduced, which take DES resource ID as input for Azure cloud configuration.

```
[admin:controller]: > configure cloud Default-Cloud
[admin:controller]: cloud> azure_configuration
[admin:controller]:
cloud:azure_configuration> des_id /subscriptions/0eebbbed-14c0-462e-99e0-daaaaaaaaa9/
resourceGroups/avi-resource-group/providers/Microsoft.Compute/diskEncryptionSets/DESavi
[admin:controller]: cloud:azure_configuration> save
[admin:controller]: cloud> save
```

### Note

- SE image and SEs with encrypted OS and Azure Managed disk have the same DES ID.
- Different DES IDs are not supported among SE VMs or Image and SE VMs.

## Multiple Azure Load Balancers for Microsoft Azure Cloud

This section explains about the multiple Azure load balancers feature for Microsoft Azure Cloud.

NSX Advanced Load Balancer deployments in Microsoft Azure leverage the Azure Load Balancer (ALB) to provide an ECMP-like, Layer 3 scale-out architecture.

By default, the Controller creates two Azure load balancers per SE group, one internal and one external. This limits the number of virtual service IPs (VIPs) or ports supported on the Service Engine. Each virtual IP and port consumes a rule.

---

**Note** Currently the number of rules per NIC is limited to 300. The revised number will periodically be updated at [Load Balancer limits](#).

---

This feature is supported for basic ALB only.

---

Multi Azure load balancers are supported within a single NSX Advanced Load Balancer SE group. Consider an instance where 150 virtual services need to be created in an SE group, each with two front-end ports, 80 and 443. This would require 300 rules, which is the allowed limit. So, a new virtual service cannot be configured in the same SE group. However, with multi load balancers feature enabled, more virtual services can be created in the SE group. The Controller creates multiple Azure load balancers and distributes Service Engines across the availability sets.

The following are the advantages of enabling this feature:

- This feature is offered along with basic load balancer at no additional costs.
- The new load balancer is automatically created as required and scaled out seamlessly without any user intervention.
- Multi Azure load balancers are configured only for the SE group and so no configuration changes are required for the remaining virtual services.

## Enabling Multi Azure Load Balancer

Follow the steps below to enable multi Azure load balancers in an Azure cloud:

- 1 Configure the Service Engine group for Azure cloud.
- 2 Set the `enable_multi_lb` option on the Service Engine group for the cloud(Optional).
- 3 Set the following knobs for multi load balancer rules to restrict the number of rules used:
  - a Maximum ruler per Azure load balancer
  - b Maximum public VIPs per Azure load balancer

```
[admin:10-X-X-X]: > configure serviceenginegroup Default-Group

[admin:10-X-X-X]: serviceenginegroup> enable_multi_lb

[admin:10-X-X-X]: serviceenginegroup> max_rules_per_lb 150

[admin:10-X-X-X]: serviceenginegroup> max_public_ips_per_lb 30
```

## Migrating to Multi Azure Load Balancers

You can migrate to multi load balancers in an Azure cloud without deleting the virtual services.

Follow the steps below, in the order provided, to migrate the existing SE groups with virtual services:

- 1 Disable all virtual services placed on the target Service Engine group.
- 2 Delete all the Service Engines from the Service Engine group.
- 3 Enable multi load balancers and configure the knobs, as explained in the section above.
- 4 Enable all virtual services.



# Installing NSX Advanced Load Balancer in Cisco CSP

# 9

Cisco Cloud Services Platform is a software and hardware platform for data center Network Functions Virtualization (NFV). This section explains how to install NSX Advanced Load Balancer on Cisco CSP.

This chapter includes the following topics:

- [Considerations for Deploying NSX Advanced Load Balancer in Cisco CSP Environments](#)
- [Installing NSX Advanced Load Balancer Controller and Service Engine in CSP](#)
- [Configuring the NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Deployment Options](#)

## Considerations for Deploying NSX Advanced Load Balancer in Cisco CSP Environments

This section explains the prerequisites, performance details of NSX Advanced Load Balancer in CSP.

### Prerequisites

Ensure that the following prerequisites are met before installing the NSX Advanced Load Balancer Controller:

- Run CSP v2.3.1 at a minimum
- Use VIRTIO as the disk type while configuring all NSX Advanced Load Balancer VNFs on CSP (Controllers and SEs)
- i40evf bonds are supported only with CSP image version 2.4.1-185

### CSP NIC Modes

The following table explains three possible NIC mapping options on CSP and the corresponding performance implications:

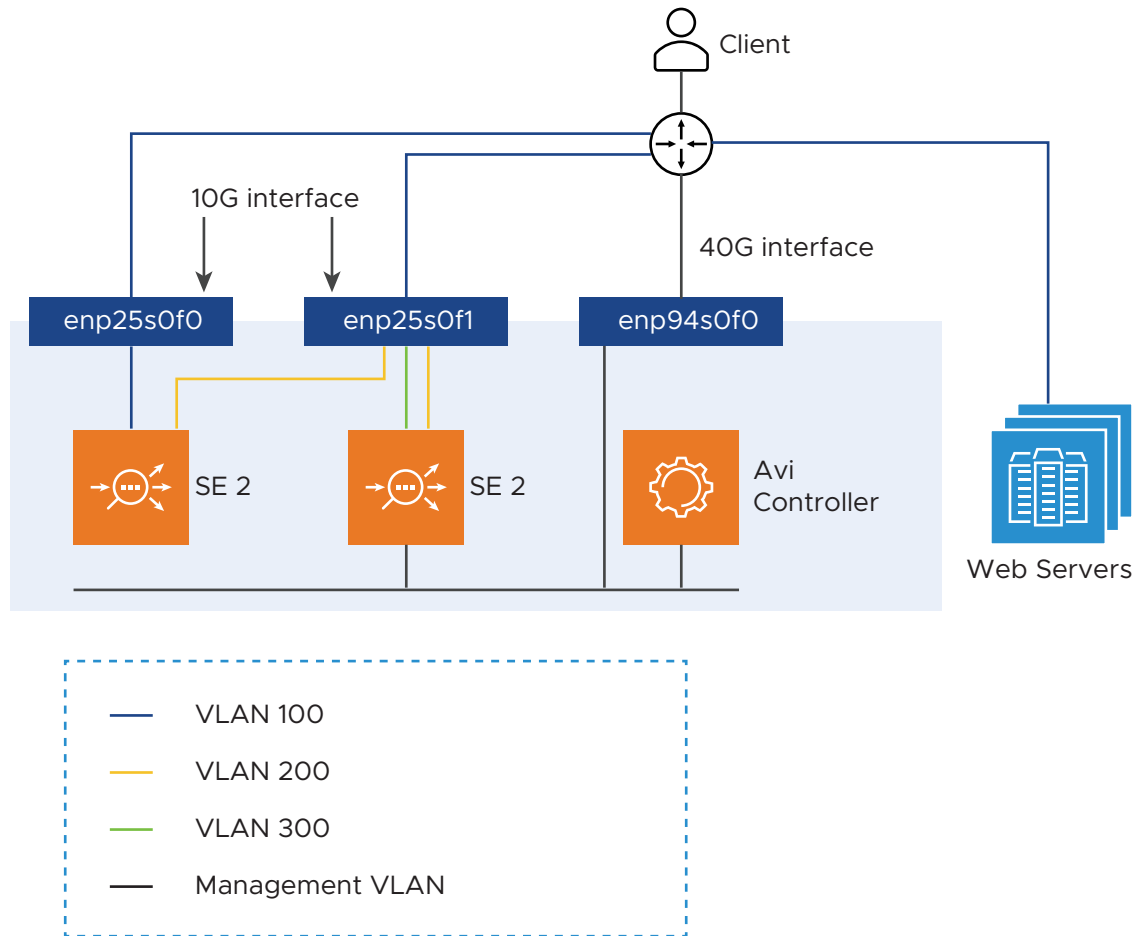
| Mode             | Description                                         | Comments                                                                                                                                                                                               | Drivers and Supported NICs                                                                                                                        |
|------------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Access mode      | Traffic switched using OVS                          | Allows physical NICs to be shared amongst VMs most generally. Performance is generally lower due to soft switch overhead.                                                                              | NA                                                                                                                                                |
| Passthrough mode | Physical NIC directly mapped to VM                  | Physical NIC is dedicated to a VM. With 1x10 Gbps pNIC per VM, a maximum of 2 VMs or 4 VMs can be created on a single CSP with 1 or 2 PCIe dual-port 10-Gbps NIC cards. Provides the best performance. | ixgbe driver supports these NICs: 82599, X520, X540, X550, X552<br>i40e driver supports these NICs: X710, XL710                                   |
| SR-IOV mode      | Virtual network functions created from physical NIC | Allows pNICs to be shared amongst VMs without sacrificing performance, as packets are switched in HW.<br>Maximum 32 virtual functions (VFs) can be configured per pNIC.                                | ixgbe-vf driver supports these NICs (and bonding): 82599, X520, X540, X550, X552<br>i40e-vf driver supports these NICs (and bonding): X710, XL710 |

## Topology

The SEs should be connected to pNICs of the CSP in passthrough (PCIe) or SR-IOV mode to leverage the DPDK capabilities of the physical NICs. For instance,

- 10 Gbps pNICs – enp25s0f0
- 40 Gbps pNICs – enp94s0f0

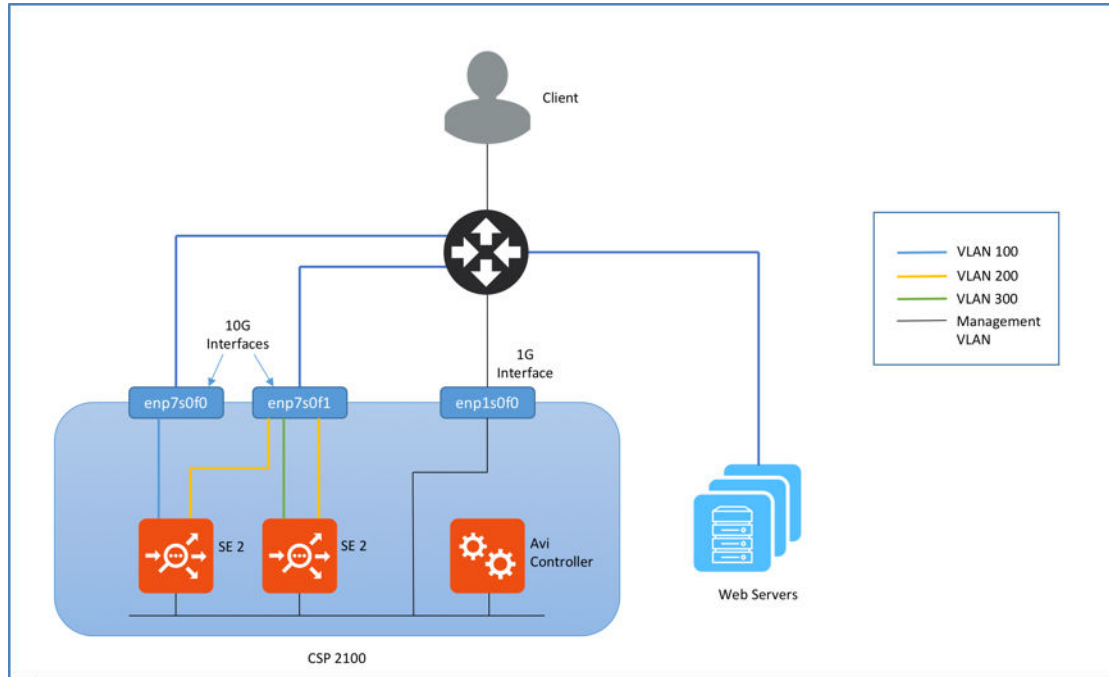
The SE can be connected to multiple VLANs on the pNICs' virtual functions (VF) in SR-IOV mode. The management network can be connected to the 10-Gbps pNIC.



## Topology of CSP 2100

The topology shown below consists of a Controller and Service Engines (SEs). To leverage the DPDK capabilities of the physical NICs, the SEs should be connected to the 10-Gbps enp7s0fx pNICs of the CSP 2100 in passthrough (PCIe) or SR-IOV mode. The SE can be connected to multiple VLANs on the pNICs' virtual functions (VF) in SR-IOV mode. The management network can be connected to the 1-Gbps pNIC.

Figure 9-1. CSP-2100 Topology



## Numad Service

`numad` is a user-level daemon that provides placement advice and process management for efficient CPU and memory utilization. On CSP servers, `numad` runs every 15 seconds and scans all candidate processes for optimization. The following are the required criteria for a candidate:

- RAM usage greater than 300 MB
- CPU utilization greater than 50% of one core

On CSP, `numad` takes each candidate process (which includes VNFs) and attempts to move either the process or its memory, so that they are on the same NUMA node, i.e., a physical CPU and its directly-attached RAM. On CSP servers, as it fails to move pages from memory, it takes about 10 to 30 seconds to move memory between NUMA nodes. This causes the VNFs processed by `numad` to hang during that duration. All processes, which includes even the VNFs, will become a candidate for `numad` after the holddown timer expires.

The `numad` service will be disabled by default for CSP servers running versions 2.2.4 and above.

---

**Note** Disabling `numad` is safe and has no adverse effects.

---

NSX Advanced Load Balancer SEs have high background CPU utilization, even when passing no traffic. This makes the NSX Advanced Load Balancer SE VNF a candidate for `numad`, which hangs the NSX Advanced Load Balancer SE VNF. This leads to various issues such as:

- Heartbeat failures
- BGP peer flapping
- Inconsistent performance

## Disabling Numad Service

You can disable `numad` service as follows:

- 1 Install Cisco CSP software
- 2 Execute the following commands from CSP CLI:

```
avinet-3# config terminal
Entering configuration mode terminal
avinet-3(config)# cpupin enable
avinet-3(config)# end
Uncommitted changes found, commit them? [yes/no/CANCEL] yes
Commit complete.
avinet-3#
```

**Note** If the CSP server is running an older version than v2.2.4, contact Cisco support to disable `numad`.

## Performance and Sizing for CSP

This section provides Cisco CSP 2100 and 5000 sizing guidelines for deploying NSX Advanced Load Balancer. Cisco CSP is generally available as a 2-socket server, with configurable options for CPU, memory, disk, and network interface cards.

### Cisco CSP 2100 Sizing Guidelines

Cisco CSP 2100 is an NFV platform based on Intel x86 and the KVM hypervisor. The Controller and Service Engines can be deployed on Cisco CSP 2100. The following sections provide sizing guidelines for common deployment scenarios. One of the CSP 2100's  $n$  vCPUs is reserved for CSP software. The rest ( $n-1$ ) are available for deploying NSX Advanced Load Balancer.

#### Deploying Service Engines on Cisco CSP 2100

NSX Advanced Load Balancer Service Engines (SEs) are deployed as virtual machines (VMs) on Cisco CSP 2100. The SEs can be as small as a 1-vCPU VM. The performance of the SEs is dependent on the number of vCPUs per SE VM (for throughput and SSL TPS) and memory (for concurrent connections). For optimal performance, SE VMs must be deployed on Cisco CSP 2100 with SRIOV enabled.

#### NSX Advanced Load Balancer SE Performance on Cisco CSP 2100

The following table outlines SE performance on Cisco CSP 2100, with Intel v3 CPU@2.4 GHz, hyper-threading disabled, and a 10-Gbps NIC with SRIOV enabled.

| SE VM Size   | L4 Performance |            | L7 Performance |            |
|--------------|----------------|------------|----------------|------------|
|              | CPS            | Throughput | RPS            | Throughput |
| 2-vCPU, 4 GB | 35,000         | 7 Gbps     | 75,000         | 5 Gbps     |

| L7 SSL Performance |            |           |              |
|--------------------|------------|-----------|--------------|
| SE VM Size         | Throughput | TPS (ECC) | TPS (RSA 2K) |
| 2-vCPU, 4 GB       | 2.4 Gbps   | 4,000     | 1,600        |

#### Note

- In general, the SSL/TLS performance (both throughput and TPS) scales linearly with number of vCPUs. For instance, 4-vCPU SE would be 2x the performance listed in table 1.
- SSL/TLS performance on Intel v4 CPU is 20%-30% higher than SSL/TLS performance Intel v3 CPU (at the same CPU clock speed).

In addition to the SEs, the Controller also can be deployed (as a VM) on Cisco CSP 2100. The Controller VM sizing is based on the system scale.

Depending on the network design, the Controller Cluster can be deployed on dedicated Cisco CSP 2100s, or can share Cisco CSP 2100s with SEs.

#### Recommended Cisco CSP 2100 Specifications

The following are recommended Cisco CSP 2100 specifications for various deployment scenarios, and applies to either CSP 2100 X1 or X2 models:

| Cisco CSP 2100 Sizing Recommendations |                            |                                                               |        |               |                                                 |
|---------------------------------------|----------------------------|---------------------------------------------------------------|--------|---------------|-------------------------------------------------|
| Scale                                 | NSX Advanced Load Balancer | CSP 2100 CPU                                                  | Memory | Storage       | NICs                                            |
| Low                                   | 19 Gbps SSL40k<br>SSL TPS  | 2.40 GHz<br>E5-264010<br>cores /<br>socket(Total 20<br>cores) | 128 GB | 4x 480 GB SSD | 2x dual-port 10G<br>NICs(total 4x 10G<br>ports) |

| Cisco CSP 2100 Sizing Recommendations |                            |                                                               |        |               |                                                 |
|---------------------------------------|----------------------------|---------------------------------------------------------------|--------|---------------|-------------------------------------------------|
| Medium                                | 27 Gbps SSL60K<br>SSL TPS  | 2.60 GHz<br>E5-269014<br>cores /<br>socket(total 28<br>cores) | 128 GB | 4x 480 GB SSD | 2x dual-port 10G<br>NICs(total 4x 10G<br>ports) |
| High                                  | 40 Gbps SSL100k<br>SSL TPS | 2.20 GHz<br>E5-269922<br>cores /<br>socket(total 44<br>cores) | 256 GB | 8x 480 GB SSD | 4x dual-port 10G<br>NICs(total 8x 10G<br>ports) |

### Note

- Maximum SE performance assumes all available vCPUs on the CSP 2100 are used for NSX Advanced Load Balancer SEs. If the Controller is deployed on the same CSP 2100, the maximum SE performance depends on the total number of vCPUs available for the SEs.
- This configuration is recommended when Cisco CSP 2100 is used to deploy multiple NFV solutions, for instance, virtual ASA.

### Creating Logical interfaces on Service Engine

The following table shows the logical interfaces that can be created on an SE:

| Logical Interface                                                       | Limit                                                                              |
|-------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Maximum number of vNICs attached to NSX Advanced Load Balancer's SE VNF | 23 (out of which vNICO would be used for NSX Advanced Load Balancer SE management) |
| Maximum number of VLANs on NSX Advanced Load Balancer SE                | 200                                                                                |
| Maximum number of VLANs per SRIOV passthrough interface                 | 64                                                                                 |

## Cisco CSP 5000 Sizing Guidelines

This section provides Cisco CSP 5000 sizing guidelines for deploying the NSX Advanced Load Balancer.

Cisco CSP 5000 is an NFV platform based on Intel x86 and the KVM hypervisor. The Controller and Service Engines can be deployed on Cisco CSP 5000.

### Deploying NSX Advanced Load Balancer Service Engines on Cisco CSP 5000

NSX Advanced Load Balancer Service Engines (SEs) are deployed as virtual machines (VMs) on Cisco CSP. The performance of the SEs depends on the number of vCPUs per SE VM (for throughput and SSL TPS) and memory (for concurrent connections). For optimal performance, SE VMs must be deployed on Cisco CSP with SRIOV enabled.

The following table outlines the core allocation for CSP:

| CSP Model | Total cores available | Cores reserved for OS | Available cores |
|-----------|-----------------------|-----------------------|-----------------|
| CSP 5216  | 16                    | 2                     | 14              |
| CSP 5228  | 28                    | 4                     | 24              |
| CSP 5436  | 36                    | 4                     | 32              |
| CSP 5444  | 44                    | 4                     | 40              |
| CSP 5436  | 56                    | 4                     | 52              |

### NSX Advanced Load Balancer SE performance on Cisco CSP 5000

The following table outlines the SE performance on Cisco CSP, Intel(R) Xeon(R) Gold 6152 CPU @ 2.10GHz, with hyper-threading disabled, and a 10-Gbps NIC with SRIOV enabled.

| L4 Performance                             |        |            | L7 Performance |            |
|--------------------------------------------|--------|------------|----------------|------------|
| SE VM Size                                 | CPS    | Throughput | RPS            | Throughput |
| 2 vcpu, 4 gig<br>(no dedicated dispatcher) | 60000  | 7 Gbps     | 100000         | 7 Gbps     |
| 4 vcpu, 4 gig<br>(dedicated dispatcher)    | 125000 | 9 Gbps     | 175000         | 8.5 Gbps   |
| 8 vcpu, 8 gig<br>(dedicated dispatcher)    | 150000 | 9 Gbps     | 350000         | 9 Gbps     |

| L7 SSL Performance (ECC)                   |      |            | L7 SSL Performance (RSA) |            |
|--------------------------------------------|------|------------|--------------------------|------------|
| SE VM Size                                 | CPS  | Throughput | RPS                      | Throughput |
| 2 vcpu, 4 gig<br>(no dedicated dispatcher) | 4000 | 3 Gbps     | 1500                     | 3 Gbps     |



| L7 SSL Performance (ECC)             |       |        | L7 SSL Performance (RSA) |        |
|--------------------------------------|-------|--------|--------------------------|--------|
| 4 vcpu, 4 gig (dedicated dispatcher) | 9000  | 5 Gbps | 3200                     | 5 Gbps |
| 8 vcpu, 8 gig (dedicated dispatcher) | 18000 | 9 Gbps | 7000                     | 9 Gbps |

### Note

- In general, the SSL/TLS performance (both throughput and TPS) scales linearly with the number of vCPUs. For instance, a 4-vCPU SE can provide performance twice the capacity as listed in the table above.
- Dedicated dispatcher CPU – Selecting this option dedicates the core that handles packet receive/transmit from/to the data network to just the dispatching function. This option is most applicable for an SE that has three or more vCPUs.

### Recommended Cisco CSP 5000 Specifications

The following are the recommended Cisco CSP 5000 specification for various deployment scenarios:

| Scale       | CSP Model | Memory | Storage  | NICs        | Maximum NSX Advanced Load Balancer SE performance |
|-------------|-----------|--------|----------|-------------|---------------------------------------------------|
| Extra small | CSP5216   | 256 GB | 3 TB SSD | 1x 40G NICs | 20 Gbps SSL 25K SSL TPS                           |
| Small       | CSP5228   | 256 GB | 3 TB SSD | 1x 40G NICs | 30 Gbps SSL 50K SSL TPS                           |
| Medium      | CSP5436   | 256 GB | 3 TB SSD | 2x 40G NICs | 40 Gbps SSL 70K SSL TPS                           |
| Large       | CSP5444   | 512 GB | 6 TB SSD | 2x 40G NICs | 50 Gbps SSL 90K SSL TPS                           |
| Extra Large | CSP5456   | 512 GB | 6 TB SSD | 2x 40G NICs | 65 Gbps SSL 115K SSL TPS                          |

### Creating Logical interfaces on Service Engine

The following table shows the logical interfaces that can be created on an SE:

| Logical Interface                                                       | Limit                                                                              |
|-------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Maximum number of vNICs attached to NSX Advanced Load Balancer's SE VNF | 23 (out of which vNIC0 would be used for NSX Advanced Load Balancer SE management) |
| Maximum number of VLANs on NSX Advanced Load Balancer SE                | 200                                                                                |
| Maximum number of VLANs per SRIOV passthrough interface                 | 64                                                                                 |

## Installing NSX Advanced Load Balancer Controller and Service Engine in CSP

This section explains the installation process of NSX Advanced Load Balancer Controller and SE.

### Installing NSX Advanced Load Balancer Controller in CSP

This section explains how to install NSX Advanced Load Balancer in Cisco CSP.

#### Uploading Controller Image

The following are the steps to upload the Controller image:

- 1 Log in to the **CSP dashboard** on a browser.
- 2 Navigate to **Configuration > Repository**.
- 3 Click the **+** sign, and browse to and select the **Controller qcow2** image.
- 4 Click **Upload**.

The Controller itself can have a day-zero YAML file before it is spun up. The YAML file needs to be imported into the repository prior to creating an image. Ensure that you have VNC access to the console. In a large deployment, this might require additional firewall rules.

---

**Note** A CSP cluster may have multiple copies (equal to the number of cluster hosts ) of the same image/YAML file. Consequently, when any deletions are required, all copies should be deleted. Typically, you can change a key (such as auth token) with the same filename and re-upload.

---

#### Configuring Controller Metadata File

To configure the Controller management Interface statically, the IP, netmask, and gateway information must be passed as a YAML file. The name of the metadata file must be in `avi_meta/*.yaml` format.

For instance, create a text file with name `avi_meta_controller.yaml` with contents as:

```
avi.mgmt-ip.CONTROLLER: "10.128.2.20"
avi.mgmt-mask.CONTROLLER: "255.255.255.0"
avi.default-gw.CONTROLLER: "10.128.2.1"
```

Here `avi.mgmt-ip.CONTROLLER` is the management IP for the NSX Advanced Load Balancer Controller, `avi.mgmt-mask.CONTROLLER` is the network mask and `avi.default-gw.CONTROLLER` is the gateway IP address for the management network.

---

**Note** Ensure that you replace the IP address in the example with the correct ones for your network. Upload this metadata file to the CSP repository.

---

## Deploying the NSX Advanced Load Balancer Controller using CSP UI

This section describes how to deploy Controller using CSP UI.

The following are the steps to deploy the Controller using the CSP UI:

### Procedure

- 1 Navigate to **Configuration > Services**.
- 2 Click **+**.

---

**Note** The disk size of any CSP image cannot be changed. To avoid deletion and recreation of the entire configuration, have an informed idea of deployment.

---

- 3 Specify the **Service Name**.
- 4 Select the host from the **Target Host Name** list. In version CSP 2.1.0, on a CSP cluster, you can select the HA host name.
- 5 Leave the **VNF Management IP** field blank. This is set using the **Day Zero Config**.
- 6 Select the **controller.qcow2** image from the **Image Name** list.
- 7 Select the Controller metadata file from the **Day Zero Config** drop-down menu.
- 8 Set the resource values for Disk, CPU and RAM.
- 9 Click **+** to add a vNIC and connect it to **enp1s0f0** in access mode.

---

**Note** If the management network is on a different VLAN, specify the VLAN number in the **VLAN** field, and click on **VLAN Tagged** to enable tagging.

---

- 10 Specify a password for the console login using VNC, if required.
- 11 Click **Deploy**.

## Deploying the NSX Advanced Load Balancer Controller using REST API

This section describes how to deploy the Controller using REST API.

CSP uses basic authentication for the REST API.

## Procedure

- 1 Use the following `curl` command to create the Controller service:

```
curl -X POST --user admin:AviCsp@2100 -H "Content-Type: application/json" -d '{
 "service":{
 "name":"Controller",
 "power":"on",
 "iso_name":"controller.qcow2",
 "day0_filename":"avi_meta_controller.yml",
 "numcpu":8,
 "memory":24576,
 "vnics":{
 "vnic":[
 {
 "nic":"0",
 "type":"access",
 "tagged":"false",
 "network_name":"enpls0f1"
 }
]
 }
 }
}' -k "https://10.8.3.106/api/running/services/"
```

- 2 The CSP should reply with status code 201 Created. To verify, get all installed services using following `curl` command:

```
curl -X GET --user username:password -H "Content-Type: application/json" -k "https://10.8.3.106/api/running/services/service/Controller"
```

## Results

The following output is generated:

```
{
{
 "vsb:service": {
 "name": "Controller",
 "uuid": "368bf2e5-7590-4efc-b19b-2a501e616933",
 "memory": 24576,
 "numcpu": 8,
 "macid": 153,
 "disk_size": "64.0",
 "iso_name": "controller.qcow2",
 "power": "on",
 "day0_filename": "avi_meta_controller.yml",
 "vnics": {
 "vnic": [
 {
 "nic": 0
 }
]
 },
 "operations": {
 "export": "/api/running/services/service/Controller/_operations/export",
 "monitor": "/api/running/services/service/Controller/_operations/monitor"
 }
 }
}
```

## Installing NSX Advanced Load Balancer Service Engine in CSP

This section explains the workflow of deploying an SE on CSP, with data NICs in SR-IOV passthrough mode.

Not every deployment will use SR-IOV, but if it is, it must be configured on the CSPs beforehand, for instance, numVFs. NSX Advanced Load Balancer supports IPv6 for SE data interfaces.

The following are the steps to install NSX Advanced Load Balancer SE:

### Procedure

- 1 On the Controller, navigate to **Infrastructure > Clouds**.
- 2 Click the download icon on **Default Cloud** row and select **Qcow2**.
- 3 Upload `se.qcow2` to the CSP repository.

### Uploading SE Metadata File

To configure SE management interface statically, the IP, netmask and gateway information must be passed as a YAML file. The name of the metadata file must be in `avi_meta/*.yaml` format.

For instance, create a text file with name `avi_meta_se.yaml` with contents as:

```
avi.mgmt-ip.SE: "10.128.2.18"
avi.mgmt-mask.SE: "255.255.255.0"
avi.default-gw.SE: "10.128.2.1"
AVICNTRL: "10.10.22.50"
AVICNTRL_AUTHTOKEN: "febab55d-995a-4523-8492-f798520d4515"
AVITENANT_UUID: 'tenant-f3fd4914-01e2-4fbf-b5bc-65b054700cee'
```

Where,

- `avi.mgmt-ip.SE` – Management IP for NSX Advanced Load Balancer SE
- `avi.mgmt-mask.SE` – Network mask
- `avi.default-gw.SE` – Gateway IP address for the management network
- `AVICNTRL` – Management IP of the NSX Advanced Load Balancer Controller

Replace the IP address in the above example with correct ones for your network.

`AVITENANT_UUID` (optional) is the UUID of the tenant on the Controller to which the SE must connect. If this field is omitted, the SE will connect to the admin tenant by default.

`AVICNTRL_AUTHTOKEN` is the authentication token used to authenticate SE-to-Controller communication.

The following are the steps to generate the authentication token:

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Click the key icon on the **Default-Cloud** row to view the authentication token key. The authentication token has a validity timeout of 1 hour by default.

3 Copy the authentication token.

Upload this metadata file to the CSP repository.

## Enabling SRIOV

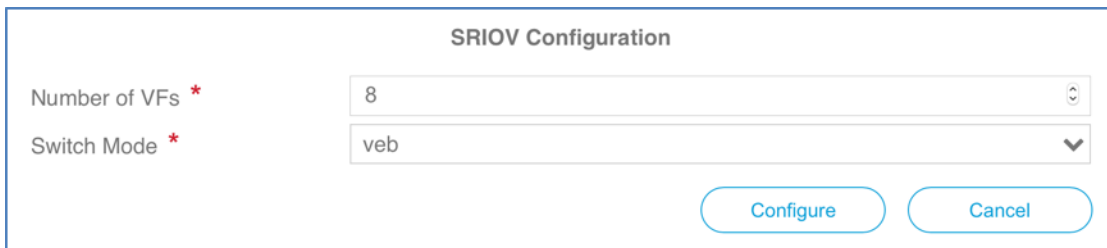
SR-IOV must be enabled on the CSP pNIC.

The following are the steps to enable SR-IOV on `enp94s0f0`:

### Procedure

- 1 Navigate to **Configuration > SRIOV Config**. Click on the settings icon for the **Enable SRIOV** option.
- 2 Under **SRIOV Configuration**, set the **Number of VFs** to a desired number and **Switch Mode** to veb.

Figure 9-2. SRIOV CSP 2100



The screenshot shows a configuration window titled "SRIOV Configuration". It contains two input fields. The first field is labeled "Number of VFs" with a red asterisk, and its value is "8". The second field is labeled "Switch Mode" with a red asterisk, and its value is "veb". At the bottom right of the window are two buttons: "Configure" and "Cancel".

- 3 Confirm the SR-IOV configuration.
- 4 Repeat the above steps to configure `enp94s0f1` for SR-IOV if required.

## Deploying Service Engine in SR-IOV Mode

This section explains about deploying SE in SR-IOV mode.

The following are the steps to deploy the SE using the CSP UI:

### Procedure

- 1 Navigate to **Configuration > Services**.

- Click **+**. Refer to the System Requirements Section for recommendations on a minimum production SE configuration.

**Figure 9-3. SRIOV SE Mode**

☒ Create Service    ☐ Create Service using Template

Name: \*

Target Host Name: \*

VNF Management IP:

Image Name: \*

☒ Day Zero Config

|   | Source File Name       | Destination File Name  | Action |
|---|------------------------|------------------------|--------|
| 1 | avi_meta_data_se-1.yml | avi_meta_data_se-1.yml |        |

Number of Cores:   
 Available Cores: 16

☒ Do you want to resize disk?

Disk Space (GB):   
 Available Disk Space (GB): 2563 | Minimum Disk Size (GB) : 10

RAM (MB):   
 Available RAM (MB): 196302

☐ NFS Storage

Disk Type: ☐ IDE    ☒ VIRTIO

- Specify the **Service Name**.
- Click **Target Host Name** and select the host from the list.
- Leave the **VNF Management IP** field blank. This is set using the Day Zero Config.
- Click **Image Name** and select the `se.qcow2` image from the list.
- Click **Day Zero Config** drop-down and select the SE metadata file.
- Set the resource values for Disk, CPU and RAM.



- 9 Click + to add a vNIC and connect it to `enp1s0f0` in access mode.

Figure 9-4. SRIOV SE Mode1

|                 |                                                                                                       |
|-----------------|-------------------------------------------------------------------------------------------------------|
| Name: *         | <input type="text" value="vnic0"/>                                                                    |
| VLAN Type:      | <input checked="" type="radio"/> Access <input type="radio"/> Trunk <input type="radio"/> Passthrough |
| VLAN:           | <input type="text" value="range 1-1000,1025-4094"/>                                                   |
| VLAN Tagged:    | <input type="radio"/> True <input checked="" type="radio"/> False                                     |
| Native VLAN:    | <input type="text" value="range 0-4095"/>                                                             |
| Model:          | <input type="radio"/> Virtio <input checked="" type="radio"/> e1000                                   |
| Network Type:   | <input type="radio"/> Internal <input checked="" type="radio"/> External                              |
| Network Name: * | <input type="text" value="enp1s0f0"/> ▼                                                               |

- 10 Click + to add a vNIC and connect it to `enp134s0f0` in SR-IOV mode.
- 11 Specify a password for console login using VNC, if required.
- 12 Click **Deploy**.
- 13 Verify that the SE is able to connect to the Controller by navigating to **Infrastructure > Dashboard** on the Controller UI (this may take a few minutes).

---

**Note** PF devices can be passed through to Service Engines. The method is the same as configuring a SR-IOV VF, with the exception that PCIE has to be selected for PF devices in place of SR\_IOV for VF devices.

---

## Configuring the NSX Advanced Load Balancer Cloud Connector

This section explains the steps to create a no-orchestrator cloud.

The following are the steps to upload the SE image in a cloud:

### Procedure

- 1 Create a no-orchestrator cloud.
- 2 Click the user-created cloud and select **Qcow2**.
- 3 Upload `se.qcow2` to the CSP repository.

## Additional Deployment Options

This section explains the additional deployment options.

## Configuring Dedicated Interfaces for HSM and ASM Communication on Cisco CSP

A Hardware Security Module (HSM) is a physical computing device that safeguards and manages digital keys for strong authentication and provides cryptoprocessing. NSX Advanced Load Balancer supports configuration of dedicated interfaces on the Controller and SEs for HSM and sideband (ASM) communication on Cisco CSP. HSM and ASM communication are supported for both existing and new NSX Advanced Load Balancer setups.

## Configuring Dedicated Interfaces for HSM Communication on NSX Advanced Load Balancer Service Engines

This section explains the configuration of dedicated interfaces for HSM communication on a new SE and an existing SE.

### Configuring Dedicated Interfaces for HSM Communication on a New NSX Advanced Load Balancer Service Engine

The dedicated HSM interfaces on the Service Engines use the following YAML configuration parameters:

- `avi.hsm-ip.SE`
- `avi.hsm-static-routes.SE`
- `avi.hsm-vnic-id.SE`

For configuration on new SEs, these parameters are provided in the day-zero YAML file.

## YAML Parameters

| YAML Parameter           | Description                                                                                                                                                                                                                                                                                                                                | Format                                                                                                    | Example                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| avi.hsm-ip.SE            | IP address of the dedicated HSM vNIC on the SE (this is NOT the IP address of the HSM)                                                                                                                                                                                                                                                     | IP-address/subnet-mask                                                                                    | avi.hsm-ip.SE:<br>10.160.103.227/24                                                            |
| avi.hsm-static-routes.SE | These are comma-separated, static routes to reach HSM devices. Even /32 routes can be provided.<br><br>If there is a single static route, provide the same and ensure the square brackets are matched. Also, if HSM devices are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [hsm network1/mask1 via gateway1, hsm network2/mask2 via gateway2 ] OR [ hsm network1/mask1 via gateway1] | avi.hsm-static-routes.SE:<br>[ 10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2] |
| avi.hsm-vnic-id.SE       | ID of the dedicated HSM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface and vNIC2 is data-out interface)                                                                                                                                                                                         | numeric vNIC ID                                                                                           | avi.hsm-vnic-id.SE: '3'                                                                        |

## Instructions

A sample YAML file for the day zero configuration on the CSP is shown below:

```
bash# cat avi_meta_data_dedicated_hsm_SE.yml
avi.mgmt-ip.SE: "10.128.2.18"
avi.mgmt-mask.SE: "255.255.255.0"
avi.default-gw.SE: "10.128.2.1"
AVICNTRL: "10.10.22.50"
AVICNTRL_AUTHTOKEN: "febab55d-995a-4523-8492-f798520d4515"
avi.hsm-ip.SE: 10.160.103.227/24
avi.hsm-static-routes.SE:[10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2]
avi.hsm-vnic-id.SE: '3'
```

Once the SE is created with the Day Zero configuration file and appropriate virtual NIC interfaces are added to the SE service instance on Cisco CSP, verify that the dedicated vNIC configuration is applied successfully and the HSM devices are reachable via this interface. In this case, interface eth3 (dedicated HSM interface) is configured with IP 10.160.103.227/24.

Login into the bash prompt of the SE and use `IP route` command and run a ping test to check reachability of the dedicated interface IP.

```
bash# ssh admin@<SE-MGMT-IP>
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.103.227 Bcast:10.160.103.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000

 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.1.0/24 via 10.160.103.1 dev eth3
10.128.2.0/24 via 10.160.103.2 dev eth3
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.103.0/24 dev eth3 proto kernel scope link src 10.160.103.227
bash# ping -I eth3 <HSM-IP>
ping -I eth3 10.128.1.51
PING 10.128.1.51 (10.128.1.51) from 10.160.103.227 eth3: 56(84) bytes of data.
64 bytes from 10.128.1.51: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for HSM Communication on the Service Engine.

The dedicated HSM interfaces on the SEs use the following configuration parameters:

- `avi.hsm-ip.SE`
- `avi.hsm-static-routes.SE`
- `avi.hsm-vnic-id.SE`

For the existing SEs, these parameters can be populated in the `/etc/ovf_config` file.

---

**Note** All parameters in this file are comma-separated and the file format is slightly different from the YAML file used for spinning up new SEs. However, the parameters and their respective formats are exactly the same as they are for new SEs.

---

## YAML Parameters

| YAML Parameter           | Description                                                                                                                                                                                                                                                                                                                                | Format                                                                                                    | Example                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| avi.hsm-ip.SE            | IP address of the dedicated HSM vNIC on the SE (this is NOT the IP address of the HSM)                                                                                                                                                                                                                                                     | IP-address/subnet-mask                                                                                    | avi.hsm-ip.SE:<br>10.160.103.227/24                                                            |
| avi.hsm-static-routes.SE | These are comma-separated, static routes to reach HSM devices. Even /32 routes can be provided.<br><br>If there is a single static route, provide the same and ensure the square brackets are matched. Also, if HSM devices are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [hsm network1/mask1 via gateway1, hsm network2/mask2 via gateway2 ] OR [ hsm network1/mask1 via gateway1] | avi.hsm-static-routes.SE:<br>[ 10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2] |
| avi.hsm-vnic-id.SE       | ID of the dedicated HSM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                        | numeric vNIC ID                                                                                           | avi.hsm-vnic-id.SE: '3'                                                                        |

## Instructions for Configuring CSP

To add a dedicated HSM vNIC on an existing SE CSP service, perform the following steps:

**Note** In the sample configuration provided below, vNIC3, the fourth NIC on the CSP service is used.

- 1 Navigate to **Configuration > Service > Action > Power Off** to power off the SE service using CSP user interface.
- 2 Add a new vNIC to the SE with desired parameters by navigating to **Configuration > Service > Action > Service Edit > Add vnic**. Provide VLAN ID, VLAN type, VLAN tagged, Network Name, Model, etc., and click on the **Submit** button.
- 3 To power on the SE service on CSP UI, navigate to **Configuration > Service > Action > Power On**.

## Instructions for Configuring NSX Advanced Load Balancer Service Engine

Perform the following steps using the Service Engine bash shell.

```
ssh admin@<SE-MGMT-IP>
bash#
bash# sudo su
bash# /opt/avi/scripts/stop_se.sh
bash# mv /var/run/avi/ovf_properties.saved /home/admin
```

**Note** Perform a move operation; do not copy this file. Edit it to provide the three comma-separated, HSM-dedicated NIC related parameters. The file is as follows:

```
bash# cat /home/admin/ovf_properties.saved
AVICNTRL: 10.128.2.18, AVICNTRL_AUTHTOKEN: 1403771c- fc59-4d76-89b2-b3c35682b342,
avi.default-gw.SE: 10.128.2.1,
avi.hsm-ip.SE: 10.160.103.227/24,
avi.hsm-static-routes.SE:[10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2],
avi.hsm-vnic-id.SE: '3',
avi.mgmt-ip.SE: 10.128.2.27, ovf_source: CSP,
uuid: FCE9B12D-A1B0-4EF3-B922-BDC2A5F8AA11

bash# cp /home/admin/ovf_properties.saved /etc/ovf_config
bash# /opt/avi/scripts/start_se.sh
```

Verify that the dedicated vNIC information is applied correctly and the HSM devices are reachable via this interface. In this sample configuration, the eth3 dedicated HSM interface is configured with IP 10.160.103.227/24.

```
bash# ssh admin@<SE-MGMT-IP>
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.103.227 Bcast:10.160.103.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.1.0/24 via 10.160.103.1 dev eth3
10.128.2.0/24 via 10.160.103.2 dev eth3
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.103.0/24 dev eth3 proto kernel scope link src 10.160.103.227
bash# ping -I eth3 <HSM-IP>
ping -I eth3 10.128.1.51
PING 10.128.1.51 (10.128.1.51) from 10.160.103.227 eth3: 56(84) bytes of data.
64 bytes from 10.128.1.51: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for ASM Communication on NSX Advanced Load Balancer Service Engines

This section explains the configuration of dedicated interfaces for ASM communication on a new SE and an existing SE.

### Configuring Dedicated Interfaces for ASM Communication on a New NSX Advanced Load Balancer Service Engines

The dedicated sideband interfaces on SEs use the following YAML configuration parameters:

- `avi.asm-ip.SE`
- `avi.asm-static-routes.SE`
- `avi.asm-vnic-id.SE`

For new SEs, these parameters can be provided in the day-zero YAML file.

| YAML Parameter                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                        | Format                                                                                                              | Example                                                                                                      |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>avi.asm-ip.SE</code>            | This is the IP address of the dedicated sideband interface on the SE (this is NOT the self IP or virtual service IP of the ASM device).                                                                                                                                                                                                                                                                                                            | IP-address/subnet-mask                                                                                              | <code>avi.asm-ip.SE:</code><br>10.160.103.227/24                                                             |
| <code>avi.asm-static-routes.SE</code> | These are comma-separated, static routes to reach the sideband ASM virtual service IP. Even /32 routes can be provided. The gateway will be the self IP of the ASM device.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched.<br><br>Also, if the ASM virtual service IPs are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [asm-vip-network1/mask1 via gateway1, asm-vip-network2/mask2 via gateway2] or [asm-vip-network1/mask1 via gateway1] | <code>avi.asm-static-routes.SE:</code><br>[169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2] |
| <code>avi.asm-vnic-id.SE</code>       | ID of the dedicated ASM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                                                                                                                                | numeric vNIC ID                                                                                                     | <code>avi.asm-vnic-id.SE: '3'</code>                                                                         |

## Instructions

A sample SE YAML file for the day-zero configuration on the CSP will be as follows:

```
bash# cat avi_meta_data_dedicated_asm_SE.yml

avi.mgmt-ip.SE: "10.128.2.18"
avi.mgmt-mask.SE: "255.255.255.0"
avi.default-gw.SE: "10.128.2.1"
AVICNTRL: "10.10.22.50"
AVICNTRL_AUTHTOKEN: "febab55d-995a-4523-8492-f798520d4515"
avi.asm-vnic-id.SE: '3'
avi.asm-static-routes.SE: [169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2]
avi.asm-ip.SE: 10.160.102.227/24
```

Once the SE is created with this day-zero configuration and appropriate virtual NIC interfaces are added to the SE service instance on CSP, verify that the dedicated vNIC configuration is applied successfully and the ASM virtual service IPs are reachable via this interface. In this case, the interface eth3 is dedicated sideband ASM interface and it is configured with IP 10.160.102.227/24.

```
bash# ssh admin@<SE-MGMT-IP>
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.102.227 Bcast:10.160.102.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.102.0/24 dev eth4 proto kernel scope link src 10.160.102.227
169.254.1.0/24 via 10.160.102.1 dev eth3
169.254.2.0/24 via 10.160.102.2 dev eth3
bash# ping -I eth3 <ASM-VIP>
ping -I eth3 169.254.1.10
PING 169.254.1.10 (169.254.1.10) from 10.160.102.227 eth3: 56(84) bytes of data.
64 bytes from 169.254.1.10: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for ASM Communication on an Existing NSX Advanced Load Balancer Service Engine

The dedicated sideband interfaces on SEs use the following configuration parameters:

- avi.asm-ip.SE
- avi.hsm-static-routes.SE
- avi.asm-vnic-id.SE



For the existing SEs, these parameters can be populated in the `/etc/ovf_config` file.

**Note** All parameters in this file are comma-separated and the file format is slightly different from the YML file used for spinning up new SEs. However, the parameters and their respective formats are exactly the same as they are for new SEs.

| YAML Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                  | Format                                                                                                              | Example                                                                                         |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| avi.asm-ip.SE            | IP address of the dedicated ASM vNIC on the SE (this is NOT the IP address of the ASM)                                                                                                                                                                                                                                                                                                                                                       | IP-address/subnet-mask                                                                                              | avi.asm-ip.SE:<br>10.160.103.227/24                                                             |
| avi.hsm-static-routes.SE | These are comma-separated, static routes to reach the sideband ASM virtual service IPs. Even /32 routes can be provided. The gateway will be the self IP of the ASM device.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if the ASM virtual service IPs are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [asm-vip-network1/mask1 via gateway1, asm-vip-network2/mask2 via gateway2] or [asm-vip-network1/mask1 via gateway1] | avi.asm-static-routes.SE:<br>[169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2] |
| avi.asm-vnic-id.SE       | ID of the dedicated ASM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                                                                                                                          | numeric vNIC ID                                                                                                     | avi.asm-vnic-id.SE: '3'                                                                         |

## Instructions for Configuring CSP

To add a dedicated ASM vNIC on an existing SE CSP service, perform the following steps:

In the sample configuration below, vNIC3, which is the fourth NIC on the CSP service, is used.

1. Navigate to **Configuration > Service > Action > Power Off** to power off the SE service on Cisco CSP.
2. To add a new vNIC to the SE with desired parameters, navigate to **Configuration > Service > Action > Service Edit > Add vnic**. Provide VLAN ID, VLAN type, VLAN tagged, Network Name, Model, etc., and click on **Submit** button.
3. To power on the SE service on CSP, navigate to **Configuration > Service > Action > Power On**.

## Instructions for Configuring NSX Advanced Load Balancer Service Engine

SSH to the SE IP and perform the following steps:

```
ssh admin@<SE-MGMT-IP>
bash#
bash# sudo su
bash# /opt/avi/scripts/stop_se.sh
bash# mv /var/run/avi/ovf_properties.saved /home/admin
```

---

**Note** Do not copy this file; move it instead. Edit it to provide the three comma-separated ASM-dedicated NIC related parameters.

---

The sample file is as follows:

```
bash# cat /home/admin/ovf_properties.saved

AVICNTRL: 10.128.2.18, AVICNTRL_AUTHTOKEN: 1403771c- fc59-4d76-89b2-b3c35682b342,
avi.default-gw.SE: 10.128.2.1,
avi.asm-ip.SE: 10.160.102.227/24,
avi.asm-static-routes.SE: [169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2],
avi.asm-vnic-id.SE: '3',
avi.mgmt-ip.SE: 10.128.2.27, ovf_source: CSP,
uuid: FCE9B12D-A1B0-4EF3-B922-BDC2A5F8AA11}

bash# cp /home/admin/ovf_properties.saved /etc/ovf_config
bash# /opt/avi/scripts/start_se.sh
```

Verify that the dedicated vNIC information is applied correctly and the ASM virtual service IPs are reachable via this interface. In this case, the interface eth3 is the dedicated ASM interface and it is configured with IP 10.160.102.227/24.

```
bash# ssh admin@<SE-MGMT-IP>
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.102.227 Bcast:10.160.102.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.102.0/24 dev eth4 proto kernel scope link src 10.160.102.227
169.254.1.0/24 via 10.160.102.1 dev eth3
169.254.2.0/24 via 10.160.102.2 dev eth3
bash# ping -I eth3 <ASM-VIP>
ping -I eth3 169.254.1.10
PING 169.254.1.10 (169.254.1.10) from 10.160.102.227 eth3: 56(84) bytes of data.
64 bytes from 169.254.1.10: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for both HSM and ASM (Sideband) Communication on the NSX Advanced Load Balancer Service Engine

This section explains how to configure dedicated interfaces for HSM and sideband (ASM) communication on a new SE.

### Configuring Dedicated Interfaces for both HSM and ASM (Sideband) Communication on a new SE

The dedicated HSM and sideband interfaces on SEs use the following configuration parameters:

- `avi.hsm-ip.SE`
- `avi.hsm-static-routes.SE`
- `avi.hsm-vnic-id.SE`

For new SEs, these parameters can be provided in the day-zero YAML file.

### YAML Parameters for HSM

| YAML Parameter                        | Description                                                                                                                                                                                                                                                                                                                                      | Format                                                                                                         | Example                                                                                                     |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <code>avi.hsm-ip.SE</code>            | IP address of the dedicated HSM vNIC on the SE (this is NOT the IP address of the HSM device)                                                                                                                                                                                                                                                    | IP-address/subnet-mask                                                                                         | <code>avi.hsm-ip.SE:</code><br>10.160.103.227/24                                                            |
| <code>avi.hsm-static-routes.SE</code> | These are comma-separated, static routes to reach HSM devices. Even /32 routes can be provided.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if HSM devices are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [ hsm network1/mask1 via gateway1, hsm network2/mask2 via gateway2 ] OR<br>[ hsm network1/mask1 via gateway1 ] | <code>avi.hsm-static-routes.SE:</code><br>[ 10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2] |
| <code>avi.hsm-vnic-id.SE</code>       | ID of the dedicated HSM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                              | numeric vNIC ID                                                                                                | <code>avi.hsm-vnic-id.SE: '3'</code>                                                                        |

## YAML Parameters for ASM

| YAML Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                   | Format                                                                                                              | Example                                                                                         |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| avi.asm-ip.SE            | IP address of the dedicated ASM vNIC on the SE (this is NOT the IP address of the ASM)                                                                                                                                                                                                                                                                                                                                        | IP-address/subnet-mask                                                                                              | avi.asm-ip.SE:<br>10.160.103.227/24                                                             |
| avi.hsm-static-routes.SE | These are comma-separated, static routes to reach the sideband ASM vips. Even /32 routes can be provided. The gateway will be the self IP of the ASM device.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if the ASM virtual service IPs are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [asm-vip-network1/mask1 via gateway1, asm-vip-network2/mask2 via gateway2] or [asm-vip-network1/mask1 via gateway1] | avi.asm-static-routes.SE:<br>[169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2] |
| avi.asm-vnic-id.SE       | ID of the dedicated ASM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                                                                                                           | numeric vNIC ID                                                                                                     | avi.asm-vnic-id.SE: '3'                                                                         |

## Instructions

A sample Service Engine YAML file for the day-zero configuration on Cisco CSP looks as follows:

```
bash# cat avi_meta_data_dedicated_asm_hsm_SE.yml
avi.mgmt-ip.SE: "10.128.2.18"
avi.mgmt-mask.SE: "255.255.255.0"
avi.default-gw.SE: "10.128.2.1"
AVICNTRL: "10.10.22.50"
AVICNTRL_AUTHTOKEN: "febab55d-995a-4523-8492-f798520d4515"
avi.hsm-ip.SE: 10.160.103.227/24
avi.hsm-static-routes.SE: [10.128.1.0/24 via 10.160.103.1, 10.128.2.0/24 via 10.160.103.2]
avi.hsm-vnic-id.SE: '3'
avi.asm-vnic-id.SE: '4'
avi.asm-static-routes.SE: [169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2]
avi.asm-ip.SE: 10.160.102.227/24
```

Once the SE is created with this day-zero configuration and appropriate virtual NIC interfaces are added to the SE service instance in CSP, verify that the dedicated vNIC configuration is applied successfully and the HSM devices and ASM virtual service IPs are reachable via the dedicated interfaces. In this sample configuration, the interface eth3 is configured as the dedicated HSM interface with IP 10.160.103.227/24 and the interface eth4 is configured as the sideband ASM interface with IP 10.160.102.227/24.

The SE requires the following 5 interfaces for this configuration:

- vNIC0: Management interface
- vNIC1: Data in interface
- vNIC2: Data out interface
- vNIC3: Dedicated HSM interface
- vNIC4: Dedicated sideband interface

To verify configuration of both the dedicated interfaces, ssh to the SE IP, run the IP route command, and perform a ping test.

```
bash# ssh admin@10.10.2.18
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.103.227 Bcast:10.160.103.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)
```

```
bash# ip route
default via 10.10.2.1 dev eth0
10.10.1.0/24 via 10.160.103.1 dev eth3
10.10.2.0/24 via 10.160.103.2 dev eth3
10.10.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.103.0/24 dev eth3 proto kernel scope link src 10.160.103.227
bash# ping -I eth3 <HSM-IP>
ping -I eth3 10.10.1.51
PING 10.10.1.51 (10.128.1.51) from 10.160.103.227 eth3: 56(84) bytes of data.
64 bytes from 10.10.1.51: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for both HSM and ASM (Sideband) Communication on an Existing NSX Advanced Load Balancer Service Engine

The dedicated sideband interfaces on the SEs use the following configuration parameters:

- avi.asm-ip.SE
- avi.hsm-static-routes.SE
- avi.asm-vnic-id.SE

For existing SEs, these parameters can be populated in the `/etc/ovf_config` file.

**Note** All parameters in this file are comma-separated and the file format is slightly different from the YAML file used for spinning up new SEs. However, the parameters and their respective formats are exactly the same as they are for the new SEs.

## YAML Parameters

| YAML Parameter           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                  | Format                                                                                                              | Example                                                                                         |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| avi.asm-ip.SE            | IP address of the dedicated ASM vNIC on the SE (this is NOT the IP address of the ASM)                                                                                                                                                                                                                                                                                                                                                       | IP-address/subnet-mask                                                                                              | avi.asm-ip.SE:<br>10.160.103.227/24                                                             |
| avi.hsm-static-routes.SE | These are comma-separated, static routes to reach the sideband ASM virtual service IPs. Even /32 routes can be provided. The gateway will be the self IP of the ASM device.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if the ASM virtual service IPs are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [asm-vip-network1/mask1 via gateway1, asm-vip-network2/mask2 via gateway2] or [asm-vip-network1/mask1 via gateway1] | avi.asm-static-routes.SE:<br>[169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2] |
| avi.asm-vnic-id.SE       | ID of the dedicated ASM vNIC and is typically 3 on CSP (vNIC0 is management interface, vNIC1 is data-in interface, and vNIC2 is data-out interface)                                                                                                                                                                                                                                                                                          | numeric vNIC ID                                                                                                     | avi.asm-vnic-id.SE: '3'                                                                         |

## Instructions for Configuring the Cisco CSP

The following are the steps to add a dedicated ASM vNIC on an existing SE CSP service. In this example, vNIC 3, which is actually the fourth NIC on the CSP service is used.

1. Navigate to **Configuration > Service > Action > Power Off** to power off the SE service on Cisco CSP.
2. To add a new vNIC to the SE with desired parameters, navigate to **Configuration > Service > Action > Service Edit**, click on **Add vNIC** and provide VLAN ID, VLAN type, VLAN tagged, network Name, Model etc., and click on the **Submit** button.

- 3 Navigate to **Configuration > Service > Action** and select **Power On** to power on the SE service on Cisco CSP.

## Instructions for Configuring the NSX Advanced Load Balancer Service Engine

Perform the following steps on the SE using bash shell.

SSH to the SE IP and perform the following steps:

```
ssh admin@<SE-MGMT-IP>
bash#
bash# sudo su
bash# /opt/avi/scripts/stop_se.sh
bash# mv /var/run/avi/ovf_properties.saved /home/admin
```

**Note** Do not copy this file; move instead. Edit it to provide the three comma-separated ASM-dedicated NIC related parameters. The file looks as follows:

```
bash# cat /home/admin/ovf_properties.saved

AVICNTRL: 10.128.2.18, AVICNTRL_AUTHTOKEN: 1403771c- fc59-4d76-89b2-b3c35682b342,
avi.default-gw.SE: 10.128.2.1,
avi.asm-ip.SE: 10.160.102.227/24,
avi.asm-static-routes.SE: [169.254.1.0/24 via 10.160.102.1, 169.254.2.0/24 via 10.160.102.2],
avi.asm-vnic-id.SE: '3',
avi.mgmt-ip.SE: 10.128.2.27, ovf_source: CSP,
uuid: FCE9B12D-A1B0-4EF3-B922-BDC2A5F8AA11}

bash# cp /home/admin/ovf_properties.saved /etc/ovf_config
bash# /opt/avi/scripts/start_se.sh
```

Verify that the dedicated vNIC information is applied correctly and the ASM virtual service IPs are reachable via this interface. In this case, the interface eth3 is the dedicated ASM interface and it is configured with IP 10.160.102.227/24.

```
bash# ssh admin@<SE-MGMT-IP>
bash# ifconfig eth3
eth3 Link encap:Ethernet HWaddr 02:6a:80:02:11:05
 inet addr:10.160.102.227 Bcast:10.160.102.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:4454601 errors:0 dropped:1987 overruns:0 frame:0
 TX packets:4510346 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:672683711 (672.6 MB) TX bytes:875329395 (875.3 MB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.27
10.160.102.0/24 dev eth4 proto kernel scope link src 10.160.102.227
169.254.1.0/24 via 10.160.102.1 dev eth3
169.254.2.0/24 via 10.160.102.2 dev eth3
bash# ping -I eth3 <ASM-VIP>
ping -I eth3 169.254.1.10
```

```
PING 169.254.1.10 (169.254.1.10) from 10.160.102.227 eth3: 56(84) bytes of data.
64 bytes from 169.254.1.10: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for HSM Communication on the NSX Advanced Load Balancer Controller

This section explains how to configure dedicated interfaces for HSM communication on a new and an existing Controller.

### Configuring Dedicated Interfaces for HSM Communication on a new Controller

For configuration on a new Controller, the following YAML parameters can be provided in the day-zero YAML file:

- `avi.hsm-ip.Controller`
- `avi.hsm-static-routes.Controller`
- `avi.asm-vnic-id.Controller`

| YAML Parameter                                | Description                                                                                                                                                                                                                                                                                                                                                                              | Format                                                                                                  | Example                                                                                                            |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>avi.hsm-ip.Controller</code>            | IP address of the dedicated HSM vNIC on the Controller (this is not the IP address of the HSM device)                                                                                                                                                                                                                                                                                    | IP-address/subnet-mask                                                                                  | <code>avi.hsm-ip.SE:</code><br>10.160.103.230/24                                                                   |
| <code>avi.hsm-static-routes.Controller</code> | These are comma-separated, static routes to reach the HSM devices from the respective Controllers. Even /32 routes can be provided.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if the HSM devices are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [hsm-network1/mask1 via gateway1, hsm-network2/mask2 via gateway2] or [hsm-network1/mask1 via gateway1] | <code>avi.hsm-static-routes.Controller:</code><br>[10.128.1.0/24 via 10.160.103.1, 10.130.1.0/24 via 10.160.103.1] |
| <code>avi.asm-vnic-id.Controller</code>       | This is the ID of the dedicated HSM vNIC and is typically 1 on CSP. vNIC0 is the management interface, which is the only interface on the Controllers by default.                                                                                                                                                                                                                        | numeric-vnic-id                                                                                         | <code>avi.hsm-vnic-id.Controller:</code><br>'1'                                                                    |



## Instructions

A sample Controller service YAML file for the day-zero configuration on the CSP is as follows:

```
bash# cat avi_meta_data_ctlr-dedicated-hsm.yml

avi.default-gw.Controller: 10.128.2.1
avi.mgmt-ip.Controller: 10.128.2.30
avi.mgmt-mask.Controller: 255.255.255.0
avi.hsm-ip.Controller: 10.160.103.230/24
avi.hsm-static-routes.Controller: [10.128.1.0/24 via 10.160.103.1, 10.130.1.0/24 via
10.160.103.1]
avi.hsm-vnic-id.Controller: '1'
```

Once the Controller is created with this day-zero configuration and additional virtual NIC interface is added to the Controller service instance on CSP, verify that the dedicated vNIC configuration is applied successfully and the HSM devices are reachable via the dedicated interface. In this case we configured eth1 as the dedicated HSM interface with IP 10.160.103.230/24.

```
bash# ssh admin@<CONTROLLER-MGMT-IP>
bash# ifconfig eth1
eth1 Link encap:Ethernet HWaddr 02:4a:80:02:11:04
 inet addr:10.160.103.230 Bcast:10.160.103.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:342620 errors:0 dropped:2855 overruns:0 frame:0
 TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:29201376 (29.2 MB) TX bytes:11230 (11.2 KB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.1.0/24 via 10.160.103.1 dev eth1
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.18
10.130.1.0/24 via 10.160.103.1 dev eth1
10.160.103.0/24 dev eth1 proto kernel scope link src 10.160.103.218
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
bash# ping -I eth1 <HSM-IP>
ping -I eth1 10.130.1.10
PING 10.130.1.10 (10.130.1.10) from 10.160.103.230 eth1: 56(84) bytes of data.
64 bytes from 10.130.1.10: icmp_seq=1 ttl=62 time=0.229 ms
```

## Configuring Dedicated Interfaces for HSM Communication on an existing NSX Advanced Load Balancer Controller

The dedicated HSM interfaces on an existing Controller uses the following YAML parameters:

- `avi.hsm-ip.Controller`
- `avi.hsm-static-routes.Controller`
- `avi.hsm-vnic-id.Controller`

For an existing Controller, these parameters can be populated in the `/etc/ovf_config` file. All the parameters in this file are comma-separated, and the file format is slightly different from the YAML file used for spinning up a new Controller. However, the parameters and their respective formats are same as they are for a new Controller.

## YAML Parameters

| YAML Parameter                   | Description                                                                                                                                                                                                                                                                                                                                                                              | Format                                                                                                  | Example                                                                                               |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| avi.hsm-ip.Controller            | IP address of the dedicated HSM vNIC on NSX Advanced Load Balancer Controller (this is not the IP address of the HSM device)                                                                                                                                                                                                                                                             | IP-address/subnet-mask                                                                                  | avi.hsm-ip.SE:<br>10.160.103.230/24                                                                   |
| avi.hsm-static-routes.Controller | These are comma-separated, static routes to reach the HSM devices from the respective Controllers. Even /32 routes can be provided.<br><br>Note: If there is a single static route, provide the same and ensure the square brackets are matched. Also, if the HSM devices are in the same subnet as the dedicated interfaces, provide the gateway as the default gateway for the subnet. | [hsm-network1/mask1 via gateway1, hsm-network2/mask2 via gateway2] or [hsm-network1/mask1 via gateway1] | avi.hsm-static-routes.Controller:<br>[10.128.1.0/24 via 10.160.103.1, 10.130.1.0/24 via 10.160.103.1] |
| avi.asm-vnic-id.Controller       | ID of the dedicated HSM vNIC and is typically 1 on CSP                                                                                                                                                                                                                                                                                                                                   | numeric-vnic-id                                                                                         | avi.hsm-vnic-id.Controller:<br>'1'                                                                    |

## Instructions for Configuring Cisco CSP

Follow these steps to add the dedicated HSM vnic interfaces on an existing Controller CSP service. In the example mentioned below, vNIC1 is configured as the dedicated HSM interface (vNIC0 is the NSX Advanced Load Balancer Controller management interface)

- Navigate to **Configuration > Services > Action** and select **Power Off** to power off the SE service on Cisco CSP.
- To add a new vNIC for the Controller with the desired parameters, navigate to **Configuration > Services > Action > Service Edit**, click on **Add vNIC** and provide VLAN ID, VLAN type, VLAN tag, network name, Model etc., and click Submit.
- Navigate to **Configuration > Services > Action** and click **Power On** to power on the Controller service on Cisco CSP.

## Instructions for Configuring the Controller

Login to the Controller bash shell and perform the following steps:

```
bash# cat avi_meta_data_ctlr-dedicated-hsm.yml
ssh admin@<Controller-MGMT-IP>
 bash#
 bash# sudo su
 bash# mv /var/run/avi/ovf_properties.saved /home/admin
```

**Note** Perform a move operation; do not copy this file. Edit it to provide the three comma-separated, HSM dedicated NIC related parameters.

The file looks like the following:

```
bash# cat /home/admin/ovf_properties.saved
{avi.default-gw.CONTROLLER: 10.128.2.1,
avi.mgmt-ip.CONTROLLER: 10.128.2.18,
avi.mgmt-mask.CONTROLLER: 255.255.255.0,
 avi.hsm-ip.CONTROLLER: 10.160.103.230/24,
 avi.hsm-static-routes.CONTROLLER: [10.128.1.0/24 via 10.160.103.1, 10.130.1.0/24 via
10.160.103.1],
 avi.hsm-vnic-id.CONTROLLER: '1',
 ovf_source: CSP, uuid: E8FEBCCD-497E-4458-A933-B8317C1D8743}

bash# cp /home/admin/ovf_properties.saved /etc/ovf_config
bash# shutdown -h now
```

Verify that the dedicated vNIC configuration is applied correctly and the HSM devices are reachable via the dedicated interface. In this case, the interface eth1 is configured as the dedicated HSM interface with IP 10.160.103.230/24.

```
bash# ssh admin@<CONTROLLER-MGMT-IP>
bash# ifconfig eth1
eth1 Link encap:Ethernet HWaddr 02:4a:80:02:11:04
 inet addr:10.160.103.230 Bcast:10.160.103.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:342620 errors:0 dropped:2855 overruns:0 frame:0
 TX packets:78 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:29201376 (29.2 MB) TX bytes:11230 (11.2 KB)

bash# ip route
default via 10.128.2.1 dev eth0
10.128.1.0/24 via 10.160.103.1 dev eth1
10.128.2.0/24 dev eth0 proto kernel scope link src 10.128.2.18
10.130.1.0/24 via 10.160.103.1 dev eth1
10.160.103.0/24 dev eth1 proto kernel scope link src 10.160.103.218
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
bash# ping -I eth1 <HSM-IP>
ping -I eth1 10.130.1.10
PING 10.130.1.10 (10.130.1.10) from 10.160.103.230 eth1: 56(84) bytes of data.
64 bytes from 10.130.1.10: icmp_seq=1 ttl=62 time=0.229 ms
```

# Installing NSX Advanced Load Balancer in Cisco ACI based Environments

## 10

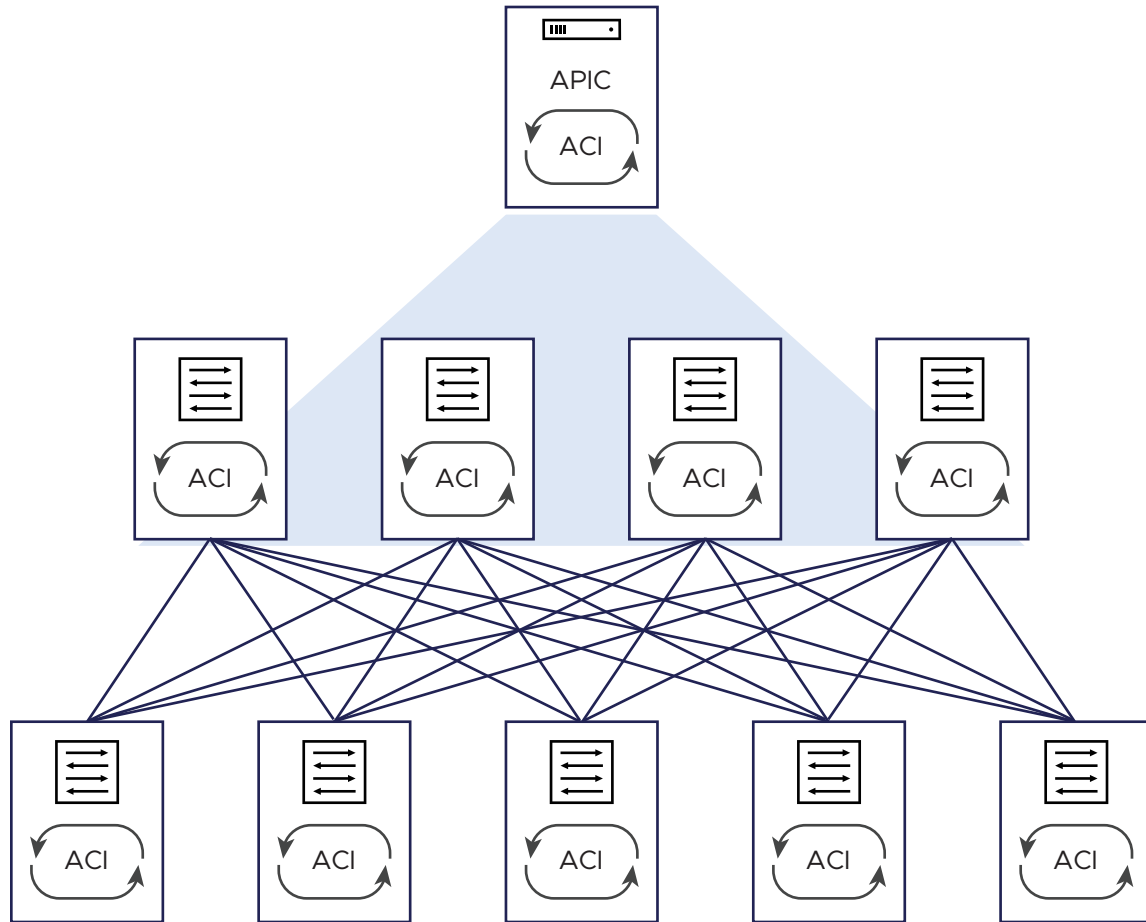
This article describes the installation process of NSX Advanced Load Balancer in Cisco ACI based environments.

This chapter includes the following topics:

- [Overview](#)
- [Networking Consideration](#)

## Overview

Cisco ® Cloud Application Centric Infrastructure (Cisco Cloud ACI) is a software-defined networking solution offered by Cisco for data centers and clouds, which helps in increasing operational efficiencies, delivering network automation, and improving security for any combination of on-premises data centers, private, and public clouds.



The important building blocks of Cisco ACI are Nexus 9000 hardware and APIC.

The APIC provides centralized policy automation and management for ACI fabric. The Controller helps in a common policy and management framework across physical, virtual, and cloud infrastructure.

ACI is based on open architecture (open APIs and standards) which integrates Layer 4-Layer 7 (L4-L7) services in the network. ACI solution offers robust implementation of multi-tenant security, quality of service (QoS), and high availability.

The following is a list of the most used terminologies in ACI:

#### **ACI fabric**

A Virtual Extensible LAN (VXLAN) overlay configured by APIC on leaf or spine switches to provide end-to-end connectivity for clients or servers.

#### **Bridge domains**

A bridge domain is a Layer 2 segment analogous to VLANs in a traditional network.

#### **Endpoint groups (EPGs)**

Endpoint groups are associated with endpoints in the network. The endpoints are identified by their domain connectivity (virtual, physical, or outside) and their connectivity method.

For instance, virtual machine port groups (VLAN, VXLAN), physical interfaces, or VLANs including virtual port channels, external VLANs, external subnets.

### Contracts

These are directional access lists between the provider and consumer EPGs. They comprise one or more filters (ACEs) to identify and allow traffic between EPGs. By default, communication between EPGs is blocked and it requires a contract to allow the traffic.

---

**Note** Intra EPG traffic is allowed by default and so no contract is required.

---

### Application network profiles

These are containers that group one or more EPGs together with their associated connectivity policies.

### Tenants

These are network-wide administrative containers, which are like logical containers for application policies.

## Networking Consideration

This section lists all the network considerations required for integrating the NSX Advanced Load Balancer with ACI.

## Software Requirements

The following table lists the minimum software requirements for NSX Advanced Load Balancer integration with ACI:

| Component                             | Version                             |
|---------------------------------------|-------------------------------------|
| NSX Advanced Load Balancer Controller | 17.2.10 or later                    |
| Cisco APIC                            | 1.03f or later                      |
| VMware vCenter                        | 5.1, 5.5, 6.0, 6.4, 6.5, 6.7 or 7.0 |

## Integration Options for NSX Advanced Load Balancer in ACI Fabric

This section discusses integrating NSX Advanced Load Balancer with ACI in different hosting infrastructures.

### VMware

#### Network policy mode with write access vCenter cloud

This is a traditional mode where EPGs for the SEs are used instead of Service Graphs within ACI. ACI is used only to provide access between the client network and virtual service network. NSX Advanced Load Balancer integrates with vCenter in write access mode. This is our recommended deployment mode.

### Network policy mode with no-access/read access vCenter cloud

This is a traditional mode where EPGs for the SEs are used instead of Service Graphs within ACI. ACI is used only to provide access between the client network and virtual service network. NSX Advanced Load Balancer will be deployed as an external L3out on VMware infrastructure with no access cloud (no integration with vCenter) or read access (read only access to Vcenter) and will perform the BGP peering with ACI.

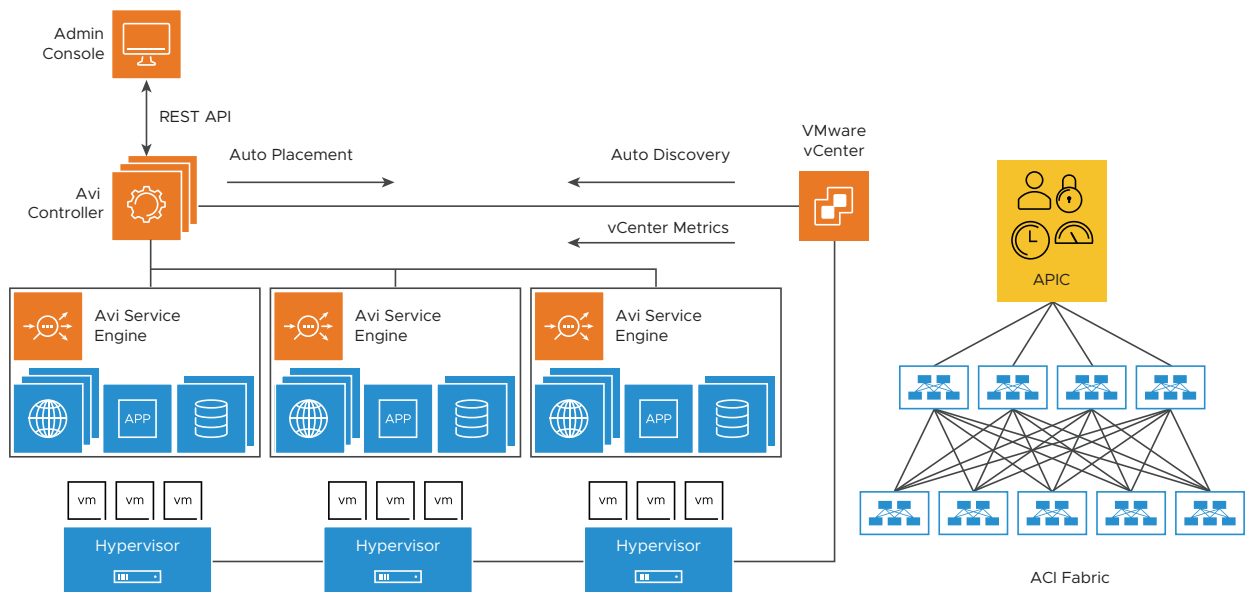
## Cisco CSP 2100/ Bare Metal Servers

### Network policy mode

This is a traditional mode where EPGs for the SEs are used instead of Service Graphs within ACI. NSX Advanced Load Balancer will be deployed as an external L3Out on Cisco CSP and will perform the BGP peering with ACI.

## Network Policy Mode with NSX Advanced Load Balancer on Write Access VMware Cloud

In this mode, NSX Advanced Load Balancer is integrated with VMware and the VMware infrastructure is used to configure the interfaces and port groups.



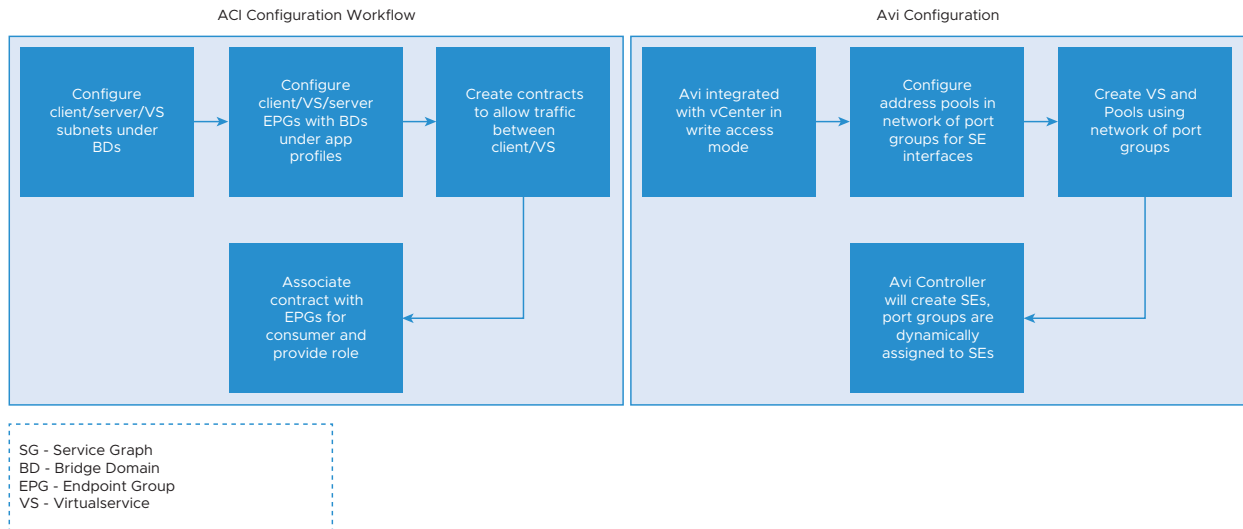
For deploying NSX Advanced Load Balancer in Network Policy Mode the following minimum ACI configurations are required:

- A bridge domain defined containing subnets to be used for Virtual Service IPs and SE interfaces.

- An EPG created referencing the previous bridge domain

As seen above, the integration is with vCenter in write access mode.

The configuration workflow for this mode is given below:



To deploy NSX Advanced Load Balancer in vCenter with write access mode, see [Chapter 14 Installing NSX Advanced Load Balancer in VMware vSphere Environments](#).

This is a traditional deployment where ACI provides access (contracts) between the clients and virtual service and the Service Engine and servers.

### Configuring APIC contracts for NSX Advanced Load Balancer

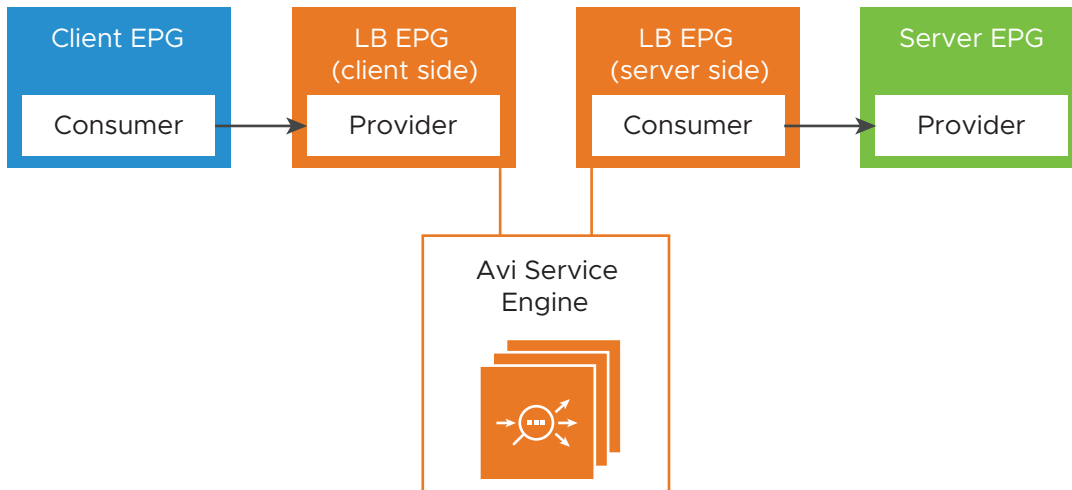
After deploying NSX Advanced Load Balancer in vCenter write access mode, you need to create contracts to allow communication between the clients and virtual service networks, and Service Engines and servers.

The contracts can be configured in ACI for the following deployment modes:

#### NSX Advanced Load Balancer Deployed in Two-Arm Mode

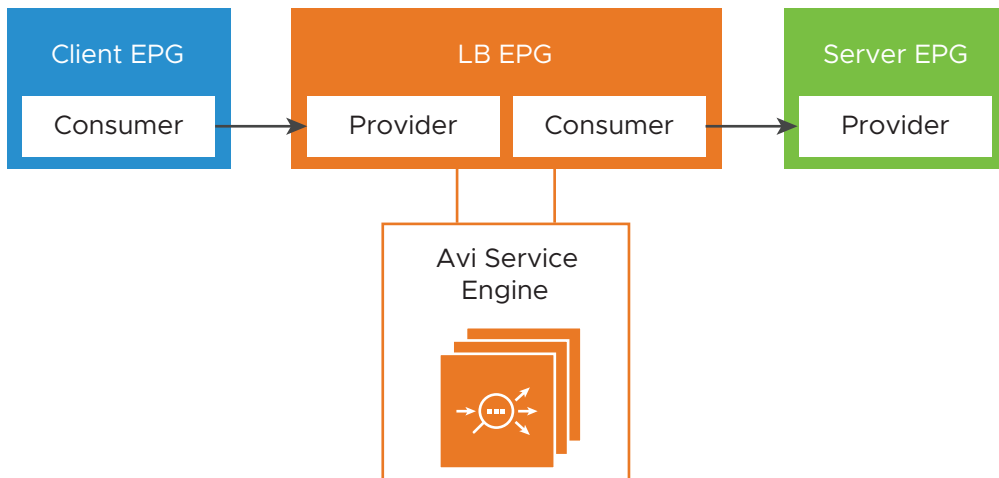
In this mode, SE connectivity to the clients and servers will use different networks. You need to create a contract to allow communication between the client EPG and virtual servers EPG and between NSX Advanced Load Balancer and the server EPG.





### NSX Advanced Load Balancer Deployed in One Arm Mode

In this mode, connectivity from the SE to the clients and servers will use the same interface. You need to create a contract to allow communication between the client EPG and virtual servers EPG and between NSX Advanced Load Balancer and the server EPG.



### Network Policy Mode with NSX Advanced Load Balancer on No Access or Read Access VMware Cloud

In this deployment, BGP is used for peering with ACI fabric and to exchange the virtual service routes. This deployment is mostly applicable for setups with no vCenter deployed or has no vCenter integration with NSX Advanced Load Balancer Controller.

In vCenter no-access cloud, the Controller has no access to vCenter resources. In vCenter read access cloud, the Controller will integrate with vCenter, but the Controller can only read the vCenter resources and not write or create any Service Engines or resources. In both cases, you need to manually deploy Service Engines.

For more information on VMware no-access or read access, see [Chapter 14 Installing NSX Advanced Load Balancer in VMware vSphere Environments](#).

To support BGP peering between the SEs hosted on vCenter and ACI fabric the SEs should be connected to the ACI fabric and configured as an external routed device in APIC. Then, APIC will consider it as an L3Out device.

### BGP Peering

For configuring BGP L3Out on Cisco APIC, please refer to the APIC configuration section in NSX Advanced Load Balancer Deployment for North-South Traffic.

To configure BGP on NSX Advanced Load Balancer Controller:

- 1 Navigate to **Infrastructure > Routing**.
- 2 Select the cloud in which BGP peering is configured.
- 3 Under BGP Peering, enter the peer IP, AS numbers, and other parameters.

For configuring BGP L3Out on Cisco APIC, please refer to the APIC configuration section in NSX Advanced Load Balancer Deployment for North-South Traffic.

To configure BGP on the NSX Advanced Load Balancer Controller:

- 1 Navigate to **Infrastructure > Routing**.
- 2 Select the cloud in which BGP peering is configured.
- 3 Under **BGP Peering**, enter the **peer IP, AS numbers**, and other parameters.

For detailed information, see [BGP Support for Scaling Virtual Services](#).

### Virtual Service Provisioning

After configuring BGP peering, proceed with creating virtual services.

To create virtual services:

- 1 In NSX Advanced Load Balancer Controller, click **Creating Virtual Service (Advanced)** option.
- 2 Enter all virtual service parameters, such as the name, IP address, pool, etc.
- 3 Click **Next** and navigate to the **Advanced** tab.
- 4 Enable **Advertise VIP via BGP** option.

**Edit Virtual Service: VMware-no-access-App**

Settings Policies Analytics Advanced

Name \* VMware-no-access-App Enabled ☒ Virtual Hosting VS ☐

**VIP Address**

VIP Address \* 15.15.15.5

**Profiles**

Application Profile \* System-HTTP

TCP/UDP Profile \* System-TCP-Proxy

WAF Policy Select WAF Policy

**Service Port** [Switch to Advanced](#)

Services 80 ☐ SSL

**Pool**

☒ Pool ☐ Pool Group

Pool VMware-no-pool

Cancel Save

---

**Edit Virtual Service: VMware-no-access-App**

Settings Policies Analytics Advanced

Fairness ☐ Throughput And Delay Fairness ☒ Throughput Fairness

**HTTP Basic Authentication**

☐ Enable HTTP Basic Authentication

**Other Settings**

Server Network Profile None

Host Name Translation a.b.com

☒ Auto Gateway ☐ Use VIP as SNAT

☒ Advertise VIP via BGP ☐ Advertise SNAT via BGP

SE Group Default-Group

SNAT IP Address x.x.x.x, x.x.x.y

☐ Remove Listening Port when VS Down

Cancel Save

After the virtual service is created, the NSX Advanced Load Balancer SE will start publishing the routes to ACI fabric.

You can check the BGP peering status and advertised routes in APIC. Alternatively, to check BGP peering and neighbor status on NSX Advanced Load Balancer, login to the SE CLI and run the commands shown below under the quagga router. See [How to Access and Use Quagga Shell using NSX Advanced Load Balancer CLI](#) for reference.

```
10-128-3-198> show ip bgp
BGP table version is 0, local router ID is 2.103.143.46
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
 i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```

 Network Next Hop Metric LocPrf Weight Path
*> 15.15.15.5/32 0.0.0.0 0 32768 i
*> 16.16.16.0/24 17.17.17.1 0 0 500 670 ?

Total number of prefixes 2

```

```

10-128-3-198> show ip bgp summary
BGP router identifier 2.103.143.46, local AS number 600
RIB entries 3, using 336 bytes of memory
Peers 1, using 4568 bytes of memory

```

| Neighbor   | V | AS  | MsgRcvd | MsgSent | TblVer | InQ | OutQ | Up/Down  | State/PfxRcd |
|------------|---|-----|---------|---------|--------|-----|------|----------|--------------|
| 17.17.17.1 | 4 | 500 | 19369   | 19373   | 0      | 0   | 0    | 01w6d10h | 1            |

```

Total number of neighbors 1

```

You can verify the virtual service routes in ACI fabric. The sample below is the output of the virtual service route learned by a border leaf in ACI fabric:

```

leaf-1# show bgp ipv4 unicast vrf VMware-NO-Access-Demo:vmware-no-vrf
BGP routing table information for VRF VMware-NO-Access-Demo:vmware-no-vrf, address family
IPv4 Unicast
BGP table version is 12, local router ID is 4.4.4.4
Status: s-suppressed, x-deleted, S-stale, d-dampened, h-history, *-valid, >-best
Path type: i-internal, e-external, c-confed, l-local, a-aggregate, r-redist, I-injected
Origin codes: i - IGP, e - EGP, ? - incomplete, | - multipath, & - backup

```

| Network          | Next Hop   | Metric | LocPrf | Weight | Path |
|------------------|------------|--------|--------|--------|------|
| *>r4.4.4.4/32    | 0.0.0.0    | 0      | 100    | 32768  | ?    |
| *>e15.15.15.5/32 | 17.17.17.2 | 0      | 0      | 500    | 600  |
| *>r16.16.16.0/24 | 0.0.0.0    | 0      | 100    | 32768  | ?    |
| *>r17.17.17.0/24 | 0.0.0.0    | 0      | 100    | 32768  | ?    |

## Network Policy Mode with NSX Advanced Load Balancer in Write Access VMware Cloud with BGP L3\_Out

In this deployment, BGP is used for peering with ACI fabric and to exchange the virtual service routes. This deployment is mostly applicable for setups where BGP auto-scaling is required for virtual service scaling on SEs.

In vCenter write access mode, the NSX Advanced Load Balancer Controller is configured with a vCenter cloud connector. The Controller has write access permissions to vCenter and handles complete automation involved in creating Service Engines and placing them in the network. The Controller also scales the Service Engines based on the configured threshold.

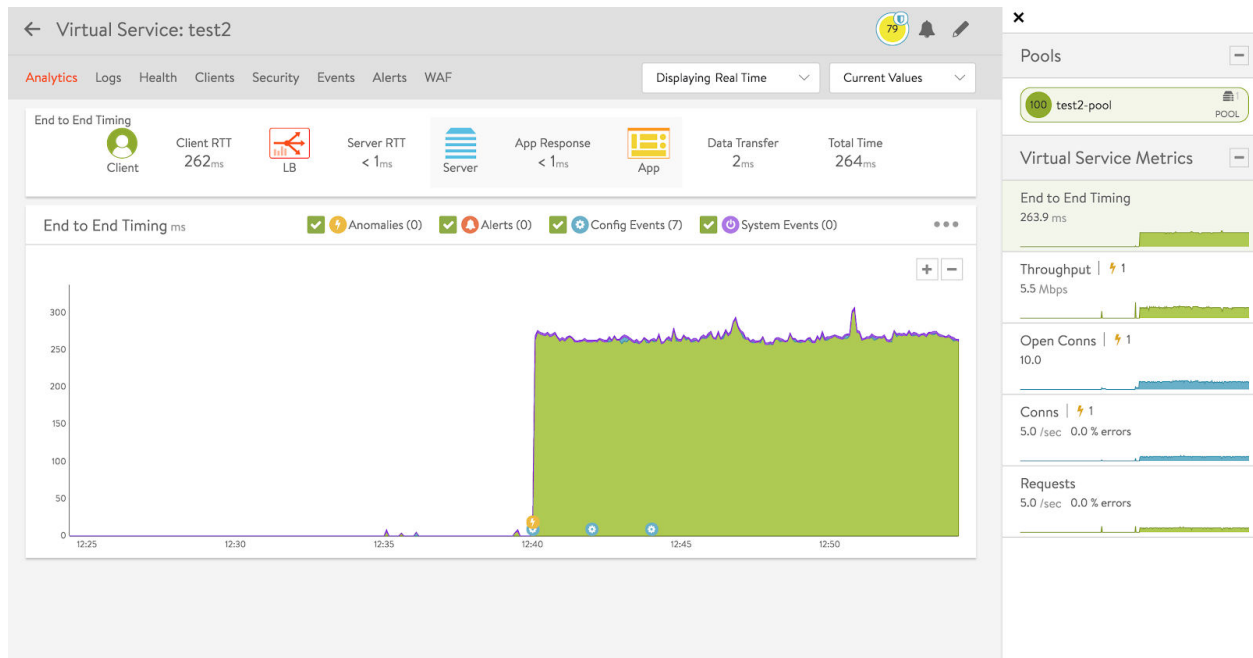
For more information, see [Deploying NSX Advanced Load Balancer Controller](#).

The BGP peering and virtual service configuration remains the same as mentioned in the [Network Policy Mode with NSX Advanced Load Balancer on No Access or Read Access VMware Cloud](#).

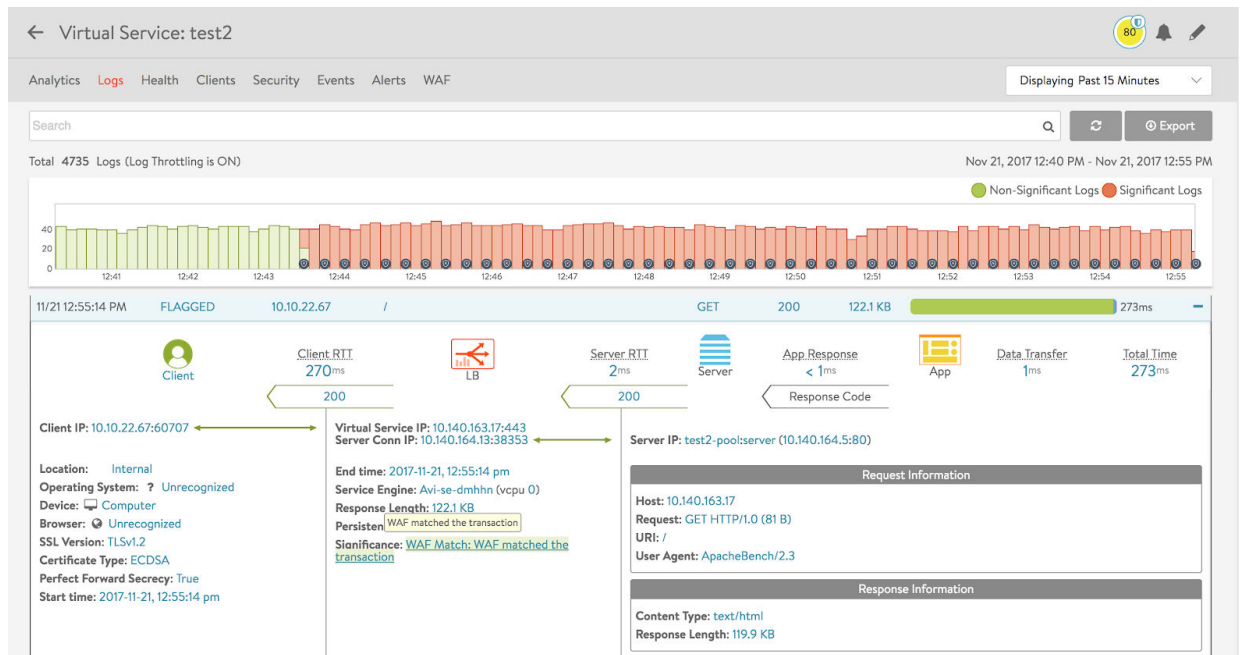
A port group with a static VLAN needs to exist in vCenter for the Service Engine Data vNICs. Use the SVI interface with static VLAN as L3out in the ACI fabric.

## Monitoring

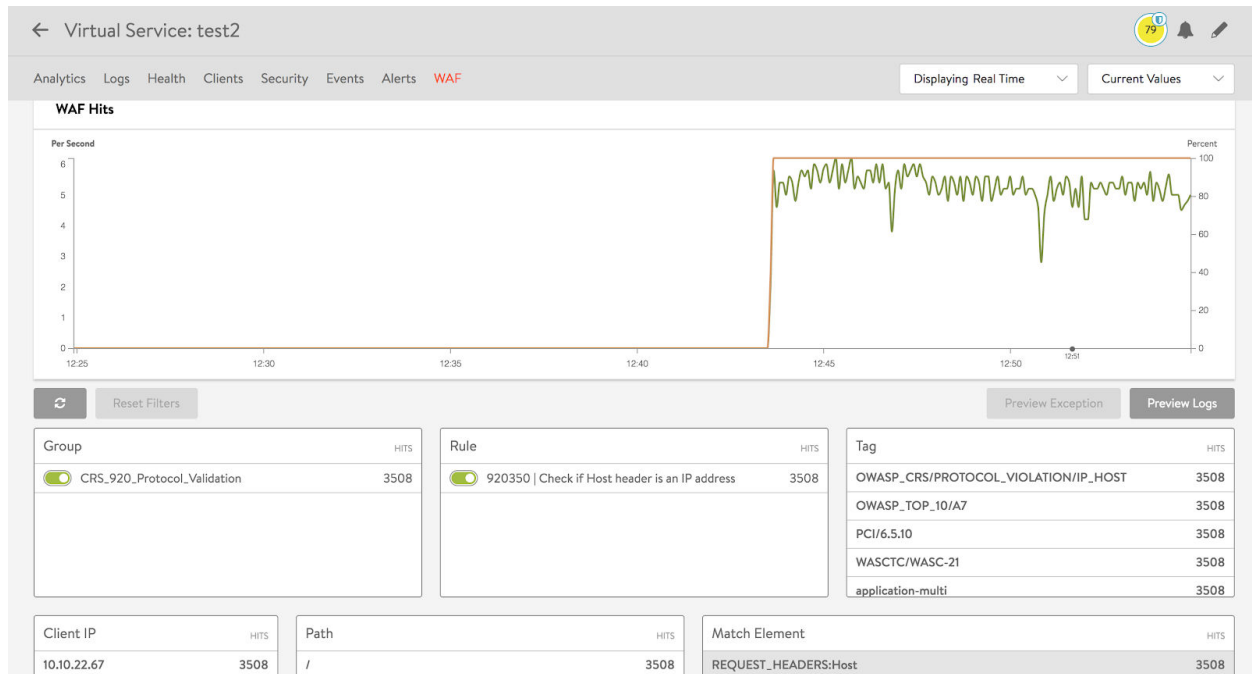
NSX Advanced Load Balancer Controller provides real-time analytics dashboards that provide application load balancing and security analytics in a single frame.



NSX Advanced Load Balancer virtual service real-time metrics provide details relating to transactions per second, delay, response times, etc.



NSX Advanced Load Balancer logs provide a detailed view of each connection, as in the displayed case, where the client/virtual service/server end-to-end communication is displayed, which is used for troubleshooting.



NSX Advanced Load Balancer WAF analytics provides information on real-time web security attacks on the virtual service.

The NSX Advanced Load Balancer Controller offers the following monitoring capabilities for Cisco ACI fabric:

- Monitors Load Balancer (SE) and application server health.
- Provides real-time application analytics.
- Protects applications against L4-L7 DDoS attacks.
- Monitors APIC EPG membership to automatically add or remove application instances from pools.
- Performs Load Balancer auto-scaling based on real-time performance metrics (CPU, memory, bandwidth, connections, latency, etc).
- Provides point-and-click simplicity for iWAF policies with central control.
- Provides granular security insights on traffic flows and rule matches to enable precise policies using iWAF.

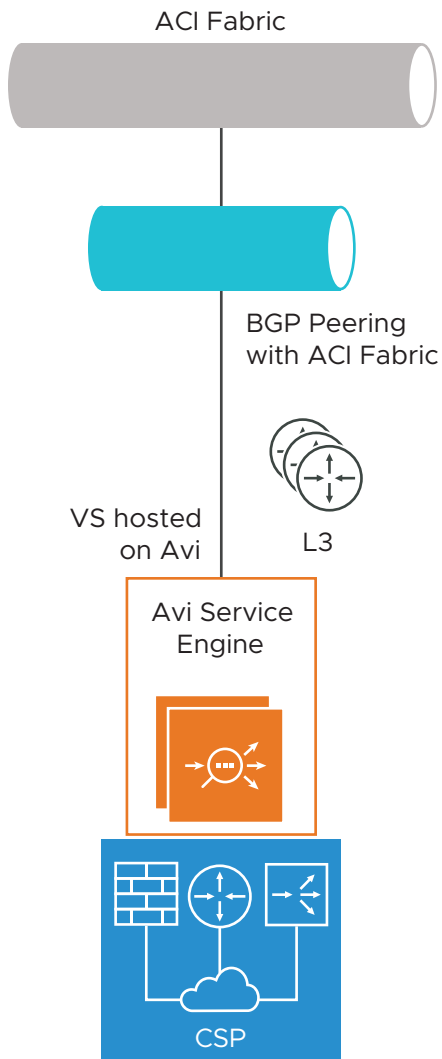
## Network Policy Mode with NSX Advanced Load Balancer on Cisco ACI 2100 as BGP L3 Outs

Cisco ACI 2100 is a turn-key hosting operating system and hardware platform optimized for Data Center NFV. The platform facilitates the quick onboarding and lifecycle management of any Cisco or 3rd party VNF.

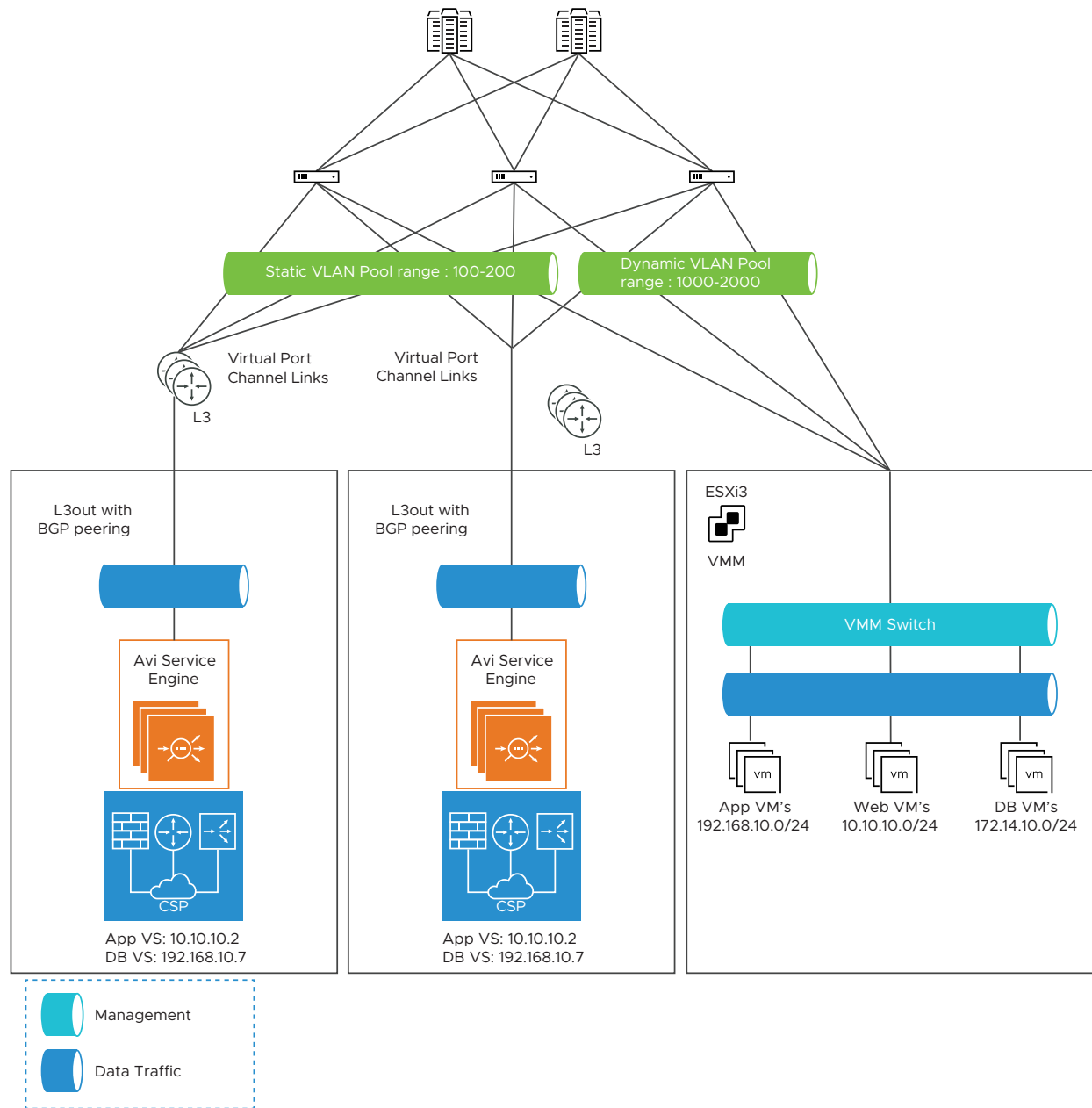
In this design option, the NSX Advanced Load Balancer will be deployed in No-orchestrator mode on Cisco ACI 2100, NSX Advanced Load Balancer SEs will be configured as BGP L3 Outs in APIC in order to exchange the virtual service routes, leaf/spine switches in ACI fabric will be learning this routes to forward the virtual service traffic to SE's.

### Logical Network Topology

The logical view of one arm mode looks like the screenshot provided below, where NSX Advanced Load Balancer SE hosted on Cisco ACI 2100 will be connected to ACI Fabric as L3out using BGP peering. The clients are sourced from ACI fabric and can access the virtual service hosted on the SE.



The following is the logical network topology of the one arm mode design with BGP peering for SEs hosted on Cisco ACI 2100.



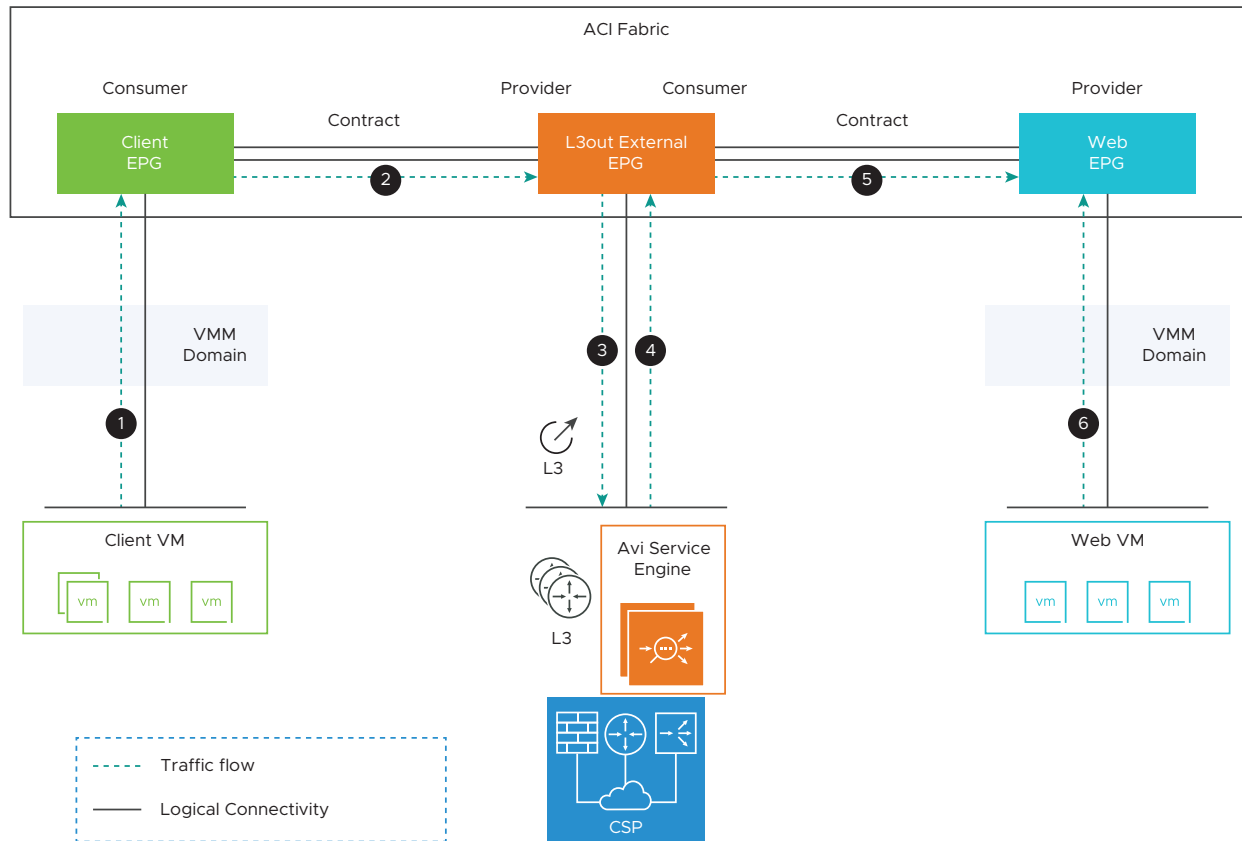
## Logical Traffic Flow

The traffic flow for client VM to access virtual service app hosted on the SE is shown as follows:

- 1 Client VM → ACI Fabric → Client EPG
- 2 Client EPG → Contract → L3out extern
- 3 L3out external EPG → ACI fabric → Avi SE
- 4 Avi SE → load balancing to back-end servers → ACI fabric → L3out external EPG
- 5 L3out external EPG → Contract → Web EPG
- 6 Web EPG → Web server VM



Return traffic follows the same path as the incoming traffic.



## Service Engine Routing

NSX Advanced Load Balancer SEs can only publish the virtual service routes. It cannot learn any routes using BGP from any of the BGP peers. For few cases, SEs need to learn the routes from the BGP peer in order to send the return traffic to an appropriate next-hop which forwards the traffic. In such cases, you can use the auto gateway option to ensure that the return traffic is sent to the same MAC address from which it was received.

For detailed information, see [Create a Virtual Service](#).

## High Availability

As BGP is used for exchanging routes, high availability is entirely dependent on BGP. By default, the Service Engines are inactive/active state. For active/standby virtual service deployment, use the local preference option in ACI fabric. This allows you to choose one SE route as the most preferred one over other SEs.

For detailed information on local preference in ACI fabric, see [L3 Config/Cisco APIC Layer 3 Configuration Guide](#).

## References

In this deployment, NSX Advanced Load Balancer Controller works in No Orchestrator mode. Hence you need to manually install Service Engines. Customers can use Ansible in order to automate the installation.

- For more details on Ansible Automation for Cisco CSP and NSX Advanced Load Balancer Deployment, see [Ansible Role NSX Advanced Load Balancer Controller CSP](#).
- The BGP peering and Virtual service configuration are same as mentioned in [Network policy Mode with VMware No Access/Read Access](#).
- For more details on NSX Advanced Load Balancer installation on Cisco CSP 2100, see [Installing NSX Advanced Load Balancer for CSP 2100](#).
- For more details on sizing guidelines, see [Sizing Guidelines](#).
- Before starting with NSX Advanced Load Balancer Deployment/Design hosted on Cisco CSP, for configuring Port Channel, to get maximum bandwidth on Service Engines, see [Port Channel for CSP 2100 for more details](#).

# Installing the NSX Advanced Load Balancer in Oracle Cloud

# 11

The NSX Advanced Load Balancer supports integration with Oracle Cloud Infrastructure via Linux server cloud. For the integration, it uses the OCI IPAM feature. Oracle hierarchical compartments are supported.

This chapter includes the following topics:

- [Considerations for Deploying the NSX Advanced Load Balancer in Oracle Cloud](#)
- [Configuring OCI Credentials](#)
- [Configuring OCI IPAM](#)
- [Configuring the Cloud Connector](#)

## Considerations for Deploying the NSX Advanced Load Balancer in Oracle Cloud

To deploy the NSX Advanced Load Balancer in the Oracle cloud, an OCI user has to be created. This user account is used to create the IPAM profile.

## Configuring OCI Credentials

An OCI user is created using the `configure cloudconnectoruser <username>` command.

To create the OCI user:

### Procedure

- 1 Log in to the CLI shell.
- 2 Execute the `configure cloudconnectoruser <username>` command.
- 3 Enter the following details:

The `oci_credentials` to enter the mentioned submode are as follows:

- `user` : The user OCID
- `key_content`: The private key content for signing AP (replace every next line with `\n` character while copying the key content in quotes)

- `pass_phrase`: The `pass_phrase` for the private key (only if key is encrypted)
- `fingerprint` – The fingerprint generated after adding the public key in the OCI console

```
admin@10-0-0-77:~$ shell
Login: admin
Password:
[admin:10-0-0-77]: > configure cloudconnectoruser ocuser
[admin:10-0-0-77]: cloudconnectoruser> oci_credentials
fingerprint API key with respect to the Public
Key

key_content Private Key file (pem file)
content

pass_phrase Pass phrase for the
key

user Oracle Cloud Id for the
User

[admin:10-0-0-77]: cloudconnectoruser> oci_credentials
```

## Results

To view the `cloudconnectoruser ocuser` configuration:

```
[admin:10-0-0-77]: > show ipamdnsproviderprofile prof1
+-----+-----+
| Field | Value |
+-----+-----+
uuid	ipamdnsproviderprofile-d67ad96c-8bbf-48ff-ab40-5580621c1c69
name	prof1
type	IPAMDNS_TYPE_OCI
oci_profile	
tenancy	ocid1.tenancy.oc1..aaaaaaaay7s6icq755xqlytpl33i7ysjzzb2kv3vk3
	itg5ilsxanrzqmsaha
region	us-phoenix-1
cloud_	ocuser
credentials_ref	
ocid1.compartment.oc1..aaaaaaa5trt72k3smsky7fz27gqlucbfa2lmy	
nshky4hl4r7gom6wcpmhrq	
vcn_id	ocid1.vcn.oc1.phx.aaaaaaaangx3fookzumnhck3st5obrruwsmxiggtx2i
	c7zoharlhwi262gla
allocate_ip_in	False
_vrf	
tenant_ref	admin
+-----+-----+
+
```

## Configuring OCI IPAM

The IPAM profile is created using the OCI credentials.

To create the IPAM profile:

### Procedure

- 1 Log in to the CLI shell.
- 2 Execute the `configure ipamdnsproviderprofile <profile name>` command.
- 3 Enter the following details:
  - type: Set the value as `IPAMDNS_TYPE_OCI`
  - oci\_profile:
    - tenancy: tenancy OCID
    - region : OCI region name
    - cloud\_credentials\_ref : The reference to the cloud connector user created in the previous section. Use the **Tab** key to list the users.
    - vcn\_compartment\_id : Compartment OCID of the the VCN
    - vcn\_id : VCN OCID

```
admin@10-0-0-77:~$ shell
Login: admin
Password:
[admin:10-0-0-77]: > configure ipamdnsproviderprofile ocprof
[admin:10-0-0-77]: > configure ipamdnsproviderprofile ocprof2
[admin:10-0-0-77]: ipamdnsproviderprofile> type ipamdns_type_oci
cloud_credentials_ref Credentials to access oracle
cloud

region Region in which Oracle cloud resource
resides

tenancy Oracle Cloud Id for tenant aka root
compartment

vcn_compartment_id Oracle cloud compartment id in which VCN
resides

vcn_id Virtual Cloud network id where virtual ip will belT
```

### Results

To view the OCI IPAM configuration:

```
[admin:10-0-0-77]: > show ipamdnsproviderprofile prof1
+-----+-----+
|Field |Value|
+-----+-----+
|Field |Value|
```

```

+-----+-----+
uuid	ipamdnsproviderprofile-d67ad96c-8bbf-48ff-ab40-5580621c1c69
name	prof1
type	IPAMDNS_TYPE_OCI
oci_profile	
tenancy	ocid1.tenancy.oc1..aaaaaaaay7s6icq755xqllytpl33i7ysjzzb2kv3vk
	itg5ilsxanrzqmsaha
region	us-phoenix-1
cloud_	ocuser
credentials_ref	
vcn_compartment	ocid1.compartment.oc1..aaaaaaa5trt72k3smsky7fz27gqlucbfa2l
_id	mynshky4hl4r7gom6wcpmhrq
vcn_id	ocid1.vcn.oc1.phx.aaaaaaaangx3fookzumnhck3st5obrruwsmxiggtx
	2ic7zoharlhwi262gla
allocate_ip_in_	False
vrf	
tenant_ref	admin
+-----+-----+

```

## Configuring the Cloud Connector

A Linux Server Cloud has to be created and configured with the OCI IPAM profile (prof1) that was created.

For detailed instructions on how to create a Linux Server Cloud, refer [Installing NSX Advanced Load Balancer in a Linux Server Cloud](#)

# Installing NSX Advanced Load Balancer in Amazon Web Services

# 12

This installation guide discusses integrating NSX Advanced Load Balancer with Amazon Elastic Compute Cloud (Amazon EC2) instances.

The Amazon EC2 is one of the services provided by Amazon Web Service (AWS). Amazon EC2 instance types comprise various combinations of memory, storage, CPU, and networking capacity. The integration of the Amazon EC2 instances with NSX Advanced Load Balancer provides flexibility to choose from various resources based on the application or user requirements.

NSX Advanced Load Balancer offers elastic application services that extend beyond load balancing to deliver real-time application and security insights, simplify troubleshooting, auto scale predictively, and enable developer self-service and automation in Amazon Web Services.

The NSX Advanced Load Balancer Controller analyzes traffic and, based on real-time analytics, can scale out / scale in SEs to load-balance traffic that varies over time.

SEs handle all data plane operations within NSX Advanced Load Balancer by receiving and executing instructions from the Controller. The SEs perform load balancing and handle client and server traffic.

This chapter includes the following topics:

- [Deployment Considerations for NSX Advanced Load Balancer in AWS](#)
- [Installing the NSX Advanced Load Balancer Controller](#)
- [Configuring the NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Deployment Options](#)

## Deployment Considerations for NSX Advanced Load Balancer in AWS

This section discusses the deployment considerations for installation of NSX Advanced Load Balancer in AWS.

### Minimum Requirements

This section lists out the minimum requirements to install the NSX Advanced Load Balancer in AWS.

For more information, see [System Requirements](#).

## Ports and Protocols

This section lists out the required ports and protocols for NSX Advanced Load Balancer integration with AWS.

NSX Advanced Load Balancer Controller and the SEs use specific ports for management and network services. The firewall should allow traffic for these ports.

For the detailed information on the required management ports, see [Ports and Protocols](#). The [Ports and Protocols](#) mentioned in the section are applicable for both Controller and SE communication with AWS Cloud.

## Installation Options and Credential Methods

This section discusses the various credential methods available for AWS integration with the NSX Advanced Load Balancer.

When deploying NSX Advanced Load Balancer within Amazon Web Services (AWS), the installation wizard prompts for input of credential information. You are not required to enter AWS Secret and Access key credentials. The credentials can be entered in either of the following forms:

- Identity and Access Management (IAM) roles
- AWS customer account key
- Use Cross-Account AssumeRole

These credential methods are used during AWS cloud configuration.

### Accounts and IAM Roles

This section discusses various IAM roles required for NSX Advanced Load Balancer integration with AWS.

#### Identity and Access Management (IAM) roles

IAM roles are the set of policies that define access to resources within AWS. The roles and the policies that define their access are defined in JSON files. This method does not require an AWS account key. Instead, the role and policy files must be downloaded from NSX Advanced Load Balancer and installed using the AWS CLI. (Download links for the role and policy files, and the required AWS CLI syntax, are provided in this article.) After setting up the IAM roles, return to this article to install the NSX Advanced Load Balancer EC2 instance. Use this method if you do not want to enter AWS credentials. Following are the mandatory IAM roles that should be configured through Amazon Management Console:

- vmimport
- AviController-Refined-Role



## IAM Role Setup for Installation into AWS

If using the IAM role method to define access for an NSX Advanced Load Balancer installation in Amazon Web Services (AWS), use the steps in this article to set up the IAM roles before beginning deployment of the NSX Advanced Load Balancer Controller EC2 instance.

| IAM Role Name                                                            | Policy Name                                                                         | Description                                                                                                                                                                                                                                                               | Required |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| vmimport <a href="#">vmimport-role-trust.json</a>                        | vmimport <a href="#">vmimport-role-policy.json</a>                                  | Enables the NSX Advanced Load Balancer SE VM to be imported into AWS. Without this IAM role, the SE cannot be launched. This role is associated with the AWS account (not with the Controller). For more details on vmimport, see <a href="#">VM Import/Export</a> guide. | Yes      |
|                                                                          | kmsimport <a href="#">avicontroller-kms-vmimport.json</a>                           | Used to create an IAM policy and attached to vmimport role, or it can be directly applied to the KMS key.                                                                                                                                                                 | Yes      |
| AviController-Refined-Role <a href="#">avicontroller-role-trust.json</a> | AviController-EC2-Policy <a href="#">avicontroller-ec2-policy.json</a>              | Enables NSX Advanced Load Balancer Controllerinstance to be installed.                                                                                                                                                                                                    | Yes      |
|                                                                          | AviController-IAM-Policy <a href="#">avicontroller-iam-policy.json</a>              | Enable access to retrieve IAM roles and policy information.                                                                                                                                                                                                               | Yes      |
|                                                                          | AviController-s3-Policy <a href="#">avicontroller-s3-policy.json</a>                | Enable S3 permissions                                                                                                                                                                                                                                                     | Yes      |
|                                                                          | AviController-R53-Policy <a href="#">avicontroller-r53-policy.json</a>              | Enables access to the AWS cloud's DNS.                                                                                                                                                                                                                                    | No       |
|                                                                          | AviController-AutoScalingGroup-Policy <a href="#">avicontroller-asg-policy.json</a> | Enables read access to the AWS cloud's Auto Scaling groups.                                                                                                                                                                                                               | No       |
|                                                                          | AviController-SQS-SNS-Policy <a href="#">avicontroller-sqs-sns-policy.json</a>      | Enables NSX Advanced Load Balancer Controller to use SNS and SQS feature for Auto Scaling groups. .Allows NSX Advanced Load Balancer Controller to receive ASG notifications when SNS and SQS features are enabled.                                                       | No       |
|                                                                          | AviController-KMS-Policy <a href="#">avicontroller-kms-policy.json</a>              | Enables the NSX Advanced Load Balancer Controller to list the encryption keys in the NSX Advanced Load Balancer UI, anddecrypt encrypted messages (required, when using SQS encryption)                                                                                   | No       |

To begin, download the JSON files for the IAM role and policies onto a host that has the AWS CLI.

Then use one of the following workflows to set up the IAM roles:

### Using the AWS CLI

The AWS CLI needs to be run from the same directory in which you save the files.

#### Procedure

##### 1 Create the VM Import Service Role.

Use the following commands to create a role name “vmimport” with the required permission.

```
aws iam create-role --role-name vmimport --assume-role-policy-document file://vmimport-
role-trust.json

aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://vmimport-role-policy.json

aws iam put-role-policy --role-name vmimport --policy-name AviController-vmimport-KMS-
Policy --policy-document file://avicontroller-kms-vmimport.json
```

##### 2 Create the required policies for the NSX Advanced Load Balancer Controller role.

AviController-Refined-Role is the role which will be attached to the Controller via the instance profile. Follow the below commands:

```
aws iam create-role --role-name AviController-Refined-Role --assume-role-policy-document
file://avicontroller-role-trust.json

aws iam create-policy --policy-name AviController-EC2-Policy --policy-document file://
avicontroller-ec2-policy.json

aws iam create-policy --policy-name AviController-S3-Policy --policy-document file://
avicontroller-s3-policy.json

aws iam create-policy --policy-name AviController-IAM-Policy --policy-document file://
avicontroller-iam-policy.json

aws iam create-policy --policy-name AviController-R53-Policy --policy-document file://
avicontroller-r53-policy.json

aws iam create-policy --policy-name AviController-ASG-Policy --policy-document file://
avicontroller-asg-policy.json

aws iam create-policy --policy-name AviController-SQS-SNS-Policy --policy-document file://
```

```
avicontroller-sqs-sns-policy.json
```

```
aws iam create-policy --policy-name AviController-KMS-Policy --policy-document file://
avicontroller-kms-policy.json
```

**Note** Following are the optional policies for AWS DNS service and the SNS-SQS feature:

- AviController-R53-Policy
- AviController-AutoScalingGroup-Policy
- AviController-SQS-SNS-Policy
- AviController-KMS-Policy (supported as of release 17.2.8)

### 3 Attach policies to the NSX Advanced Load Balancer Controller role.

Once the policies (AviController-EC2-Policy, AviController-R53-Policy, AviController-IAM-Policy, etc.) are created (in Step 2), attach them to the AviController-Refined-Role.

```
aws iam attach-role-policy --role-name AviController-Refined-Role --policy-arn
"arn:aws:iam::123456789012:policy/AviController-EC2-Policy" aws iam attach-role-policy
--role-name AviController-Refined-Role --policy-arn "arn:aws:iam::123456789012:policy/
AviController-R53-Policy" aws iam attach-role-policy --role-name AviController-Refined-
Role --policy-arn "arn:aws:iam::123456789012:policy/AviController-AutoScalingGroup-Policy"
aws iam attach-role-policy --role-name AviController-Refined-Role --policy-arn
"arn:aws:iam::123456789012:policy/AviController-SNS-Policy" aws iam attach-role-policy
--role-name AviController-Refined-Role --policy-arn "arn:aws:iam::123456789012:policy/
AviController-SQS-Policy" aws iam attach-role-policy --role-name AviController-Refined-
Role --policy-arn "arn:aws:iam::123456789012:policy/AviController-ASG-Notification"
aws iam attach-role-policy --role-name AviController-Refined-Role --policy-arn
"arn:aws:iam::123456789012:policy/AviController-KMS-Policy"
```

**Note** Following are the optional policies for AWS DNS service and SNS-SQS feature:

- AviController-R53-Policy
- AviController-AutoScalingGroup-Policy
- AviController-SQS-SNS-Policies
- AviController-KMS-Policy

### 4 Create an instance profile and apply this instance profile to the EC2 role.

```
aws iam create-role --role-name AviController-Refined-Role --assume-role-policy-document
file://avicontroller-role-trust.json

aws iam create-policy --policy-name AviController-EC2-Policy --policy-document file://
avicontroller-ec2-policy.json

aws iam create-policy --policy-name AviController-S3-Policy --policy-document file://
avicontroller-s3-policy.json

aws iam create-policy --policy-name AviController-IAM-Policy --policy-document file://
avicontroller-iam-policy.json
```

```
aws iam create-policy --policy-name AviController-R53-Policy --policy-document file://
avicontroller-r53-policy.json

aws iam create-policy --policy-name AviController-ASG-Policy --policy-document file://
avicontroller-asg-policy.json

aws iam create-policy --policy-name AviController-SQS-SNS-Policy --policy-document file://
avicontroller-sqs-sns-policy.json

aws iam create-policy --policy-name AviController-KMS-Policy --policy-document file://
avicontroller-kms-policy.json
```

---

### Note

- The **AWS put-role-policy** command creates an inline policy in the role (as opposed to an attached policy).
  - Make sure to replace “123456789012” with the applicable AWS account ID.
- 

### Using AWS Web Interface

The various roles and the associated policies mentioned in the previous section can be created using the AWS web interface (AWS management console) too.

This section discusses configuration steps for the following mandatory policies and the associated roles:

- vmimport policy
- vmimport role (associated with vmimport policy)
- AviController-Refined-Role
- AviController-EC2-Policy (associated with AviController-Refined-Role)
- AviController-IAM-Policy (associated with AviController-Refined-Role)

Follow the same steps to create the optional policies as required.

## Procedure

### 1 Creating vmimport policy.

- a Log in to the AWS console using your AWS customer account, select **Policies**.
- b Select **Create policy**, select **JSON tab**, copy and paste the content from the JSON file (vmimport-role-policy.json), and click **Review Policy**.

aws Services Resource Groups

### Create policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

This policy validation failed and might have errors converting to JSON : The policy must have at least one statement For more information about the IAM policy grammar, see [AWS IAM Policies](#)

Visual editor JSON Import managed policy

```

1 {
2 "Version": "2012-10-17",
3 "Statement": []
4 }

```

aws Services Resource Groups

```

1 {
2 "Version": "2012-10-17",
3 "Statement": [
4 {
5 "Effect": "Allow",
6 "Action": [
7 "s3:ListBucket",
8 "s3:GetBucketLocation"
9],
10 "Resource": "*"
11 },
12 {
13 "Effect": "Allow",
14 "Action": [
15 "s3:GetObject"
16],
17 "Resource": "*"
18 }
19]
20 }

```

Cancel Review policy

- c Provide the name for the policy (vmimport), the description (optional), click **Create Policy**. It is mandatory to use the name of the vmimport policy as vmimport

### 2 Creating vmimport role and associating it with the vmimport policy.

- a Select **Roles**, then click **Create role**.
- b Select type of trusted identity (**AWS Service**), choose the service (**EC2**) that will use this role, and click **Next: Permissions**.
- c Select the policy created in the previous step (vmimport policy), and click **Next: Review**.

- d Provide **Role name**, **Role description**, and click **Create role**.
- e Once the role is created, the AWS web interface will exhibit the message.

---

**Note** For vmimport role, Trust relationships should be edited. Navigate to the **Trust relationships** tab, click **Edit**, and copy the content of vmimport-role-trust.json (from the table mentioned in the beginning) to the **JSON** tab, and click **Update Trust Policy**.

---

### 3 Creating AviController-Refined-Role.

- a To Create Policies, select the **Policies** option on the AWS web interface, and click **Create Policy**.
- b Select the **JSON** tab, copy the content from the JSON file (avicontroller-role-policy.json), paste it in the JSON box, and click **Review Policy**.
- c Provide the name for the policy (AviController-EC2-Policy), the description (optional), and click **Create Policy**.
- d Once the policy is successfully created, the AWS web interface will exhibit the message.

---

**Note**

- Follow the steps mentioned above to create AviController-IAM-Policy. Choose the policy name and the JSON file as mentioned in the table provided at the beginning of the article.
  - Based on the requirement, create the other optional policies as well. For example, If NSX Advanced Load Balancer will use the AWS DNS service, create a policy named “AviController-R53-Policy” and copy-and-paste the contents of the avicontroller-role-53-policy.json file into the **Policy Document** field.
- 

### 4 Creating Role and Associating with the Required Policies.

- a Select **Roles**, and click **Create Role**.
- b Select type of trusted identity (**AWS Service**), choose the service (**EC2**) that will use this role, and click **Next: Permissions**.
- c select the policy created in the previous step (AviController-EC2-Policy), and (AviController-IAM-Policy), and select **Next: Preview**.
- d Provide the role name (AviController-Refined-Role), the description (optional), and click **Create role**.
- e Once the role is created, the AWS web interface will exhibit the message.

## Results

The new roles should be on the list.

---

**Note** There are 2 ways, an AWS cloud can be created in NSX Advanced Load Balancer, namely, Using Access/Secret key and Using IAM roles of the Controller.

Both of the methods mentioned above, require a `vmimport` role to be present. However, while using the Access/Secret key method, the user whose keys are used must have all the necessary permissions for carrying out all the operations done in NSX Advanced Load Balancer. `AviController-Refined-Role` needs to be present, if we choose using IAM roles of the Controller option. If the `AviController-Refined-Role` role is created using AWS CLI, then, an instance-profile is required as created in step-4. But, if the role is created using AWS GUI, then it is not required to create instance-profile separately, as it is automatically created along with the role.

---

## Scoping AWS Permissions

This article discusses the steps to create policy files for deployments where the scope of IAM roles have to be restricted to specific resources or types.

### Overview

The policy files provided in this reference use a broader scope to apply the permissions.

The scope of action in an AWS policy can be restricted in two ways:

- Explicitly specifying the resource(s)
- Specifying a condition Resource(s) are specified using their ARN (Amazon Resource Name).

Below are a couple of notations to specify ARN.

- `arn:partition:service:region:account-id:resource`
- `arn:partition:service:region:account-id:resourcetype/resource`

ARN has the following components:

Partition — Partition the aws resource is in. It can have 3 values

- `aws` – Public AWS partition
- `aws-cn` – AWS China
- `aws-us-gov` – AWS Government Cloud

Service — The service namespace like S3, EC2, KMS, IA.

Region — The region resources are in.

Account- { "Sid": "VisualEditor1", "Effect": "Allow", "Action": [ "ec2:RebootInstances", "ec2:TerminateInstances", "ec2:StartInstances", "ec2:StopInstances" ], "Resource": "\*", "Condition": { "StringLike": { "ec2:ResourceTag/AVICLOUD\_UUID": "\*" } } }id — The ID of the AWS account that owns the resource.

Resource, resourcetype/resource — This part points to the actual resource.

These AWS resources can be uniquely identified by their names. These resources are used to restrict scope of the IAM permissions. For example, `arn:aws:s3:::avi` or `arn:aws:iam:::role/vmimport`.

But, for the resources that can not be uniquely identified using resource name like EC2 resources, ARN is made up using resource id, and since a resource ID is not known in advance, these resources can be used to restrict IAM permissions. In this case, the conditions in the policy document are used to restrict permissions.

In the example snippet shown below, the tag key for `AviCloud_UUID` is used to restrict the permission of a user or a role. This policy makes sure that a user or a role can only perform the specified operations on the resources that have the resource tag key set as `AVICLOUD_UUID`.

```
{ "Sid": "VisualEditor1", "Effect": "Allow", "Action": ["ec2:RebootInstances",
"ec2:TerminateInstances", "ec2:StartInstances", "ec2:StopInstances"], "Resource": "*",
"Condition": { "StringLike": { "ec2:ResourceTag/AVICLOUD_UUID": "*" } }}
```

### Scoping AWS Permissions Using JSON Policies

This section discusses various tags for the IAM policies which can be modified or restricted in the JSON policies as per the deployment requirements.

| IAM Policies                                  | Tag Values which can be modified                   |
|-----------------------------------------------|----------------------------------------------------|
| <a href="#">avicontroller-ec2-policy.json</a> | <code>cloud_name:cloud_uuid</code>                 |
| <a href="#">AviController-S3-policy.json</a>  | <code>new_se_prefix</code>                         |
| <a href="#">AviController-R53-policy.json</a> | <code>hosted_zone_id</code>                        |
| <a href="#">AviController-IAM-policy.json</a> | <code>custom-avi-&lt;service(ec2/s3...)&gt;</code> |
| <a href="#">avicontroller-kms-policy.json</a> | <code>key_id_to_be_used</code>                     |

- 1 [avicontroller-ec2-policy.json](#) — It is used to create an IAM policy without any modification. Permissions for this policy are restricted based on the resource tag key. For further restriction, modify the policy file and add the desired value for the tag key. In this case, the permissions apply only to the resources that have the same tag key and the tag value. Tag value should be in the format `cloud_name:cloud_uuid`. Replace the specific tag value as per the deployment.
- 2 [AviController-S3-policy.json](#) — Bucket name is derived from the Service Engine name prefix, configured in the Service Engine Group. The default SE name prefix is NSX Advanced Load Balancer so the bucket name starts with `avi` (converting the prefix to lower case). If the SE prefix value is changed, use the same value but in lower case. Note: Replace the `new_se_prefix` value as per the deployment.
- 3 [AviController-R53-policy.json](#) — Use this policy to restrict R53 permissions. The JSON file is updated with the ID of the Route53-hosted zone. The selected zone is used for registration and de-registration of VIPs and FIPs. In the example snippet shown below, the policy restricts the above 2 permissions to `hosted_zone_id`. Replace the `hosted_zone_id` value as per the deployment.



- 4 [AviController-IAM-policy.json](#)— This policy file must be updated with correct `role_name`, `instance_profile_name` (same as the role name), and the common prefix of created policies using IAM Role Setup for Installation into AWS. In the example snippet shown below, for an NSX Advanced Load Balancer Controller role name – `CustomAviRole` , policies are created with the common prefix – `custom-avi-<service(ec2/s3...)>`. Replace the prefix value as per the deployment.
- 5 [avicontroller-kms-policy.json](#) — This policy is used to enable S3/EBS encryption. Permissions on KMS Encryption keys can be managed by using both resource-based policies or IAM policies. To restrict IAM policies, a specific key ID can be provided. Or, the same permissions can be directly applied to KMS key under the Key Policy section. Replace the `key_id_to_be_used` value as per the deployment.

## AWS User Cross-Account AssumeRole

This section discusses the AWS User Cross-Account AssumeRole options available as the credential method for NSX Advanced Load Balancer deployment with AWS.

NSX Advanced Load Balancer supports deployment of Amazon Web Services (AWS) with multiple AWS accounts utilizing the IAM AssumeRole functionality.

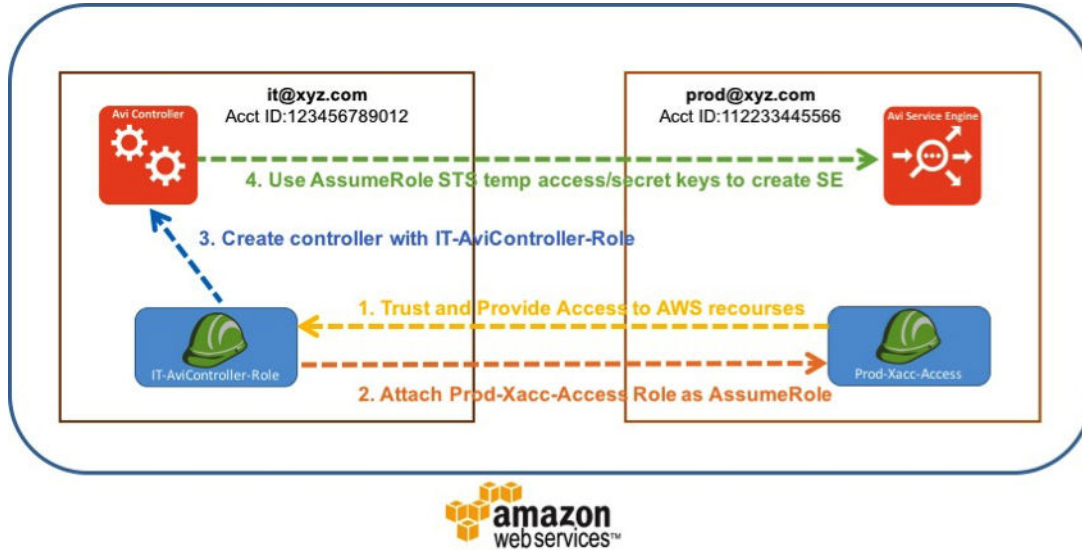
IAM roles provide access across AWS accounts to the AWS resources/API from the respective accounts, instead of sharing user Access Key ID and Secret Access Key from different accounts.

While creating the AWS cloud-type, new option of Use Cross-Account AssumeRole is available on the NSX Advanced Load Balancer. The Use Cross-Account AssumeRole feature can be enabled if the AWS cloud needs to be created in an AWS account other than the one that hosts the Controller.

### Use Case of Cross-Account Assume Role

Consider a hypothetical organization XYZ Corp with multiple AWS account IDs: IT – 123456789012 Prod – 112233445566.

Figure 12-1. Cross-Account Assume Role



In a general deployment, the XYZ Corp would require an Access Key ID and Secret Access Key to create NSX Advanced Load Balancer Service Engine cloud in AWS for each account. Sharing the AWS access keys for respective accounts would be a security concern with this.

In addition to this, it would be cumbersome to track all the keys and update them. Instead AWS IAM Roles can be used in every account. The IAM roles feature provides access to the AWS resources or API access across AWS accounts.

### Enabling User Cross-Account AssumeRole

The Use Cross-Account AssumeRole feature can be enabled during the cloud creation step.

Follow the steps mentioned below to enable the feature:

#### Prerequisites

The prerequisite configuration is required on AWS to set up the IAM user or roles to access other accounts using [Assume Role](#).

#### Procedure

1. Navigate to **Infrastructure > Cloud**, click **Create**. Provide the desired name of the cloud, and select Amazon Web Services as the **Cloud Infrastructure Type**, and click on **Next**.

- 2 Enable the *Use Cross-Account Assume Role* checkbox, available under the *AWS User Credentials* section.

The screenshot shows the 'New Cloud: test' configuration window in the VMware NSX Advanced Load Balancer interface. The window is divided into three steps: Step 1: Select Cloud, Step 2: Infrastructure (current), and Step 3: VPC/Network/Encryption. In Step 2, the 'AWS User Credentials' section is highlighted with a red box. This section contains two radio buttons: 'Use Access/Secret Key' (selected) and 'Use AWS Identity and Access Management (IAM) Roles of Controller'. Below these are fields for 'Access Key ID' and 'Secret Access Key'. At the bottom of this section, the 'Use Cross-Account AssumeRole' checkbox is visible and highlighted with a red box. The 'Access Key ID' field contains the text 'Access Key ID' and the 'Secret Access Key' field contains the text 'Secret Access Key'.

### Configuring User Cross-Account AssumeRole

This section discusses the steps required to configure User Cross-Account Assume Role.

The configuration steps are divided into the following main points:

- Creating a role using AWS user interface.
- Granting access to the role using AWS user interface.
- Configuring NSX Advanced Load Balancer to use User Cross-Account Assume Role for AWS accounts.

### Creating Role Using AWS User Interface

This section explains the steps to setup a Prod-Xacc-Access in the Prod AWS account.

#### Prerequisites

AWS accounts require access to AWS resources or APIs. In this example, the NSX Advanced Load Balancer Controller is hosted in the IT account (AWS account id – 123456789012) and the NSX Advanced Load Balancer Service Engine cloud provides data path services in the Prod account (AWS account id – 112233445566). Use the account IDs and resource ARNs that are applicable to your environment, while following this guide. Cross-account setup is explained in Delegate Access Across AWS Accounts Using IAM Role.





## Procedure

- 1 In Prod account, set up the Prod-Xacc-Access role which will be a cross-account role. Navigate to **IAM > Roles** and click on **Create New Role**. Select Another AWS account, and provide **Account ID**, and click on **Next:Permissions**. Enter the **AWS account ID** of the AWS account which can assume this role. In this example, it is the IT account (AWS account ID – 123456789012). You can choose **Require MFA** based on your requirement.

### Create role

1 2 3

#### Select type of trusted entity

|                                                                                                                                |                                                                                                                                               |                                                                                                                                         |                                                                                                                                            |
|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
|  <b>AWS service</b><br>EC2, Lambda and others |  <b>Another AWS account</b><br>Belonging to you or 3rd party |  <b>Web identity</b><br>Cognito or any OpenID provider |  <b>SAML 2.0 federation</b><br>Your corporate directory |
|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|

Allows entities in other accounts to perform actions in this account. [Learn more](#)

#### Specify accounts that can use this role

Account ID\*  ⓘ

- Options**
- ☐ Require external ID (Best practice when a third party will assume this role)
  - ☐ Require MFA ⓘ

\* Required

[Cancel](#)

[Next: Permissions](#)

- 2 Select the policies required by the Prod-Xacc-Access role to create the NSX Advanced Load Balancer SE for providing NSX Advanced Load Balancer functionality, and click **Review**. The following are the policies attached to this role in this reference section:
  - AviController-EC2-Policy
  - AviController-IAM-XAccess-Policy
  - AviController-R53-Policy
  - AviController-S3-Policy

- 3 Provide the **Role name** (Prod-Xacc-Access), **Role description** (optional), and click **Create Role**.

Create role

123

Review

Provide the required information below and review this role before you create it.

Role name\*

Prod-Xacc-Access3

Use alphanumeric and '+=, @ \_ .' characters. Maximum 64 characters.

Role description

The role for cross-account access

Maximum 1000 characters. Use alphanumeric and '+=, @ \_ .' characters.

Trusted entities

The account 123456789015

\* Required

Cancel

Previous

Create role

To summarize, for Prod-Xacc-Access role, the role ARN will be displayed as `arn:aws:iam::112233445566:role/Prod-Xacc-Access`. Ensure that the format is: `arn:aws:iam::account-id:role/role-name`.

### Granting Access to Role

Once the Prod-Xacc-Access is set up in the Prod AWS account (112233445566), any user or role in the IT AWS account (123456789012) will be able to assume the role with appropriate configuration, as explained in the steps given in this section:

Create the policy Cross-Account-AssumeRole-policy, which will be attached to the IT-AviController-Role role.

### Prerequisites

Create the IT-NSX Advanced Load Balancer Controller-Role role in the IT AWS account (123456789012) to assume the role of Prod-Xacc-Access in the Prod AWS Account (112233445566). This is the role to which the Controller instance will be attached.

### Procedure

- 1 Navigate to *IAM > Policies*, and click *Create Policy*. Configure the policy with `sts:AssumeRole` action and provide the resource reference for Prod-Xacc-Access role's ARN, which in this example is `arn:aws:iam::112233445566:role/Prod-Xacc-Access`.

```
{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::112233445566:role/Prod-Xacc-Access"
 }
}
```

- 2 Navigate to the *JSON* tab, provide the syntax mentioned above, and click on *Review Policy* to save the policy. For attaching multiple accounts, use the following policy syntax:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::<ACCOUNT-ID1>:role/<Role-Name-1>"
 },
 {
 "Effect": "Allow",
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::<ACCOUNT-ID1>:role/<Role-Name-2>"
 },
 {
 "Effect": "Allow",
 "Action": "sts:AssumeRole",
 "Resource": "arn:aws:iam::<ACCOUNT-ID2>:role/<Role-Name>"
 }
]
}
```

- 3 Navigate to **IAM > Roles**, click **Create Role**. Select **AWS Service**, select **Amazon EC2** as the role type, and click **Next:Permissions**.

- 4 Under Attach-Policy, select the Cross-Account-AssumeRole-Policy created earlier and also the AviController-EC2-Policy, which is required for the Controller's internal operations.

**Create role** 1 2 3

Choose one or more policies to attach to your new role.

Create policy

Filter: Policy type  Showing 3 results

|                                     | Policy name                               | Attachments | Description                                                   |
|-------------------------------------|-------------------------------------------|-------------|---------------------------------------------------------------|
| <input type="checkbox"/>            | cross-account-policy                      | 1           |                                                               |
| <input type="checkbox"/>            | AmazonVPCCrossAccountNetworkInterfaceO... | 0           | Provides access to create network interfaces and attach th... |
| <input checked="" type="checkbox"/> | Cross-Account-AssumeRole-Policy           | 3           | Policy to assume the role of other AWS accounts - created...  |

\* Required Cancel Previous Next: Review

**Create role** 1 2 3

Choose one or more policies to attach to your new role.

Create policy

Filter: Policy type  Showing 14 results

|                                     | Policy name                           | Attachments | Description                          |
|-------------------------------------|---------------------------------------|-------------|--------------------------------------|
| <input type="checkbox"/>            | AviController-Role-Policy             | 1           |                                      |
| <input type="checkbox"/>            | AviController-AIO                     | 1           | all-in-one avicontroller role policy |
| <input type="checkbox"/>            | AviController-ASG-Notification        | 3           |                                      |
| <input type="checkbox"/>            | AviController-AutoScalingGroup-Policy | 4           |                                      |
| <input checked="" type="checkbox"/> | AviController-EC2-Policy              | 17          | policygen-201512171511               |

\* Required Cancel Previous Next: Review

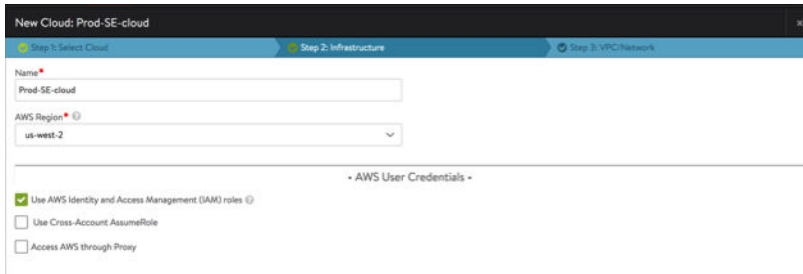
- 5 Provide the Role name as IT-AviController-role, the description (optional), and click **Create role**. Verify if both policies are now attached.

### Configuring NSX Advanced Load Balancer for Cross-Account Role

After completing the prerequisite setup, you can configure the IAM role for NSX Advanced Load Balancer Controller as IT-AviController-Role by following the steps mentioned in AWS Installation Guide. Skip the cloud creation steps and choose *No Orchestrator* during the set-up. Ensure that the VPCs and subnets are configured in AWS, so that NSX Advanced Load Balancer Controller management interface and Service Engine's management networks will be reachable from other accounts.

## Procedure

- 1 Create the AWS cloud by navigating to **Infrastructure > Clouds** and click on **Create**. Choose the appropriate region and select the checkbox for **Use AWS Identity and Access Management (IAM) roles**. This will ensure that the IT-AviController-Role is attached to the NSX Advanced Load Balancer Controller when it is launched. Both IAM role and access/secret key can be used for cross-account role given the role/user has the necessary permissions (cross-account policy).



- 2 Select the checkbox for Use Cross-Account AssumeRole, if the cloud has been set up in another AWS account. However, in this case, the NSX Advanced Load Balancer SE cloud is created in the Prod AWS account (112233445566) from the NSX Advanced Load Balancer Controller hosted in IT AWS account (123456789012). As the cross-account AssumeRole has already been set up for IT-AviController-Role, on selecting the checkbox, the back-end APIs will fetch the associated AssumeRole accounts and their roles and display them in the drop-down menu. If there were no AssumeRoles attached, then the list would have been empty. There would be a text box that can be used to enter the ARN of the role for which the Controller instance's IAM role (in our case, IT-AviController-Role) can assume the role.
- 3 Select the ARN for the account and role, where the SE targets will be deployed.
- 4 If the role has appropriate access and is correctly setup, NSX Advanced Load Balancer Controller will fetch the AWS account details and configuration's VPC networks. Similarly, this will continue for the older SE AWS cloud setup.
  - Cloud setup will progress, and the NSX Advanced Load Balancer SE AMI will be copied to the target account.
  - Once the transfer is completed, the cloud status will move to Cloud ready for Virtual Service placement.
- 5 Virtual services can now be configured on this cloud by following the steps mentioned at [Creating Virtual Service](#).

## AWS Customer Account Key

AWS Customer Account Key is a unique authentication key associated with the AWS account.

Access credentials are needed by the NSX Advanced Load Balancer Controller to communicate with AWS APIs. AWS cloud configuration with NSX Advanced Load Balancer SaaS Controller only supports the Use Access/Secret Key credentials method. For detailed information on using access key on AWS, see [Managing Access Keys for Your AWS Account User](#).



## Encryption

This section covers various encryption options available for NSX Advanced Load Balancer integration with AWS.

### SNS-SQS Encryption

Server-Side Encryption (SSE) of Amazon Simple Queue Service (SQS) message queues is supported by NSX Advanced Load Balancer. Encrypting a queue does not encrypt backlogged messages, nor does turning off encryption remove encryption from backlogged messages. SQS queue encryption is supported only in 3 AWS regions as of the time of this writing: US EAST (N. Virginia), US EAST (Ohio), and US WEST (Oregon).

#### Prerequisites

This section covers the prerequisites for NSX Advanced Load Balancer integration with AWS.

##### Prerequisites

For the NSX Advanced Load Balancer Controller to work with encrypted SQS queues and other artifacts of Amazon Simple Notification Service (SNS), either the user whose access/secret key is used or the `AviController-Refined-Role` must have the following policies attached to it:

- `AviController-SQS-Policy`
- `AviController-SNS-Policy`
- `AviController-KMS-Policy`

The `AviController-Refined-Role` must be able to decrypt received messages when polling SQS queues. For this, the `AviController-KMS-Policy` must be updated to include within it a write action, `kms:Decrypt`. JSON files for this role and policy are shown in the [IAM Role Setup for Installation into AWS](#) section.

#### Customer-managed Customer Master Keys

This section discusses about the customer-managed CMKs and adding permissions to it.

The primary resources in the AWS key management service are Customer Master Keys (CMK)s. Customer-managed CMKs are CMKs the user creates, manages, and uses. It is in contrast with AWS-managed CMKs, which are created, managed, and used on the user's behalf by an AWS service that is integrated with AWS KMS.

This includes enabling and disabling the CMK, rotating its cryptographic material, and establishing the IAM policies and key policies that govern access to the CMK, and also using the CMK in cryptographic operations.

SSE of an SQS queue is done using a customer managed CMK, and an SNS topic must be able to make use of that encryption key to encrypt/decrypt a message that it wants to send to the queue. For this, the encryption key's policy must be modified to allow SNS service to work with it.

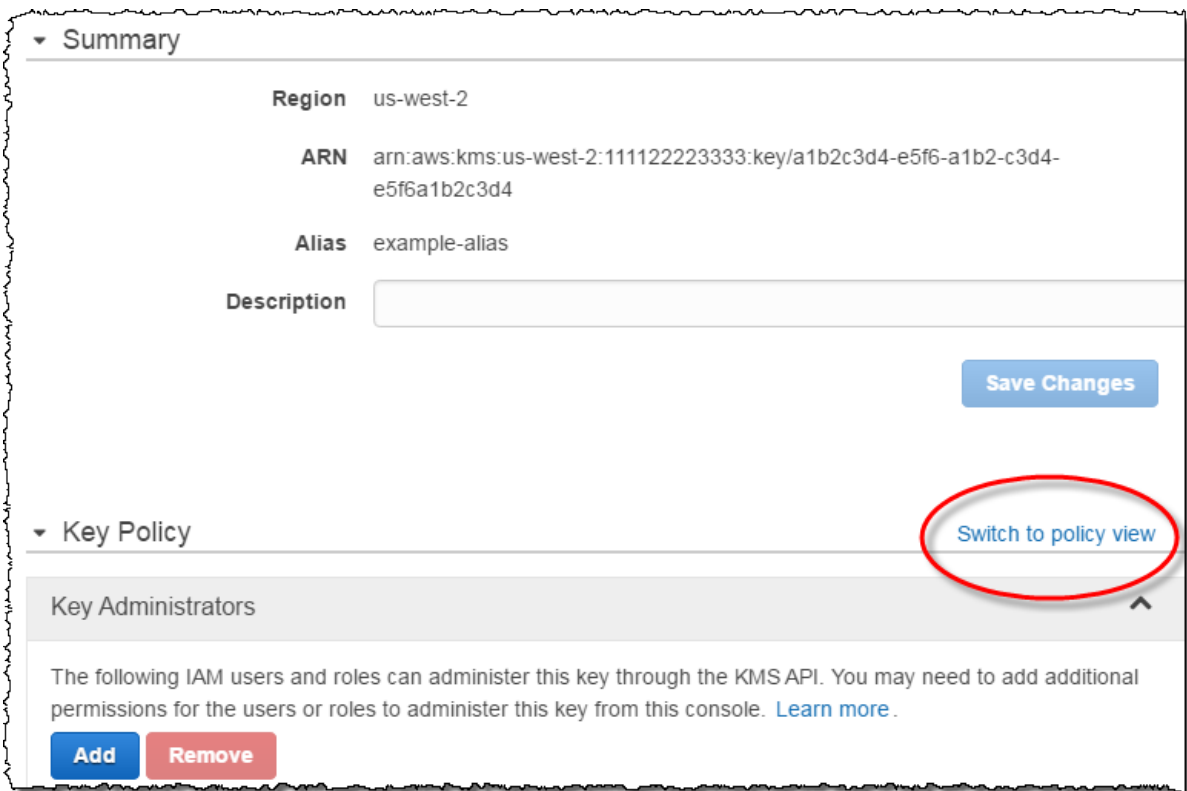
## Adding Permissions to Customer-managed CMKs

Sign in to the AWS Management Console and open the AWS Identity and Access Management (IAM) console. Follow the steps mentioned below:

### Prerequisites

### Procedure

- 1 In the left navigation pane, choose Encryption keys.
- 2 For Region, choose the appropriate AWS Region.
- 3 Choose the alias of the CMK whose key policy document you want to edit.
- 4 On the Key Policy line, choose Switch to policy view.



a

a

## 5 Add following statement in the key policy.

a

```
{
 "Sid": "Allow SNS to use CMK",
 "Effect": "Allow",
 "Principal": {
 "Service": "sns.amazonaws.com"
 },
 "Action": [
 "kms:GenerateDataKey*",
 "kms:Decrypt"
],
 "Resource": "*"
}
```

## 6

### Results

#### Example:

#### What to do next

### Configuring SQS Encryption Using NSX Advanced Load Balancer UI

Enabling SQS queue encryption through the NSX Advanced Load Balancer UI is available only for the following AWS regions: US East (N. Virginia) US East (Ohio) US West (Oregon) During the cloud creation steps, select one of the AWS regions mentioned above. Follow the steps mentioned below to enable SQS queue encryption through the UI.

Enabling SQS queue encryption through the NSX Advanced Load Balancer UI is available only for the following AWS regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)

During the cloud creation steps, select one of the AWS regions mentioned above.

Follow the steps mentioned below to enable SQS queue encryption through the NSX Advanced Load Balancer UI:

#### Procedure

- 1 Navigate to **Infrastructure > Clouds**, click on **Create** to create a new cloud or use the **Edit** icon to edit the existing cloud.

- 2 Select **VPC/Network/Encryption** option, and enable the checkbox for **Enable Simple Queue Service (SQS) for Autoscale Groups Monitoring** available under the AWS VPC and Availability Zones section.

Edit Cloud: awsCloud

Infrastructure VPC/Network/Encryption

Availability Zone\* <sup>?</sup> us-west-2a SE Management Network\* <sup>?</sup> 2A-nw-10 - 10.144.10.0/24

Availability Zone\* <sup>?</sup> us-west-2b SE Management Network\* <sup>?</sup> 2B-nw-10 - 10.144.74.0/24

Availability Zone\* <sup>?</sup> us-west-2c SE Management Network\* <sup>?</sup> 2C-nw-10 - 10.144.138.0/24

+ Add Availability Zone

☒ Enable Simple Queue Service (SQS) for Autoscale Groups Monitoring <sup>?</sup>

☐ Allow wildcard access to SEs <sup>?</sup>

• DNS Settings •

Cancel Save

- 3 Select *Use Encryption for SQS Queue* available under the *Encryption* section, and select value for *AWS KMS Master Key ARN ID* from the drop-down menu as shown below.

Edit Cloud: awsCloud

Infrastructure VPC/Network/Encryption

☐ None ☒ Amazon Route 53 <sup>?</sup> ☐ DNS Profile <sup>?</sup>

• Encryption •

☒ Use Encryption for SE S3 Bucket <sup>?</sup> ☒ Use Encryption for SE AMI/EBS volumes <sup>?</sup>

AWS KMS Master Key ARN ID\* <sup>?</sup> aws/s3 - d0f8aa45-bceb-4698-a053-2ef65ca59333 AWS KMS Master Key ARN ID\* <sup>?</sup> aws/ebs - 8a6ee249-cdc5-499d-af31-d69a24ba81b2

☒ Use Encryption for SQS Queues <sup>?</sup>

AWS KMS Master Key ARN ID\* <sup>?</sup> sqs-key - 7f455e16-d3aa-43b3-a64b-dbd984e73791

• Other Settings •

Cancel Save

### Configuring SQS Encryption Using NSX Advanced Load Balancer CLI

This section elaborates on the steps to configure SQS encryption using CLI.

Substituting your cloud name and key\_id as appropriate, type these CLI commands into the NSX Advanced Load Balancer shell prompt:

## Procedure

- ◆ Login to the NSX Advanced Load Balancer shell prompt and use the configure cloud cloud name.

a

```
[admin:controller]: > configure cloud cloudAWS [admin:controller]: cloud>
aws_configuration
[admin:controller]: cloud:aws_configuration> sqs_encryption
[admin:controller]: cloud:aws_configuration:sqs_encryption> mode
aws_encryption_mode_sse_kms
[admin:controller]: cloud:aws_configuration:sqs_encryption> master_key <key_id>
[admin:controller] : cloud:aws_configuration:sqs_encryption> save
```

## EBS Encryption

NSX Advanced Load Balancer supports EBS and S3 encryption using AWS SSE-KMS which encrypts the Amazon Machine Image (AMI). The section explains Amazon EBS encryption which is a solution offered to encrypt EBS volumes.

Encrypting EBS volumes and attaching it to the supported instance type encrypts the data inside the volume, all data moving between the volume and the instance, all snapshots created from the volume, and all volumes created from those snapshots.

The data at rest within an Amazon S3 data center can be protected using AWS KMS. Server-side encryption is one way to use AWS KMS, in which you can protect your data using the customer master key.

The three different modes of service-side encryption are:

- SSE-S3 – Amazon S3 manages the data and master encryption keys.
- SSE-C – User manages the encryption key.
- SSE-KMS – AWS manages the data key, but the user manages the master key in AWS KMS. For complete information on AWS KMS, refer to [How Amazon Simple Storage Service \(Amazon S3\) Uses AWS KMS](#).

NSX Advanced Load Balancer supports EBS and S3 encryption using AWS SSE-KMS which encrypts the Amazon Machine Image (AMI). For detailed information on AMI, see [Amazon Machine Images \(AMI\)](#).

## Configuring AWS Encryption on NSX Advanced Load Balancer

On deploying the NSX Advanced Load Balancer Controller instance in the AWS cloud, an Amazon Machine Image (AMI) is generated and uploaded to an Amazon Simple Storage Service (S3) bucket within the account. This Controller AMI is used to deploy the Service Engines as required.

Enabling encryption, encrypts both the Controller and SE AMIs. As explained earlier, this encryption is done for the EBS volume and S3 bucket. Enabling encryption does not dynamically update any existing SEs and is applied only to the newly launched SEs.

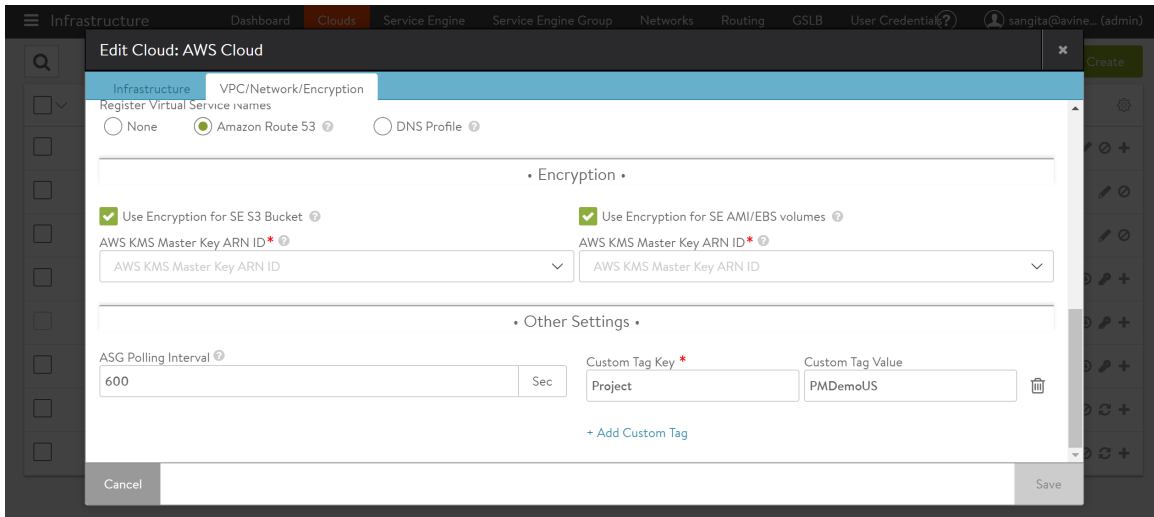
## Configuring EBS Encryption Using NSX Advanced Load Balancer UI

This section explains the steps to configure EBS Encryption using NSX Advanced Load Balancer UI.

The following are the steps to configure EBS Encryption Using NSX Advanced Load Balancer UI:

### Procedure

- 1 To enable the encryption on UI, navigate to **Infrastructure > Clouds**, and select the **AWS cloud** to enable encryption for. Click the **Edit** icon.
- 2 In the AWS User Credentials section, enable the checkbox for **Use Encryption** for SE S3 Bucket to encrypt S3 bucket used during the Service Engine image upload and enable the checkbox for **Use Encryption** for SE AMI/EBS volumes to encrypt Service Engine AMI, snapshot, or volume.
- 3 Select **SSE KMS** from the dropdown menu for **Encryption Mode**. For the **AWS KMS Master Key ARN ID** field, choose one of the relevant options:
  - a If the given credentials or Controller role has sufficient permissions to read the list of the keys, then they will be displayed in a drop-down menu. Choose the displayed option.
  - b The key ARN can be entered manually in the Customer Master Key (CMK) format `arn:aws:kms:AWS-Region:AWS-Account-ID:key/CMK-key-ID`. If left blank, the default KMS CMK of the service will be used.



### Results

Example:

## What to do next

### Note

- Most instance types are supported for EBS encryption. For complete information, see [Amazon EBS Encryption](#).
- The S3 bucket encryption feature requires VMimport.
- As a part of cloud orchestration, NSX Advanced Load Balancer Controller will upload and manage either an unencrypted or encrypted Service Engine AMI based on the **Use Encryption for SE AMI/EBS volumes** option.

## Enabling EBS Encryption Using NSX Advanced Load Balancer CLI

This section explains the steps to configure EBS Encryption using NSX Advanced Load Balancer CLI.

To enable the encryption using CLI, enter the Controller bash and enter the following options under the cloud configuration mode:

### Procedure

- ◆ `configure cloud aws_cloud aws_configuration s3_encryption [mode | key]  
ebs_encryption [mode | key].`

## What to do next

- Entering the mode keyword enables the SSE KMS mode of AWS encryption mode.
- Entering the key keyword allows you to enter the AWS KMS ARN ID of the master key for encryption.

## Enabling EBS Encryption Using NSX Advanced Load Balancer CLI.

This section explains the steps to configure EBS Encryption Using NSX Advanced Load Balancer CLI.

To enable the encryption using CLI, enter the Controller bash and enter the following options under the cloud configuration mode:

- Entering the mode keyword enables the SSE KMS mode of AWS encryption mode.
- Entering the key keyword allows you to enter the AWS KMS ARN ID of the master key for encryption.

```
configure cloud aws_cloud
 aws_configuration
 s3_encryption [mode | key]
 ebs_encryption [mode | key]
```

## Security Groups

This article discusses how to use security groups on the NSX Advanced Load Balancer to achieve additional flexibility and security in AWS clouds deployments.

By default, the NSX Advanced Load Balancer creates and manages a single security group (SG) for the SE. This SG manages the ingress/egress rules for the SE's management- and data-plane traffic. In certain customer environments, it may be required to provide custom SGs to be also be associated with the SEs management- and/or data-plane vNICs.

For the recommended security groups for AWS deployment, see [Recommended Security Group Rules for AWS Deployment](#).

### Default Security Group Rules

This section discusses the default security groups created by NSX Advanced Load Balancer.

The following are the rules which are added to the default security groups created by NSX Advanced Load Balancer:

- Data rules – Rules to open ports to communicate with virtual service.
- Management rules – This is for NSX Advanced Load Balancer Controller to SE communication. The following are the rules required for management communication. To enable SSH on port 22 To enable ping for all ICMP-IPv4 packets.
- Tunneling rules – Custom Protocol EtherIP (97), Custom Protocol CPHB (73), and Custom Protocol 63 (63).

The following are the different options available for the default security group. Each of the NSX Advanced Load Balancer created rules are added only to the security groups it created.

- ingress\_access\_mgmt
- ingress\_access\_data
- custom\_securitygroups\_mgmt
- custom\_securitygroups\_data

### Ingress Access for Management Interface

The following table lists behaviour and the possible values for the `ingress_access_mgmt` option:

| Possible Values        | Behaviour                                                      |
|------------------------|----------------------------------------------------------------|
| SG_INGRESS_ACCESS_NONE | Management rules are not set up                                |
| SG_INGRESS_ACCESS_ALL  | Management rules are setup with source IP address as 0.0.0.0/0 |
| SG_INGRESS_ACCESS_VPC  | Management rules are setup with source IP address as VPC CIDR  |



| Possible Values        | Behaviour                                                |
|------------------------|----------------------------------------------------------|
| SG_INGRESS_ACCESS_NONE | Data rules are not set up                                |
| SG_INGRESS_ACCESS_ALL  | Data rules are setup with source IP address as 0.0.0.0/0 |
| SG_INGRESS_ACCESS_VPC  | Data rules are setup with source IP address as VPC CIDR  |

The following are the limitations of the default security groups created by NSX Advanced Load Balancer:

- One security group is created per SE and AWS allows only 500 security groups per account.
- The source IP address for all the data and management traffic is set to either (0.0.0.0) or (VPC CIDR). There is no control to allow or disallow certain network only.
- AWS automatically allows all outbound traffic through security groups.
- NSX Advanced Load Balancer supports custom security group option, which allows customers to create their own security group. The custom security groups are attached to the SE, in addition to the default security groups. The default security groups are not of much use if the custom security group is in use.

## Installing the NSX Advanced Load Balancer Controller

To install NSX Advanced Load Balancer in AWS, deploy an EC2 instance of the Controller and run the setup wizard.

Installing the Controller consists of the following steps:

- Deploying an EC2 Instance
- Controller Initial Configuration

### Deploying an EC2 Instance

The following are the steps to deploy an EC2 instance:

#### Procedure

- 1 Access Amazon Web Services (AWS) using <https://aws.amazon.com> and log in using your AWS credentials.
- 2 Go to the NSX Advanced Load Balancer page on AWS Marketplace. Click **Continue** to start the AMI deployment process. You can either perform a 1-Click launch or a Manual Launch via the EC2 console, API, or CLI.

- 3 The subsequent steps discuss the Manual Launch (EC2 Console) process:
- Based on the deployment scale considerations, choose an appropriate instance type.
  - Select the appropriate VPC from the **Network** drop-down menu and select the network from the Subnet drop-down menu. This is the subnet in which the Controller will get the IP address for the management NIC.
  - Also, select the **Enable termination protection** option.
  - If installing with an IAM role instead of an AWS customer account key, select IAM role if you have created as explained in Credential Method. In this example we have used the *IAM Role AviController-Refined-Role*

1. Choose AMI 2. Choose Instance Type 3. Configure Instance Details 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

### Step 3: Configure Instance Details

Purchasing option ⓘ ☐ Request Spot instances

---

**Network** ⓘ vpc-19295f7c (10.144.0.0/16) | AVI-WEST2-VPC ⓘ [Create new VPC](#)

**Subnet** ⓘ subnet-d9ea5280(10.144.137.0/24) | 2C-nw-9 | us-west-2c 236 IP Addresses available ⓘ [Create new subnet](#)

**Auto-assign Public IP** ⓘ Use subnet setting (Disable) ⓘ

**Placement group** ⓘ No placement group ⓘ

---

**IAM role** ⓘ AviController-Refined-Role ⓘ [Create new IAM role](#)

---

**Shutdown behavior** ⓘ Stop ⓘ

**Enable termination protection** ⓘ ☒ Protect against accidental termination

**Monitoring** ⓘ ☐ Enable CloudWatch detailed monitoring  
[Additional charges apply.](#)

**EBS-optimized instance** ⓘ ☒ Launch as EBS-optimized instance

**Tenancy** ⓘ Shared - Run a shared hardware instance ⓘ  
[Additional charges will apply for dedicated tenancy.](#)

---

▼ **Network interfaces** ⓘ

| Device | Network Interface       | Subnet            | Primary IP  | Secondary IP addresses |
|--------|-------------------------|-------------------|-------------|------------------------|
| eth0   | New network interface ⓘ | subnet-d9ea5280 ⓘ | Auto-assign | <a href="#">Add IP</a> |

[Cancel](#)
[Previous](#)
[Review and Launch](#)
[Next: Add Storage](#)

- 4 In the Size (G/B) field, enter 128 to allocate 128 GB to the Controller instance, and go to the “Next: Tag Instance” option.

- Enter a name for this Controller instance and create a security group that allows traffic through the firewall, to allow communication between the Controller and the Service Engines (SEs). To know more, see [Security Groups in AWS](#).

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group  
☐ Select an existing security group

Security group name:   
 Description:

| Type            | Protocol | Port Range | Source             |
|-----------------|----------|------------|--------------------|
| SSH             | TCP      | 22         | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 443        | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 8443       | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 80         | Anywhere 0.0.0.0/0 |
| Custom UDP Rule | UDP      | 123        | Anywhere 0.0.0.0/0 |

Add Rule

**Warning**

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

Cancel Previous **Review and Launch**

- Select SSD as the storage type. (This enhances the responsiveness of the Controller web interface).

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group  
☐ Select an existing security group

Security group name:   
 Description:

| Type            | Protocol | Port Range | Source             |
|-----------------|----------|------------|--------------------|
| SSH             | TCP      | 22         | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 443        | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 8443       | Anywhere 0.0.0.0/0 |
| Custom TCP Rule | TCP      | 80         | Anywhere 0.0.0.0/0 |
| Custom UDP Rule | UDP      | 123        | Anywhere 0.0.0.0/0 |

Add Rule

**Warning**

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

**Boot from General Purpose (SSD)**

General Purpose (SSD) volumes provide the ability to burst to 3000 IOPS per volume, independent of volume size, to meet the performance needs of most applications and also deliver a consistent baseline of 3 IOPS/GiB.

☒ Make General Purpose (SSD) the default boot volume for all instance launches from the console going forward (recommended).

☐ Make General Purpose (SSD) the boot volume for this instance.

☐ Continue with Magnetic as the boot volume for this instance.

Free tier eligible customers can get up to 30GB of General Purpose (SSD) storage.

**Next**

Cancel Previous **Review and Launch**

- Review your EC2 instance, and click **Launch**.

## 8 Select the **Key Pair Settings** and select **Launch Instance**.

- a If you don't have key pair, create new key pair. After downloading the key pair, change the permissions to **"400"** (`chmod 400 ".pem"`) to do SSH.
- b If you have key pair, select a key pair for AML authentication.

## 9 The deployment status of the NSX Advanced Load Balancer Controller EC2 instance into AWS is displayed. When the instance is ready (status as *running*), you can access the instance using a private or public IP address. Wait for all checks to pass before setting up the Controller.

The screenshot displays the AWS Management Console interface for the EC2 Dashboard. On the left, a navigation menu lists various AWS services, with 'INSTANCES' expanded. The main panel shows a table of EC2 instances. One instance, 'Avi Controller 16.2-9091' (ID: i-5f94b782), is highlighted. Its status is 'running', and its status checks are '2/2 checks ...' (indicated by a green checkmark). The 'Private IP' address, '10.144.137.13', is highlighted in a red box. Below the table, the instance details are shown, including the private IP address.

## NSX Advanced Load Balancer Controller Initial Configuration

Use the following steps to complete the initial configuration for the Controller.

### Procedure

- 1 Log in to the SSH and use the `sudo /opt/avi/scripts/initialize_admin_user.py` command to configure the admin password for the first time login to the Controller.
- 2 Once the admin password is configured, point your browser to the Controller IP address and provide the input for remaining UI options as mentioned below.
  - a DNS and NTP Settings (This information can be entered later on too).
  - b Provide the desired email address for sending alerts from the controller (Can be set up at a later stage).
  - c Select **No Orchestrator** to complete the initial configuration.
  - d Continue by clicking **No** for Support multiple Tenants (Multi-tenancy can be enabled later).

- 3 Once the setup is completed, the browser will automatically refresh to the NSX Advanced Load Balancer Controller dashboard. This completes the installation process. The Controller is now ready for deploying virtual services. The next step is to configure AWS cloud on NSX Advanced Load Balancer.

## Initial Configuration Workflow for NSX Advanced Load Balancer Controller Deployed in AWS

This section discusses the initial configuration workflow for the Controller deployed in AWS

The NSX Advanced Load Balancer Controller does not have the initial password creation option when deployed on AWS. It is required to log in using an SSH key installed while creating the Controller in AWS clouds and set the password. This enhancement also introduces a change in the workflow to create the Controller cluster in AWS.

### Configuring Admin Password for the NSX Advanced Load Balancer Controller Set-up in AWS

Once the Controller is instantiated, execute the following command script to set password for the admin user:

```
local@Avi-Dev:~# ssh -i avitest-aws-key.pem admin@10.10.1.1 "sudo /opt/avi/scripts/
initialize_admin_user.py --password Password@123"

Avi Cloud Controller

Avi Networks software, Copyright (C) 2013-2017 by Avi Networks, Inc.
All rights reserved.

Version: 18.2.3
Date: 2019-05-01 22:43:26 UTC
Build: 9063
Management: 10.10.1.1/24 UP
Gateway: 10.10.1.10 UP

[node1.controller.local] Executing task 'sync_linux_one_user'
DEPLOY_OVERRIDES {}
[node1.controller.local] sudo: export PYTHONPATH=/opt/avi/python/lib && /opt/avi/scripts/
linux_user_maintenance.py --sync-user '{"username": "admin", "uid": 2000, "is_sudoer": true,
"controller": true, "unix_crypt_password": "abc12345", "delete": false}'
[node1.controller.local] out: The user `admin' is already a member of `sudo'.
[node1.controller.local] out:

[node1.controller.local] Executing task 'sync_cli_user'
Resetting password for user admin.
Password reset complete
local@Avi-Dev:~#
```

If the browser is pointed to the Controller IP address, a login screen is available with instructions to set up the admin password. Once the password is configured, the admin user can log in via the NSX Advanced Load Balancer UI.

Once the authentication is successful, the user is provided with the initial configuration workflow as usual.

### Changes for Cluster Set-up

While configuring the Controller cluster in AWS, the admin password for each node of the cluster is required as shown below:

### Edit Controller Configuration

#### Cluster Information

Controller Cluster IP ?

---

#### Cluster Nodes

| Hostname/IP * ?                            | Name                                       | Password ?                              | Public IP ?                              |
|--------------------------------------------|--------------------------------------------|-----------------------------------------|------------------------------------------|
| <input type="text" value="10.161.101.32"/> | <input type="text" value="10.161.101.32"/> | <input type="text" value="Password"/>   | <input type="text" value="Public IP"/>   |
| <input type="text" value="Hostname/IP ?"/> | <input type="text" value="Name"/>          | <input type="text" value="Password ?"/> | <input type="text" value="Public IP ?"/> |
| <input type="text" value="Hostname/IP ?"/> | <input type="text" value="Name"/>          | <input type="text" value="Password ?"/> | <input type="text" value="Public IP ?"/> |

This is the same password for the admin account which is discussed in the previous section.

## Configuring NSX Advanced Load Balancer Controller Cluster in AWS

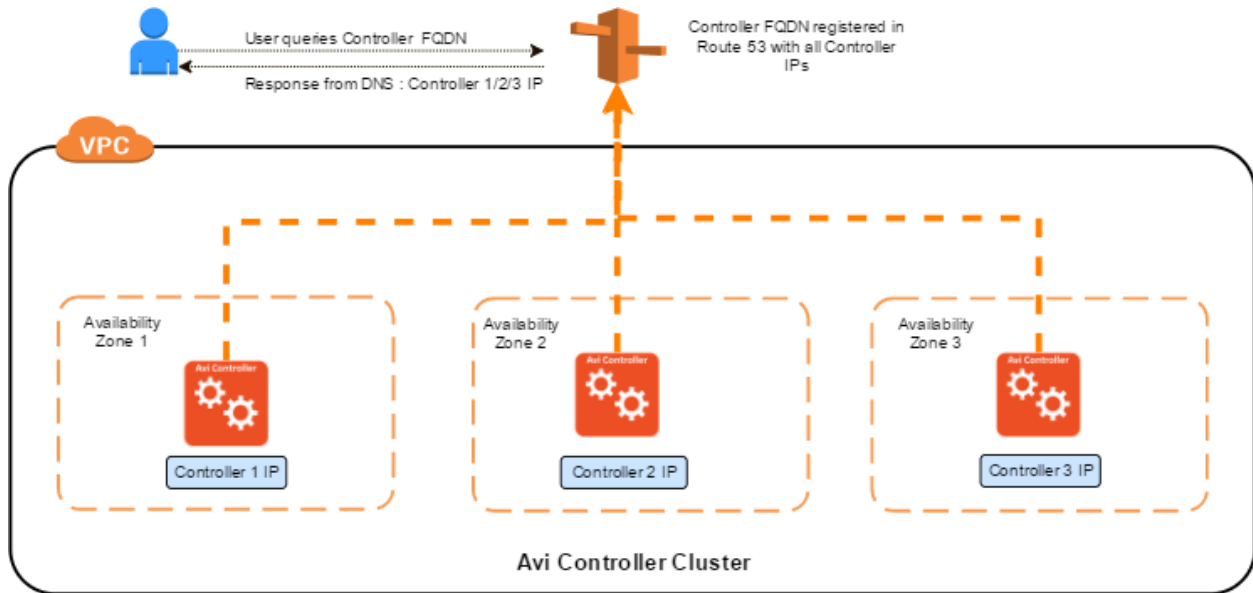
This section discusses the steps to configure the Controller Cluster in AWS.

To provide high availability (HA) for the Controller, add two additional Controller nodes to create a 3-node Controller cluster. For detailed instructions to configure an NSX Advanced Load Balancer, see [Deploying an NSX Advanced Load Balancer Controller Cluster](#).

In AWS environments, AWS Availability Zones (AZs) provide redundancy and separate fault domains. All AWS regions support a minimum of two AZs. To leverage the HA provided by AWS AZs, it is recommended to deploy different Controller instances of a cluster in different AZs.

## Managing a Controller Cluster across AZs

Each Controller receives an IP address from a different subnet given that an AWS subnet does not span across AZs.



In this scenario, it is recommended to create an FQDN in AWS Route 53 and associate all three Controller IPs with this FQDN. In addition, Route 53 health checks can be used in conjunction with multivalue routing when the FQDN is added to a public zone. This ensures that only healthy controller IPs are returned.

For AWS deployments where Controllers are on different subnets, Route 53 configuration with health checks enables resolving the Cluster's domain name to a Controller IP address directly. For detailed information on cluster configuration in AWS, see the [Controller Cluster Configuration in AWS](#).

## Publishing NSX Advanced Load Balancer Private IP DNS Records to Route 53

This section explains how to publish NSX Advanced Load Balancer private IP DNS Records to Route 53.

A Virtual Service on an NSX Advanced Load Balancer deployed in an AWS cloud can have two IP addresses:

- A private IP address, also known as a Virtual IP address (VIP).
- An external IP address, also known as floating IP address.

When Route 53 is enabled in the AWS cloud, the VIP is registered with the private hosted zone while the floating IP address is registered with the public hosted zone of AWS. The fully qualified domain names (FQDNs) registered in a private hosted zone are reachable only within the cloud's VPC.

### Use Case

The typical use cases of this feature are listed below:

Where the applications reside outside of AWS but need to use Route 53 as a DNS provider.

If there are a large number of AWS VPCs, currently it is required to associate the private hosted zone with every VPC to enable DNS resolution. By enabling this new feature, this tedious configuration is no longer required.

### Configuring Route 53

This feature is enabled by setting the `publish_vip_public_zone` flag (present in the AWS configuration) value to true through the CLI. This feature is available both when the `cloud-type` is set to AWS, and when the cloud is non-AWS but uses AWS DNS IPAM.

This feature is enabled by setting the `publish_vip_public_zone` flag (present in the AWS configuration) value to true through the CLI. This feature is available both when the `cloud-type` is set to AWS, and when the cloud is non-AWS but uses AWS DNS IPAM.

Log in to the NSX Advanced Load Balancer and follow the below steps when the `cloud-type` is set to AWS:

```
[admin:<controller-ip>]: > configure cloud <cloud_name>
[admin:<controller-ip>]: cloud> aws_configuration
[admin:<controller-ip>]: cloud:aws_configuration> publish_vip_to_public_zone
[admin:<controller-ip>]: > save
```

Log in to the NSX Advanced Load Balancer and follow the below steps when AWS DNS is used as a DNS Service:

```
[admin:<controller-ip>]: > configure ipamdnsproviderprofile <name>
[admin:<controller-ip>]: ipamdnsproviderprofile> aws_profile
[admin:<controller-ip>]: ipamdnsproviderprofile:aws_profile> publish_vip_to_public_zone
[admin:<controller-ip>]: ipamdnsproviderprofile:aws_profile> save
```

### Note

- To register the private VIP and the public external IP address (floating IP address), you must have public- and private-hosted zones of the same name, for example, `abc.foo.com`.
- To register the private IP address in a public-hosted zone, you must set `publish_vip_public_zone`, and choose that zone when defining the virtual service.

## Configuring the NSX Advanced Load Balancer Cloud Connector

This section discusses how to create a cloud configuration of type Amazon Web Services, so that the NSX Advanced Load Balancer spins up Service Engines in the required availability zones.

### Configuring the NSX Advanced Load Balancer Cloud Connector Using AWS Shared Key

This section covers the steps to configure the Cloud connector using AWS shared key.



The following are the steps to configure the cloud connector using AWS shared key:

## Prerequisites

## Procedure

- 1 Log in to the NSX Advanced Load Balancer UI and navigate to **Infrastructure > Clouds**, provide the desired name, and select the cloud type as **Amazon Web Services**. Click **Next**.
- 2 Use the drop-down menu to select **AWS Region**. Enable the **Access AWS through Proxy** if there is a custom proxy between your corporate network and AWS. Provide the details for Proxy Host and Proxy Port if required.
- 3 Access credentials are needed by the Controller to communicate with AWS API. Provide **Access Key ID**, and **Secret Key ID**. Click **Next**.
- 4 Configure SE Management Network. This is the subnet in which the Controller will place the management vNIC of SEs. The management network of SE should be reachable from the Controller management IP address.
  - a Select the **Availability Zone**, and **SE Management Network**.
  - b Select **Template Service Engine Group** from the drop-down menu from **Service Engine** section.
  - c Select the desired DNS Settings, and other settings as required. Click **Save** once all the desired options are selected.

**Edit Cloud: awscloud**

Infrastructure VPC/Network/Encryption

☒ Free Unused Elastic IP Address

Select SE Management Network for required Availability Zones

Availability Zone: us-west-1b

SE Management Network: 172.31.0.0/20 - 172.31.0.0/20

+ Add Availability Zone

☐ Enable Simple Queue Service (SQS) for Autoscale Groups Monitoring

• Service Engine •

Template Service Engine Group: Default-Group

• DNS Settings •

Register Virtual Service Names

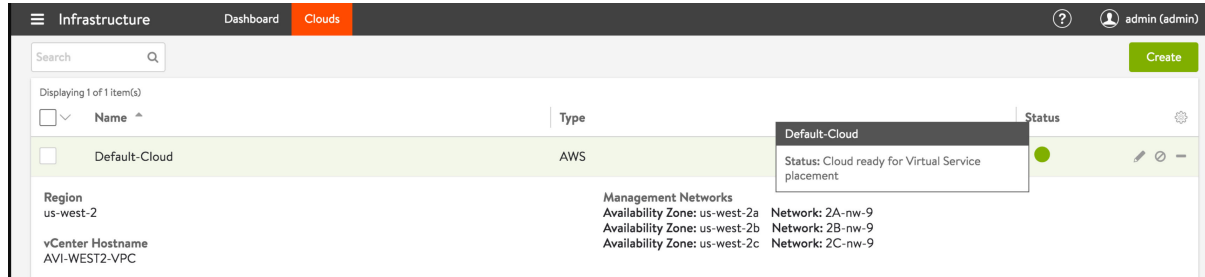
☒ None ☐ Amazon Route 53 ☐ DNS Profile

Cancel Save

## What to do next

- To verify the installation, navigate to **Infrastructure > Clouds**, click **Default-Cloud**, and then click the **Status** button.
- Status should turn green, indicating the installation was successful.

Figure 12-2.

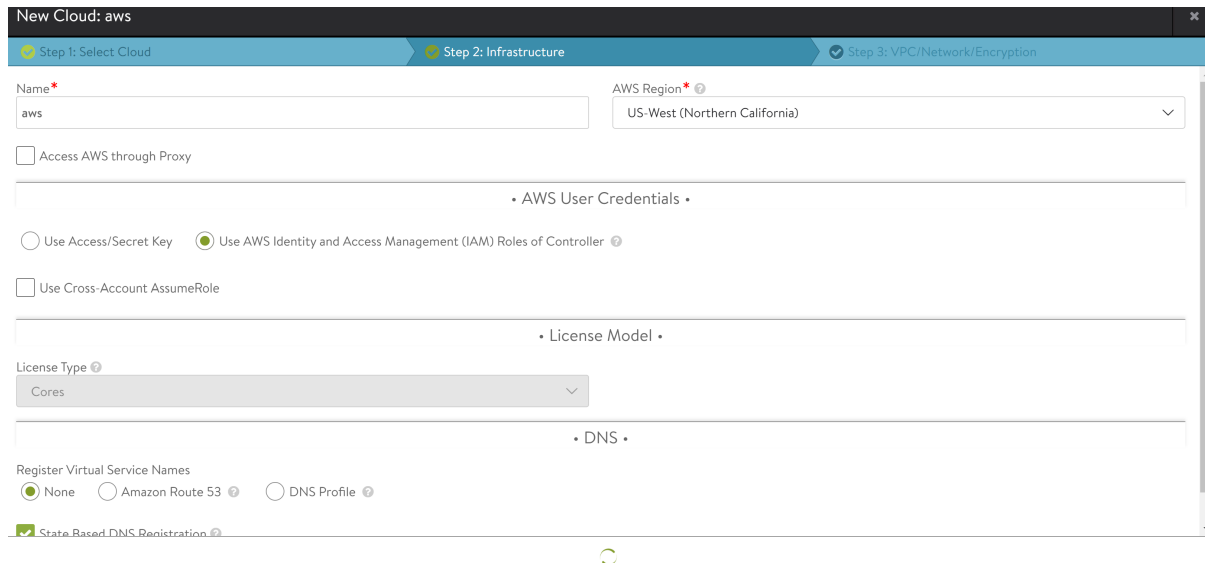


## Configuring the NSX Advanced Load Balancer Cloud Connector Using IAM Roles

This section covers the steps to configure the Cloud connector using IAM Roles.

### Procedure

- 1 Log in to the NSX Advanced Load Balancer UI and navigate to **Infrastructure > Clouds**, provide the desired name, and select the cloud type as **Amazon Web Services**. Click **Next**.
- 2 Use the drop-down menu to select **AWS Region**. Enable the **Access AWS through Proxy** if there is a custom proxy between your corporate network and AWS. Provided details for **Proxy Host** and **Proxy Port** if required.
- 3 Access credentials are needed by the Controller to communicate with AWS API. Select the option **Use AWS Identity and Access Management (IAM) Roles of Controller** to use IAM roles for the authentication. For detailed information on configuring IAM roles, see [Accounts and IAM Roles](#)



- 4 The remaining steps are similar to the steps mentioned in the previous section.

## Configuring the NSX Advanced Load Balancer Cloud Connector Use Cross-Account AssumeRole

This section discusses the steps to configure the Cloud Connector using User Cross-Account AssumeRole.

After completing the prerequisite setup, you can configure the IAM role for the Controller as IT-AviController-Role by following the steps mentioned in AWS Installation Guide. Skip the cloud creation steps and choose *No Orchestrator* during the setup. Ensure that the VPCs and subnets are configured in AWS, so that Controller management interface and Service Engine's management networks will be reachable from other accounts.

### Procedure

- 1 Create the AWS cloud by navigating to **Infrastructure > Clouds** and click on **Create**. Choose the appropriate region and select the checkbox for **Use AWS Identity and Access Management (IAM) roles**. This will ensure that the IT-AviController-Role is attached to the Controller when it is launched. Both IAM role and access/secret key can be used for cross-account role given the role/user has the necessary permissions (cross-account policy).

- 2 Select the checkbox for **Use Cross-Account AssumeRole**, if the cloud has been set up in another AWS account. However, in this case, the SE cloud is created in the Prod AWS account (112233445566) from the Controller hosted in IT AWS account (123456789012). As the cross-account AssumeRole has already been set up for IT-AviController-Role, on selecting the checkbox, the back-end APIs will fetch the associated AssumeRole accounts and their roles and display them in the drop-down menu. If there were no AssumeRoles attached, then the list would have been empty. There would be a text box that can be used to enter the ARN of the role for which the Controller instance's IAM role (in our case, IT-AviController-Role) can assume the role.
- 3 Select the ARN for the account and role, where the SE targets will be deployed.
- 4 If the role has appropriate access and is correctly setup, the Controller will fetch the AWS account details and configuration's VPC networks. Similarly, this will continue for the older SE AWS cloud setup.
  - Cloud setup will progress, and the SE AMI will be copied to the target account.
  - Once the transfer is completed, the cloud status will move to Cloud ready for Virtual Service placement.

- 5 Virtual services can now be configured on this cloud by following the steps mentioned at [Creating Virtual Service](#).

## Multi AZ Support for AWS

This section explains the Multi-AZ feature available for AWS.

The Multi-AZ feature addresses load balancing back-end servers of an application (virtual service) spread across multiple AZs in AWS with a VIP in each AZ. Without this feature, the user has to configure separate virtual services, one in each AZ. NSX Advanced Load Balancer centrally configures the Multi-AZ virtual service and combines analytics and logs across multiple VIPs of the same virtual service into a single consolidated view for the application.

## Service Discovery Using IPAM and DNS

This section covers the IPAM and DNS options available for NSX Advanced Load Balancer when integrated with AWS.

For more information on using IPAM and DNS on NSX Advanced Load Balancer, see [Service Discovery Using IPAM and DNS](#).

## IPAM Provider AWS

This section explains the AWS IPAM feature available while integrating NSX Advanced Load Balancer with AWS.

NSX Advanced Load Balancer integrates with Amazon Web Services (AWS) for providing IPAM services to applications running on instances in AWS.

---

### Note

- If the cloud type is AWS, there is support for IPAM, by default, without the need for a separate IPAM configuration.
- A separate IPAM configuration (as described below) is required only for cases where AWS provides the infrastructure service for other clouds, for instance, Mesos Cluster running on AWS instances.
- AWS IPAM is supported only for North-South IPAM Provider.

---

To use AWS as the IPAM provider, one of the following types of credentials is required:

- Identity and Access Management (IAM) roles
- AWS customer account key

### AWS IPAM Configuration Using IAM Role

If using the IAM role method to define access for NSX Advanced Load Balancer installation in AWS, use these steps to set up the IAM roles before beginning deployment of the Controller EC2 instance:

Select AWS IPAM as type and **Use IAM Roles** as shown below:

The screenshot shows a configuration window titled "New IPAM/DNS Profile: aws-northsouth-dns". It contains the following fields and options:

- Name:** A text input field containing "aws-northsouth-dns".
- Type:** A dropdown menu showing "AWS Route 53 DNS".
- AWS Profile Configuration:**
  - Two radio buttons: "Use IAM Roles" (selected) and "Use Access Keys".
  - Region:** A dropdown menu showing "US-West (Oregon)".
  - Two checkboxes: "Access AWS through Proxy" (unchecked) and "Use Cross-Account AssumeRole" (unchecked).
- Next:** A green button at the bottom right.

- 1 Click **Next** to configure VPCs.
- 2 A drop-down of available VPCs in that region is displayed.
- 3 Select the appropriate VPC.
- 4 A drop-down of availability zones (AZ) in that region and a corresponding list of networks in each AZ are displayed. For multi-AZ virtual service applications, configure at least one network from each AZ for IPAM.
- 5 Click **Save**.

New IPAM/DNS Profile: aws-northsouth-ipam

Name\* ?

aws-northsouth-ipam

Type ?

AWS IPAM

AWS Profile Configuration

VPC\* ?

AVI-WEST2-VPC - 10.144.0.0/16

Availability Zone ?

us-west-2a

✕

▼

Usable Network ?

2A-nw-1

✕

▼

🗑️

Availability Zone ?

us-west-2b

✕

▼

Usable Network ?

2B-nw-1

✕

▼

🗑️

Availability Zone ?

us-west-2c

✕

▼

Usable Network ?

2C-nw-1

✕

▼

🗑️

Add Usable Network

Previous


Save


### AWS IPAM Configuration Using Access Key

To configure AWS IPAM using Access Key, select AWS IPAM as the **Type** , then select **Use Access Keys** and enter the following information:

- Access Key ID: AWS customer key ID
- Secret Access Key: customer key
- Select the AWS region into which the VIPs will be deployed
- Select **Access AWS through Proxy**, if access to AWS endpoints requires a proxy server

**New IPAM/DNS Profile: aws-northsouth-ipam**

Name\*  aws-northsouth-ipam

Type  AWS IPAM


---

**AWS Profile Configuration**


☐ Use IAM Roles ☒ Use Access Keys

Access Key ID\* Secret Access Key\*

..... .....

Region\* US-West (Oregon) 

☐ Access AWS through Proxy

☐ Use Cross-Account AssumeRole 

**Next**

- 1 Select **Use Cross-Account AssumeRole**, if the AWS credentials or role is being leveraged to access across accounts. Click **Next** to configure VPCs.
- 2 A drop-down of available VPCs in that region is displayed.
- 3 Select the appropriate VPC.
- 4 A dropdown of availability zones (AZ) in that region and a corresponding list of networks in each AZ are displayed. For multi-AZ virtual service applications, configure at least one network from each AZ for IPAM.
- 5 Click **Save**.

New IPAM/DNS Profile: aws-northsouth-ipam

Name\* ?

aws-northsouth-ipam

Type ?

AWS IPAM

AWS Profile Configuration

VPC\* ?

AVI-WEST2-VPC - 10.144.0.0/16

|                                                                                |                                                                                       |
|--------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <div>Availability Zone ?</div> <div>us-west-2a</div> <div>✕</div> <div>▼</div> | <div>Usable Network ?</div> <div>2A-nw-1</div> <div>✕</div> <div>▼</div> <div>🗑</div> |
| <div>Availability Zone ?</div> <div>us-west-2b</div> <div>✕</div> <div>▼</div> | <div>Usable Network ?</div> <div>2B-nw-1</div> <div>✕</div> <div>▼</div> <div>🗑</div> |
| <div>Availability Zone ?</div> <div>us-west-2c</div> <div>✕</div> <div>▼</div> | <div>Usable Network ?</div> <div>2C-nw-1</div> <div>✕</div> <div>▼</div> <div>🗑</div> |

Add Usable Network

Previous

Save

## DNS Provider AWS

This section discusses about the DNS services provided by the NSX Advanced Load Balancer.

NSX Advanced Load Balancer integrates with Amazon Web Services (AWS) to provide DNS services to applications running on instances in AWS. AWS Cloud in NSX Advanced Load Balancer supports AWS DNS by enabling `route53_integration` in the cloud configuration and doesn't require this DNS profile configuration. A separate DNS provider configuration (as described below) is required only for cases where AWS provides the infrastructure service for other clouds (e.g., Mesos Cluster running on AWS instances). AWS DNS is supported only for North-South DNS Provider.

## Configuring Security Groups

Follow the steps mentioned in this section to create security groups using SE group level configuration.



It is recommended to create a custom security group at the SE group level and disable the default security group creation. `disable_avi_sg_creation` is the flag to disable the default security group creation by NSX Advanced Load Balancer .

### Note

- Once the option to create the default security group is disabled, NSX Advanced Load Balancer does not create any new security group.
- By default, rules for management interface, data interface, and tunnelling protocols are not added to the custom security groups. These rules are created manually. This is equivalent to setting the value for the `ingress_access_data` option and `ingress_access_mgmt` option to None.
- If the `disable_avi_sg_creation` option is set on an existing cloud, it applies only to the newly created Service Engines and virtual services. The existing security groups are not deleted automatically.

```
admin@controller:~$ shell
Login: admin
Password:
[admin:controller]: > configure serviceenginegroup Default-Group
[admin:controller]: serviceenginegroup> disable_avi_sg_creation
```

## Recommended Security Group Rules for AWS Deployment

The following are the recommended rules to be configured when using an user-created security group or a custom security group on AWS.

### Management Rules

The rules mentioned below is required for NSX Advanced Load Balancer Controller to SE communication (management interface traffic).

| Type        | Protocol | Port Range | Source                                                                                                                                                                                          |
|-------------|----------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SSH         | TCP      | 22         | 0.0.0.0/0 is the default value. This indicates SSH is enabled from anywhere. This value is configured as per requirement to restrict SSH access from a specific network, subnet, or IP address. |
| ICMP - IPv4 | ICMP     | N/A        | Same as above                                                                                                                                                                                   |

### Data Rules

Data rules include ports to which any virtual service (VIP/FIP) is listening. The table below exhibits an example for HTTP communication on port 80:

| Type        | Protocol | Port Range | Source                                                                                                                                                                                  |
|-------------|----------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTP        | TCP      | 80         | 0.0.0.0/0 is the default value. This indicates SSH is enabled from anywhere. This value is configured as per requirement to restrict SSH from a specific network/subnetwork/IP address. |
| ICMP - IPv4 | ICMP     | N/A        | Same as above                                                                                                                                                                           |

## Tunneling Protocols

The following table exhibits custom ports required for communication between NSX Advanced Load Balancer and AWS.

| Type              | Protocol | Port Range | Source   |
|-------------------|----------|------------|----------|
| Custom Protocol   | 73       | all        | VPC CIDR |
| Customer Protocol | 97       | all        | VPC CIDR |

It is recommended to create the AWS tags and security groups at the time of SE creation (when virtual services are deployed to the SE Group). If you have updated these settings, you can delete the SEs and they will be automatically re-created with the new settings.

## Additional Deployment Options

This section gives information on the additional deployment options supported by the NSX Advanced Load Balancer.

### Multi AZ Support for AWS

This section explains multiple availability zones support for AWS.

The Multi-AZ feature addresses load balancing back-end servers of an application (virtual service) spread across multiple AZs in AWS with a VIP in each AZ. Without this feature, the user has to configure separate virtual services, one in each AZ. NSX Advanced Load Balancer centrally configures the Multi-AZ VS, and combines analytics and logs across multiple VIPs of the same VS into a single consolidated view for the application.

Refer to the below diagram. The application's virtual service shares a common pool of nine servers (SRV1 - SRV9) in a single pool spanning the three availability zones (AZ-1, AZ-2, AZ-3). The servers are accessed via 3 VIPs (VIP1, VIP2, VIP3), one per AZ.

In a Multi-AZ deployment, it is not required to have Active/Active HA for VIPs within each AZ. The Multi-AZ deployment automatically provides the HA by having SE instances across AZs. Also, the NSX Advanced Load Balancer supports non-disruptive upgrades (SE in each AZ is upgraded one at a time, keeping the VS always up during the upgrade). Hence the recommended HA mode is N+M with a buffer of 0.

## DNS

For the Multi-AZ feature, the virtual service must be configured with an FQDN. NSX Advanced Load Balancer integrates with DNS as well as Route 53 in AWS. One or the other is a pre-requisite for the Multi-AZ feature. In both cases, all the VIPs for the VS are automatically populated in the DNS. It is recommended to configure DNS with a consistent hashing algorithm to minimize cross-AZ traffic (AWS charges extra for inter-AZ traffic).

## Server Selection

Any [load balancing algorithms](#) of NSX Advanced Load Balancer are available for server selection. In the current release, it does not automatically localize traffic on a VIP to the pool servers in the same AZ. This enhancement will be added in a future release.

## Operational Status

Each individual VIP generates an event when it is UP/DOWN; this information can be used to determine the health of the VIP. Starting with 17.1.2, when a VIP is down, NSX Advanced Load Balancer automatically withdraws that VIP from DNS (be it DNS or Route 53). When the VIP is up again, the DNS is updated automatically as well.

| VS oper_status | VIP oper_status           |
|----------------|---------------------------|
| down           | if no VIP is UP           |
| up             | if any of the VIPs are UP |

In the virtual service list shown below, the IP addresses of the member VIPs are revealed.

Virtual service health on a per-VIP basis can be viewed in the virtual service's submenu, as shown below.

Clicking the **Scale Out**, **Scale In**, or **Migrate** button appearing in the above display results in windows with drop-down menus that enable selection of a particular VIP to be scaled or migrated.

## Creating a Multi-VIP VS in the UI

The user can use the Multi-AZ feature by specifying more than one VIP in the list within the VIP Address section of the Settings tab of the VS editor. Two VIPs are selected in the example below by specifying two networks from which they will be auto-allocated. In addition, its FQDN needs to be registered with Route 53.

## DPDK support on AWS native AMI SE

This section explains the DPDK support for AWS.

Starting with NSX Advanced Load Balancer release 20.1.1, DPDK support on AWS native AMI SE is available at the SE level.

The following options are available under the SE Group boot-up properties for the DPDK support:

- `se_dpdk_pmd`: This configuration knob enables the DPDK mode when the value of the `se_dpdk_pmd` parameter is set to 1.
- `max_queues_per_vnic`: The `max_queues_per_vnic` parameter in SE-group properties allows configuring the maximum number of queues per dispatcher.

---

**Note** DPDK is disabled by default.

DPDK is supported on AWS instances that support the Elastic Network Adapter (ENA). For the detailed information on the `max_queues_per_vnic` option, see [Multiple Queues per Dispatcher](#).

---

# Installing NSX Advanced Load Balancer in Nutanix Acropolis Based Environments

# 13

NSX Advanced Load Balancer supports running in the Nutanix Acropolis hypervisor orchestrated by Prism. This section discusses the steps required to install the NSX Advanced Load Balancer in this environment.

This chapter includes the following topics:

- [Considerations for Deploying NSX Advanced Load Balancer in Nutanix Environments](#)
- [Installing the NSX Advanced Load Balancer in Nutanix Acropolis](#)

## Considerations for Deploying NSX Advanced Load Balancer in Nutanix Environments

This section provides information on the hardware and software requirements for deploying the NSX Advanced Load Balancer in Nutanix environments.

### Hardware Requirements for installing the NSX Advanced Load Balancer in Nutanix Acropolis

The following are the requirements for installing the Controller and SEs for deployment:

- For the Controller:
  - Atleast 8 vCPUs and 24 GB memory
  - A NIC for the management network. This is the network used for accessing the web interface. The Controller also uses this network to communicate with the SEs.
- For the SEs:
  - Atleast 2 vCPUs and 2 GB memory
  - A NIC for the management network as the first interface, and additional NICs for the data networks

For more information on the minimum recommendations for deployment of the NSX Advanced Load Balancer, see [System Limits](#).

## Software Requirements for installing the NSX Advanced Load Balancer in Nutanix Acropolis

The following versions are tested and qualified for the NSX Advanced Load Balancer:

- Hypervisor: Nutanix AHV (Version 2018.05.01 - Community License)
- AOS version: ce-2018.05.01-stable
- NCC Version: 3.5.1
- LCM Version: 1.4.f2bd889

## Installing the NSX Advanced Load Balancer in Nutanix Acropolis

To install the NSX Advanced Load Balancer in Nutanix Acropolis based environments:

- 1 Deploy the Controller
- 2 Deploy the Service Engine

### Deploying the Controller through Prism

Follow the steps mentioned below for deploying the Controller through the Prism Image Service:

**Procedure**

- 1 Upload the NSX Advanced Load Balancer Controller image to the Prism image service:
  - a Click the **Settings** button and go to **Image Configuration** to upload the files.
  - b In the **Create Image** screen, select the **IMAGE TYPE** as Disk and select the **CONTAINER**.

**Create Image** ? X

**NAME**  
Avl\_Controller

**ANNOTATION**

**IMAGE TYPE**  
DISK

**CONTAINER**  
default-container-42076

**IMAGE SOURCE**

☐ From URL

☒ Upload a file **Browse...** controller-16.2.1-9013.qcow2

Cancel Save

- c Create a virtual machine (VM) instance for the Controller from Prism with the following details:
    - NAME: Desired name

- vCPUs: 1
- Number of Cores per vCPU: 8 (or more)
- Memory: 24 GB
- Disks:
  - Remove CD-ROM
  - Add new Disk
  - Operation: Clone from Image Service
  - Image: Select the name of the image created in the previous steps
  - Size: Default to 64 GB
- NIC: vLAN. Select a network that will be used for management. This network is used for accessing the Controller and for Controller-to-Service Engine communication. For production environments this is a static address.

---

**Note** For configuring networks and IP addresses for the Controller and SEs, usually networks should be created beforehand. For more details, follow the instructions in [Nutanix Acropolis 101: Creating VMs](#).

---

Select that network when adding a vNIC to a Controller or SE. Based on how the network was originally set up, the IP address will be static or picked up via DHCP and assigned to the vNIC.

- 2 Power on the Controller VM. During the initial boot-up, the Controller initializes databases and runs through internal setup tasks (may take several minutes).
- 3 Connect to the NSX Advanced Load Balancer UI via `https://[IP Address]`.
- 4 Set up the the Controller. From the NSX Advanced Load Balancer UI setup wizard, perform the initial configuration of the Controller. Provide a password, DNS, and set the **Infrastructure Type** to **No-Orchestrator**.

## Deploying the Service Engine

Follow the steps given below to deploy the service engine:

### Procedure

- 1 From the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Clouds**.
- 2 Click the **download** icon shown on the right.
- 3 Select **qcow2** as the **Type of SE Image**. The Controller creates the new SE image. This may take several minutes, depending on disk performance. Once the image is created it will automatically be downloaded to your browser.



- 4 Navigate to **Infrastructure > Clouds** and click the key icon to generate a new token. Copy the token, which will be used at the end of the next step.



**Note** The new SE will be required to authenticate to the Controller by presenting a valid token that is generated by the Controller and placed on the SE. This token may be used only once, and will expire if not used within one hour.

- 5 Create the SE VM from Prism.

- vCPUs: 1
- Cores per vCPU: 2
- Memory: 2 GB
- Disks:
  - Remove CD-ROM
  - Add Disk: Operation: Clone from Image
  - Service Image: Select the SE image name
  - Size: 10 GB
- Network Adapters:
  - Create one NIC in the management network used by the Controller. This NIC is reserved for management.
  - Create additional NICs as necessary for access to client and server networks for load balancing traffic.

**Note** While the management network may be used for load-balanced traffic, it will require a second NIC configured for that network.

- Custom Scripts: Paste the following information into this section. This info is used to tell the SE about the Controller IP address and the auth token it should present. The auth token is already generated.

```
AVICNTRL: [Controller IP address] AVICNTRL_AUTHTOKEN: [Auth token]
```

**Note** In NSX Advanced Load Balancer, no-access cloud is supported for Nutanix Acropolis.

Provide value for the following attributes in addition to the Controller IP address and the authentication token:

- SE default gateway
- SE management IP address
- Subnet mask for SE's management IP address

```
AVICNTRL: [Controller IP address] AVICNTRL_AUTHTOKEN: [Auth token] avi.default-gw.SE: [SE
Default Gateway] avi.mgmt-ip.SE: [SE Management IP] avi.mgmt-mask.SE: [SE Management IP
mask]
```

- 6 Power on the SE VM.
- 7 To verify connectivity between the SE and Controller, navigate to **Infrastructure > Dashboard** and check the SE icon.
- 8 Repeat as necessary to create any additional SEs.

## Deploying VRF on Nutanix

When the NSX Advanced Load Balancer runs on Nutanix, you can create new VRFs, if required, and move interfaces to it.

For more information, see [VRF support for vCenter Deployments](#).

# Installing NSX Advanced Load Balancer in VMware vSphere Environments

# 14

This section explains how to integrate NSX Advanced Load Balancer into a VMware vCenter cloud.

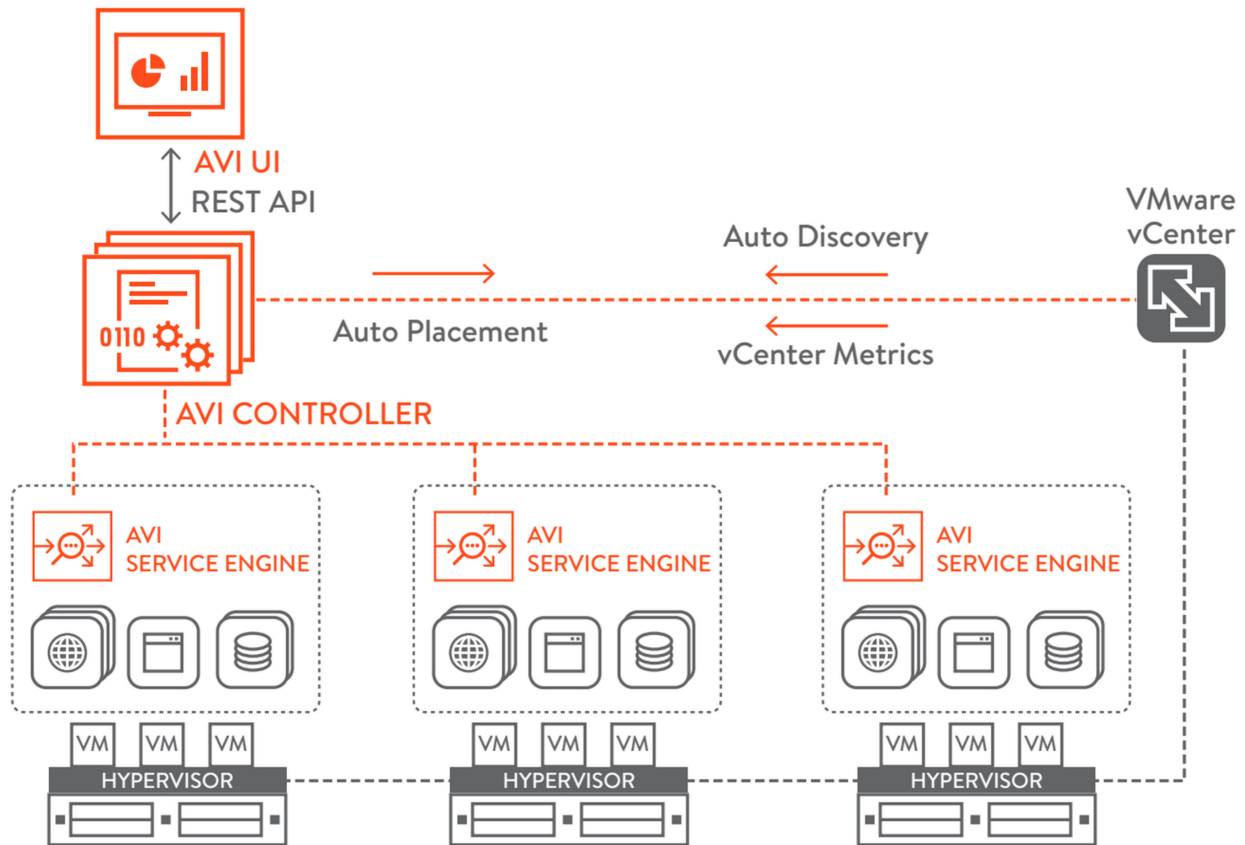
This chapter includes the following topics:

- [Overview](#)
- [Installation Considerations](#)
- [Deploying NSX Advanced Load Balancer Controller in VMware vCenter](#)
- [Configuring NSX Advanced Load Balancer Cloud Connector](#)
- [Additional Configuration Options](#)

## Overview

NSX Advanced Load Balancer is a software-based solution that provides real-time analytics and elastic application delivery services. It optimizes core web functions, including SSL termination and load balancing.

NSX Advanced Load Balancer runs on virtual machines (VMs) managed by VMware vCenter. When deployed into a vCenter-managed VMware cloud, it performs as a fully distributed, virtualized system consisting of a Controller and SEs each running as a VM.



The NSX Advanced Load Balancer platform is built on software-defined architectural principles that separate the data plane and control plane. The product components include:

**NSX Advanced Load Balancer Controller (Control Pane)** - The NSX Advanced Load Balancer Controller stores and manages all policies related to services and management. Through vCenter, the NSX Advanced Load Balancer Controller discovers VMs, data centers, networks, and hosts. Based on this auto-discovered information, virtual services can quickly be added using the web interface. To deploy a virtual service, the NSX Advanced Load Balancer Controller automatically selects an ESX server, spins up an SE (described below), and connects it to the correct networks (port groups).

**Note** Controllers need access to the desired ESXi hosts (over port 443) to allow the Controller-to-vCenter communication.

The NSX Advanced Load Balancer Controller can be deployed as a single VM or as a high availability cluster of 3 NSX Advanced Load Balancer Controller instances, each running on a separate VM. A single NSX Advanced Load Balancer Controller cluster supports multiple concurrent vCenter clouds.

**NSX Advanced Load Balancer SE (Data Plane)** - Each NSX Advanced Load Balancer SE runs on its virtual machine. It provides the application delivery services to end-user traffic, and also collect real-time end-to-end metrics for traffic between end-users and applications.

## Installation Considerations

This section explains the installation consideration for NSX Advanced Load Balancer with VMware vSphere.

### Points to Consider

- NSX Advanced Load Balancer can be deployed with a VMware cloud in either no access, read access, or write access mode. Each mode is associated with different functionality and automation, and also requires different levels of privileges for the Controller within VMware vCenter.

### vCenter Account Requirements

During the initial Controller setup, a vCenter account must be entered to allow communication between the Controller and the vCenter. The vCenter account must have the privileges to create new folders in the vCenter. This is required for Service Engine creation, which then allows virtual service placement.

### VMware User Role for NSX Advanced Load Balancer

NSX Advanced Load Balancer manages the lifecycle of the load balancer within each cloud. In VMware write access cloud, the Controller requires vCenter URL, username, and password to establish a connection with the vCenter portal. With this, the Controller discovers the vCenter managed objects to build an internal relation graph. As a part of the load balancer lifecycle management, SE is created and port groups are added and (or) removed from the virtual machines.

On deploying vCenter cloud, NSX Advanced Load Balancer is not provided the root credentials for security reasons. On creating the cloud in NSX Advanced Load Balancer, the vCenter user is assigned certain roles that allow the Controller to manage the load balancer lifecycle. The user is mapped to two roles during the role configuration on vCenter. One of the roles is applied at the vCenter root level and another at the folder level where the Service Engines are created by the Controller.

For detailed information on creating the required roles and the permissions, see [vSphere Permissions and User Management Tasks](#).

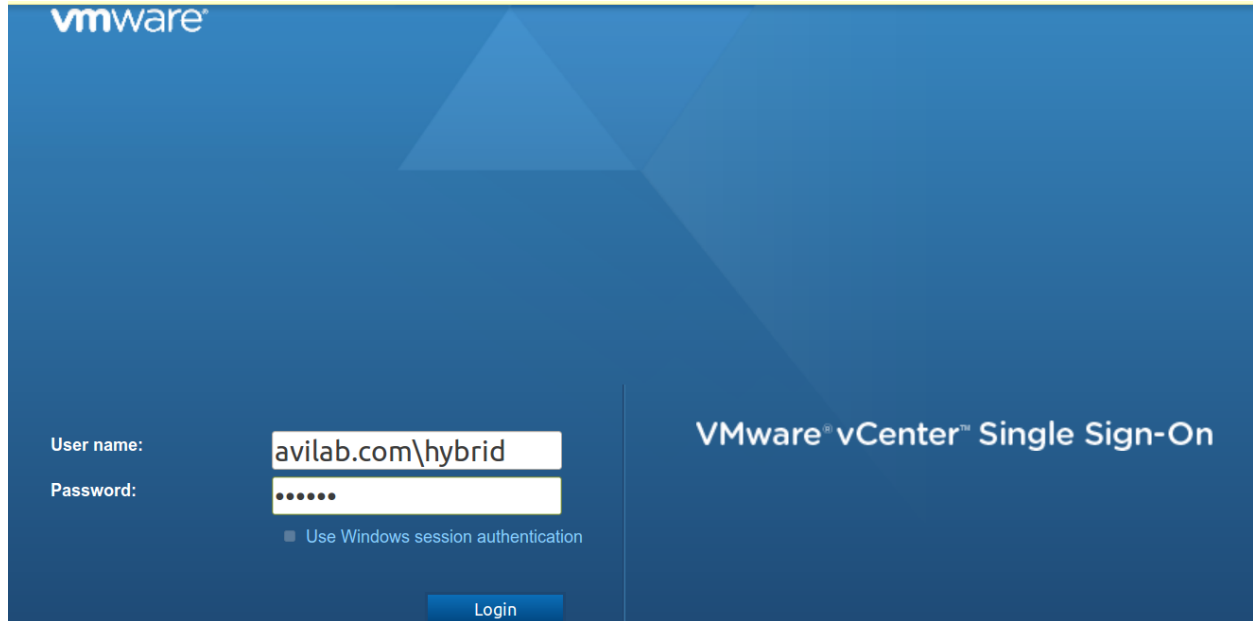
The following section discusses defining role privileges for two roles AviRole1 and AviRole2 that are assigned to the vCenter user.

### Configuring Role Settings

This topics explains how to configure role settings.

In the example below, avilab.com is part of the LDAP, and avilab.com\hybrid is the user. Login to VMware vCenter as the hybrid user.

Figure 14-1. For 6 - 6.5 versions



Login to VMware vCenter as follows:

Figure 14-2. For 6.7 and 7 versions



### Root Folder Level Role

This section explains how to configure a root folder level role.

AviRole1 is the role applied at the root folder level, which allows the assigned user to:

- Deploy SE in a data center.

- Create virtual NIC for the SE.
- Discover all available networks with Read Only access.
- Discover the best possible host to deploy the Service Engine in Read Only mode.

To configure the role settings:

- 1 Navigate to **AdministrationRoles**.
- 2 Locate the Avi role name - **AviRole1**.

Apply this role to the root level of the vCenter object hierarchy for giving the NSX Advanced Load Balancer Controller access to discover vCenter resources.

Figure 14-3. For 6 - 6.5 versions

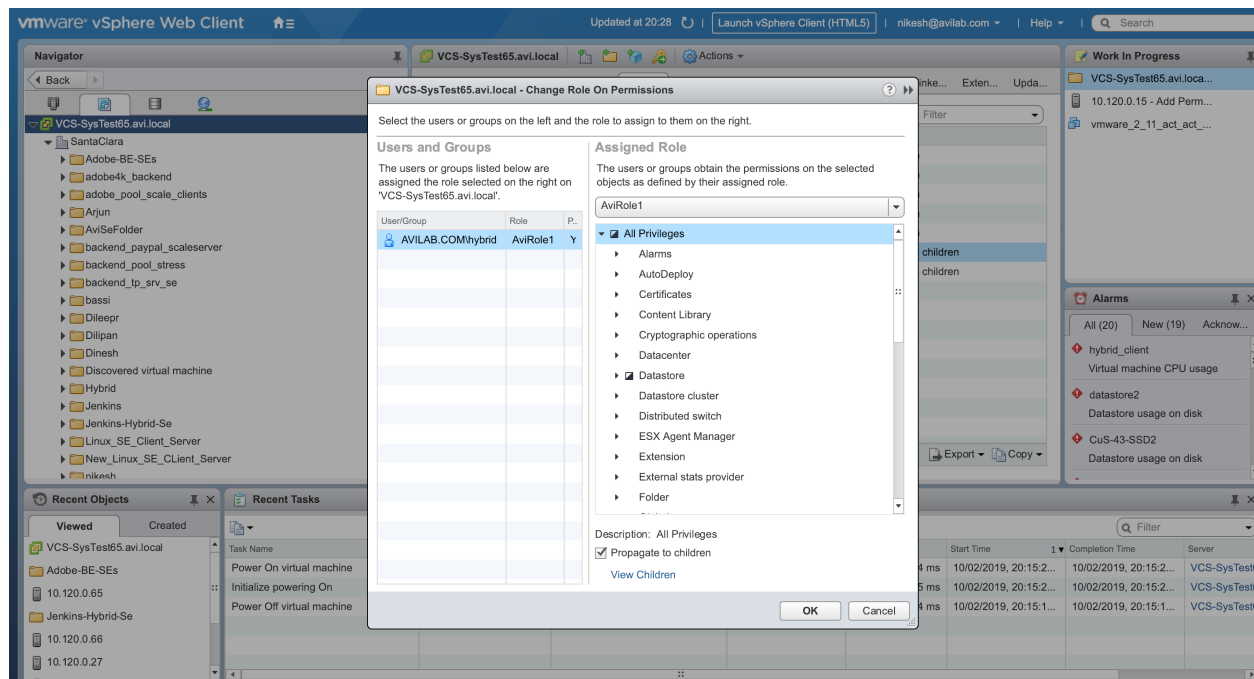
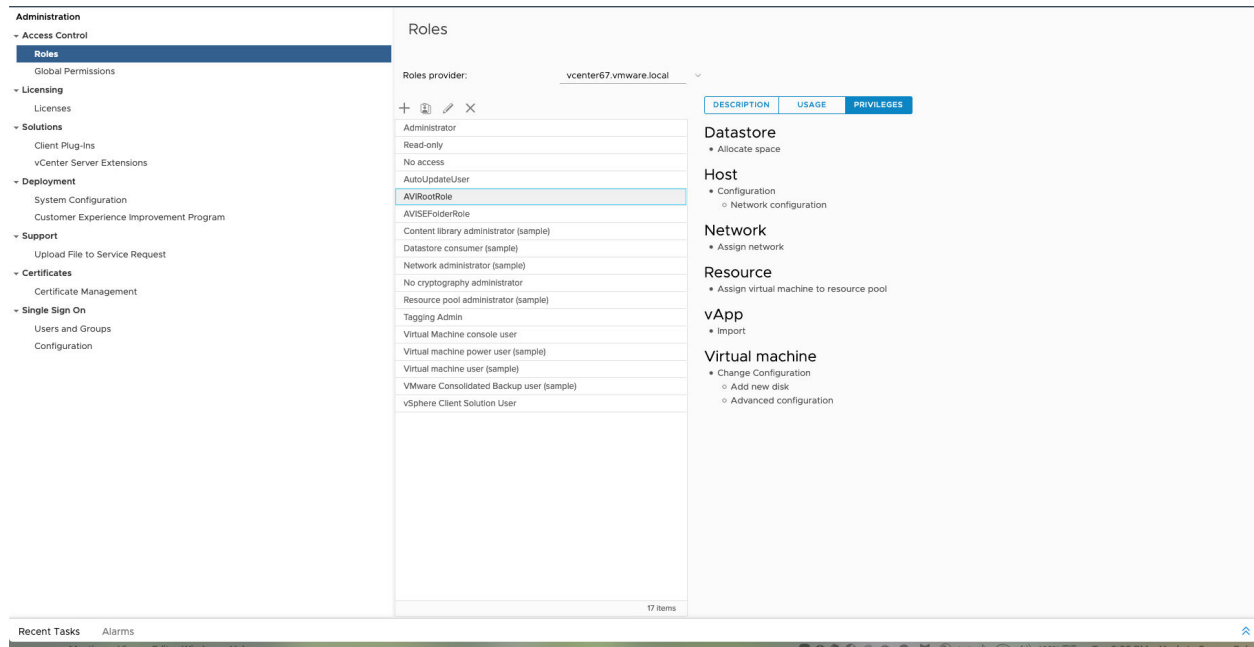


Figure 14-4. For 6.7 and 7 versions



Under **PrivilegeAll Privileges**, define the following parameters for this role:

- Datastore settings
- Network configuration
- Resource
- Virtual machine configuration
- vApp import



## Procedure

- 1 Navigate to **Datastore**. Expand the list and click the checkbox for **Allocate space**.

Figure 14-5. For 6 - 6.5 versions

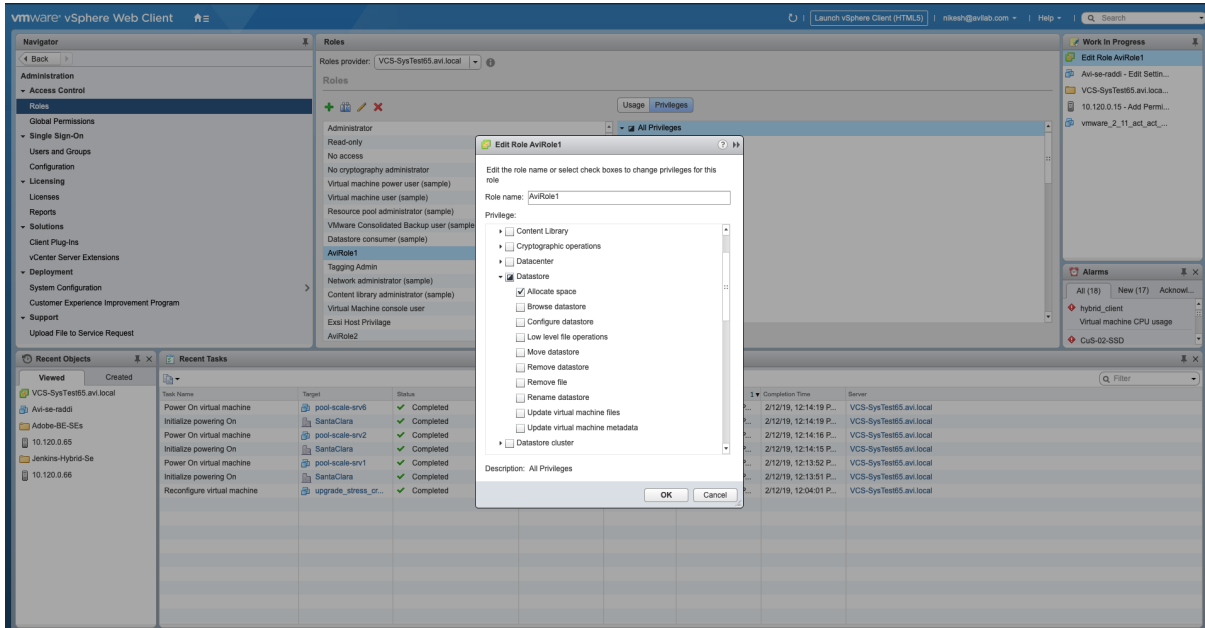
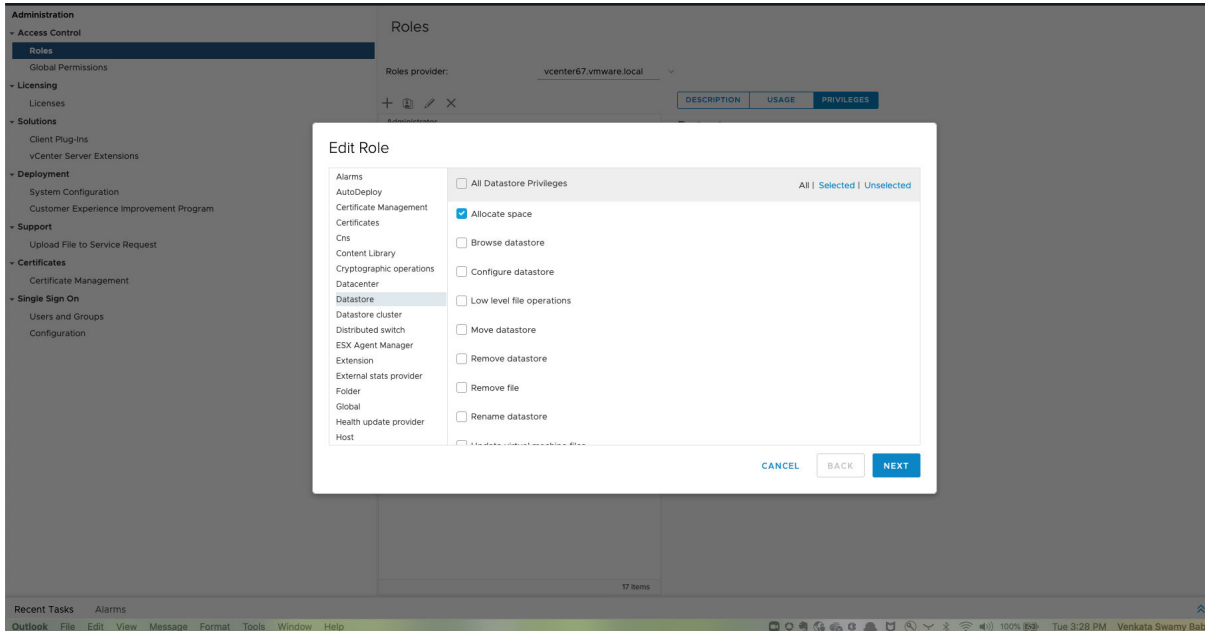


Figure 14-6. For 6.7 and 7 versions



- 2 Navigate to **Host** and select **Configuration**. Expand the list and click the checkbox for **Network configuration**.

Figure 14-7. For 6 - 6.5 versions

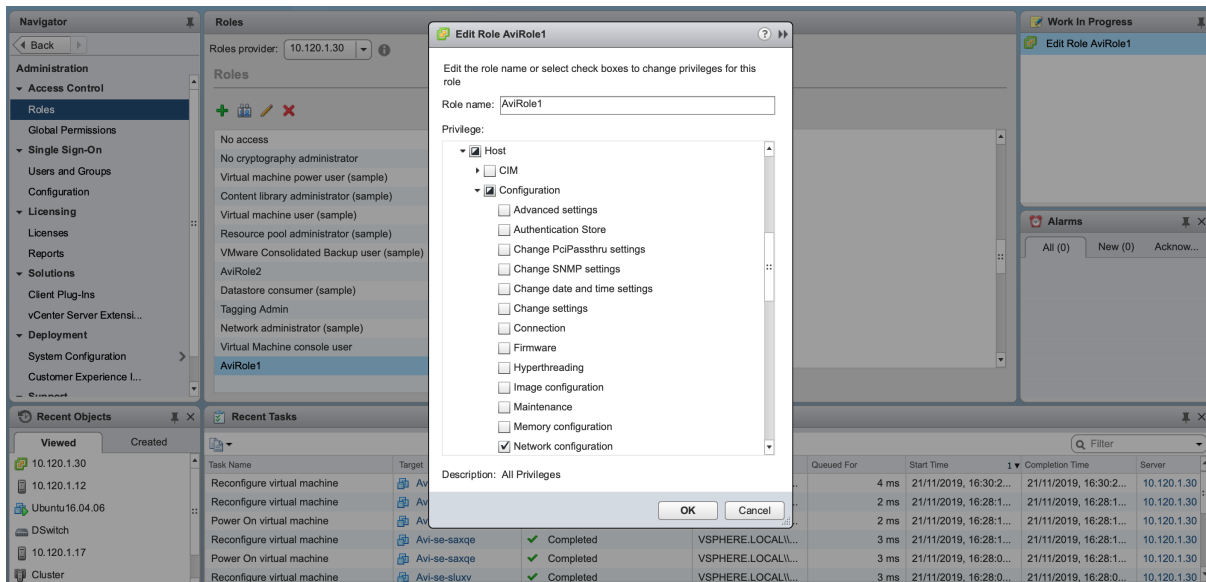
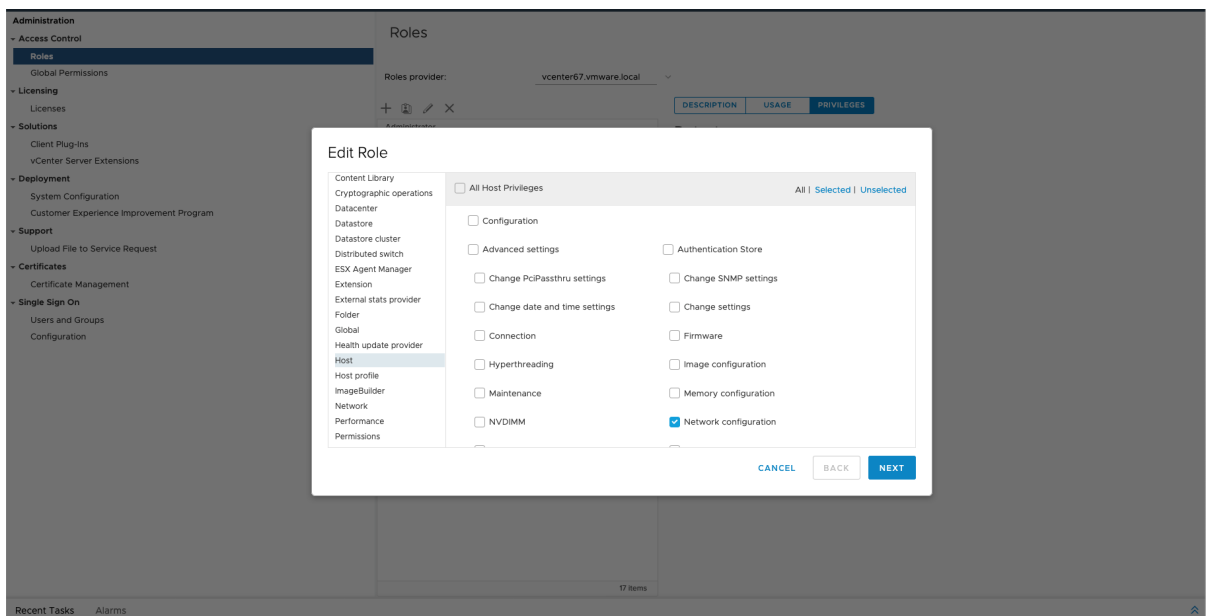


Figure 14-8. For 6.7 and 7 versions



### 3 Navigate to **Network** and select **Assign network**.

Figure 14-9. For 6 - 6.5 versions

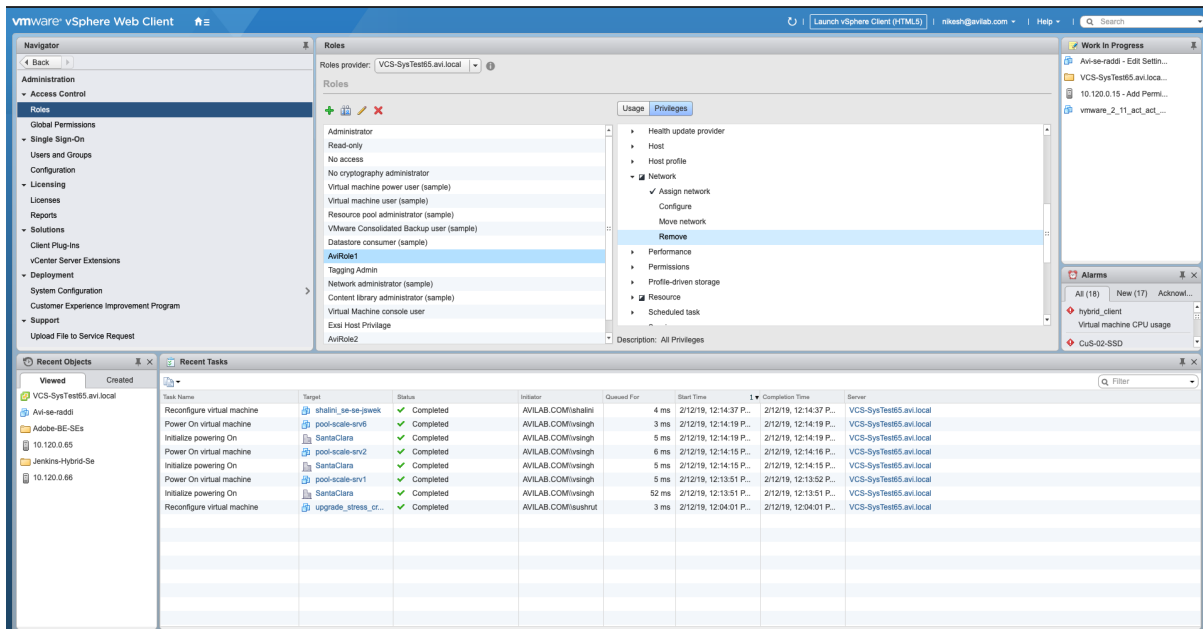
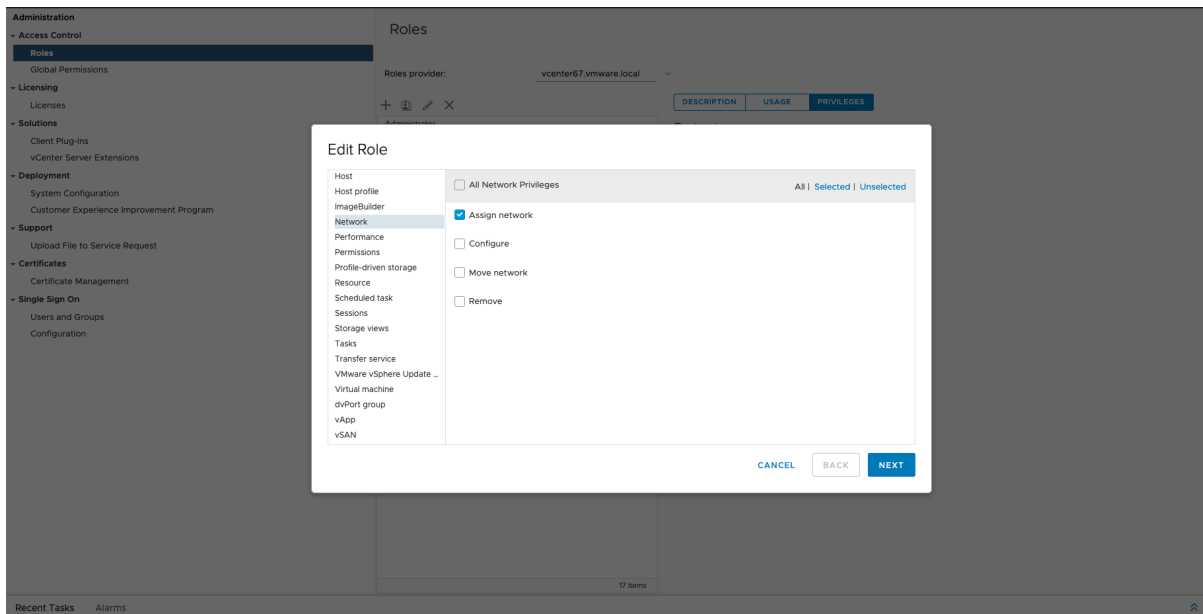


Figure 14-10. For 6.7 and 7 versions



#### 4 Navigate to **Resource** and select **Assign virtual machine to resource pool**.

Figure 14-11. For 6 - 6.5 versions

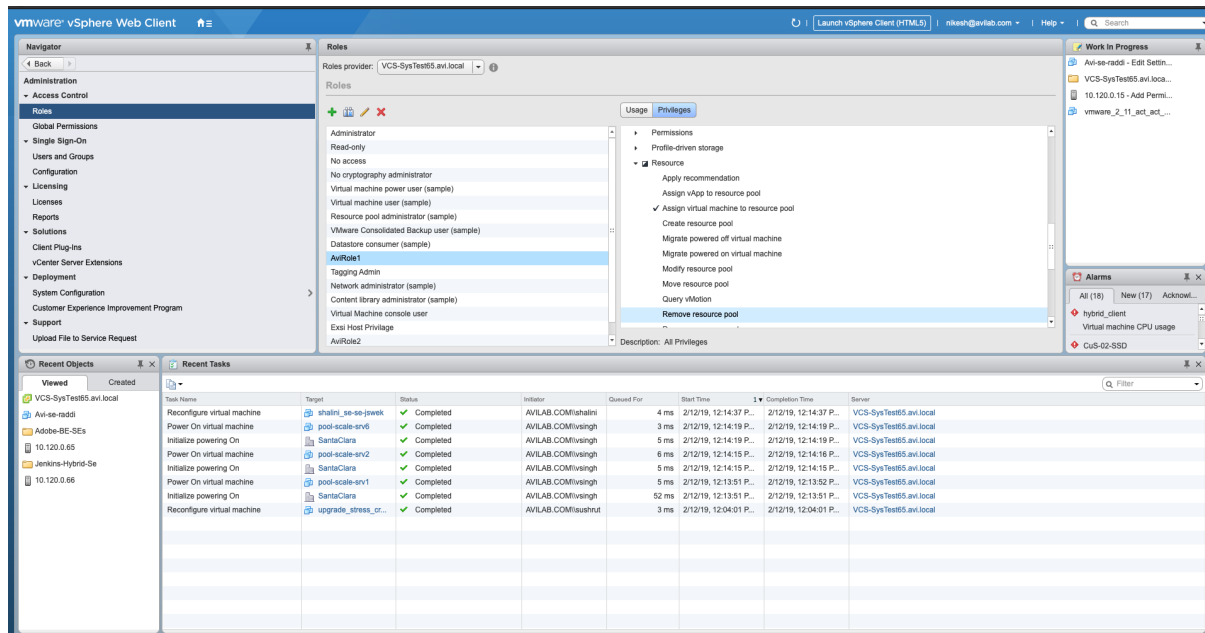
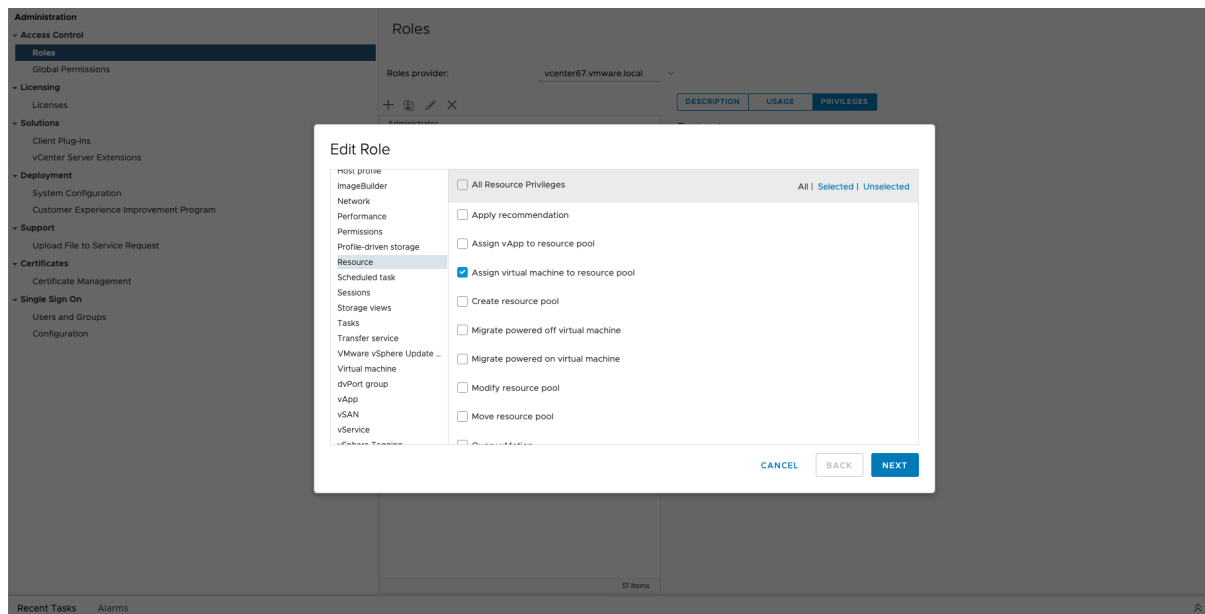


Figure 14-12. For 6 - 6.5 versions



#### 5 Navigate to **Virtual machine > Configuration** and select the following options:

- Add new disk
- Advanced

Figure 14-13. For 6 - 6.5 versions

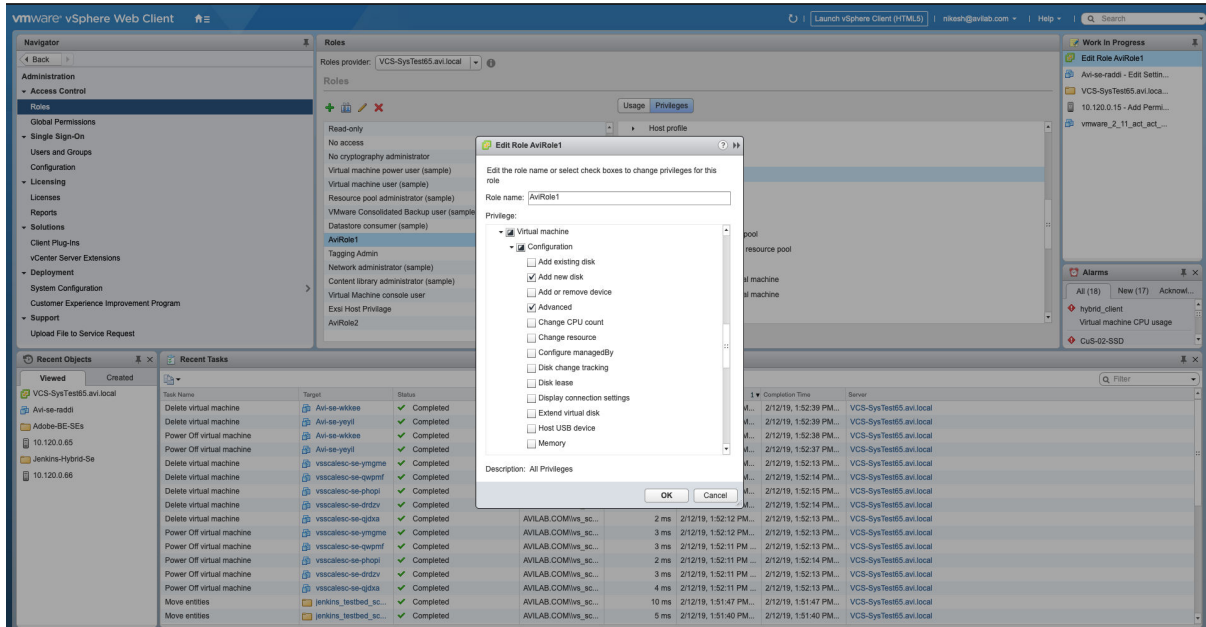
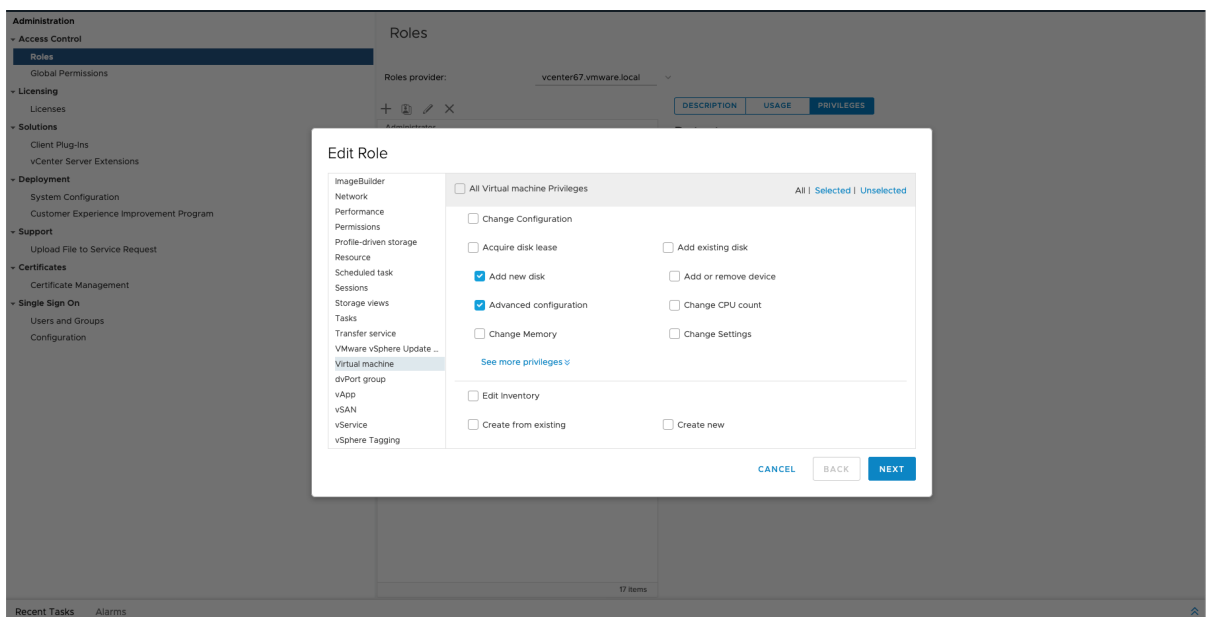


Figure 14-14. For 6.7 and 7 versions



## 6 Navigate to **vApp** and select **Import**.

Figure 14-15. For 6 - 6.5 versions

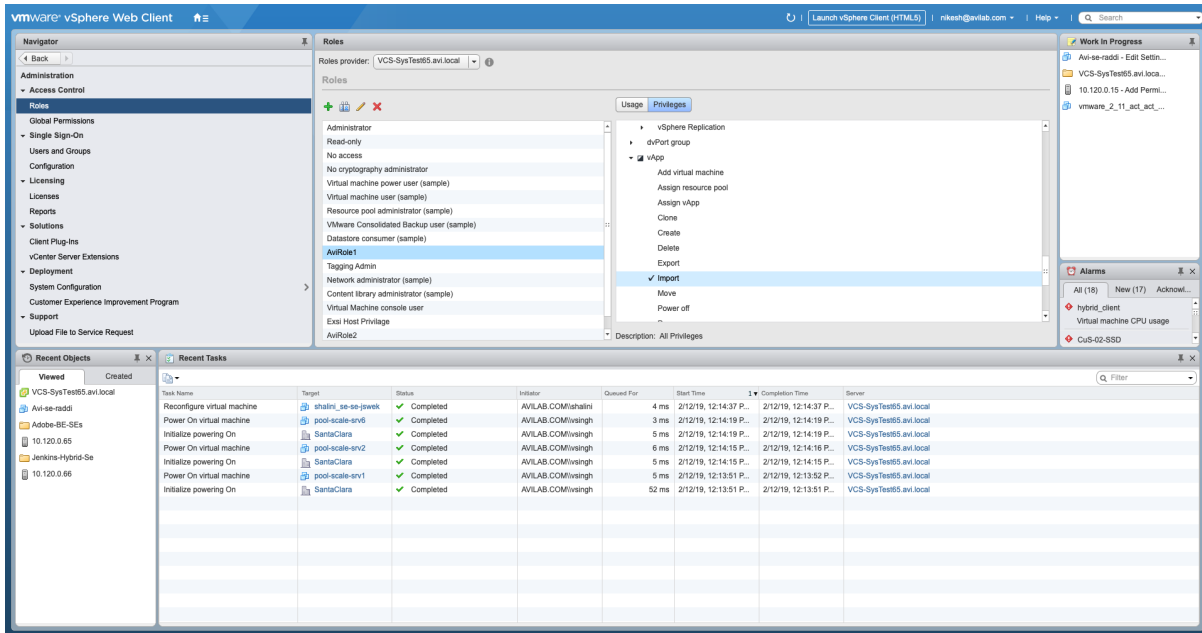
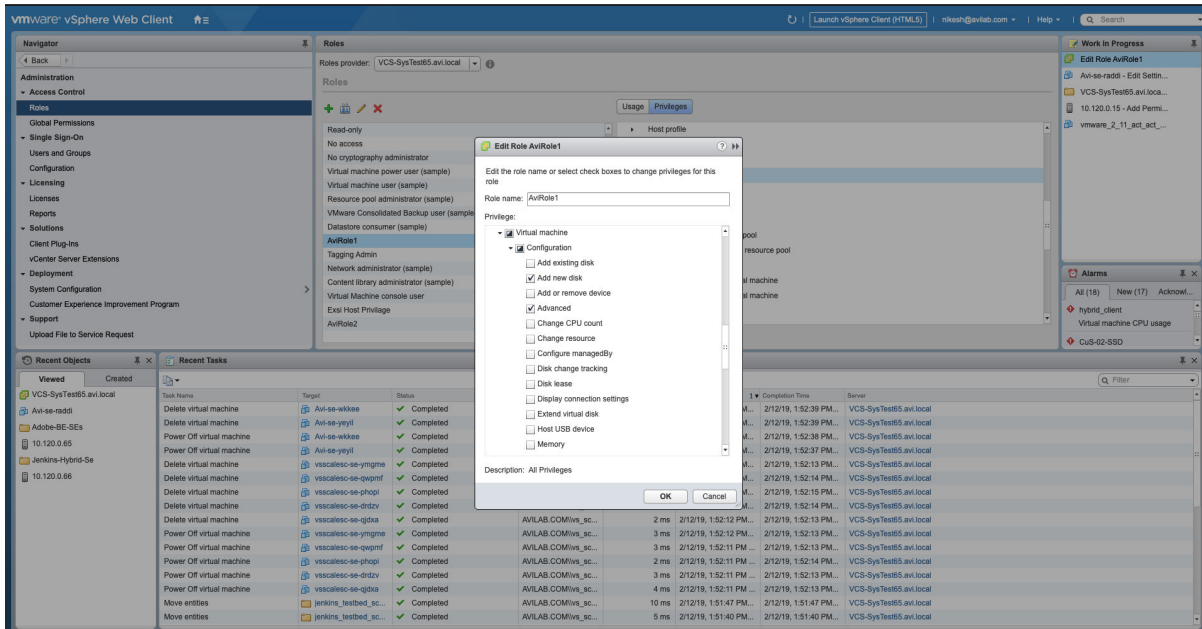


Figure 14-16. For 6.7 and 7 versions



### SE Creation Folder Level Role

AviRole2 role is applied at the folder level where the SEs are created by the NSX Advanced Load Balancer Controller.

With this role, the user is restricted to perform all operations on a Service Engine within a particular folder and is not allowed to edit any resources outside the specific folder.

This role is required for the user to access the datastore, host, and networking settings that allow creating the Service Engine.

Under **Privilege > All Privileges**, define the following parameters for this role:

- 1 Datacenter settings
- 2 Datastore settings
- 3 Distributed switch configuration
- 4 Host configuration
- 5 Network, performance, virtual machine, and vApp import settings

#### Procedure

- 1 Navigate to **Datacenter** by expanding the list and click the checkbox for:
  - Network protocol profile configuration
  - Query IP pool allocation
  - Release IP allocation

Figure 14-17. For 6 - 6.5 versions

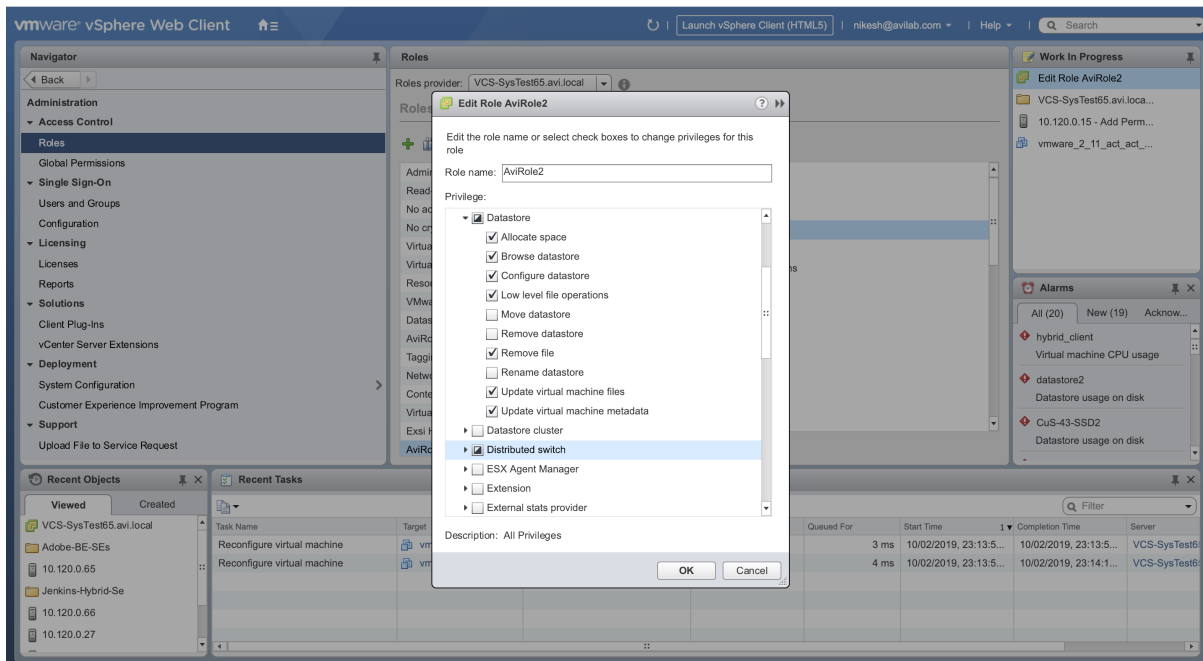
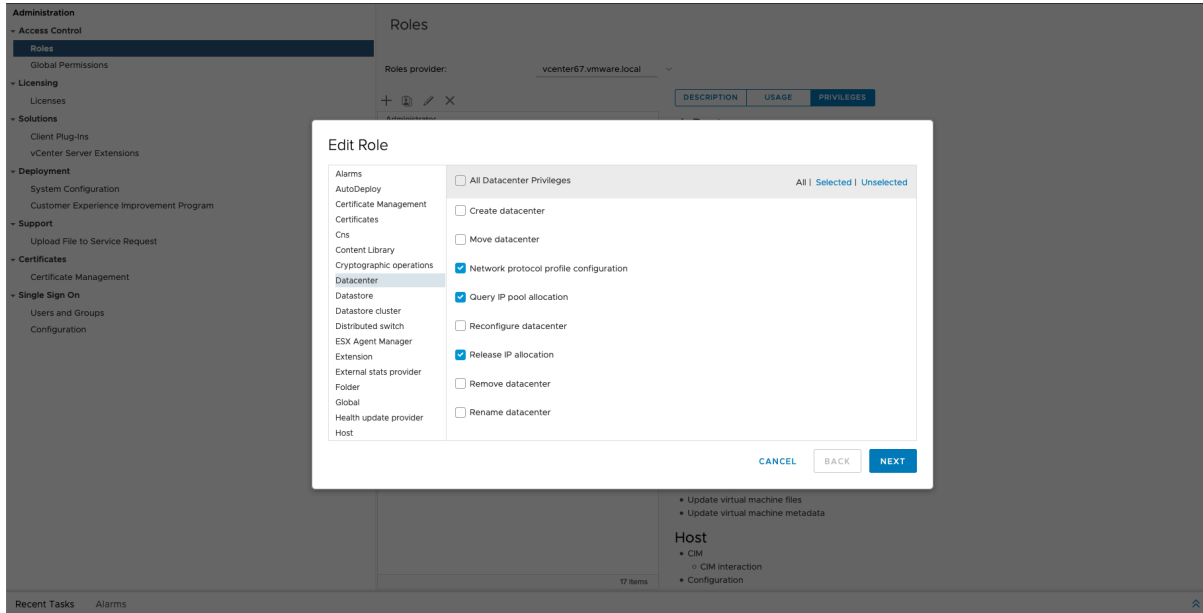


Figure 14-18. For 6.7 and 7 versions



2 Navigate to **Datastore** by expanding the list and click the following checkboxes:

- Allocate space
- Browse datastore
- Configure datastore
- Low-level file operations
- Remove file
- Update virtual machine files
- Update virtual machine metadata



Figure 14-19. For 6 - 6.5 versions

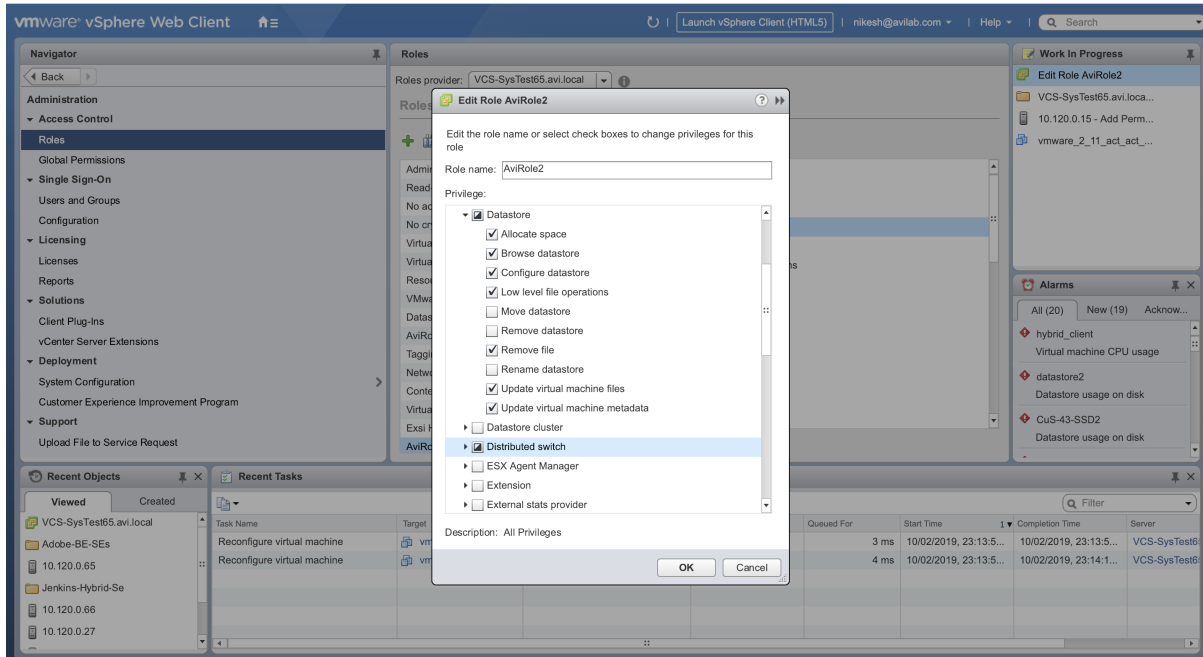
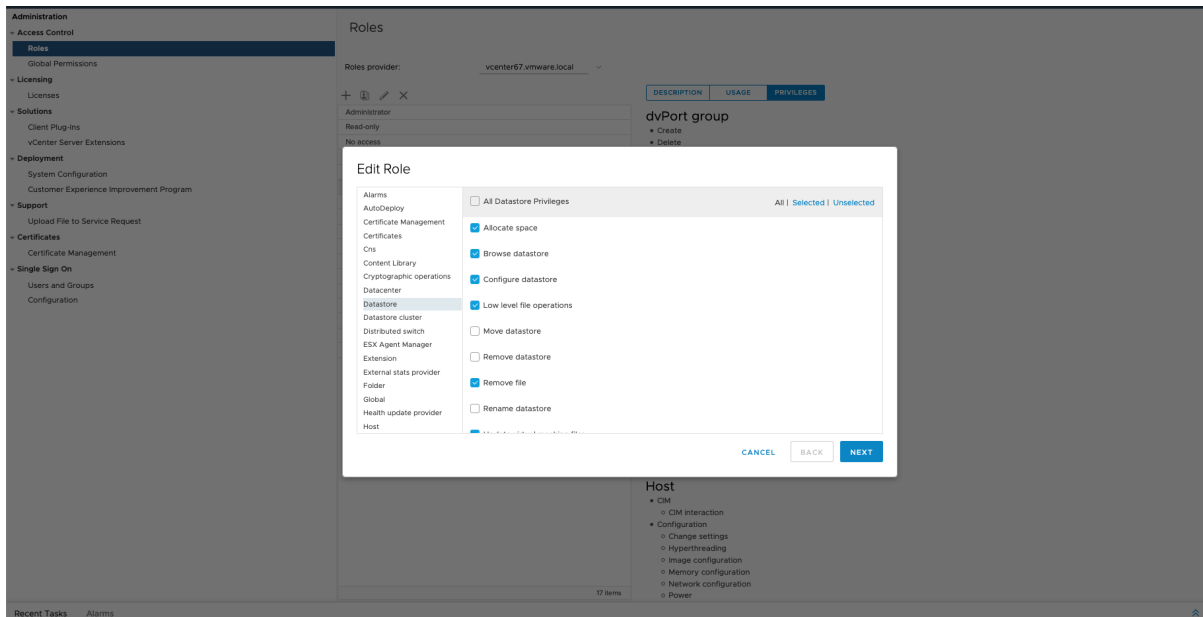


Figure 14-20. For 6.7 and 7 versions



### 3 Navigate to **Distributed switch** by expanding the list and click the checkbox for:

- Create
- Host operation
- Modify
- Network I/O control operation
- Policy operation

- Post configuration operation
- Port setting operation

Figure 14-21. For 6 - 6.5 versions

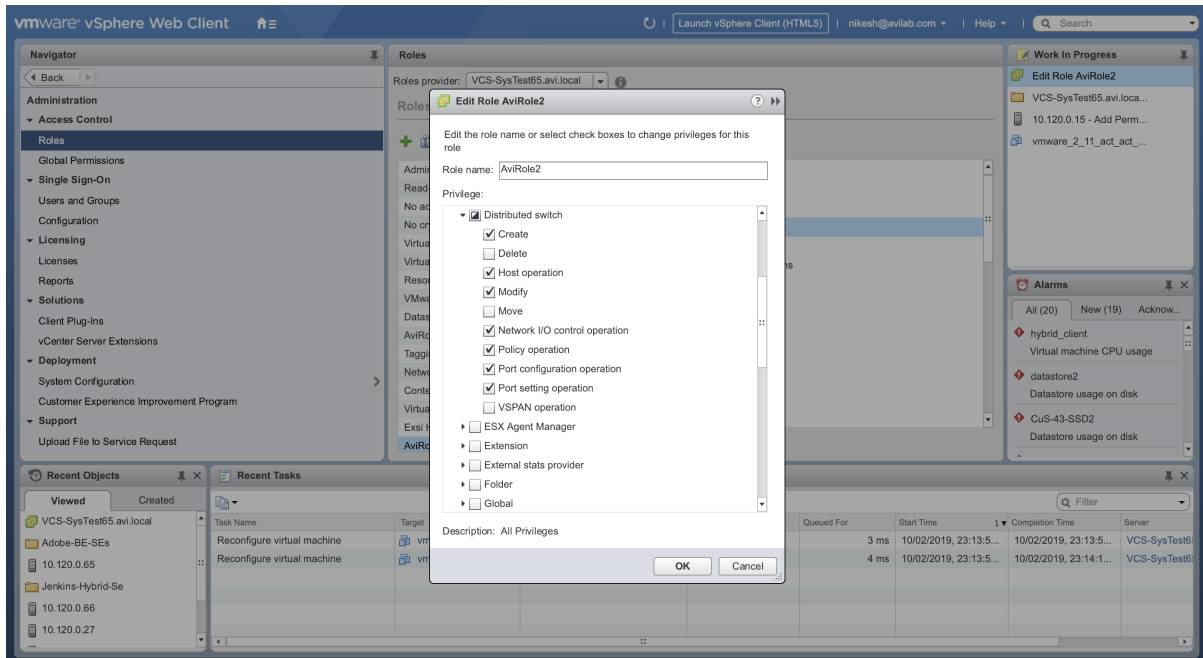
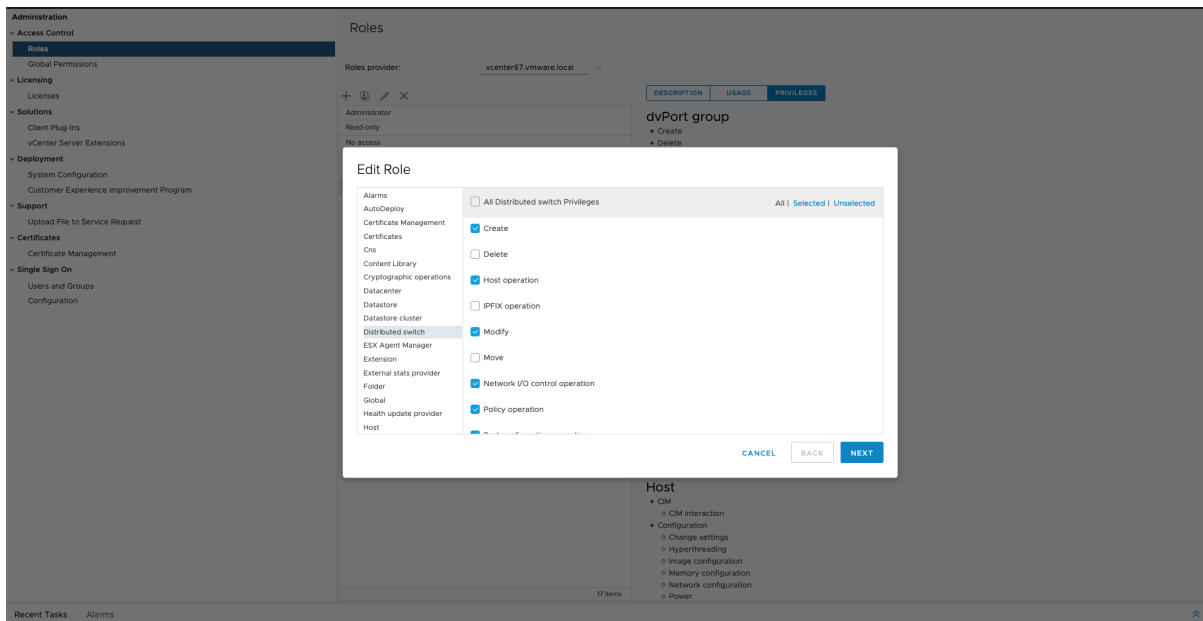


Figure 14-22. For 6.7 and 7 versions



#### 4 Navigate to Host by expanding the list and click the checkbox for:

- CIM
- Local operations

- Inventory Under Configuration, select the following options:
  - Change settings
  - Hyperthreading
  - Image configuration
  - Memory configuration
  - Network configuration
  - Power
  - System Management
  - System resources
    - Virtual machine autostart configuration

Figure 14-23. For 6 - 6.5 versions

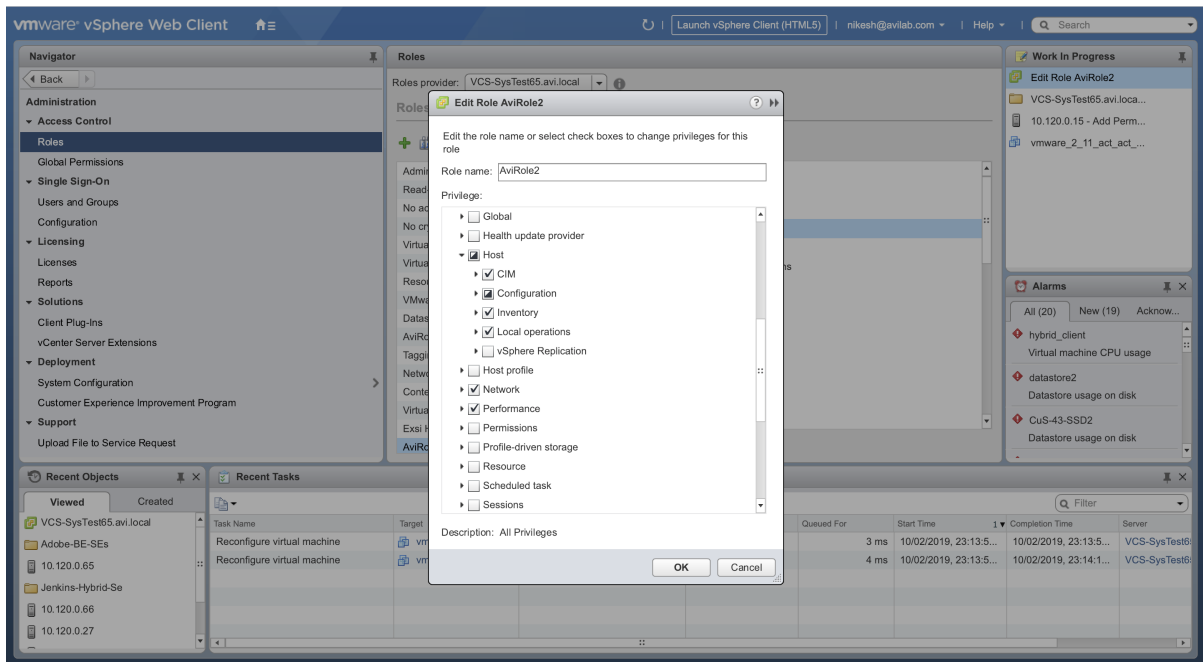
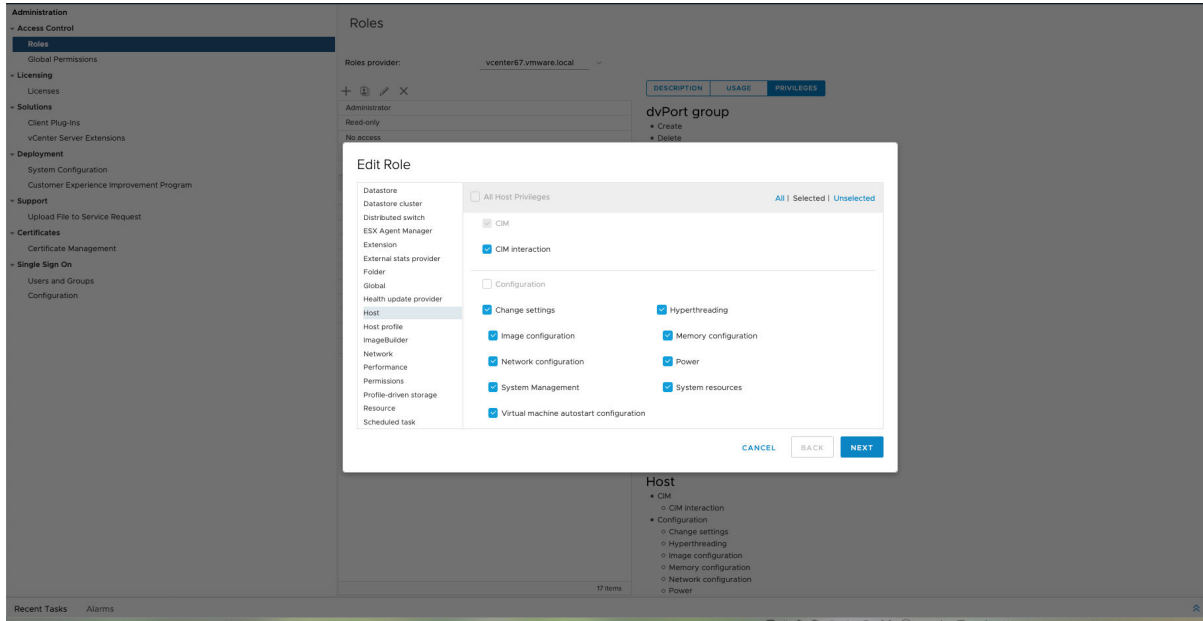


Figure 14-24. For For 6.7 and 7 versions



5 Select the checkbox. Enable all parameters under the following categories:

- Network
- Performance
- Tasks
- Virtual machine
- dvPort group
- vApp

Figure 14-25. For 6 - 6.5 versions

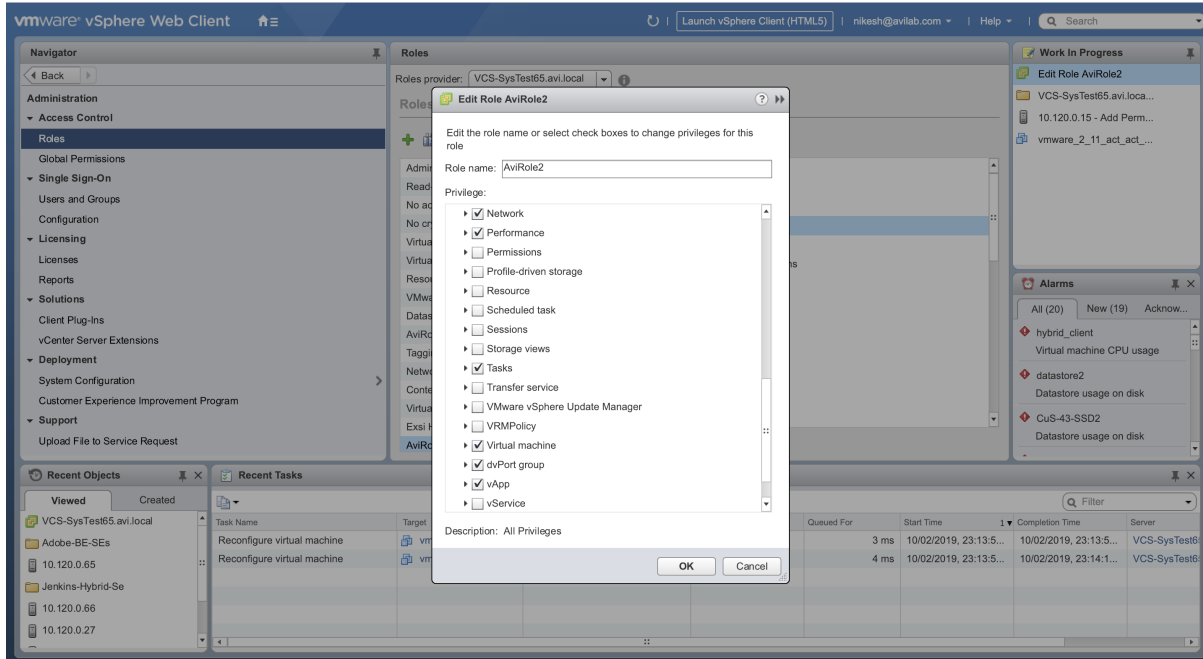
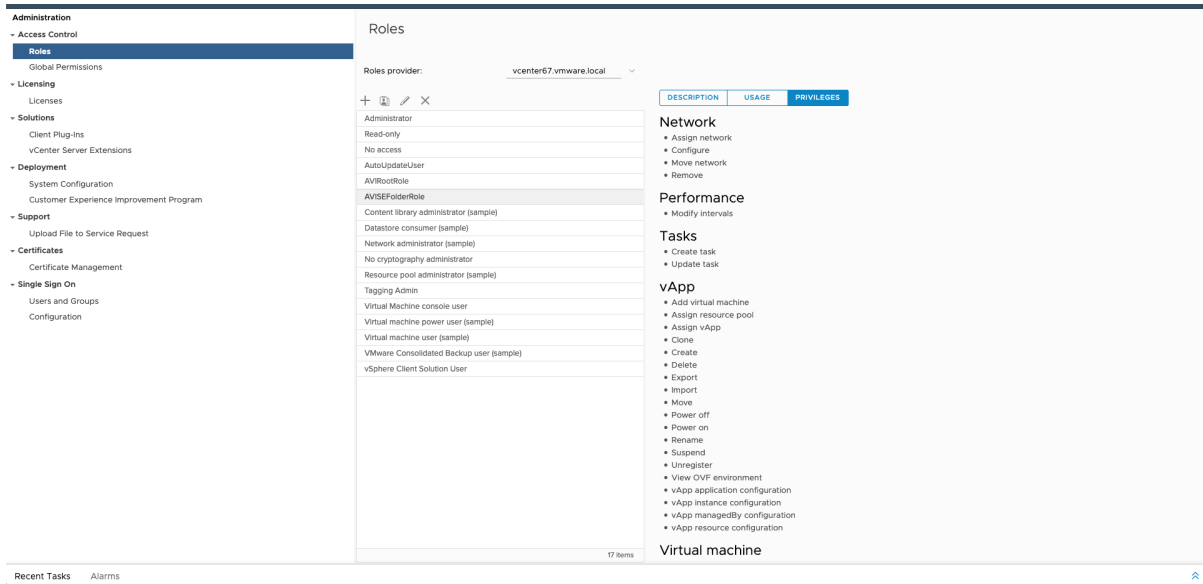


Figure 14-26. For 6.7 and 7 versions

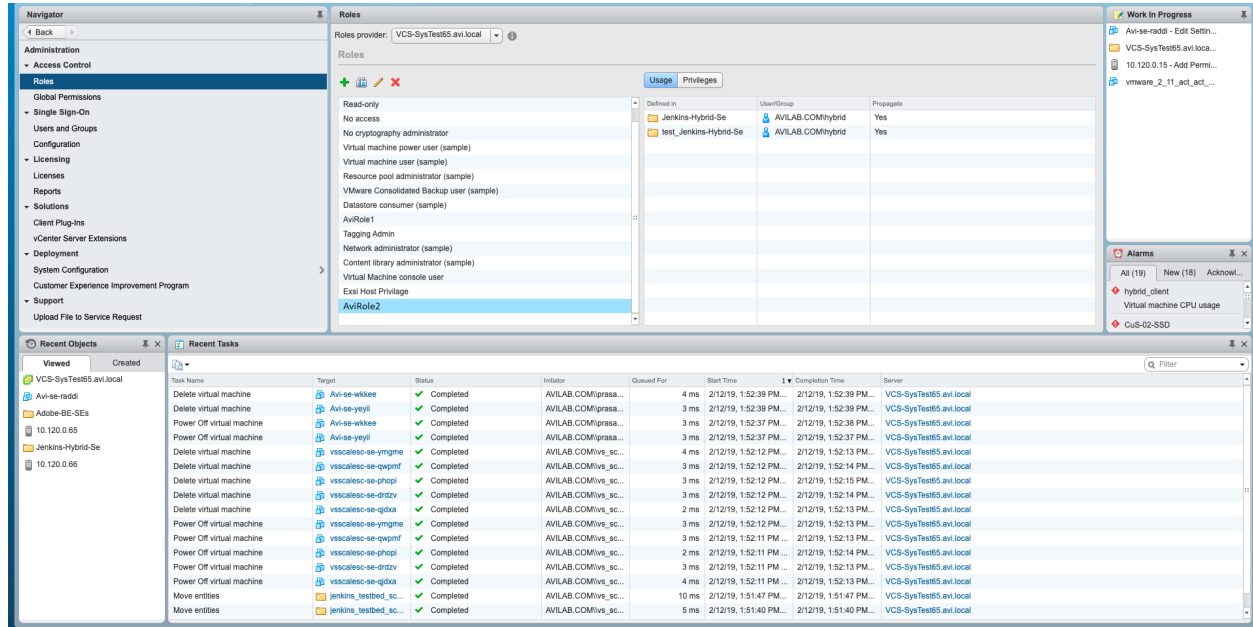


## Results

Assign AviRole2 to a folder that is defined for the Service Engine creation as shown in the example below:

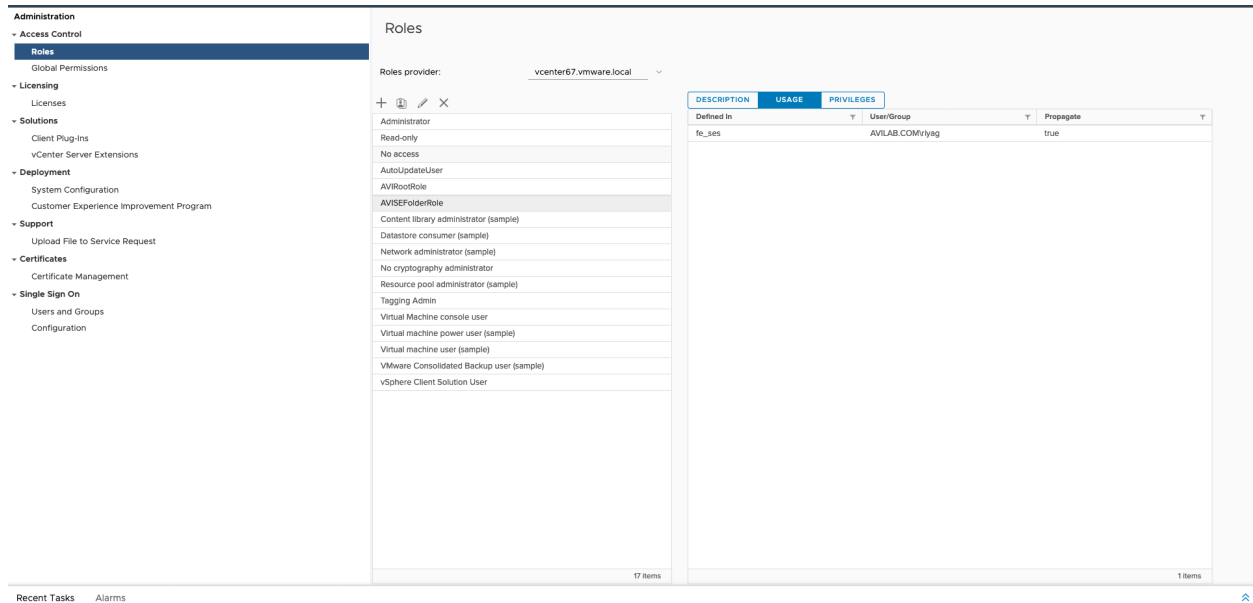
The folder name used in this example is Jenkins-Hybrid-Se.

Figure 14-27. For 6 - 6.5 versions



Assign AviRootRole1 to a folder that is defined for the Service Engine creation as shown in the example below. The folder name used in this example is FE-Se. The diagram is as follows:

Figure 14-28. For 6.7 and 7 versions



In NSX Advanced Load Balancer UI, navigate to **Infrastructure > Service Engine** and enter the folder name from the previous step under the **Service Engine Folder** field.

Edit Service Engine Group: act\_act-SE\_GRP

Basic Settings
Advanced

Service Engine Name Prefix
vmware\_2\_11\_act\_act\_act-SE\_GRP

Delete Unused Service Engines After
120
Min

Service Engine Folder
Jenkins-Hybrid-Se

Host & Data Store Scope

Host Scope Service Engine within
Any
Cluster
Host

Data Store Scope for Service Engine Virtual Machine
Any
Local
Shared

Cluster
Include
Exclude
Cumulus

Advanced HA & Placement

Buffer Service Engines
0

Scale per Virtual Service
2
Minimum
4
Maximum

Override Management Network
Select a Network

☐ CPU socket Affinity
☐ Dedicated dispatcher CPU

Cancel
Save

## Displaying vCenter Information

The following examples show the NSX Advanced Load Balancer Controller CLI commands used for displaying vCenter outputs:

```
[admin:jenkins-hybrid-controller]: > show vinfra vcenter 10.1.1.1-SantaClara datastores
```

| Field      | Value                                               |
|------------|-----------------------------------------------------|
| datacenter | datacenter-2-cloud-81cxxxxx-5bxx-46xx-89xx-5fexxxxx |

```
[admin:jenkins-hybrid-controller]: > show vinfra vcenter 10.1.1.1-SantaClara redis
```

| Name                  | Inventory State            | Progress |
|-----------------------|----------------------------|----------|
| 10.10.2.11-SantaClara | VCENTER_DISCOVERY_COMPLETE | 100      |
| 10.10.2.5-SantaClara  | VCENTER_DISCOVERY_COMPLETE | 100      |

```
[admin:jenkins-hybrid-controller]: > show vinfra vcenter 10.1.1.1-SantaClara hostresources
```

| Name                | Managed Object Id | Host Scale | Num Se | Se Fail | Se Success |
|---------------------|-------------------|------------|--------|---------|------------|
| 10.160.5.23         | host-603          | 2558       | -      | -       | -          |
| 10.160.5.24         | host-588          | 1217       | -      | -       | -          |
| cum-esx-9.avi.local | host-5526         | 431        | -      | -       | -          |
| cum-esx-8.avi.local | host-5513         | 543        | -      | -       | -          |

## Software Requirement

This section covers the software requirement for the VMware vCenter for the integration with NSX Advanced Load Balancer.

The following are the VMware vCenter versions supported in NSX Advanced Load Balancer:

| NSX Advanced Load Balancer Controller | VMware Ecosystems                 |
|---------------------------------------|-----------------------------------|
| 20.1.x                                | 5.1, 5.5, 6.0, 6.4, 6.5, 6.7, 7.0 |

NSX Advanced Load Balancer only works with the following switches in a vCenter cloud:

- Standard Virtual Switch (vSwitch)
- Distributed Virtual Switch (vDS)

Also, the deployment of the Controller and SE OVA directly on an ESX host (for a no-access or other cloud-connector types) is not supported. They must be deployed using the vCenter UI, as the deployment requires Open Virtualization Format (OVF) properties to be configured.

- VMXNET3 network adapter is required for SE to operate in DPDK mode in VMware
- NSX-T v 2.5. and 3.0
- VMware NSX 6.2.4+

## Ports and Protocols

This section explains the protocol ports used by the NSX Advanced Load Balancer for management communication.

For the protocols and ports used between communication between NSX Advanced Load Balancer and VMware vSphere, see [Ports and Protocols](#).

## IP Address Requirements

The NSX Advanced Load Balancer Controller requires only one management IP address. Administrative commands are configured on the Controller by accessing it using this IP address.

The management IP address is also used by the Controller to communicate with other SEs. This IP address for all Controllers within a cluster should belong to the same subnet. For more information, see [Controller Cluster IP](#) document.

Each NSX Advanced Load Balancer Service Engine requires one management IP address, a virtual service IP address, and an IP address that faces the pool network.

For quick deployments, DHCP is recommended over the static assignment for NSX Advanced Load Balancer SE management and the pool network IP address allocation.

---

**Note** Use a static IP for the NSX Advanced Load Balancer Controller management address, unless your DHCP server can preserve the assigned IP address permanently.

---



The virtual service IP address is manually specified while creating the load balancing application. You can automate the virtual service IP address allocation by integrating it with an IPAM service. For more information, see IPAM and DNS Support.

NSX Advanced Load Balancer load balances the traffic with VIP address: port as its destination across the members (servers) within the pool.

## VRF

Virtual Routing Framework (VRF) is a method of isolating traffic within a system. This is also referred to as a routing domain within the load balancer community.

### NSX Advanced Load Balancer VRF Support for vCenter Deployments

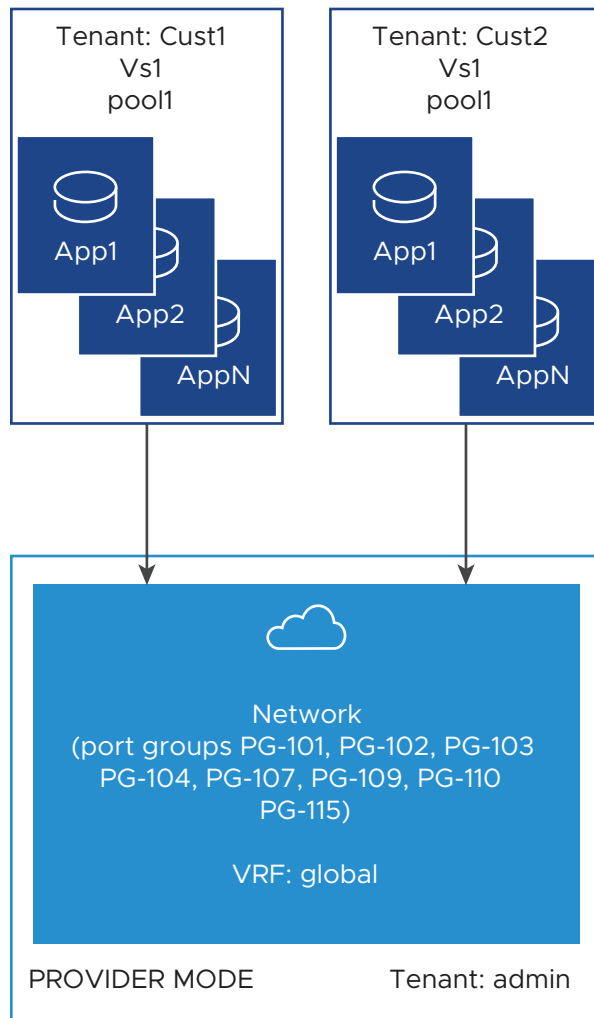
Virtual Routing Framework (VRF) is a method of isolating traffic within a system. This is also referred to as a routing domain within the load balancer community.

In NSX Advanced Load Balancer deployments with VMware vCenter, all port groups discovered from vCenter are placed into a single VRF context, by default, called Global. VRF contexts simplify virtual service deployment by organizing the port groups discovered from vCenter into subsets.

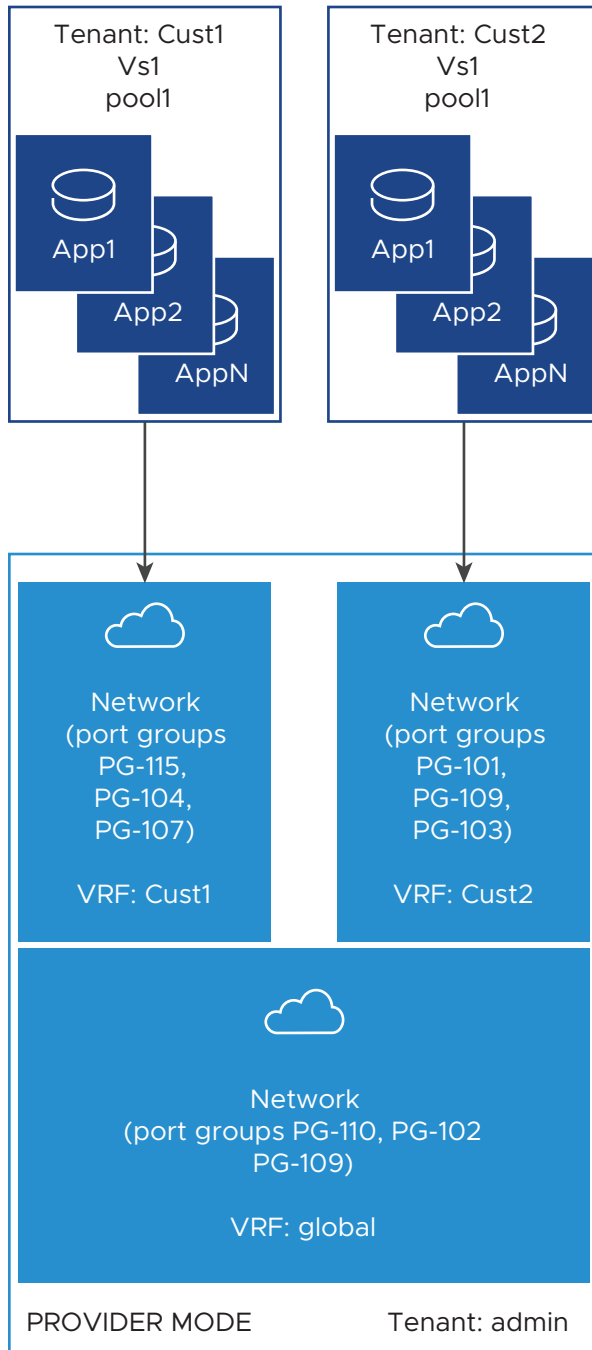
If multiple VRF contexts are configured, the workflow for creating a virtual service begins with selecting the VRF in which the virtual service is placed. The web interface presents only the networks in the selected VRF context as valid targets for placing that virtual service.

#### Global VRF and Admin Tenant

When a VMware cloud is created on the NSX Advanced Load Balancer Controller, NSX Advanced Load Balancer adds all of the port groups learned from vCenter into a VRF named Global in the admin tenant.



Additional VRF contexts can be created in the admin tenant, and individual port groups can be moved from the global VRF into the other VRF contexts. The VRF contexts and their port groups remain in the admin tenant but are available for selection by other tenants when they create virtual services, as shown in the following example:



### NSX Advanced Load Balancer Provider Mode

When integrated with a vCenter-managed cloud, NSX Advanced Load Balancer operates in Provider mode. This is not the same as a virtual routing context. Provider mode is an NSX Advanced Load Balancer deployment mode wherein all the cloud's network resources — the packet plumbing — remain in the admin tenant and cannot be moved. To configure VRF contexts and move port groups into them, the NSX Advanced Load Balancer user must have write privileges for the admin tenant.

## Configuring VRF

This section gives steps for creating VRF contexts on the Service Engine Group, moving port groups to the VRF contexts, and creating virtual services in those contexts.

---

### Note

- This section assumes that the Controller has already been installed, and that initial configuration of the Controller has been performed using the setup wizard.
  - The NSX Advanced Load Balancer user must have write privileges for the admin tenant. These steps can be performed only from the admin tenant.
- 

## IPAM and DNS

Virtual Routing Framework (VRF) is a method of isolating traffic within a system. This is also referred to as a routing domain within the load balancer community.

For more information, see [Service Discovery Using IPAM and DNS](#).

## Controller and VMware Communication

The NSX Advanced Load Balancer Controller must be able to communicate with vCenter and all ESX hosts that contribute to the deployment. If the communication fails, then the NSX Advanced Load Balancer Controller will not be able to spawn SEs.

Similarly, if the ESX hosts have DNS names, the NSX Advanced Load Balancer Controller must point to the DNS server used by the ESX hosts to avoid the names resolving to different IP addresses.

## Deploying NSX Advanced Load Balancer Controller in VMware vCenter

NSX Advanced Load Balancer Controller is deployed using write access mode, read access mode, or the no access mode.

### Modes of Deployment

Depending on the level of vCenter access provided, NSX Advanced Load Balancer can be deployed in a VMware cloud in the following modes:

#### Write access mode

This mode requires a vCenter user account with write privileges. The Controller automatically spins up the Service Engines as needed, and accesses vCenter to discover information about the networks and VMs.

#### Read access mode

This mode requires a vCenter user account with read privileges. The Controller accesses vCenter to discover information about the networks and VMs. The Service Engines are spun up and connected to the networks by NSX Advanced Load Balancer and the vCenter administrator.

### No access mode

The Controller does not access vCenter. The NSX Advanced Load Balancer and vCenter administrator manually deploy the Service Engines, define networks and interface IP addresses, and map the Service Engines to the correct networks.

---

**Note** In NSX Advanced Load Balancer, IPv6 is supported for VMware vCenter.

---

Write access is the recommended deployment mode. It is the quickest and easiest way to deploy and offers highest levels of automation between NSX Advanced Load Balancer and vCenter.

In read access mode,

- The Controller can discover networks and VMs.
- The Controller does not automatically deploy the Service Engines or connect them to the networks.
- The Controller can provide the properties of the SE VM analytics.
- vCenter's OVF property Controller cluster UUID for the Controller must be set for the SEs to connect to the correct Controller cluster.
  - Navigate to **Infrastructure > Clouds** and click the key icon, as shown in the screenshot below, to retrieve the Controller cluster UUID.

In no access mode,

- The Controller does not access vCenter and does not automatically deploy the SEs or connect them to the networks.
- The SE deployment and network placement are performed by NSX Advanced Load Balancer and vCenter administrators.
- The Controller does not provide the VM properties of the SE VM analytics. However, it continues to provide virtual service analytics.
- vCenter's OVF property **Controller Cluster UUID** for the Controller must be set.

Avi-se-umxxb - Edit Settings

Virtual Hardware VM Options SDRS Rules vApp Options

Product: Avi Service Engine  
Version: 16.3.2  
Vendor: Avi Networks, Inc

Application: 8 settings

IP Address of the Avi Controller: 10.10.25.223

Avi Service Engine Type: For Avi Networks, Inc internal use. Do not change  
UNIFIED ADMIN:1cbb3980-3566-4854-ab7e-538016491975,AVICLOUD\_UUID:cloud-231270e1-fdf8-47b8-9248-b0ab9cd65dc1,AVIBACKWARDCOMP:True

Authentication token for Avi Controller: Authentication token used by Avi Controller to verify this instance.

Controller Cluster UUID for Avi Controller: Unique identifier for an Avi Controller  
**cluster-005056b093d0**

Management Interface IP Address: IP address for the Management Interface. Leave blank if using DHCP. Example: 192.168.10.4  
10.128.3.191/24

Management Interface Subnet Mask: Subnet mask for the Management Interface. Leave blank if using DHCP. Example : 24 or 255.255.255.0

Default Gateway: The Default Gateway for the Service Engine. Leave blank if using DHCP.  
10.128.3.1

Compatibility: ESXi 5.0 and later (VM version 8)

OK Cancel

## Deploying NSX Advanced Load Balancer Controller

Follow the steps given below to deploy NSX Advanced Load Balancer in a vCenter managed VMware cloud.

The steps to deploy a Controller consists of the following. These steps are common across all the modes:

- 1 Deploying NSX Advanced Load Balancer Controller OVA
- 2 Performing the NSX Advanced Load Balancer Controller initial setup
- 3 (For static IP assignment) Configuring IP address pools
- 4 Verifying Installation

### Deploying the NSX Advanced Load Balancer Controller OVA

This task explains how to deploy NSX Advanced Load Balancer Controller OVA.

Use the client to deploy NSX Advanced Load Balancer Controller OVA file by following the steps mentioned below:

#### Prerequisites

Log into the vCenter server through a vCenter client.

#### Procedure

- 1 Click **File** in the top menu and select **Deploy OVF Template**.

**2** Follow the Deploy OVA Template wizard instructions:

- a Choose **Thick Provision Lazy Zeroed** for disk format.
- b Choose a port group for **Destination Networks in Network Mapping**.  
This port group will be used by the NSX Advanced Load Balancer Controller to communicate with vCenter.
- c Specify the management IP address and default gateway. In the case of DHCP, leave this field empty.
- d Leave the **Sysadmin login authentication key** field blank.

**3** Power on the VM.**4** Repeat these steps to deploy three NSX Advanced Load Balancer Controller VMs to create an Controller cluster set-up.**Performing the NSX Advanced Load Balancer Controller Initial setup**

You can change or customize settings following initial deployment using the NSX Advanced Load Balancer Controller's web interface.

To perform the NSX Advanced Load Balancer Controller initial setup:

**Prerequisites**

Navigate to the NSX Advanced Load Balancer Controller on your browser.

---

**Note** While the system is booting up, a blank web page or a 503 status code may appear. Wait for about 5 to 10 minutes and then follow the instructions below for the setup wizard.

---

**Procedure****1** Configure the basic system settings:

- a Administrator account
- b DNS and NTP server information
- c Email or SMTP information

**2** Set the infrastructure type to VMware.**3** Enter the vCenter settings:

- a vCenter credentials – To create SEs, the vCenter account must have privileges to create new folders in vCenter
- b vCenter IP address
- c Permissions – Select Write
- d Integration with Cisco APIC – Leave unselected or disabled

- e Data center – NSX Advanced Load Balancer will be deployed here
  - f IP allocation method for the networks where the pools and virtual services will be located – DHCP or Static (Wizard screen example below shows DHCP.)
- 4 Configure NSX Advanced Load Balancer SE Settings:
- a Management network
  - b IP allocation method for management network
  - c **Support Multiple Tenants**– Select **No**.

One vNIC out of the 10 NSX Advanced Load Balancer SE vNIC is for management network connection. The other vNICs are data vNICs.

For the IP allocation method, enter a subnet address and a range of host addresses within the subnet, in the case of static address assignment. NSX Advanced Load Balancer will assign addresses from this range to the NSX Advanced Load Balancer SE data interfaces.

### Results

If the management and pool networks use DHCP, the deployment procedure is complete with this step. In case of static IP address allocation, you need to configure an IP address pool as explained in the next section.

### Verifying the Configuration

This task explains how to verify the configuration.

To verify the installation:

### Procedure

- 1 Navigate to **Infrastructure > Clouds**.
- 2 Click **Default-Clouds**.
- 3 Click the **Status** button.

### Results

If the status is green, then the installation is a success.

## Configuring NSX Advanced Load Balancer Controller Cluster

This section discusses the steps for configuring a Controller cluster.

To configure the Controller cluster, login to the first NSX Advanced Load Balancer Controller, configured in the previous section, and follow the steps below:

- 1 Navigate to **Administration > Controller > Nodes**.
- 2 Click **Edit**.
- 3 The IP of the first NSX Advanced Load Balancer Controller will be present. Enter the IP addresses and names of the remaining two Controllers.



- 4 Assign a static IP as **Controller Cluster IP**.
- 5 Click **Save**.

For more information, see [Deploying an NSX Advanced Load Balancer Controller Cluster](#).

## Communication Between NSX Advanced Load Balancer Controller and VMware vSphere

This section discusses the communication between NSX Advanced Load Balancer Controller and VMware vSphere.

The NSX Advanced Load Balancer Controller must be able to communicate with vCenter and all ESX hosts that contribute to the deployment. If the communication fails, then the NSX Advanced Load Balancer Controller will not be able to spawn SEs.

Similarly, if the ESX hosts have DNS names, then the NSX Advanced Load Balancer Controller must point to the DNS server used by the ESX hosts to avoid the names resolving to different IP addresses.

## Deploying an NSX Advanced Load Balancer Controller Cluster

This section shows the web interface pop-up and CLI commands for deploying an NSX Advanced Load Balancer Controller cluster.

To deploy an NSX Advanced Load Balancer Controller cluster, first deploy a single NSX Advanced Load Balancer Controller node (the leader), and then add the follower nodes to the leader. (Ensure that after using the setup wizard to install NSX Advanced Load Balancer on the follower nodes, you do not make any other configuration changes on these nodes.) For more information about initial cluster deployment, see [Deploying an NSX Advanced Load Balancer Controller Cluster](#).

## Configuring NSX Advanced Load Balancer Cloud Connector

This section describes the steps to configure the NSX Advanced Load Balancer cloud connector in different access modes.

### Configuring NSX Advanced Load Balancer Cloud Connector in Write Access Mode

SE installation in write access mode is automatic.

- 1 From the NSX Advanced Load Balancer Controller UI, navigate to the **Infrastructure > Clouds** page on the NSX Advanced Load Balancer Controller on your browser.
- 2 Select Create to configure a new cloud. Select *VMware* as the orchestrator and provide name the Cloud.

### 3 Enter the vCenter settings.

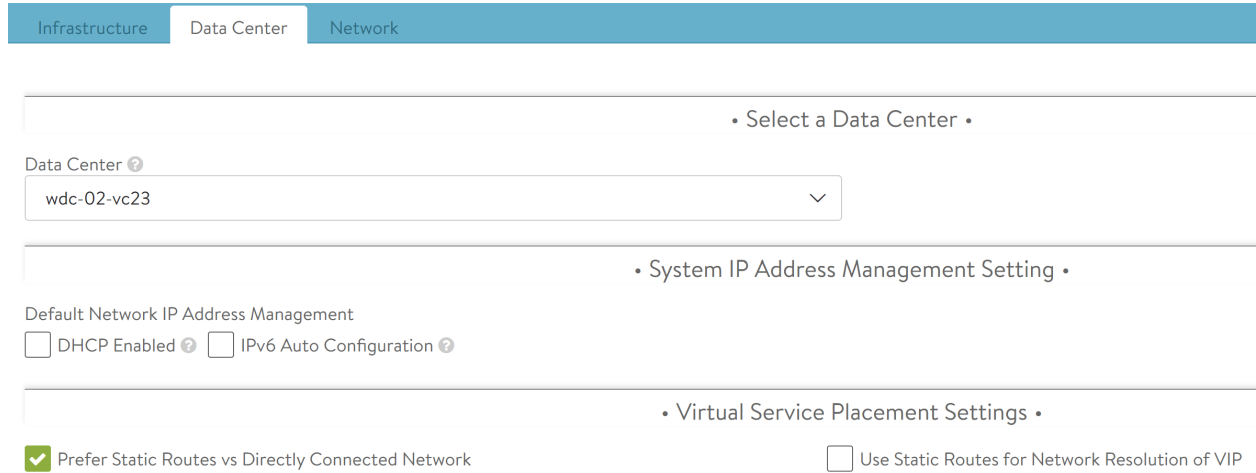
- vCenter credentials – To create SEs, the vCenter account must have privileges to create new folders in vCenter.
- vCenter FQDN/IP address
- Access Permissions – Select Write
- Set *SDN Integration Settings* to None

**Figure 14-29. VCenter Settings**

The screenshot shows the 'VCenter Settings' configuration page in the VMware NSX Advanced Load Balancer interface. The page has a blue header with tabs for 'Infrastructure', 'Data Center', and 'Network'. The 'Infrastructure' tab is active, showing the 'vcenter' configuration. The 'vcenter' field is set to 'wdc-02-vc23.oc.vmware.com'. The 'Password' field is masked with asterisks. The 'Access Permission' section has 'Read' and 'Write' buttons, with 'Write' selected. The 'SDN Integration Settings' section has two radio buttons: 'None' (selected) and 'VMware NSX'. The 'License Model' section has a 'License Type' dropdown menu set to 'Cores'. The 'IPAM/DNS' section has two fields: 'IPAM Profile' set to 'vmware\_ipam' and 'DNS Profile' set to 'avi\_dns'. Both fields have expand/collapse and edit icons. At the bottom, there is a checkbox for 'State Based DNS Registration' which is checked.

If the given vCenter credentials authenticate successfully with the vCenter Address given, then the Controller discover the DataCenters managed by this vCenter. Select the followings:

- Data center – The SEs will be deployed in this DataCenter
- IP allocation method for the networks where the pools and virtual services will be located – DHCP or Static. (Defaults to static)

**Figure 14-30. VCenter Data Center Settings**


Infrastructure Data Center Network

• Select a Data Center •

Data Center ?

wdc-02-vc23

• System IP Address Management Setting •

Default Network IP Address Management

☐ DHCP Enabled ? ☐ IPv6 Auto Configuration ?

• Virtual Service Placement Settings •

☒ Prefer Static Routes vs Directly Connected Network ☐ Use Static Routes for Network Resolution of VIP

Select the Management network PortGroup.

- This PortGroup will be mapped to the management network vNIC on the SEs
- This PortGroup should be configured to have IP connectivity to the Controllers
- IP addresses for management can be either DHCP or static
  - Static IP addresses can be configured in the Static IP address pool

Note : The example snapshot shows static IP allocation

Figure 14-31. VCenter Network

New Cloud: WorkloadDomain1

Infrastructure Data Center Network

• Select Management Network •

Management Network \* ?

Mgmt\_Arista\_2

• Service Engine •

Template Service Engine Group ?

None

• IP Address Management for Management Network •

☐ DHCP Enabled ? ☐ IPv6 Auto Configuration ?

IP Subnet \* ?

192.168.100.0/24

Add Static IP Address Pool \* ?

192.168.100.10-192.168.100.100

+

Default Gateway

192.168.100.1

Cancel Complete

Click **Complete** to create a vCenter Cloud on NSX Advanced Load Balancer.

Figure 14-32. VCenter Final Configuration

☰

Infrastructure

Dashboard

Clouds

Service Engine

Service Engine Group

Networks

Routing

GSLB

admin

⋮

⌵

Q

Create

| <input type="checkbox"/> ⌵     | Name ⌵          | Type                              | Status                                      | ⚙ |
|--------------------------------|-----------------|-----------------------------------|---------------------------------------------|---|
| <input type="checkbox"/>       | Default-Cloud   | None                              | ●                                           |   |
| <input type="checkbox"/>       | WorkloadDomain2 | VMware vCenter/vSphere ESX        | ●                                           |   |
| <input type="checkbox"/>       | WorkloadDomain1 | VMware vCenter/vSphere ESX        | ●                                           |   |
| vCenter Hostname<br>10.10.2.10 |                 | Username<br>root                  | IP Address Management<br>Static IP Address  |   |
| Data Center<br>10GTest         |                 | Access Permission<br>WRITE_ACCESS | Management Network<br>vSwitch:Mgmt_Arista_2 |   |

## Configuring NSX Advanced Load Balancer Cloud Connector in No Access Mode and Read Access Mode

This section discusses the steps to configure the NSX Advanced Load Balancer cloud connector in No access mode an read access mode.

## Downloading NSX Advanced Load Balancer Service Engine on OVA

This section discusses the steps to download the SE on OVA.

- 1 The OVA image file for Service Engines is embedded in the Controller image. The NSX Advanced Load Balancer SE OVA image can be downloaded using the web interface or the API.
  - a Using the NSX Advanced Load Balancer UI, navigate to **Infrastructure > Cloud**, click the button and select **se.ova** to download the OVA image.
- 2 Using the API, navigate to *http://avi-ctrl-ip/api/fileservice/seova*, where avi-ctrl-ip is the IP address of the NSX Advanced Load Balancer Controller.
- 3 After you install the OVA and before you power the Controllers on, edit the hardware resources and change the CPU, memory, and disk to the minimum recommended values for production. For more information, see [NSX Advanced Load Balancer Controller Sizing](#).

## Deploying NSX Advanced Load Balancer Service Engine OVA

The section explains the steps required to deploy the SE OVA.

### Procedure

- 1 In vCenter, click **File** in the top menu and choose Deploy OVF Template.
- 2 Follow the Deploy OVA Template wizard instructions:
  - a Select **Thick Provision Lazy Zeroed** for disk format.
  - b Select the port groups for the SE network connections. The SE has ten vNICs. Connect the first vNIC to the management network. Connect the other vNICs to the data network.
  - c For the management connection, choose a port group that will allow the SEs to communicate with the Controller. The SE can be connected to up to nine data networks. Choose a port group in the destination networks for each source network, where you can host the virtual services and pools. The Controller expects the SE's data vNICs to be connected to virtual service and pool networks.
  - d Specify the Controller IP address.
  - e Navigate to **Infrastructure > Cloud** to copy the authentication token.
  - f Specify the management IP address and default gateway. In the case of DHCP, leave this field empty.
  - g The field for the Controller cluster UUID must be left blank.
- 3 In the VM properties menu, connect the SE data vNICs that are required to reach a virtual service network and pool network to the port groups. Leave the unused vNICs disconnected.

- 4 Note down the following information: MAC address of the vNICsIP subnet of the port group. This information will be used to identify the SE interfaces, as the Controller does not have access to vCenter and so cannot associate the SE's interface names with VMware's interface names.
  - a MAC address of the vNICs
  - b IP subnet of the port group
- 5 MAC address of the vNICsIP subnet of the port group.

#### What to do next

Repeat the above steps for at least one more Service Engine. By default, two NSX Advanced Load Balancer SEs are required for deploying a virtual service.

### Configuring NSX Advanced Load Balancer Service Engine Interfaces

This section discusses the steps to configure the NSX Advanced Load Balancer SE interfaces.

NSX Advanced Load Balancer Service Engine requires an IP address in each of the virtual service networks and server networks. This process is automatic in write access and read access mode. For no access mode, follow the steps below:

- 1 On NSX Advanced Load Balancer UI, navigate to **Infrastructure > Service Engine**, and select the SE that was deployed in the previous section.
- 2 Find the interface that matches the list of MAC addresses that were noted down during the SE deployment.
- 3 Enable the DHCP option for the interface, if it is available. Otherwise, provide a static IP address as explained in the next section.

Repeat the above steps for all connected interfaces of the virtual service and server networks.

### Configuring NSX Advanced Load Balancer Cloud Connector in Read Access Mode

This section discusses the steps to configure the cloud connector in the read access mode..

- 1 From the NSX Advanced Load Balancer Controller UI, navigate to the **Infrastructure > Cloud** page on the NSX Advanced Load Balancer Controller on your browser.
- 2 Select **Create** to configure a new cloud. Select *VMware* as the orchestrator and provide name the Cloud.
- 3 Enter the vCenter settings.
  - vCenter credentials – To create Service Engines, the vCenter account must have privileges to create new folders in vCenter.
  - vCenter FQDN/IP address
  - Access Permissions – Select Read
  - Set *SDN Integration Settings* to None.

Figure 14-33. VCenter Read Access Mode

The screenshot shows the 'New Cloud: test' configuration window. The progress bar indicates four steps: Step 1: Select Cloud, Step 2: Infrastructure (active), Step 3: Data Center, and Step 4: Network. The 'Name' field is set to 'test'. The 'vCenter / vSphere Login' section includes a 'Username' field with 'admin', a 'vCenter Address' field with '10.10.21.1', and a 'Password' field with masked characters. There are 'Read' and 'Write' buttons for 'Access Permission'. The 'SDN Integration Settings' section has radio buttons for 'None' (selected) and 'VMware NSX'. The 'License Model' section has a 'License Type' dropdown menu set to 'Cores'. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons.

Perform the remaining steps as mentioned in the previous section.

## Configuring NSX Advanced Load Balancer Cloud Connector in No Access Mode

This section discusses the steps to configure the cloud connector in the No Access mode.

- 1 From the NSX Advanced Load Balancer Controller UI, navigate to the **Infrastructure > Cloud** page on the NSX Advanced Load Balancer Controller on your browser.
- 2 Select Create to configure a new cloud. Select *No Orchestrator* and provide name of the Cloud.

Figure 14-34. No Access Cloud

The screenshot shows the 'New Cloud: No Access Mode' configuration window. The progress bar indicates two steps: Step 1: Select Cloud (active) and Step 2: DHCP Settings. The 'Name' field is set to 'No Access Mode'. The 'Cloud Infrastructure Type' section displays a row of icons for VMware vCenter/vSphere ESX, OpenStack, Amazon Web Services, Linux, Microsoft Azure, Google Cloud Platform, and 'No Orchestrator' (which is highlighted). Below this, the 'Orchestrator' is set to 'None'. At the bottom, there are 'Cancel' and 'Next' buttons.

- 3 Select the required options for the followings:
  - a IP address
  - b Routes Option
  - c IPAM Settings
  - d DNS options

## Configuring IP Address Pools

This task explains how to configure IP address pools.

---

**Note** This section is applicable only for static IP address allocation.

---

Configure IP address pools for networks hosting NSX Advanced Load Balancer SEs by following the steps mentioned below:

### Procedure

- 1 Navigate to **Infrastructure > Cloud > Default-Cloud > Network**.
- 2 Find a port group and IP subnet on which the DHCP service is not available.
- 3 Select the port group by clicking the edit icon.
- 4 Select **Static** under **Network IP Address Management**.
- 5 Select the **IP Subnet** by clicking the edit icon.
- 6 Enter the static IP address or the range of IP addresses.

## Additional Configuration Options

This section covers the following configuration options:

- Configuring VRF
- Configuring NSX Advanced Load Balancer IPAM and DNS

### Configuring VRF

This section focuses on configuring VRF for VMware vSphere deployments.

Following are the steps to configuring VRF:

#### Creating VRF Contexts

This section explains the process to create VRF contexts.

The following are the steps to create VRF contexts:

### Procedure

- 1 Navigate to **Infrastructure > Routing**.



- 2 Click the cloud name to select the cloud. The cloud name, by default, will display as **Default-Cloud** if the VMware vCenter cloud is the only one configured.
- 3 In the **VRF Context** tab, click **Create**.
- 4 Specify the name of the VRF context and click **Save**.

## Creating Networks to the VRF Contexts

This task explains how to create networks to the VRF contexts.

To create networks to the VRF contexts:

### Procedure

- 1 Navigate to **Infrastructure > Networks**.
- 2 Select cloud by clicking the **Select Cloud** drop-down menu.
- 3 Select the **DHCP** checkbox in the **IP Address Management** section.
- 4 Click **Save**.

Edit Network Settings: test

Name <sup>\*</sup>

test

• IP Address Management •

☒ DHCP Enabled ☒ IPv6 Auto Configuration

+ Add Subnet

• Network IP Subnets •

Q

Displaying 0 items

| <input type="checkbox"/> IP Subnet | Type | IP Address Pool |
|------------------------------------|------|-----------------|
| No items found                     |      |                 |

Cancel Save

## Creating Virtual Services in a VRF

This section explains the process to create virtual services in a VRF.

The following are the steps to create virtual services in a VRF:

### Procedure

- 1 Navigate to **Applications > Dashboard**
- 2 Click on **Create Virtual Service**

- 3 Select **Basic Setup** option
- 4 Click on the cloud name to select the cloud
- 5 Click on **Next**
- 6 Select the VRF context from the list and click on **Next**
- 7 Specify a name for the virtual service, virtual IP address (VIP) and other properties of the virtual service.
- 8 Click on **Save**

**Note** The steps to create a virtual service in a VRF can be performed from the admin tenant or from another tenant.

## NSX Advanced Load Balancer IPAM and DNS

This section discusses the native IPAM and DNS support for the various clouds.

| Provider             | NSX Advanced Load Balancer |     |          |     |              |               |
|----------------------|----------------------------|-----|----------|-----|--------------|---------------|
|                      | Infoblox                   |     | Internal |     | Cloud-native |               |
| Cloud Infrastructure | IPAM                       | DNS | IPAM     | DNS | IPAM         | DNS           |
| VMware vCenter       | Yes                        | Yes | Yes      | Yes | NA           | NA (Not Used) |
| No access cloud      | Yes                        | Yes | Yes      | Yes | Yes          | No            |

- When creating virtual services in OpenStack or AWS cloud, a separate configuration for IPAM is not needed/allowed, since the cloud configuration has support for IPAM natively in NSX Advanced Load Balancer.
  - When you select 'Default', the cloud's IPAM/DNS support is accepted without additional action on the part of the administrator.
  - NSX Advanced Load Balancer supports Route 53 when AWS is the cloud provider configuration in NSX Advanced Load Balancer.
  - 'Not Used' means, although the cloud supports DNS, NSX Advanced Load Balancer does not use it.
- When creating a virtual service in the Linux Server cloud in AWS/GCP environment, you can use the cloud-native IPAM solution of AWS/GCP.
- NSX Advanced Load Balancer DNS service can be used with all these clouds.

### IPAM and DNS Provider (Infoblox)

The Controller integrates with Infoblox's RESTful Web API (WAPI) for IPAM and DNS services.

These API calls are initiated by the Controller and directed to the Infoblox Grid Master IP address, or virtual IP address (VIP), if has been deployed in a HA pair. This integration enables the NSX Advanced Load Balancer to automate the allocation of IP addresses as well as the creation and deletion of host objects in DNS as new virtual services are created/deleted in the NSX Advanced Load Balancer environment.

It is assumed that all interested subnets and domain names (zones) have been configured in the Infoblox server for consumption by NSX Advanced Load Balancer. That said, when configuring Infoblox DNS and IPAM profiles, it is possible to be selective, as the next section will show.

Infoblox IPAM and DNS profiles should be independently defined and configured. For a given cloud, the permitted Infoblox combinations are:

| IPAM Provider | DNS Provider                                            |
|---------------|---------------------------------------------------------|
| Infoblox IPAM | None, Infoblox DNS, NSX Advanced Load Balancer internal |
| Any           | Infoblox DNS                                            |