August 2022

NSX-T® to NSX® Advanced Load Balancer™ (Avi) Migration Tool

How to Guide

vmware®

This page was intentionally left blank.

NSX-T to NSX Advanced Load Balancer (Avi) Migration Tool – How to Guide

Revision Control

Version	Date	Reason
1.0	09/01/2022	Doc Initial release

Overview

NSX-T to AVI Migration Tool is an Open-Source tool developed by the AVI Development team to help NSX-T users seamlessly migrate their Virtual Services to NSX Advanced Load Balancer (AVI), enabling them to get all the benefits that NSX Advanced Load Balancer can deliver, such as elastic load balancing, application security, autoscaling, container networking, and web application firewall.

In this document, for the sake of simplicity, you will find that the terms NSX ALB, NSX Advanced Load Balancer, and AVI will be used indistinctly to refer to NSX Advanced Load Balancer (formerly AVI Vantage) solution.

Links detailing procedures, and KB articles can be found in the KB Articles section of this document.

Requirements to run the tool

While running the tool from an Avi controller with the NSX-T to Avi Migration Tool embedded, all dependencies will be already satisfied, and thus this chapter can be skipped. Controllers from versions 22.1.2 and 21.1.6 are distributed with the tool embedded into the alb-sdk directory.

To run the tool from any other client machine, you must ensure that the dependencies are fulfilled. To do so, the following pieces of software need to be installed:

Python3

Pip3 (updated)

Python packages:

- avimigrationtools **
- VMware NSX Policy SDK *
- VMware vSphere SDK *

Additional references include cloning the repository, upgrading pip, and installing the needed packages and SDKs in chapter "Git and Python references" under Additional References.

* NSX Policy API SDK and VMware vSphere SDK can be downloaded from github.com/vmware/alb-sdk/python/avi/migrationtools/lib/ or from VMware Customer Connect (VMware NSX-T Data Center/Drivers & Tools/Automation Tools and SDK(s)/VMware NSX-T Data Center Automation SDK 3.2.1 for Python. **A Customer Connect account is required to download from Customer Connect.**

** Pip source https://pypi.org/project/avimigrationtools/. To get a list of all the packages that are installed to fulfill dependencies of pip package avimigrationtools, visit https://github.com/vmware/alb-sdk/blob/eng/python/avi/migrationtools/setup.py

NSX Advanced Load Balancer Architecture Overview and Components

The NSX Advanced Load Balancer architecture is based on two main components:

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

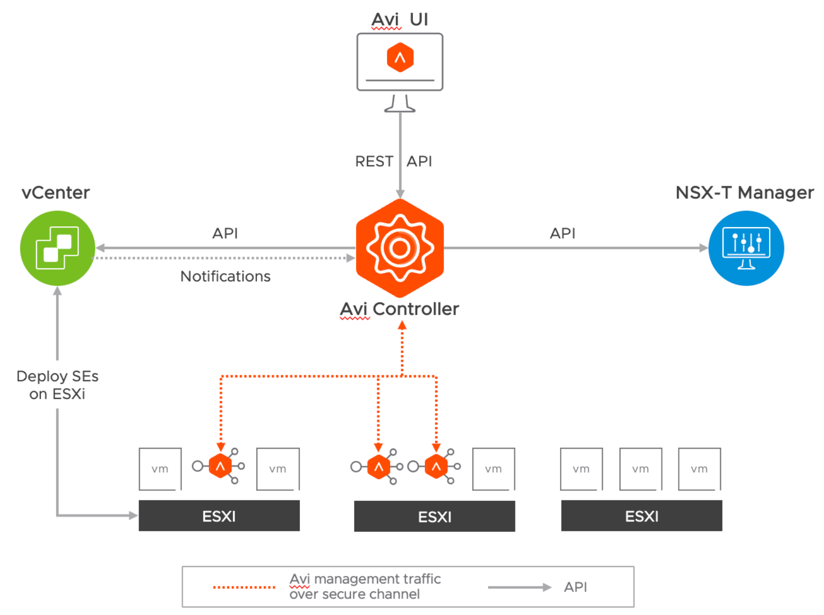
Although NSX Advanced Load Balancer is a multi-cloud automated solution, NSX-T cloud is the only cloud type supported by the migration tool, hence this document will work with NSX-T cloud only (NSX-T Cloud Connector).

Configuring an Avi NSX Cloud implies that the Avi Controller will establish communication channels with the VMware NSX-T Manager node to handle networking and firewall rules, and with VMware vCenter to deploy and manage Service Engines into vSphere nodes.

The NSX Advanced Load Balancer Controller needs to be deployed by the administrator by deploying an OVA image directly into the vCenter server. As an alternative, the Avi controller can be deployed from NSX Manager UI (See KB Articles section).

The Service Engines (SE) are deployed by the controller when needed, using vCenter and NSX API to accomplish that task. The "shape" of those Service Engines will be based on a definition previously made using Service Engine Groups. SE Groups work as templates, defining certain parameters that will be inherited by the SEs, such as HA mode (A/S, A/A, N+M), the amount of CPU, RAM, Storage, etc. A step-by-step guide to deploy and configure the NSX ALB Controller can be found in the KB section of this document.

The NSX Advanced Load Balancer Controller only has a management interface, but the Service Engines have 2 kinds of interfaces, a management interface and data plane interfaces. In this document, you will find instructions to create the NSX infrastructure needed to allow the SE Management Interfaces to communicate with the network, using a dedicated segment and a T1 router.



Ports and protocols used to establish communications can be found in the KB Section. Remember that the Avi Controller will need a user and password to access NSX-T Manager and vCenter. Details of the requirements for that user can be found in the KB Section.

You can find step-by-step guides explaining how to create an NSX-T Cloud in the Avi Controller and configure the T1 LR and segments on NSX in the KB section.

Pre-requisites to migrate from NSX-T to NSX ALB using Migration Tool

Some pre-requisites need to be fulfilled before running the NSX-T to Avi Migration Tool. Refer to the KB section for links to AVI documentation related to those pre-requisites.

- A deployed NSX Advanced Load Balancer (Avi) Controller (for instructions, see details in the previous section) with a license key applied. The migration tool does not apply or generate license keys.
- An Avi NSX-T Cloud needs to be configured for each type of transport zone (VLAN / Overlay)
- An SE Group needs to be configured before migrating NSX-T configurations. The default SE group can be used without modifications unless you have “Preserve Client IP (no-SNAT)” use cases.

- If Virtual Services with SNAT deactivated were configured in NSX-T (a feature named Preserve Client IP by Avi), an SE Group with the HA mode configured as Active/Standby must be available to support this use case. See Preserve Client IP for NSX-T Overlay in the KB Section.
- Sizing needs to be done before moving traffic to Avi to avoid traffic disruption. See KB Section for sizing docs.
- For Overlay Segments, Avi will configure a static route, advertised as a Tier 1 load balancer route (t1l flag) to T0, on the T1 LR configured on AVI NSX Cloud Connector. Proper NSX-T routing configuration is important to allow route redistribution of load balancer-flagged routes inside of NSX-T and from NSX-T to upstream routers if using BGP at T0. See Create an AVI NSX-T Cloud on KB Section for more details
- For VLAN segments using an external router as the gateway, BGP is available to avoid using static routing. See Avi BGP Support in the KB section for more details. If BGP is used, add 'bgp_peer_configured_for_vlan' as True for 'default_params_file' migration script argument

  If static routing is used, the user will need to modify the configured next-hop on the gateway router for the Virtual IP static routes.
- The machine where the tool is planned to be executed will need to have:
  - Network access to NSX-T Manager using SSL (TCP/443) protocol.
  - Network access to NSX-T Manager using SSH (TCP/22) protocol.
  - NSX-T Manager credentials
  - NSX-T Manager root credentials (if are different than admin user)
  - Network access to NSX Advanced Load Balancer Controller using SSL (TCP/443) protocol.
  - NSX Advanced Load Balancer Controller credentials

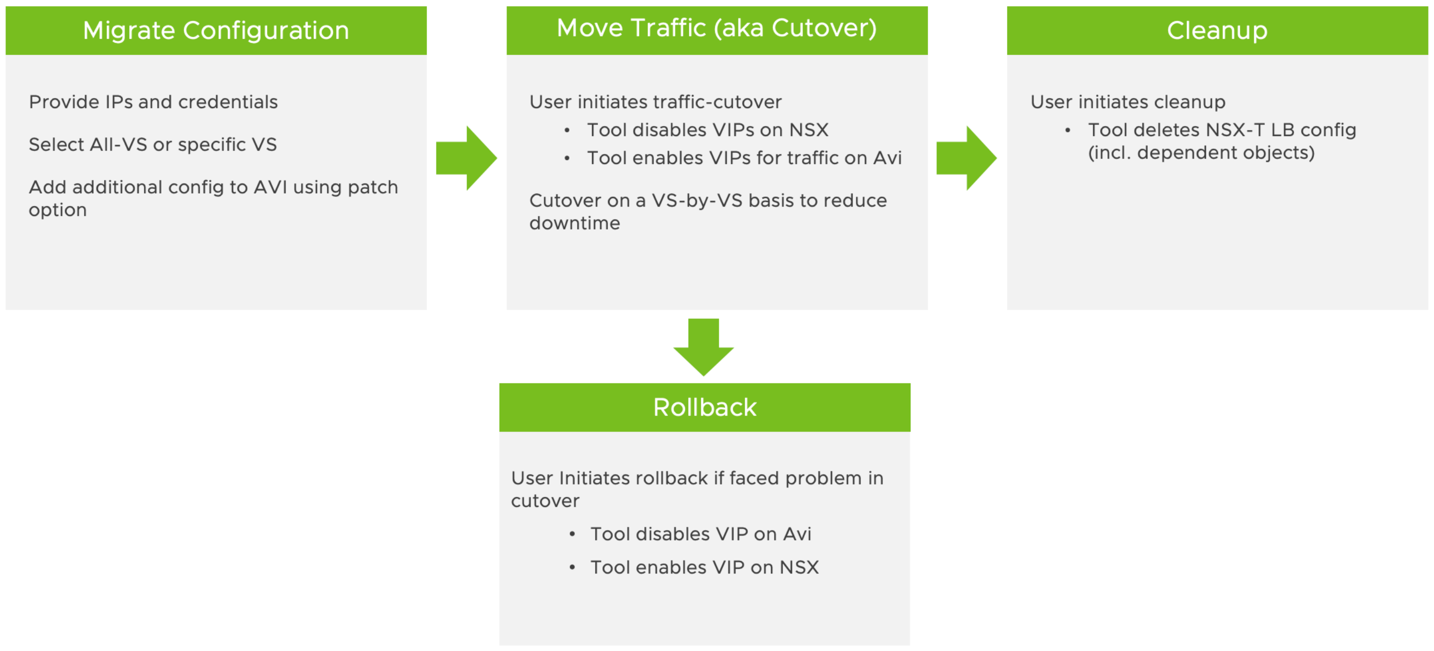
Known Caveats

Virtual Services belonging to LBs relying on T1 LR that have a combination of linked segments and service interfaces configured may not be completely migrated

User Workflows

The tool proposes a three-phase migration

- Translate configurations (Migrate)
- Migrate traffic from NSX-T LB to AVI (Cutover)
  - Rollback if something is not working as expected
- Clean residual configuration at NSX-T LB (Cleanup)


Migrate Configuration
Provide IPs and credentials
Select All-VS or specific VS
Add additional config to AVI using patch option
Move Traffic (aka Cutover)
User initiates traffic-cutover
Tool disables VIPs on NSX
Tool enables VIPs for traffic on Avi
Cutover on a VS-by-VS basis to reduce downtime
Cleanup
User initiates cleanup
Tool deletes NSX-T LB config (incl. dependent objects)
Rollback
User Initiates rollback if faced problem in cutover
Tool disables VIP on Avi
Tool enables VIP on NSX

Since the tool is a cli tool, all the scripts need to run using console. In the sake of simplicity, running the script from the Avi Controller shell is recommended if running a Controller version that has the tool embedded.

Do you have any suggestion? Feel free to contact us at avi-migration-help@groups.vmware.com

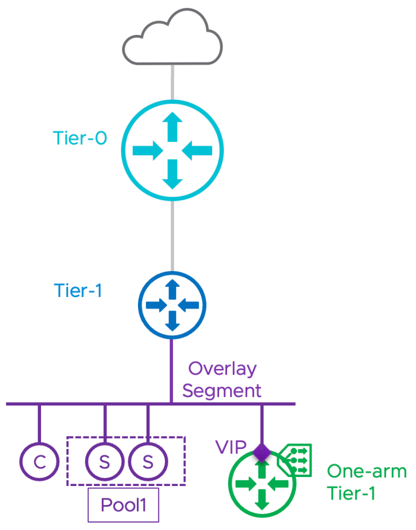
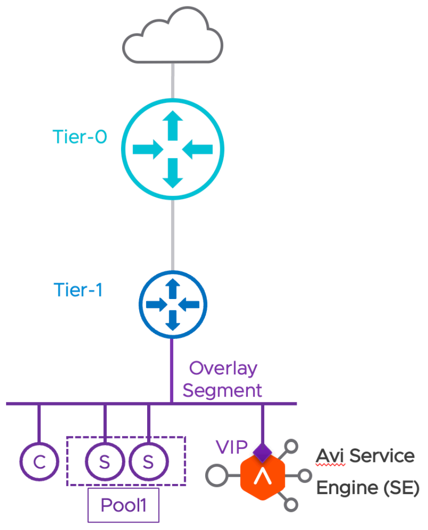
Supported Topologies

For NSX-T to Avi Migration Tool Beta version, the following topologies are supported and tested. If you have questions on different topologies, or if you would like to suggest new topologies, please contact avi-migration-help@groups.vmware.com.

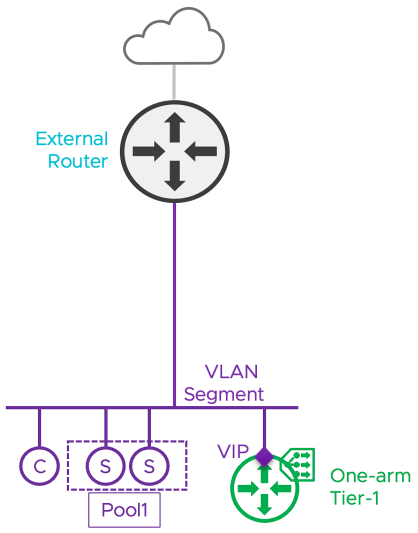
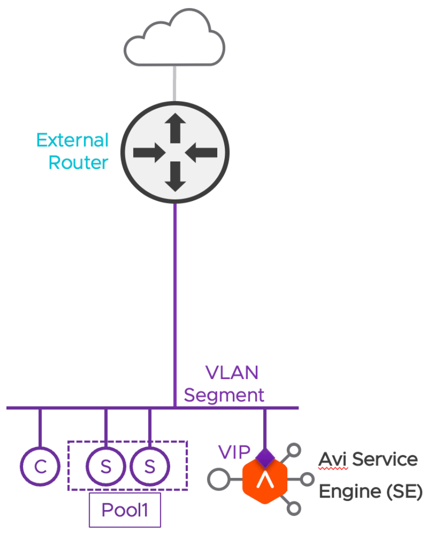
In this chapter, different topologies supported by the tool will be presented, showing on the left side how the topology is configured using NSX-T Load Balancer and how the topology will be after moving the service to AVI.

One-Arm Topologies:

- **One-Arm using a Service Interface** to connect Load Balancer inside of T1-LR to an **Overlay** segment

Topology using NSX-T Load Balancer	Topology using NSX Advanced Load Balancer
 Tier-0 Tier-1 Overlay Segment C S S Pool1 VIP One-arm Tier-1	 Tier-0 Tier-1 Overlay Segment C S S Pool1 VIP Avi Service Engine (SE)

- **One-Arm using a Service Interface** to connect Load Balancer inside of T1-LR to a **VLAN** segment.

Topology using NSX-T Load Balancer	Topology using NSX Advanced Load Balancer
 External Router VLAN Segment C S S Pool1 VIP One-arm Tier-1	 External Router VLAN Segment C S S Pool1 VIP Avi Service Engine (SE)

Note: In this topology, the user must program the data path from the external router to VIP (usually a static route to VIP or BGP peering). See Avi BGP support in the KB section for more information.

In-line topologies:

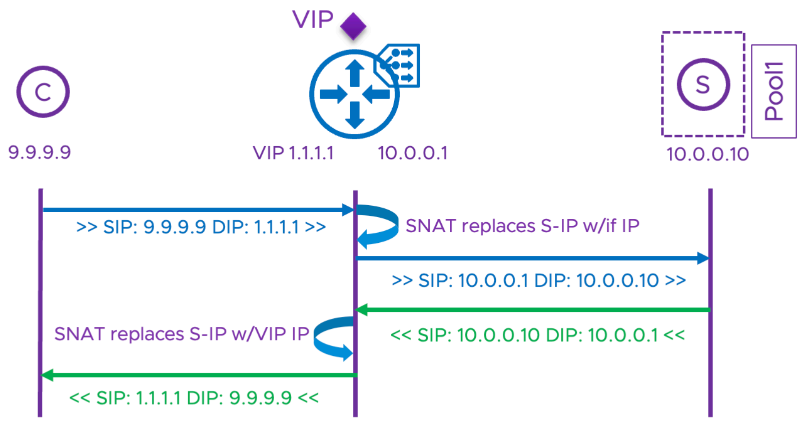
Relying on the fact that in NSX-T, T1 routers are in the data path and are used as default gateways for the members of the server pool, NSX-T was able to build in-line load balancing topologies.

When SNAT Translation is activated on the NSX-T Server Pool object, the load balancer in the T1 Router will replace the client IP address, either by T1 IP Address if Automap is selected or by a user-specified IP if “IP Pool” is selected.

Regardless of the type of SNAT configured, a new connection will be created between the Load Balancer and the server member of the Server Pool.

In the image below, a sequence diagram shows how the SNAT process replaces the client-IP when the packet flows from client to server, and server-IP is replaced with VIP when the packet flows in the opposite direction.

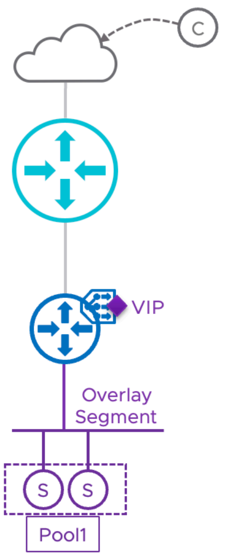
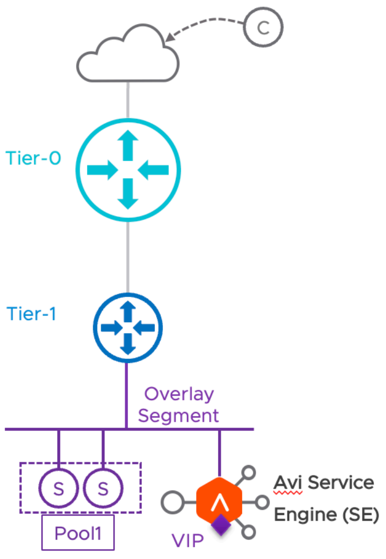
Without this last translation, the customer would receive a packet with a source IP that does not match what is expected. Besides that, if a firewall is in the traffic path and the used protocol is TCP, that firewall could find a packet flagged with SYN-ACK that does not match with any SYN packet and therefore drop the packets.


VIP
C
9.9.9.9
VIP 1.1.1.1
10.0.0.1
S
Pool1
10.0.0.10
>> SIP: 9.9.9.9 DIP: 1.1.1.1 >>
SNAT replaces S-IP w/if IP
>> SIP: 10.0.0.1 DIP: 10.0.0.10 >>
<< SIP: 10.0.0.10 DIP: 10.0.0.1 <<
SNAT replaces S-IP w/VIP IP
<< SIP: 1.1.1.1 DIP: 9.9.9.9 <<

Thanks to SNAT, the load balancer can be either in-line or one-arm and be placed in almost any network segment since only reachability is required to work. When SNAT is not activated, that is a very different story. Since the server from the pool will receive the IP packets with the original IP address in the source field, the returning packet will be crafted with that IP as the destination IP into the IP header, and the standard routing rules will be followed by that host. The packet will be relayed in layer 2 to its gateway.

Since Avi SE does not run inside of the T1 router as the NSX Load Balancer does, an in-line topology seems to be impossible to reproduce. To overcome that issue by relying on the NSX technology, Avi with NSX Cloud Connector can generate equivalent topologies even when the Avi SE is not configured as the default gateway for the servers. Bear in mind that this situation will only happen when SNAT is deactivated.

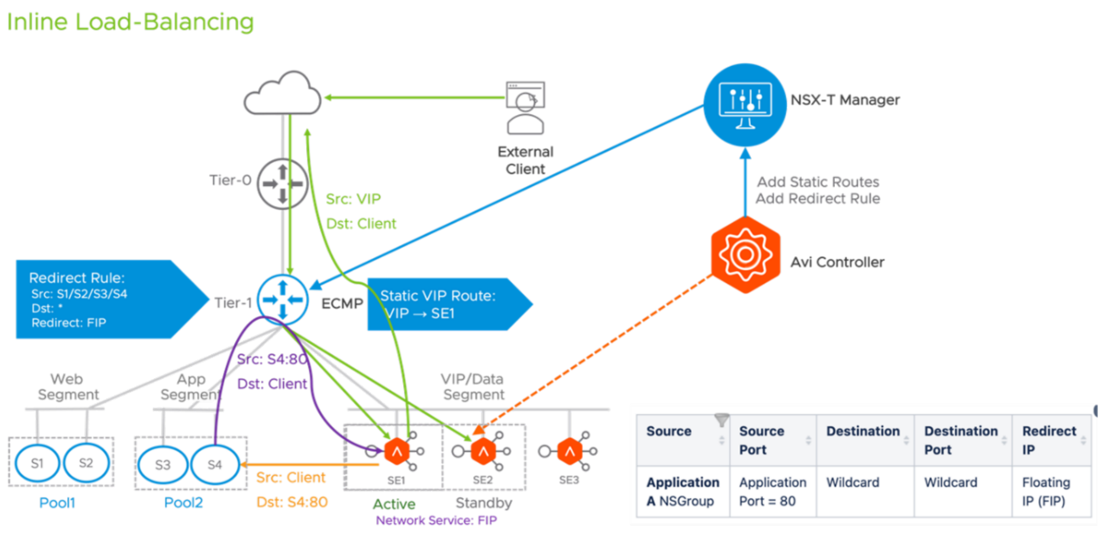
- **In-line topology with SNAT activated,** connecting a load balancer on a T1 LR to an **Overlay segment**.
    - **A client** can be either outside of NSX (N/S traffic), connected to a segment linked to a different T1 LR, connected to the same segment where server pool is, or even a different segment linked to same T1 LR. Last 2 scenarios require **mandatory SNAT** when configured in NSX-T environments.

Topology using NSX-T Load Balancer	Topology using NSX Advanced Load Balancer
 C Tier-0 Tier-1 VIP Overlay Segment S S Pool1	 C Tier-0 Tier-1 Overlay Segment S S Pool1 VIP Avi Service Engine (SE)

- In-line topology with SNAT deactivated, connecting a load balancer on a T1 LR to an Overlay segment

This topology relies on the “Preserve Client IP for NSX-T Overlay” configuration. More details can be found on the KB Section.

Detail of Traffic Flow for the returning traffic, from Server to Client (See purple flow):



Source	Source Port	Destination	Destination Port	Redirect IP
Application A NSGroup	Application Port = 80	Wildcard	Wildcard	Floating IP (FIP)

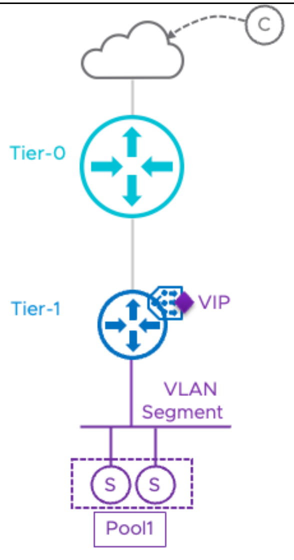
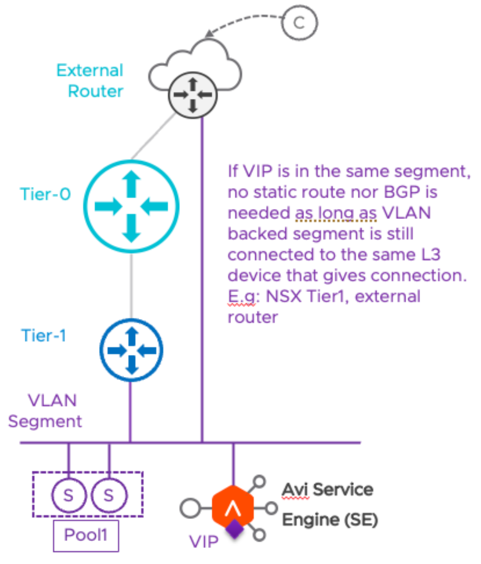
- **In-line topology with SNAT activated,** connecting a load balancer on a T1 LR and linked to a **VLAN segment**.
  - **A Client** can be either outside of NSX (N/S traffic), connected to a segment linked to a different T1 LR, connected to the same segment where the server pool is, or even a different segment linked to the same T1 LR. The last 2 scenarios require **mandatory SNAT** when configured in NSX-T environments.

Some NSX Background, to connect a T1 LR to a VLAN segment, a Service Interface is created and used on the T1 LR

When a VIP is migrated to Avi, the controller automatically connects a service engine to the corresponding VLAN segment.

If the VIP IP is part of the same subnet from the VLAN segment, no extra steps are needed, and ARP will take care of the data path. Depending on the scenario, the T1 LR or the external router will have an ARP entry for the VIP IP.

If the VIP IP is not part of the same subnet, static routes are needed to point the VIP IP to the Avi SE Interface. A BGP Session can be established as a more dynamic alternative. See the KB section for more information about BGP at Avi.

Topology using NSX-T Load Balancer	Topology using NSX Advanced Load Balancer
 C Tier-0 Tier-1 VIP VLAN Segment S S Pool1	 C External Router If VIP is in the same segment, no static route nor BGP is needed as long as VLAN backed segment is still connected to the same L3 device that gives connection. E.g: NSX Tier1, external router Tier-0 Tier-1 VLAN Segment S S Pool1 VIP Avi Service Engine (SE)

- **In-line topology with SNAT deactivated** connecting a load balancer on a T1 LR and linked to a **VLAN segment**.

To simulate the inline behavior with a one-arm load balancer (like Avi SE), **Preserve client IP** must be enabled to avoid translating the source address (SNAT), and a floating IP needs to be configured on Avi SEs to receive returning traffic from the pool's servers (to act as gateway).

For In-line topologies using VLAN segments, the redirection used for inline overlay scenarios can't be configured because of a limitation of NSX-T (See NSX-T Requirements for East-West Network Redirection in KB section).

This means that NSX-T to Avi Migration Tool can move the load balancer configuration from NSX-T to Avi but can't redirect the returning traffic to the Avi SE. To get this topology working, the server's default gateway must be pointed to the Avi SE Floating IP.

As with the last scenario, when the VIP is migrated to Avi, the controller automatically connects a service engine to the corresponding VLAN segment and **does not add** any route into the NSX T1 router.

If the VIP IP is part of the same subnet from the VLAN segment, no extra steps are needed, and ARP will take care of the data path. Depending on the scenario, the T1 LR or the external router will have an ARP entry for the VIP IP.

If the VIP IP is not part of the same subnet, static routes are needed to point the VIP IP to the Avi SE Interface. A BGP Session can be established as a more dynamic alternative. See the KB section for more information about BGP at Avi.

How to run

Migrate Configuration

To migrate the configuration from NSX-T to AVI, run the script called nsxt_converter.py. Be prepared to use the following information as arguments. Password can be passed as arguments but setting those as env variables are recommended.

- --nsxt_ip
- --nsxt_user
- --nsxt_password (the password needs to be provided either here or as an env variable)
- --alb_controller_ip
- --alb_controller_user
- --alb_controller_password (the password needs to be provided either here or as an env variable)

Linux / Mac example: (passwords set using env variables):

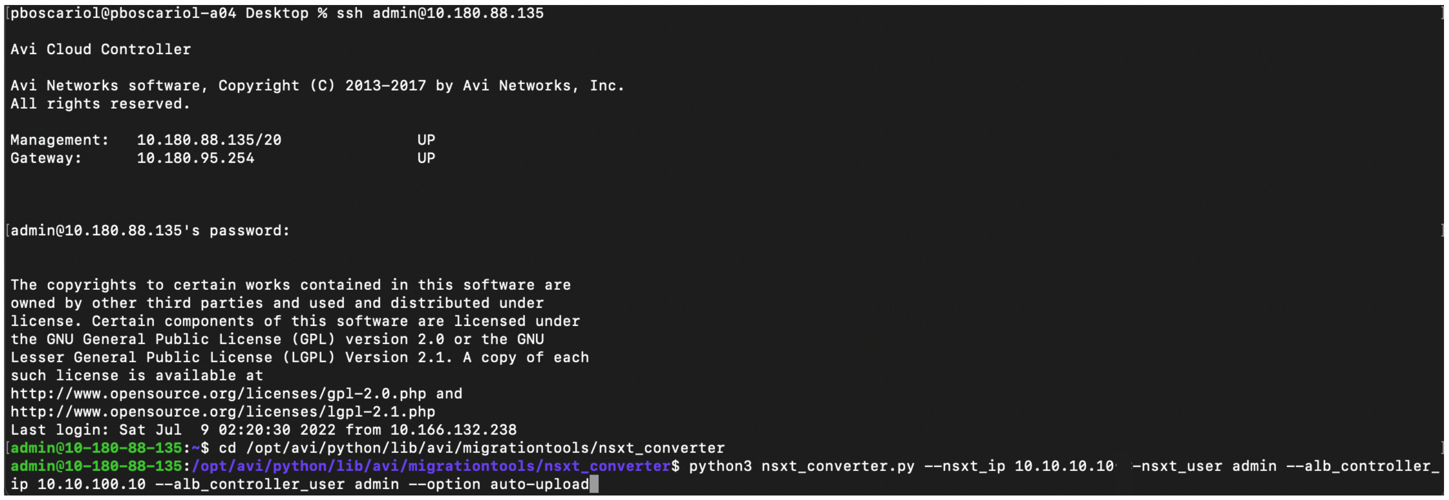
export alb_controller_password=AviPassword123!

export nsxt_password=NSXPassword123!

python3 nsxt_converter.py **--nsxt_ip** 10.10.10.10 **--nsxt_user** admin **--alb_controller_ip** 10.10.100.10 **--alb_controller_user** admin --option auto-upload

Running the Migration tool.

1. Go to the directory where the alb-sdk tool is located and change the directory to the nsxt_converter directory:
    a. If running from Avi Controller:
        i. Connect to the controller using an SSH client and authenticate as 'admin'
        ii. Change directory to tool location: cd /opt/avi/python/lib/avi/migrationtools/nsxt_converter
    b. If running the tool from your own device:
        i. Windows: open CMD and go to the directory where you stored the Avi SDK
        ii. Linux/Mac: open terminal and go to the directory where you stored the Avi SDK.
2. Set env variables with passwords (recommended)
3. Based on your specific needs, plan the arguments and values needed and run the script using python3
    a. python3 nsxt_converter.py <args>



```
[pboscariol@pboscariol-a04 Desktop % ssh admin@10.180.88.135

Avi Cloud Controller

Avi Networks software, Copyright (C) 2013-2017 by Avi Networks, Inc.
All rights reserved.

Management:   10.180.88.135/20               UP
Gateway:      10.180.95.254                  UP



[admin@10.180.88.135's password:


The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php
Last login: Sat Jul  9 02:20:30 2022 from 10.166.132.238
[admin@10-180-88-135:~$ cd /opt/avi/python/lib/avi/migrationtools/nsxt_converter
admin@10-180-88-135:/opt/avi/python/lib/avi/migrationtools/nsxt_converter$ python3 nsxt_converter.py --nsxt_ip 10.10.10.10 -nsxt_user admin --alb_controller_
ip 10.10.100.10 --alb_controller_user admin --option auto-upload
```

After running, the tool will deposit output files inside of the folder <tool-path>/output/<NSX-T Mgt IP>/

Example:

```
.
├── 10.180.84.202
│   ├── input
│   │   └── config.json
│   └── output
│       ├── <NSX-T Manager IP>_discovery_data.xlsx
│       ├── avi_config.json
│       ├── avi_config_create_object.yml
│       ├── avi_config_delete_object.yml
│       ├── nsxt-report-ConversionStatus.xlsx
│       └── state.json
└── converter.log
```

config.json: JSON file containing the configuration pulled from NSX-T Manager.

<NSX-T Manager IP>_discovery_data.xlsx: Excel file containing a summary of Virtual Servers and Pools pulled from NSX-T, and their status

avi_config.json: JSON file containing translated configuration

avi_config_create_object.yml: Ansible playbook with the Avi configuration that can be used to upload the configuration to Avi controller.

avi_config_delete_object.yml: Ansible playbook with the Avi configuration that can be used to remove the configuration from Avi controller.

state.json: JSON file containing the arguments received when the script was executed (Does not store passwords)

nsxt-report-ConversionStatus.xlsx: Excel file that shows a report of all the processed objects and their translation status (SUCCESSFUL, PARTIAL, NOT_IN_USE, SKIPPED, ERROR). Objects not marked as SUCCESSFUL should be reviewed in case those needs manual configuration to get a working configuration.

Traffic Cutover

After uploading configuration to the Avi controller and verifying that all Virtual Services migrated are ready to receive traffic, another script is available to smoothly disable services in NSX-T and enable recently migrated services into Avi.

This tool takes care of that traffic migration by detaching the load balancer from the NSX-T Virtual Server and switching the Virtual Server to Admin State = Disabled, and then enabling the "Traffic enabled" checkbox from VS configuration in Avi.

This change triggers NSX-T Cloud connector in Avi to communicate with NSX-T to do the needed plumbing to get traffic pointed to VS configured on the Avi SE or SEs. This process is done per-vs to ensure the same downtime to all the migrated Virtual Services.

Note: For Virtual Services using NSX-T VLAN Cloud Connector, neither Avi nor the NST-T to Avi migration tool does configurations on the external router/routers. As was mentioned in the Topologies section, manual intervention is needed to point traffic to the right Service Engine.

Running the Traffic Cutover tool

This script directly reads the file state.json generated by the migration script and located into the output folder in order to save you having to complete some of the arguments. The script will get that information from the JSON file.

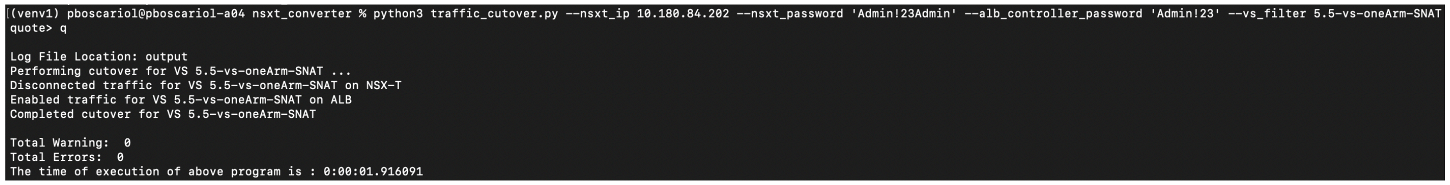
Mandatory Arguments:

- --alb_controller_password
- --vs_filter
- --nsxt_ip
- --nsxt_password

1. Go to the directory where alb-sdk tool is located and change directory to the nsxt_converter directory:
    a. If running from Avi Controller:
        i. Connect to the controller using an SSH client and authenticate as 'admin'
        ii. Change directory to tool location: cd /opt/avi/python/lib/avi/migrationtools/nsxt_converter
    b. If running the tool from your own device:
        i. Windows: open CMD and go to the directory where you stored the Avi SDK
        ii. Linux/Mac: open terminal and go to the directory where you stored the Avi SDK.
2. Based on your specific needs, plan the arguments and values needed and run the script using python3
    a. python3 <args>

Example:

```
python3 traffic_cutover.py --nsxt_ip 10.180.84.202 --nsxt_password 'Password123!' --alb_controller_password
    'Password123!' --vs_filter 5.5-vs-oneArm-SNAT
```



```
[(venv1) pboscariol@pboscariol-a04 nsxt_converter % python3 traffic_cutover.py --nsxt_ip 10.180.84.202 --nsxt_password 'Admin!23Admin' --alb_controller_password 'Admin!23' --vs_filter 5.5-vs-oneArm-SNAT
quote> q

Log File Location: output
Performing cutover for VS 5.5-vs-oneArm-SNAT ...
Disconnected traffic for VS 5.5-vs-oneArm-SNAT on NSX-T
Enabled traffic for VS 5.5-vs-oneArm-SNAT on ALB
Completed cutover for VS 5.5-vs-oneArm-SNAT

Total Warning:  0
Total Errors:  0
The time of execution of above program is : 0:00:01.916091
```

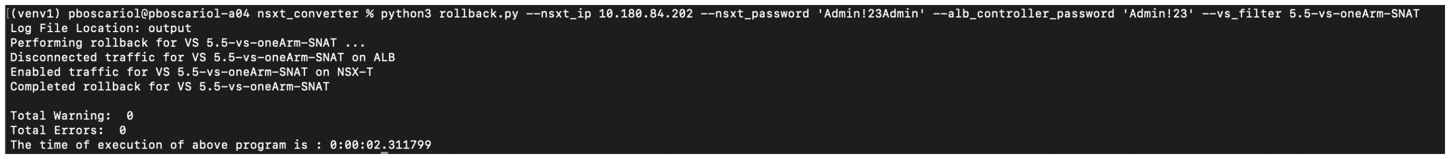
Rollback

The Rollback tool is available to run the opposite process than Traffic Cutover does. This means that the tool will uncheck "Traffic Enabled" checkbox in Avi VS, then re-connect the VS to the LB in NSX-T and activate the Admin Status knob. This work is done in the same way as Traffic Cutover does, on a per-VS basis.

Note: For Virtual Services using NSX-T VLAN Cloud Connector, neither Avi nor the NST-T to Avi migration tool does configurations on the external router/routers. As was mentioned in the Topologies section, manual intervention is needed to point traffic to the right Service Engine.

Example:

python3 rollback.py --nsxt_ip 10.180.84.202 --nsxt_password 'Password123!' --alb_controller_password 'Password123!' --vs_filter 5.5-vs-oneArm-SNAT



```
[(venv1) pboscariol@pboscariol-a04 nsxt_converter % python3 rollback.py --nsxt_ip 10.180.84.202 --nsxt_password 'Admin!23Admin' --alb_controller_password 'Admin!23' --vs_filter 5.5-vs-oneArm-SNAT
Log File Location: output
Performing rollback for VS 5.5-vs-oneArm-SNAT ...
Disconnected traffic for VS 5.5-vs-oneArm-SNAT on ALB
Enabled traffic for VS 5.5-vs-oneArm-SNAT on NSX-T
Completed rollback for VS 5.5-vs-oneArm-SNAT

Total Warning:  0
Total Errors:  0
The time of execution of above program is : 0:00:02.311799
```

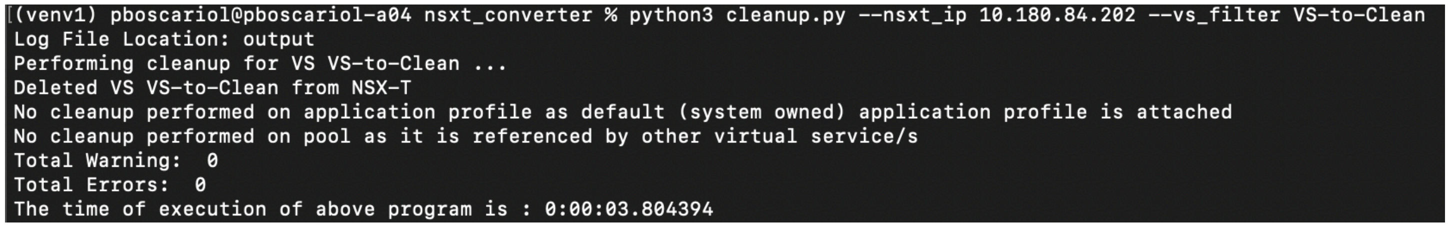
Cleanup

The cleanup tool is available to aid users in cleaning the residual configuration in NSX-T after configuration migration and traffic cutover. It is highly recommended to keep configurations in NSX-T until user acceptance has completed and rollback will be needed.

All objects connected to the mentioned Virtual Server will be cleaned only if they are not being used by other services.

Example:

python3 cleanup.py --nsxt_ip 10.180.84.202 --vs_filter VS-to-Clean



```
[(venv1) pboscariol@pboscariol-a04 nsxt_converter % python3 cleanup.py --nsxt_ip 10.180.84.202 --vs_filter VS-to-Clean
Log File Location: output
Performing cleanup for VS VS-to-Clean ...
Deleted VS VS-to-Clean from NSX-T
No cleanup performed on application profile as default (system owned) application profile is attached
No cleanup performed on pool as it is referenced by other virtual service/s
Total Warning:  0
Total Errors:  0
The time of execution of above program is : 0:00:03.804394
```

Additional References

Git and Python references

Example – Cloning avi-sdk GitHub repository

From a user defined directory, run the following command to clone the repository. The repo will be cloned on that directory.

```
user@avinetworks AviNetworks % git clone https://github.com/vmware/alb-sdk.git
Cloning into 'alb-sdk'...
remote: Enumerating objects: 95282, done.
remote: Counting objects: 100% (1614/1614), done.
remote: Compressing objects: 100% (280/280), done.
remote: Total 95282 (delta 1099), reused 1547 (delta 1089), pack-reused 93668
Receiving objects: 100% (95282/95282), 147.03 MiB | 10.99 MiB/s, done.
Resolving deltas: 100% (69481/69481), done.
user@avinetworks AviNetworks %
```

Example - Updating pip

```
user@avinetworks % pip3 install --upgrade pip
       Collecting pip
         Using cached pip-22.2.2-py3-none-any.whl (2.0 MB)
       Installing collected packages: pip
         Attempting uninstall: pip
           Found existing installation: pip 20.2.3
           Uninstalling pip-20.2.3:
             Successfully uninstalled pip-20.2.3
       Successfully installed pip-22.2.2
```

Example – Installing avimigrationtools pip package

```
user@avinetworks % pip3 install avimigrationtools
       Collecting avimigrationtools
         Using cached avimigrationtools-22.1.1.post3.tar.gz (530 kB)
         Preparing metadata (setup.py) ... done
       /////====== Output omitted ======/////
       Successfully installed ConfigParser-5.2.0 MarkupSafe-2.1.1 appdirs-1.4.4 argparse-1.4.0
       avimigrationtools-22.1.1.post3 avisdk-22.1.1.post3 bcrypt-3.2.2 certifi-2022.6.15 cffi-1.15.1
       charset-normalizer-2.1.0 cryptography-37.0.4 ecdsa-0.18.0 et-xmlfile-1.1.0 idna-3.3 jinja2-
       3.1.2 linecache2-1.0.0 networkx-2.8.5 nose-1.3.7 nose-html-reporting-0.2.3 nose-testconfig-
       0.10 numpy-1.23.2 openpyxl-3.0.10 pandas-1.4.3 paramiko-2.11.0 pexpect-4.8.0 ptyprocess-0.7.0
       pyOpenssl-22.0.0 pycparser-2.21 pycrypto-2.6.1 pynacl-1.5.0 pyparsing-3.0.9 python-dateutil-
       2.8.2 pytz-2022.2.1 pyyaml-6.0 requests-2.28.1 six-1.16.0 traceback2-1.4.0 unittest2-1.1.0
       urllib3-1.26.11 xlsxwriter-3.0.3 xmltodict-0.13.0
```

Example – Installing vapi and nsx SDKs from a custom directory

Place all the *.whl files to be installed into the same folder.

```
user@avinetworks % ls VMware-SDK-Files
user@avinetworks % tree VMware-SDK-Files
       VMware-SDK-Files
       ├── nsx_policy_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
       ├── nsx_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
```

```
        ├── vapi_common_client-2.29.0-py2.py3-none-any.whl
        └── vapi_runtime-2.29.0-py2.py3-none-any.whl
        0 directories, 4 files

## WARNING # pip will install any *whl file ##

Change to the directory where the whl files are located (in this example is called “VMware-SDK-
Files”)
user@avinetworks % cd VMware-SDK-Files
user@avinetworks VMware-SDK-Files % pip3 install *.whl
Processing ./nsx_policy_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
Processing ./nsx_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
Processing ./vapi_common_client-2.29.0-py2.py3-none-any.whl
Processing ./vapi_runtime-2.29.0-py2.py3-none-any.whl
        /////====== Output omitted ======/////
Successfully installed nsx_policy_python_sdk-4.0.0.1.0-py2.py3-none-any.whl nsx_python_sdk-4.0.0.1.0-
py2.py3-none-any.whl vapi_common_client-2.29.0-py2.py3-none-any.whl vapi_runtime-2.29.0-py2.py3-none-
any.whl
```

Example – Installing vapi and nsx SDKs from the cloned repository

```
Change to the directory where the repository was cloned.
Change to the directory alb-sdk/python/avi/migrationtools/lib/

user@avinetworks lib % pip3 install *.whl
Processing ./nsx_policy_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
Processing ./nsx_python_sdk-4.0.0.1.0-py2.py3-none-any.whl
Processing ./vapi_common_client-2.29.0-py2.py3-none-any.whl
Processing ./vapi_runtime-2.29.0-py2.py3-none-any.whl
        /////====== Output omitted ======/////
Successfully installed nsx_policy_python_sdk-4.0.0.1.0-py2.py3-none-any.whl nsx_python_sdk-4.0.0.1.0-
py2.py3-none-any.whl vapi_common_client-2.29.0-py2.py3-none-any.whl vapi_runtime-2.29.0-py2.py3-none-
any.whl
```

Available Arguments

Arguments marked with a red star (*) are mandatory

Migration Tool

- --help

  Displays help on the console. Shows syntax help, provides examples, and details about how to use arguments.

- --ansible

  When activated, creates an ansible file as part of the output

- --ansible_skip_types

  Accepts a comma-separated list of objects desired to be skipped during conversion

Example:

“--ansible_skip_types DebugController,ServiceEngineGroup” will skip debugcontroller and serviceengine objects

- **--ansible_filter_types**

  Accepts a comma-separated list of AVI object types to include during conversion.

  Example:

  --ansible_filter_types VirtualService, Pool will do ansible conversion only for Virtualservice and Pool objects

- **--alb_controller_ip (*)**

  Use this argument to tell the tool the IP of the NSX ALB (AVI) Controller.

  Example:

  --alb_controller_ip 1.2.3.4

- **--alb_controller_version**

  Indicates to the tool which version of NSX ALB is being used. Default value is 21.1.4

  Example:

  --alb_controller_version 21.1.4

- **--alb_controller_user (*)**

  Indicates to the tool the username to use to access NSX ALB controller.

  Example:

  --alb_user admin

- **--alb_controller_password**

  Indicates to the tool the password to use to get access to NSX ALB controller.

  This parameter can be set using environment variable “alb_controller_password”

  Example:

  --alb_password 'Password123!'

  -- or --

  export alb_controller_password=Password123! (Set the env variable)

  echo $alb_controller_password (read the env variable)

- **--alb_controller_tenant**

  Indicates the tool on which tenant to create the configurations. The default value is “admin”

- **--cloud_tenant**

  Indicates the tenant’s name where the NSX-t cloud is created. The default value if not set is “admin”

- --default_params_file

  Indicates the tool the absolute path for NSX-T default params file. See usage examples for more details.

- --nsxt_ip **(*)**

  Use this argument to tell the tool the IP of the **NSX-T Manager**.

  Example:

  ```
  --nsxt_ip 10.180.84.202
  ```

- --nsxt_user **(*)**

  Indicates to the tool the username to use to get access to **NSX-T Manager**

  Example:

  ```
  --nsxt_user admin
  ```

- --nsxt_password

  Indicates to the tool the password to use to get access to **NSX-T Manager.**

  This parameter can be set using environment variable "nsxt_password"

  Example:

  ```
  --nsxt_password 'Password123!'
  ```

  -- or --

  ```
  export nsxt_password='Password123!' (Set the env variable)
  echo $nsxt_password (read the env variable)
  ```

- --nsxt_port

  Indicates the tool which port should use to connect with NSX-T Manager. The default value is 443.

  Example:

  ```
  --nsxt_port 443
  ```

- --not_in_use

  Indicates to the tool that an object is not in use

- --no_object_merge

  Indicates to the tool to avoid merging duplicated objects

- --output_file_path

  Indicates the output folder where the user desires the tool to place output files.

- --option {cli-upload,auto-upload}

  Indicates the tool if the user desires to have the config auto uploaded to NSX ALB, or just have the config generated and placed into the output folder. No need to put the selected option between quotes.

  Example:

  ```
  --option auto-upload
  ```

- --patch PATCH

  Patch is a tool that allows the user to add additional configuration to the NSX ALB generated configuration, at the time that is generating / applying the translated configuration from NSX-T.

  The file with the additional configuration needs to be formatted using YAML, and the expected filename is patch.yaml

  Example:

  ```
  nsxt_converter.py --patch test/patch.yaml
  ```

- --prefix

  Indicates the tool to add the specified prefix to the objects created on NSX ALB.

  As an example, if we set "legacy" as a prefix,

  Example:

  ```
  --prefix legacy
  ```

  With this setting, a VS called "Frontend-App-X" on NSX will be called "legacy-Frontend-App-X" after being migrated to NSX ALB (AVI).

- --segroup

  Indicates the tool to use a custom SE Group. The Default value is Default-Group

- -- ssh_root_password

  Indicates the tool to use a specific password to connect to NSX-T Manager using SSH.

  This parameter can be set using environment variable "ssh_root_password"

  Example:

  ```
  --ssh_root_password 'Password123!'
  ```

  -- or --

  ```
  export ssh_root_password=Password123! (Set the env variable)
  echo $ssh_root_password (read the env variable)
  ```

- --traffic_enabled

  Indicates the tool to enable traffic on all the VS after being deployed into NSX ALB. The default is FALSE.

- --vs_filter

  Accepts a comma-separated list of VS that wants to be migrated. Any VS not listed will **not** be migrated to NSX ALB.

  Note: If vs_filter is used, and the Patch file modifies VS Name, VS names in vs-filter should match the new name.

  Example:

  ```
  --vs_filter test_vs
  ```

- --vs_level_status

  Indicates the tool to add columns of VS reference and Overall Skipped settings in the Status Excel Sheet delivered as output.

- --vs_state {enable,deactivate}

  Defines the default state of the virtual services configured at Avi Controller after conversion. The default is "disabled" (deactivate).

Usage Examples

- Using --option to auto upload config to the controller after conversion:

  ```
  nsxt_converter.py --option auto-upload
  ```

- Using --vs_state option to set the desired state of a VS after conversion to Avi (default value is disable).:

  ```
  nsxt_converter.py --vs_state enable
  ```

- Using --alb_controller_version option to provide controller version for getting output in respective controller format.

  ```
  nsxt_converter.py --alb_controller_version 21.1.4
  ```

- Using --no_object_merge option to avoid tool merging two same objects (based on their attribute values except name) :

  ```
  nsxt_converter.py --no_object_merge
  ```

- Using --patch to add additional configuration to Avi on top of converted configuration:

  ```
  nsxt_converter.py --patch test/patch.yaml
  ```

Example of the patch.YAML file to change pool's name:

```
Pool:
  - match_name_regex: POOL_1
    patch:
      name: Pool_1
```

- Using --vs_filter to filter the VSes to migrate from NSX-T to Avi

  ```
  nsxt_converter.py --vs_filter test_vs
  ```

- Using --ansible_skip_types to skip to **exclude** Avi objects from the tool-created Ansible file. Argument receives a CSV list of Avi Object Types to skip during conversion

  ```
  nsxt_converter.py --ansible --ansible_skip_types DebugController, ServiceEngineGroup
  ```

- Using --ansible_filter_types to **only include** Avi objects into the tool-created Ansible file. Argument receives a CSV list of Avi Object Types to skip during conversion

  ```
  nsxt_converter.py --ansible --ansible_filter_types virtualservice, pool
  ```

- Using --prefix to add a prefix to the name of objects converted to Avi. Applicable when adding multiple configurations to the same controller, to identify the source.

  ```
  nsxt_converter.py --prefix abc
  ```

- Using --not_in_use to remove objects that are not in use

  ```
  nsxt_converter.py --not_in_use
  ```

- Using --vs_level_status to add the VS level status into the output's excel sheet.

  ```
  nsxt_converter.py --vs_level_status
  ```

- Using --segroup to add or change SE Group used by VS

  ```
  nsxt_converter.py --segroup segroup_name
  ```

- Using --default_params_file to set default parameters to the migration. The file's expected format is JSON.

  ```
  nsxt_converter.py --default_params_file test/default_params.json
  ```

Example file:

```
    {
   "bgp_peer_configured_for_vlan": true,
   "network_service": {
     "no-OneArm-PB": "1.2.3.4",
     "T1_PB-DATA-Migration-Test-floating-ip": "192.168.223.160"
   }
 }
```

Cutover Tool

Arguments marked with a red star (*) are mandatory

- **--alb_controller_ip**

  Use this argument to tell the tool the IP of the NSX ALB (AVI) Controller.

  Example:

  ```
  --alb_controller_ip 1.2.3.4
  ```

- **--alb_controller_version**

  Indicates to the tool which version of NSX ALB is being used.

  Example:

  ```
  --alb_controller_version 21.1.4
  ```

- **--alb_controller_user**

  Indicates to the tool the username to use to get access to the NSX ALB controller.

  Example:

  ```
  --alb_user admin
  ```

- **--alb_controller_password**

  Indicates to the tool the password to use to get access to the NSX ALB controller.

  This parameter can be set using the environment variable “alb_controller_password”

  Example:

  ```
  --alb_password 'Password123!'
  ```

  -- or --

  ```
  export alb_controller_password=Password123! (Set the env variable)
  echo $alb_controller_password (read the env variable)
  ```

- --nsxt_ip **(*)**

  Use this argument to tell the tool the IP of the **NSX-T Manager**.

Example:

--nsxt_ip 10.180.84.202

- --nsxt_user

  Indicates to the tool the username to use to get access to **NSX-T Manager**

  Example:

  --nsxt_user admin

- --nsxt_password

  Indicates to the tool the password to use to get access to **NSX-T Manager.**

  This parameter can be set using the environment variable "nsxt_password"

  Example:

  --nsxt_password 'Password123!'

  -- or --

  export nsxt_password='Password123!' (Set the env variable)

  echo $nsxt_password (read the env variable)

- --nsxt_port

  Indicates the tool which port should use to connect with NSX-T Manager.

  Example:

  --nsxt_port 443

- --output_file_path

  Indicates the output folder in which the user desires the tool to place output files into.

- --vs_filter (*)

  Accepts a comma-separated list of VS that wants to be migrated. Any VS not listed will **not** be migrated to NSX ALB.

  Note: If vs_filter is used, and the Patch file modifies VS Name, VS names in vs-filter should match the new name.

  Example:

  --vs_filter test_vs

Cleanup Tool

Arguments marked with a red star (*) are mandatory

- --nsxt_ip (*)

  Use this argument to tell the tool the IP of the **NSX-T Manager**.

  Example:

  --nsxt_ip 10.180.84.202

- --nsxt_user

  Indicates to the tool the username to use to get access to **NSX-T Manager**

  Example:

  --nsxt_user admin

- --nsxt_password

  Indicates to the tool the password to use to get access to **NSX-T Manager.**

  This parameter can be set using the environment variable "nsxt_password"

  Example:

  --nsxt_password 'Password123!'

  -- or --

  export nsxt_password='Password123!' (Set the env variable)

  echo $nsxt_password (read the env variable)

- --nsxt_port

  Indicates the tool which port should use to connect with NSX-T Manager.

  Example:

  --nsxt_port 443

- --output_file_path

  Indicates the output folder into which the user desires the tool to place output files.

- --vs_filter **(*)**

  Accepts a comma-separated list of VS that wants to be migrated. Any VS not listed will **not** be migrated to NSX ALB.

  Note: If vs_filter is used, and the Patch file modifies VS Name, VS names in vs-filter should match the new name.

  Example:

  --vs_filter test_vs

Acronyms

PSO: Professional Services Offering

SE: NSX Advanced Load Balancer (Avi) Service Engine

SEG: NSX ALB Service Engine Group

ALB: Advanced Load Balancer

T1 LR: T1 Logical Router

KB Articles

Ports and Protocols: https://ports.esp.vmware.com/home/NSX-Advanced-Load-Balancer

Deploy Avi Controller: https://avinetworks.com/docs/21.1/avi-nsx-t-integration/#deploying-the-avi-controller-ova

Deploy Avi Controller from NSX Manager UI: https://docs.vmware.com/en/VMware-NSX-T-Data-Center/3.2/installation/GUID-3D1F107D-15C0-423B-8E79-68498B757779.html

Users and role requirements: https://avinetworks.com/docs/21.1/avi-nsx-t-integration/#configuring-nsx-t-role-requirements

Configure a T1 LR and a segment for AVI SE Management: https://avinetworks.com/docs/21.1/avi-nsx-t-integration/#configuring-management-networking-for-se

Create an Avi NSX-T Cloud: https://avinetworks.com/docs/21.1/avi-nsx-t-integration/#creating-an-nsx-t-cloud

Service Engine Groups (SE Groups): https://avinetworks.com/docs/21.1/configuration-guide/infrastructure/#service-engine-group

Preserve Client IP for NSX-T Overlay: https://avinetworks.com/docs/21.1/preserve-client-ip-nsxt-overlay/

Sizing Service Engines: https://avinetworks.com/docs/21.1/sizing-service-engines/

Avi BGP Support: https://avinetworks.com/docs/21.1/bgp-support-for-virtual-services/

Preserve Client IP for NSX-T Overlay: https://avinetworks.com/docs/21.1/preserve-client-ip-nsxt-overlay/

NSX-T Requirements for East-West Network Redirection: https://docs.vmware.com/en/VMware-NSX-T-Data-Center/3.2/administration/GUID-21F88A86-DC1B-4AF9-B12B-0665D1A61285.html#requirements-for-eastwest-network-introspection-0

FAQ

Are VMware PSO (Professional Services) included with the tool?

No, this tool does not include PSO credits to be used with. This is an open-source tool that is free to use.

Does the tool require PSO to be used?

No, this is free to use but working with VMware professional services is highly recommended.

Should I use the tool directly on my prod environment?

You certainly can, but we recommend familiarizing yourself with the tool in a lab environment before going to prod.

Is this tool supported under my support contract?

No. This tool is an open-source tool and is not currently being supported by the usual VMware support channels.

Do I still need PSO now that this tool is available?

No, but working with VMware professional services is highly recommended.

Do I need to buy products to get the rights to use the tool?

No, but access to VMware Customer Connect is required to download the NSX-T SDK from VMware's website. As an alternative, the whl files were uploaded to the GitHub repository. Check the KB section for more information about how to download and install.

How can I suggest changes/enhancements to this tool?

Contact us at avi-migration-help@groups.vmware.com.