White Paper: **June 2022**

How to migrate from legacy load balancers to NSX Advanced Load Balancer (Avi)

Proven migration tools and professional services simplify the process to modernize load balancers

vmware®

Table of contents

Introduction

Applications are the currency of digital business, and enterprises are committed to speeding up their ability to roll out new apps and updates. Legacy load balancing architectures are rigid, hard to manage and automate, and not designed for the cloud and containers. With a software-defined architecture, load balancers are finally aligned with the speed & agility requirements placed on your team. This document outlines the procedure to migrate from an existing F5 load balancing to Avi using the migration tools package. It is divided into the following sections:

1. Introduction to the migration tools package
2. Migration process overview
3. Migration tools usage with various use cases
4. Cutover practices
5. Tools/Utilities to leverage for a successful migration
6. Other migration related resources
7. Migration FAQs

Overview

Avi Overview

Avi delivers multi-cloud application services including a software load balancer, intelligent web application firewall, and container ingress services. The Avi platform ensures fast, scalable, and secure application experience with consistent capabilities in any private or public cloud. The modern load balancing architecture provides cloud-native automation based on machine learning and end-to-end observability needed to help bring applications into enterprise production environments. Avi helps customers including 25% of Fortune 50 companies accelerate to multi-cloud while delivering applications across data centers and clouds with 90% faster provisioning and 50% lower total cost of ownership.

Legacy LB to Avi Config Converter Tool

Avi migration tools is a python package that provides an automated method of migrating from existing load balancing solution to the Avi load balancing solution. The package contains

1. Config conversion tool - To facilitate migration from F5 LTM and/or NetScaler ADC based deployments to Avi
2. Config patch utility - To facilitate bulk configuration modifications to Avi objects
3. VS filter utility - Useful for filtering VS and its related configuration from the converter configuration file in json format

F5 config converter takes input from an F5's bigip.conf file, and outputs Avi configuration in JSON format, which can be uploaded to Avi Controller for migrating applications and settings.

There are two ways to input F5 configuration

1. Command Line: Parse the bigip.conf file directly (Does not include certs and keys)
2. Download from F5: Download the bigip.conf file (and/or certs/keys) directly from the F5 LTM device.

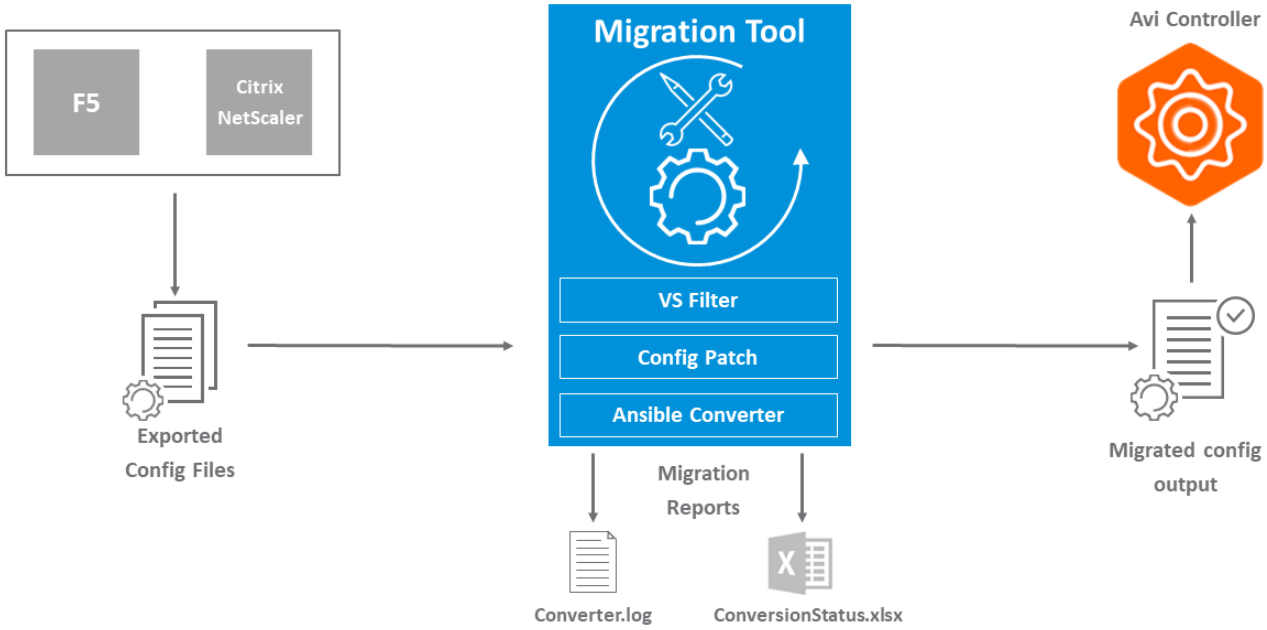
Upon successful run the migration converter outputs the results in a directory with following contents:

- **avi_config_create_object.yml**: Ansible playbook containing the converted F5 configuration set to **create** the objects.
- **avi_config_delete_object.yml**: Ansible playbook containing the converted F5 configuration set to **delete** the objects.
- ConversionStatus.xlsx: Status of the conversion. It is an Excel file with each row representing F5 configuration object and its migration status and corresponding Avi configuration.
- **Output.json**: This is the ouput of the migrated configuration in Avi Vantage configuration JSON format. This configuration can be uploaded to the Avi Vantage controller.
- **converter.log**: This is troubleshooting log for F5 Converter useful for debugging and logging purpose.
- **Output.yaml**: Optional, Using the flag --ansible, the script will output a working playbook to import all the objects directly into Avi

Further details about the Avi F5 Converter Tool are available on the Avi F5 Converter Tool Github page

The goal of the migration tool is to accelerate Avi field adoption in brownfield deployments. This is aided by the following capabilities

- Automation - Automatically convert and preserve existing application behavior
- Reporting - Provide easy to understand reporting on the conversion status
- Utilities - Provide ability to perform seamless cutovers and customizations



Migration Process

The migration process is broadly categorized into the following stages

1. Project planning & scheduling
2. Automated config conversion
3. Customization
4. Validation and functional testing
5. Cutover and Go-live

Each stage is associated with a set of attributes to be considered in preparation to the actual migration. A stage-wise breakdown of the different attributes and their implications on the migration is given below. These attributes also intend to serve as a set of prerequisites or a checklist to be discussed before getting to the hands-on migration phase.

Stage-1: Planning

In this stage, following tasks are expected to be completed

- Kick off meeting
- Team identification
- Migration project plan
- Infrastructure readiness assessment

Below is a list of attributes to be considered for this stage

SL NO	Attribute	Description	Implication	Flags/Settings
1	Migration Strategy	Determine the overall migration strategy. Common strategies based on field experience are as follows • Device by device • Multiple devices in parallel • Based on applications • Geo/Region based	• Project planning	• --vs_filter <VS1,VS2,VS3,VS4> • --ansible_skip_types <Object_Type> • --ansible_filter_types <Object_Type> • --ansible
2	Consolidation Strategy	Determine whether multiple existing ADCs will be consolidated to a single instance of Avi deployment.	• Project planning • Config import and staging* • SE Sizing • Licensing	• --cloud_name <Cloud_Name> • --tenant <Tenant_Name> • --segroup <SE_Group_Name> • --vrf <VRF_name>
3	Object Naming Strategy & Object Merging	Determine whether object names will be retained in Avi deployment. It is also possible to merge similar objects (for example, profiles with different names but exact same configs/properties, (excludes vs, vsvip, pool and poolgroup objects)) during the config conversion.	• Project planning • Config conversion	• --prefix <Prefix> • --no_object_merge
4	Re-IP Strategy	Determine whether VIPs will be same between the old and new deployments.	• Project planning • Validation • Cutover strategy	• NA
5	Strategy for unused/inactive VSs	Determine the strategy for unused/disabled/inactive VSs. Migration tool provides the option to ignore/skip such VSs during the config conversion thereby enabling automated decommissioning of stale configuration	• Project planning • Config conversion	• --not_in_use
6	'avitools' container usage	Determine whether 'avitools' container will be leveraged for migration. Alternative is to use the converter utility packaged as part of the Controller https://github.com/vmware/nsx-advanced-load-balancer-tools	• Project planning • Config conversion • Validation	• NA
7	Availability of config files	Ensure availability of all required config files. In case of multiple partitions on F5, ensure all partition config files are available.	• Project planning • Config conversion	• --partition_config
8	Tenancy	Determine provider mode vs tenant context mode of deployment	• Infrastructure readiness	• --tenant <Tenant_Name> • --segroup <SE_Group_Name> • --vrf <VRF_name>

***If there are overlaps in IPs/Object names between different ADCs, this can cause issues during config conversion and importing if consolidating ADCs**

Stage-2: Conversion & Staging

In this stage, following tasks are expected to be completed

- Review existing load balancer configuration
- Select apps for migration
- Configuration conversion strategy
- Configuration conversion
- Onboard converted configuration (optional)

Below is a list of attributes to be considered for this stage

SL NO	Attribute	Description	Implication	Flags/Settings
1	Feature gaps/Unsupported config	Discuss unsupported configuration, feature gaps and associated workaround/alternatives. Current migration tool unsupported configs are: WAF/ASM, GTM, iRules, external health monitors, authentication (SAML, LDAP, SSO), VSs with multiple certificates, SNAT pool/NAT Policy, system level configurations (Routing, DNS, NTP, SNMP, backup etc)	• Config conversion strategy	• Unsupported configuration will be documented in the ConversionStatus excel file generated by the migration tool
2	Avi deployment specifics	Determine Avi deployment specifics regarding the properties below • Controller version • Cloud • SE Group • Tenant • VRF • SNAT	• Config conversion	• --controller_version <Controller_Version> • --cloud_name <Cloud_Name> • --tenant <Tenant_Name> • --segroup <SE_Group_Name> • --vrf <VRF_name> • --convertsnat
3	SSL Certificates	Determine if the migration tool gets access to the certificates on the existing ADC. If not, auto_created self-signed certificates are bound to the applications during the automated config conversion	• Config conversion	• --input_folder_location <Input_Folder_Location> • --f5_passphrase_file <Passphrase_File_Name>
4	Migration checklist	Leverage the migration checklist to verify the converted config F5 Migration Manual Verification Checklist	• Config conversion	• NA

Stage-3: Customization

In this stage, the following tasks are expected to be completed

- Creation of application policies
- Conversion of iRules, traffic manipulation rules and custom health monitors
- Onboard additional & customized configuration
- Advanced iRules conversion to DataScript(s)

Majority of the iRules can be converted to Avi policies. Advanced iRules are implemented using Avi native scripting engine (DataScript functionality).

Below is a list of attributes to be considered for this stage

SL NO	Attribute	Description	Implication	Flags/Settings
1	In use iRules	Determine a scope in-use iRules to VSs	• Conversion of iRules	• --custom_config <Converted_iRules_Yaml_File>
2	Datascript library	Leverage DataScript Library for advanced customization	• Advanced customization	NA

Stage-4: Validation and functional testing

In this stage, following tasks are expected to be completed

- Verify imported configuration
- Verify health monitor operations
- Functional testing of applications
- Identify cutover and rollback strategies

Below is a list of attributes to be considered for this stage

SL NO	Attribute	Description	Implication
1	Verify imported config	Leverage the verification checklist to verify imported config is in line with old config and the tool generated bigip-ConversionStatus.xlsx	• Verify imported configuration
2	Verify backend connectivity	Manually verify backend connectivity between the SE(s) and application servers	• Verify health monitor operations
3	Verify health monitor operations	Leverage traffic enabled flag and BGP VIP advertisement options (under BGP routing settings and VS settings) to verify health monitor operations	• Verify health monitor operations
4	'avitools' utilities	Leverage utilities packaged with 'avitools' container such as cURL for application(s) validation	• Functional testing of applications
5	Validate application(s) functionality	Engage application owners for a functional validation/testing plan in line with actual production usage of the application	• Functional testing of applications

Stage-5: Cutover and Go-live

In this stage, following tasks are expected to be completed

- Perform application traffic cutover
- UAT and sign-off from application owners

Below is a list of attributes to be considered for this stage

SL NO	Attribute	Description	Implication
1	Cutover strategy	Determine the cutover strategy based on IP retention & L2 vs L3 scale out • Big bang - Cutover at once for all apps • Based on applications • Reuse of existing IP space or new subnets	• Application traffic cutover
2	Rollback strategy	Determine the rollback strategy, mainly for critical apps	• Application traffic cutover
3	Release system to end users	Handoff to administrators. Train Avi admin/operator on • Viewing & interpreting logs and analytics on Avi UI dashboard • Collecting common logs & tech-support for both Controllers and SEs • Opening support cases	• Sign-off and release system to end users
4	Migration file Cleanup	The migration tool will create several files and folders containing certificates and keys used during the migration. It is a good idea to clean up/secure these files and folders to keep the certificates and keys secure	• Clean up and security

Migration procedure

Use case-1: Basic migration with single F5 config file

In this example, a single config file contains the entire configuration. Conversion is performed to Avi config with the following considerations:

- 'avitools' container is already installed on a linux jumpbox and leveraged for migration
- F5 config version – v11 or greater
- Ansible playbooks for config create & config delete
- Controller version - 21.1.3 or greater
- Cloud - Default-cloud
- No object merge required
- Remove unreferenced/dangling/stale objects

Example Command:

```
python3 /usr/local/bin/f5_converter.py -f bigip_v11.conf --ansible --controller_version 21.1.3 --cloud_name Default-cloud --
no_object_merge --not_in_use
```

See Appendix for "Use case 1 CLI sample output"

Use case-2: Migration with partitions/tenancy

In this example, there are 3 config files for common partition, 'Prod' partition and 'nonProd' partition. Conversion is performed to Avi config with the following considerations:

- 'avitools' container is already installed on a Linux jumpbox and leveraged for migration
- Tenants 'Prod' and 'nonProd' should be created on Avi corresponding to 'Prod' and 'nonProd' partitions
- 'Prod' and 'nonProd' applications should be hosted on 'Prod' and 'nonProd' SE Groups respectively
- Ansible playbooks for config create & config delete
- Controller version - 21.1.3
- Cloud - VMC-Cloud
- No object merge required
- Remove unreferenced/dangling/stale objects

Example Command:

```
python3 /usr/local/bin/f5_converter.py -f bigip-common-int.conf --controller_version 21.1.3 --cloud_name VMC-Cloud -convertsnat
--no_object_merge --not_in_use -o Output-Dir1 --partition_config bigip-Prod-int.conf --ansible --segroup Prod-SE-Group

python3 /usr/local/bin/f5_converter.py -f bigip-common-int.conf --controller_version 21.1.3 --cloud_name VMC-Cloud --convertsnat
--no_object_merge --not_in_use -o Output-Dir2 --partition_config bigip-nonProd-int.conf --ansible --segroup nonProd-SE-Group
```

See Appendix for "Use case 2 CLI sample output"

Use case-3: Migration with certificates and keys in local directory

In this example, a single config file contains the entire configuration but instead of downloading certs and keys directly from the F5, we will call a local directory containing them. **Certificates and keys will need to be downloaded from the F5 due to the F5 naming schema. Original certificate and key files names will no longer match and will not map correctly in Ansible playbook.** Conversion is performed to Avi config with the following considerations:

- 'avitools' container is already installed on a linux jumpbox and leveraged for migration
- F5 config version – v16
- Ansible playbooks for config create & config delete
- Controller version – 21.1.3
- Cloud - Default-cloud
- No object merge required
- Remove unreferenced/dangling/stale objects
- Call "-l" flag to map local certificates and keys to ansible playbooks

Example Command:

```
f5_converter.py -f bigip_v16.conf --ansible --controller_version 21.1.3 --cloud_name Default-cloud --no_object_merge --
not_in_use -l certs_and_keys/ -o Output-Dir3
```

See Appendix for "Use case 3 CLI sample output"

Migration tool generated files

The "converter.log" file will detail the converter script operations and highlight possible converting issues with WARNING or ERROR flags. Investigate for any possible issues. "bigip-ConversionStatus.xlsx" is vital to understanding which objects were fully converted, partially converted, and not converted, as well as which objects require manual intervention and migration.

Understanding 'ConversionStatus' report file

The output of config conversion using the F5 converter tool is as excel file 'ConversionStatus.xls' that serves as a report for the migration process. This file consists of two sheets, 'Status Sheet' and 'Pivot Sheet'.

The 'Status Sheet' consists of each row representing F5 configuration object, its migration status and corresponding Avi configuration.

The 'Pivot Sheet' provides a summary mapping various statuses to F5 object types and the associated number of occurrences.

Refer to the table below to interpret the key statuses in the report file and some common examples contributing to each status type

SL NO	Status	Description	Common Examples
1	Successful	Object was completely converted to Avi equivalent config and similar behavior may be expected	• Profiles (Application, TCP/IP, SSL, Persistence) • Health monitors • Virtual Servers • Pools
2	Partial	Not all settings within the object could be fully converted to Avi equivalent config. Converter provides list of the skipped settings as part of the status of that object.	• Profiles (Application, TCP/IP, SSL, Persistence) • Health monitors • Virtual Servers • Pools
3	Skipped	Whole object was skipped due to no equivalent config on Avi or post migration action may be required	• Stale and un-referenced objects • Profiles (spdy profile for instance) • SNAT pools (if SNAT config conversion is not required)
4	Missing File	Key/Certificate or External health monitor script file not found to convert the object to Avi equivalent config	• SSL certificates and keys • External health monitors
5	Not Applicable	Config that does not have any impact on traffic and Avi doesn't not support it	• Hardware settings • 'node' object on F5 • 'virtual-address' object on F5
6	Not Supported	Settings/configuration not supported by Avi or not supported by F5 migration tool.	• iRules • SNAT Translation • Analytics profiles • ASM Policy
7	Indirect	Indirect mapping of the feature/object to Avi equivalent config	Usually sub-settings/sub-properties of a parent object such as Application profile and Health monitor
8	External Monitor	Type of monitor not supported on NSX Advanced Load Balancer. Need an external health monitor script	NA

Uploading Ansible playbooks to controller

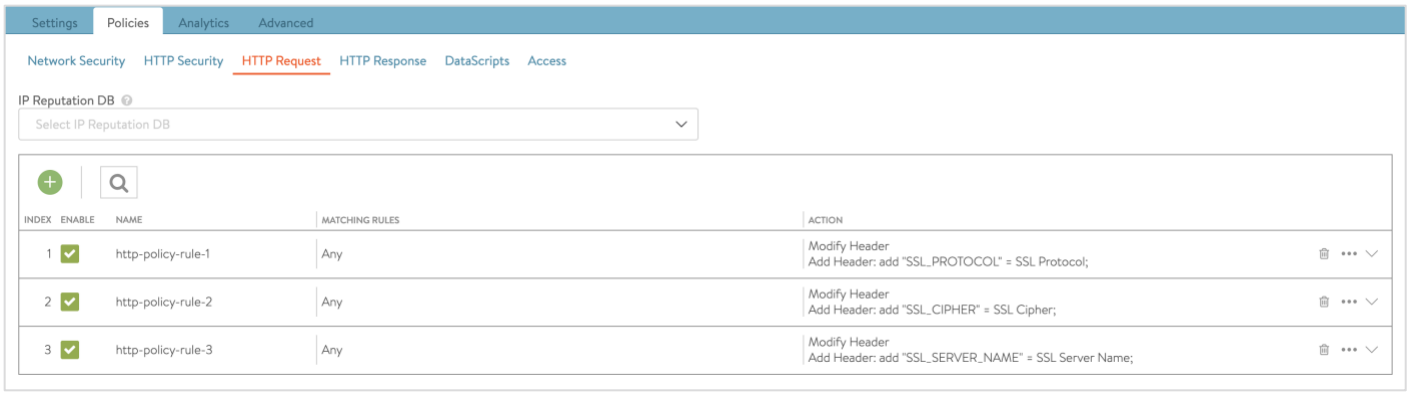
The converted config in yaml format may now be uploaded to the Avi controller using the following command - `ansible-playbook avi_config_create_object.yaml -e 'controller=<controller-ip> username=<username> password=<password>'`

Working with iRules

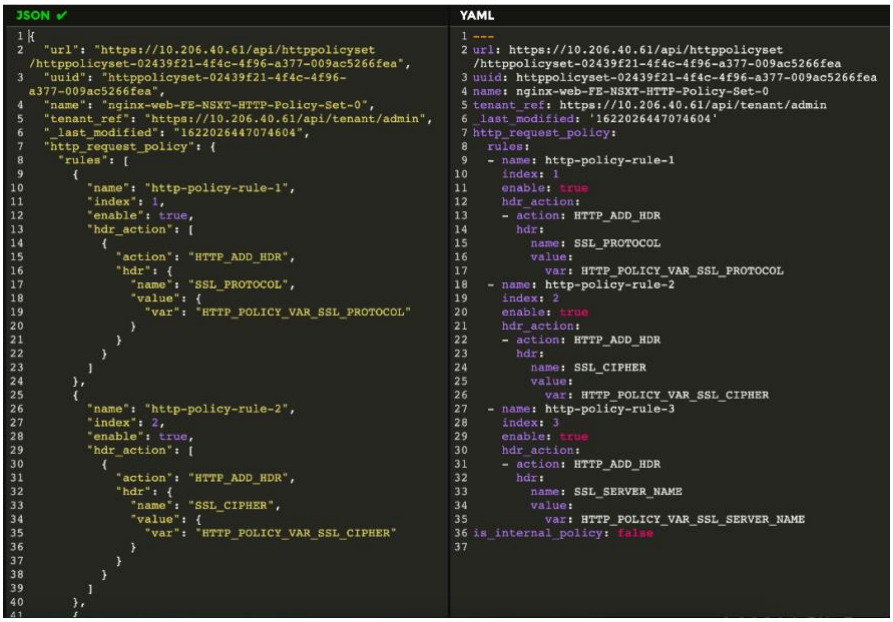
Most of the iRules on F5 can be converted to Avi policies during the migration process. Advanced iRules are implemented using Avi native scripting engine (DataScript functionality). The DataScript library available on github, can be leveraged for migrating some of the advanced iRules/functionality to Avi during migration.

Following procedure can be used to perform automated migration of iRules to Avi policies

1. Verify the iRules that need to be migrated.
2. Login to the Avi Controller. Navigate to Application > Virtual Services and edit one of the Virtual Services to configure an application policy
3. Under the edit Virtual Service workflow, navigate to Policies and select appropriate policy, for instance an HTTP request policy
4. Configure the HTTP request policy per details of the iRule in subject and save



5. Steps 6-8 are the manual way of creating the custom_config file that contains the converted iRules. You can also run the custom_config script and skip directly to step 9 if desired.
6. Retrieve the JSON equivalent of previously configured HTTP request policy by navigating to `<https://<controller_ip>/api/httppolicyset>` and clicking on the HTTP request policy name and copying **raw** JSON
7. Convert the HTTP request policy in JSON format to YAML format using a JSON to YAML converter, for instance https://www.json2yaml.com/



8. Use the YAML format of the HTTP request policy and build a custom config iRule YAML file like below. Ensure first line of the custom config iRule YAML file is always 'irule_custom_config:'. This is required by the F5 converter tool to interpret the custom config appropriately

```
irule_custom_config:


- rule_name: Insert_SSL_Info
  type: HTTPPolicySet
  avi_config:
    http_request_policy:
        rules:
        - name: http-policy-rule-1
          index: 1
          enable: true
          hdr_action:
          - action: HTTP_ADD_HDR
            hdr:
              name: SSL_PROTOCOL
              value:
                var: HTTP_POLICY_VAR_SSL_PROTOCOL
        - name: http-policy-rule-2
          index: 2
          enable: true
          hdr_action:
          - action: HTTP_ADD_HDR
            hdr:
              name: SSL_CIPHER
              value:
                var: HTTP_POLICY_VAR_SSL_CIPHER
        - name: http-policy-rule-3
          index: 3
          enable: true
          hdr_action:
          - action: HTTP_ADD_HDR
            hdr:
              name: SSL_SERVER_NAME
              value:
                var: HTTP_POLICY_VAR_SSL_SERVER_NAME
      is_internal_policy: false
```

9. Re-run the migration tool against the F5 config file as in the previous examples, this time calling the --custom_config <file_name> flag to call the iRule playbook. This will create a resulting playbook with required parameters from the custom config.

```
python3 /usr/local/bin/f5_converter.py -f bigip-common-int.conf --controller_version 21.1.3 --cloud_name VMC-Cloud --
convertsnat --no_object_merge --not_in_use -o Output-Dir1 --partition_config bigip-Prod-int.conf --ansible --segroup
Prod-SE-Group --custom_config converted_irules.yml
```

10. The converted config in YAML format may now be uploaded to the Avi controller using the following command -

```
ansible-playbook –e 'controller=<controller-ip> username=<username> password=<password>' avi_config_create_object.yaml
```

Config patch module

The 'config_patch.py' module of the migration tools bundle provides the ability to perform patch updates on the migration tool generated json configuration file.

```
In the first example, a bulk object patch is performed using the module to set the 'enabled' flag to false and 'traffic_enabled'
flag to true on all VSs. This option is particularly useful in validating the health-monitor operation between the SE and
servers prior to cutover.
root@avitools:/opt/avi/output# tree
.
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- converter.log
`-- patch.yaml

0 directories, 6 files

root@avitools:/opt/avi/output# cat avi_config_create_object.yml | grep traffic_enabled
      traffic_enabled: false
      traffic_enabled: false
      traffic_enabled: false
      traffic_enabled: false
      traffic_enabled: false
<truncated>

root@avitools:/opt/avi/output# cat patch.yaml
VirtualService:
- match_name: ".*"
  patch:
    enabled: False
    traffic_enabled: True

root@avitools:/opt/avi/output# python3 /usr/local/bin/config_patch.py -c bigip_v11-Output.json -p patch.yaml
```

See Appendix for Config patch module CLI sample output

In the second example, a bulk object patch is performed using the module to enable 'X-Forwarded-For' option under all L7 application profiles.

```
root@avitools:/opt/avi/output# tree
.
|-- app-profile-patch.yaml
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- converter.log
`-- patch.yaml

0 directories, 7 files

root@avitools:/opt/avi/output# cat app-profile-patch.yaml
ApplicationProfile:
  - match_name: ".*"
    patch:
      http_profile:
        xff_enabled: true

root@avitools:/opt/avi/output# python3 /usr/local/bin/config_patch.py -c bigip_v11-Output.json -p app-profile-patch.yaml
```

See Appendix for second example Config patch module CLI sample output

Convert Patched JSON to Ansible

After patching, you will need to convert patched JSON file to ansible. Use the avi_config_to_ansible script to complete this.

```
root@avitools:/opt/avi/output# tree
.
|-- app-profile-patch.yaml
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log
`-- patch.yaml

0 directories, 8 files

root@avitools:/opt/avi/output# python3 avi_config_to_ansible.py -c bigip_v11-Output.json.patched --controller_version 21.1.3
```

You will now see two new files, avi_config.yml and avi_config_delete.yml

```
root@avitools:/opt/avi/output# tree
.
|-- app-profile-patch.yaml
|-- avi_config_create_object.yml
|-- avi_config.yml
|-- avi_config_delete.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log
`-- patch.yaml

0 directories, 10 files
```

Traffic cutover approach

Identifying the right strategy will help in minimizing downtime during the application cutover. Following factors need to be considered to identify the right cutover strategy

1. Legacy VIP reachability
2. Avi VIP reachability
3. Re-IP (Whether or not VIP changes from old ADC to NSX Advanced Load Balancer)

Traffic cutover scenarios

SL NO	Legacy VIP Reachability	Avi VIP reachability	Re-IP	Modifications on legacy ADC	Modifications on Avi
1	L2 (L3 upstream and ADC share a subnet)	L2 (L3 upstream and SE share a subnet)	No	• Disable ARP for VIP • Disable VS	• Enable VS
2	L2 (L3 upstream and ADC share a subnet)	L3 RHI via BGP	No	• No changes needed	• Advertise VIP via BGP (Injects /32 route to VIP into the L3 upstream)
3	L3 static route (/24 route on L3 upstream)	L3 RHI via BGP	No	• No changes needed	• • Advertise VIP via BGP (Injects /32 route to VIP into the L3 upstream)
4	L3 RHI via BGP (/32 RHI route in L3 upstream)	L3 RHI via BGP (/32 RHI route in L3 upstream)	No	• Disable BGP advertisement for VIP (disables RHI as well)*	• Advertise VIP via BGP (Injects /32 route to VIP into the L3 upstream)
5	Any	Any	Yes - Modify Corp DNS to point FQDN for the application to the new IP (VIP) on Avi	• No changes needed	• No changes needed

*For this scenario, it is also possible to achieve the cutover by using BGP traffic engineering (for instance using Local Preference) instead of choosing to disable BGP advertisement for VIP on legacy ADC

Other migration related resources

1. Migration Hands on Lab (Migration Section starts on module 7)
2. Migration How to Video Series
3. GitHub Datascript Library

Migration FAQs

Migration Tool Questions

1. Where do I get the migration tools?
   a. Avitools Docker container (Preferred Option)
   b. VMware Github
2. Can I get the migration done without using the 'avitools' container?
   a. Yes, however the avitools container contains all the tools and dependencies to run them
3. Where can I get the F5 converter tool on the Avi Controller?
   a. /opt/avi/python/lib/avi/migrationtools/f5_converter/f5_converter.py
4. How do I call multiple partitions when running the tool?
   a. You can use partition flag to merge partitions in the resulting conversion
      i. --partition_config PARTITION_CONFIG <comma separated partition config files>
5. Is there an option to select a specific partition from F5 config?
   a. No, but this feature will be available in future releases.
6. How do I migrate only a subset of virtual services on an F5 instead.
   a. The migration tool will migrate all of the virtual services in the config unless you use the --vs_filter flag
      i. Example: f5_converter.py -f bigip.conf --vs_filter vs1,vs3,vs5
7. Does the migration tool convert F5 oneconnect?
   a. Yes, the migration tool will convert oneconnect to Connection Multiplex in the application profile
8. What does -cmd mean?
   a. Connection Multiplex Disabled. Oneconnect is a separate parameter on F5, where on Avi connection multiplex is a part of the application profile. If you have two VSs on F5 that have the same application profile but one VS has oneconnect enabled, the migration tool will duplicate two application profiles. One will have connection multiplex enabled and one will have connection multiplex disabled.
      i. Example Application Profiles
         1. http (Connection Multiplex Enabled)
         2. http-cmd (Connection Multiplex Disabled)
9. Why do I see default.cert-auto_created under VSs config?
   a. This object is created as a place holder during offline config conversion for all SSL certificates that can't be exported out of F5, once the converted file is uploaded to the controller, the user can replace the auto_created objects with valid ones.
10. I see SFTP Error ("Garbage packet received") when I try to run f5_converter.py command?
    a. This is a generic alert, and you have to check scp permission from F5 to Avi, or use root user on f5.

11. Why do I have duplicated pools after migration?
    a. Pool sharing on F5 is supported differently on Avi. Pools sharing is supported with these restrictions. Any use case outside of these restrictions, will result in the migration tool creating a duplicate pool.
12. Why do I see different ciphers used in Avi than I do in F5?
    a. By default, the migration tool configures the default Avi cipher suite for all virtual services over what was configured in F5. This typically causes no issues in traffic flow after cutover to Avi. For critical applications and special use cases, you may want to validate and test the "Accepted Ciphers Strings" now used with your application on Avi.

Config Import Questions

13. I migrated everything into the wrong cloud/tenant/se group/vrf etc, how do I fix it?
    a. Use the avi_config_delete_object.yml or avi_config_delete.yml (if you patched the file) to delete the objects. Update the reference to the correct one and push the playbook to the controller again
        i. Example.
            1. Wrong cloud reference: cloud_ref: /api/cloud?name=Default-Cloud
            2. Correct reference: cloud_ref: /api/cloud?name=VMware-Cloud

iRule Conversion Questions

14. Will F5 iRules be converted automatically?
    a. No, iRules must be converted manually. The migration tool can map converted iRules, now policies/DataScript(s), to all applicable Virtual Services
15. For iRule migration, can we configure the Policies before or independently of the Virtual Service migration/creation?
    a. Yes, you can create standalone policies via the CLI.
        i. https://avinetworks.com/docs/latest/http-policy-reuse/
16. How do I configure multiple indexes in a http policy in the custom_config file?
    a. https://avinetworks.com/application-delivery-how-to-videos/
17. Can we deploy a global policy on Avi that can be triggered before matching any Virtual Service?
    a. No

Appendix

F5 Migration Manual Verification Checklist

(Checklist hits major configurations but is not an all-inclusive list)

- Virtual Services
  - IP Address
  - Port
  - Application Profile
  - TCP/UDP Profile
  - SSL Profile
- Application Profile
  - General Tab
    - Type
      - HTTP, L4 etc
    - Connection Multiplex (F5 oneconnect)
    - X-Forwarded-For (General Tab)
  - Security tab
    - Secure HTTP
      - Verify whether SSL everywhere features need to be enabled
- SSL Profile
  - Accepted Ciphers
  - SSL Certificate name
- TCP Profile
- Pool
  - Persistence
  - Health Monitor
  - Default Server Port
  - Servers
    - Ports
- Health Monitor
  - Timers
  - Type
    - If HTTP/HTTPS
      - Client Request Data
        - Verify string is in correct order
        - Verify all /r/n are present with no white spaces
  - Response Code
  - Health Monitor Port
  - Server Data Response
- Persistence
- Verify whether pool has it or not

Use case 1 CLI sample output

```
aviadmin@lab-mgmt1:~$ docker ps
CONTAINER ID        IMAGE                          COMMAND
CREATED             STATUS              PORTS
NAMES
423a4238eb03        avinetworks/avitools:21.1.3    "bash"
6 months ago        Up 6 months
avitools
<truncated>

aviadmin@lab-mgmt1:~$ docker exec -it avitools bash

root@avitools:/opt/avi# ls -l bigip_v11.conf
-rw-r--r-- 1 root root 85001 Mar 12 08:52 bigip_v11.conf

root@avitools:/opt/avi# python3
/usr/local/bin/f5_converter.py -f bigip_v11.conf --ansible
--controller_version 21.1.3 --cloud_name Default-cloud --
no_object_merge --not_in_use
Log File Location: output
AVI sdk version: 21.1.3 Controller Version: 21.1.3
Parsing Input Configuration...
Progress |############----------------------------------
-| 25.4%
<truncated>
Progress
|################################################-| 100.0%

Converting Profiles ...
Progress |-------------------------------------------------
-| 1.3%
Progress |#------------------------------------------------
-| 2.6%
Progress |#------------------------------------------------
-| 3.9%
Progress |##-----------------------------------------------
-| 5.3%
Progress |###----------------------------------------------
-| 6.6%
Progress |###----------------------------------------------
-| 7.9%
<truncated>
Progress
|################################################-| 98.7%
Progress
|#################################################| 100.0%
Converting Monitors...
Progress |#------------------------------------------------
-| 3.8%
Progress |###----------------------------------------------
-| 7.7%
Progress |#####----------------------------------------------
-| 11.5%
<truncated>
Progress |################################################-
-| 96.2%
Progress
|#################################################| 100.0%
Converting Pools...
Progress |##-----------------------------------------------
-| 4.8%
Progress |####---------------------------------------------
-| 9.5%
Progress |#######------------------------------------------
-| 14.3%
<truncated>
Progress |################################################--
-| 95.2%
Progress
|#################################################| 100.0%
Converting Persistence Profiles...
Progress |###----------------------------------------------
-| 7.7%
Progress |#######------------------------------------------
-| 15.4%
<truncated>
Progress |################################################---
-| 92.3%
Progress
|#################################################| 100.0%
Converting VirtualServices ...
Progress |#------------------------------------------------
-| 3.8%
Progress |###----------------------------------------------
-| 7.7%
Progress |#####----------------------------------------------
-| 11.5%
<truncated>
Progress |################################################-
-| 96.2%
Progress
|#################################################| 100.0%
Converting Data groups...
Progress |########################----------------------
-| 50.0%
Progress
|#################################################| 100.0%
Cloning cross tenant objects...
Progress |-------------------------------------------------
-| 1.1%
Progress |#------------------------------------------------
-| 2.2%
Progress |#------------------------------------------------
-| 3.3%
Progress |##-----------------------------------------------
-| 4.4%
<truncated>
Progress |################################################-
-| 97.8%
Progress
|################################################-| 98.9%
Progress
|#################################################| 100.0%
SKIPPED: 24
SUCCESSFUL: 79
NOT APPLICABLE: 59
ERROR: 0
PARTIAL: 76
DATASCRIPT: 0
Writing Excel Sheet For Converted Configuration...
Progress |-------------------------------------------------
-| 0.3%
Progress |-------------------------------------------------
-| 0.6%
Progress |-------------------------------------------------
-| 1.0%
Progress |-------------------------------------------------
-| 1.3%
Progress |-------------------------------------------------
-| 1.6%
Progress |-------------------------------------------------
-| 1.9%
<truncated>
Progress
|################################################-| 99.0%
Progress
|################################################-| 99.4%
Progress
|################################################-| 99.7%
Progress
|#################################################| 100.0%
Total Objects of ApplicationProfile : 37
Total Objects of NetworkProfile : 24
Total Objects of SSLProfile : 28
Total Objects of PKIProfile : 0
Total Objects of SSLKeyAndCertificate : 17
Total Objects of ApplicationPersistenceProfile : 10
Total Objects of HealthMonitor : 22
Total Objects of IpAddrGroup : 3
Total Objects of StringGroup : 5
Total Objects of HTTPPolicySet : 11
```

```
Total Objects of VrfContext : 2
Total Objects of PoolGroup : 0
Total Objects of PriorityLabels : 0
Total Objects of Pool : 27
Total Objects of VirtualService : 25
Total Objects of VSDataScriptSet : 0
Total Objects of NetworkSecurityPolicy : 1
Total Objects of VsVip : 23
Total Objects of Tenant : 1
Converted Output Location: output/bigip_v11-Output.json
Conversion Started For Ansible Create Object...
Progress |--------------------------------------------------
-| 1.1%
Progress |#-------------------------------------------------
-| 2.2%
Progress |#-------------------------------------------------
-| 3.3%
Progress |##------------------------------------------------
-| 4.4%
<truncated>
```

```
Progress |################################################-
-| 96.7%
Progress |################################################-
-| 97.8%
Progress
|################################################-| 98.9%
Progress
|################################################| 100.0%
Total Warning:  38
Total Errors:  0
root@avitools:/opt/avi# cd output
root@avitools:/opt/avi/output# tree
.
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
`-- converter.log

0 directories, 5 files
root@avitools:/opt/avi/output#
```

Return to use case 1

Use case 2 CLI sample output

```
root@avitools:/opt/avi/Partition-Ex-2# ls -l
total 52
-rwxrwxr-x 1 1000 1000 20510 Mar 15 08:50 bigip-Prod-int.conf
-rwxrwxr-x 1 1000 1000  3487 Mar 15 08:50 bigip-common-int.conf
-rwxrwxr-x 1 1000 1000 21332 Mar 15 08:50 bigip-nonProd-int.conf

root@avitools:/opt/avi/Partition-Ex-2# python3 /usr/local/bin/f5_converter.py -f bigip-common-int.conf --controller_version
21.1.3 --cloud_name VMC-Cloud --convertsnat --no_object_merge --not_in_use -o Output-Dir1 --partition_config bigip-Prod-int.conf
--ansible --segroup Prod-SE-Group

<Output Omitted>

root@avitools:/opt/avi/Partition-Ex-2# tree Output-Dir1
Output-Dir1
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip-common-int-ConversionStatus.xlsx
|-- bigip-common-int-Output.json
`-- converter.log

0 directories, 5 files

root@avitools:/opt/avi/Partition-Ex-2# python3 /usr/local/bin/f5_converter.py -f bigip-common-int.conf --controller_version
21.1.3 --cloud_name VMC-Cloud --convertsnat --no_object_merge --not_in_use -o Output-Dir2 --partition_config bigip-nonProd-
int.conf --ansible --segroup nonProd-SE-Group


<Output Omitted>

root@avitools:/opt/avi/Partition-Ex-2# tree Output-Dir2
Output-Dir2
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip-common-int-ConversionStatus.xlsx
|-- bigip-common-int-Output.json

-- converter.log

0 directories, 5 files
root@avitools:/opt/avi/Partition-Ex-2#
```

Return to use case 2

Use case 3 CLI sample output

```
root@avitools:/opt/avi# f5_converter.py -f bigip_v16.conf --ansible --controller_version 21.1.3 --cloud_name Default-cloud --
no_object_merge --not_in_use -l certs_and_keys/ -o Output-Dir3

Log File Location: Output-Dir3

AVI sdk version: 21.1.3 Controller Version: 21.1.3

Parsing Input Configuration...


<Output omitted>

root@avitools:/opt/avi# tree Output-Dir3
Output-Dir3
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v16-ConversionStatus.xlsx
|-- bigip_v16-Output.json
`-- converter.log

0 directories, 5 files
```

Return to use case 3

patch module CLI sample output

```
root@avitools:/opt/avi/output# tree
.
|-- avi_config_create_object.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- converter.log
`-- patch.yaml

0 directories, 6 files
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# cat
avi_config_create_object.yml | grep traffic_enabled
      traffic_enabled: false
      traffic_enabled: false
      traffic_enabled: false
<truncated>
      traffic_enabled: false
      traffic_enabled: false
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# cat patch.yaml
VirtualService:
- match_name: ".*"
  patch:
    enabled: False
    traffic_enabled: True
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# python3
/usr/local/bin/config_patch.py -c bigip_v11-Output.json -p
patch.yaml
[2021-03-15 13:59:27,469] DEBUG [config_patch.__init__:44]
input patch {'VirtualService': [{'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}]}
Conversion For Patching of objects started...
[2021-03-15 13:59:27,473] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:EngVIP with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_2_up with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:F5-VIP-80-001 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:F5-VIP-443-001 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:F5-VIP-443-002 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:F5-VIP-443-004 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,474] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:F5-VIP-Forwarding with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:Opcito-vs with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:Opcito-vs-1 with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:dns_vs_up with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}

root@avitools:/opt/avi/output# tree
.
|-- avi_config_create_object.yml
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:gtmlistener1 with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:l4_vs_up with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:my-test-asm-philippe with patch
{'match_name': '.*', 'patch': {'enabled': False,
'traffic_enabled': True}}
[2021-03-15 13:59:27,475] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:splunk-harsh with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:test-asm-sideband with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:testmr with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:testvip with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:testvip-http-80 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs1 with patch {'match_name': '.*', 'patch':
{'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs10.90.114.202 with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_1_up with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,476] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_target_vs with patch {'match_name': '.*',
'patch': {'enabled': False, 'traffic_enabled': True}}
[2021-03-15 13:59:27,477] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_http_policy_share_1 with patch
{'match_name': '.*', 'patch': {'enabled': False,
'traffic_enabled': True}}
[2021-03-15 13:59:27,477] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_http_policy_share_2 with patch
{'match_name': '.*', 'patch': {'enabled': False,
'traffic_enabled': True}}
[2021-03-15 13:59:27,477] DEBUG
[config_patch.apply_obj_patch:182] patching
VirtualService:vs_custome_vrf with patch {'match_name':
'.*', 'patch': {'enabled': False, 'traffic_enabled': True}}
Progress
|################################################| 100.0%
```

```
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log
`-- patch.yaml


0 directories, 7 files
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# cat bigip_v11-Output.json.patched | grep traffic_enabled
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,
            "traffic_enabled": true,
            "enabled": false,

root@avitools:/opt/avi/output# python3 /usr/local/bin/avi_config_to_ansible.py -c bigip_v11-Output.json.patched
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# tree
.
|-- avi_config.yml
|-- avi_config_create_object.yml
|-- avi_config_delete.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log
`-- patch.yaml

0 directories, 9 files
root@avitools:/opt/avi/output#
```

Return to patch section

Second example Config patch module CLI sample output

```
root@avitools:/opt/avi/output# tree
.
|-- app-profile-patch.yaml
|-- avi_config_create_object.yml
|-- avi_config_delete.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- converter.log
`-- patch.yaml

0 directories, 8 files
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# cat app-profile-patch.yaml
ApplicationProfile:
  - match_name: ".*"
    patch:
      http_profile:
        xff_enabled: true

root@avitools:/opt/avi/output# python3
/usr/local/bin/config_patch.py -c bigip_v11-Output.json -p
app-profile-patch.yaml
[2021-03-15 14:24:30,380] DEBUG [config_patch.__init__:44]
input patch {'ApplicationProfile': [{'match_name': '.*',
'patch': {'http_profile': {'xff_enabled': True}}}]}
Conversion For Patching of objects started...
[2021-03-15 14:24:30,385] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:http-cmd with patch {'match_name': '.*',
'patch': {'http_profile': {'xff_enabled': True}}}
[2021-03-15 14:24:30,385] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:hdrInsert-cmd with patch {'match_name':
'.*', 'patch': {'http_profile': {'xff_enabled': True}}}
[2021-03-15 14:24:30,385] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:testhttp-cmd with patch {'match_name':
'.*', 'patch': {'http_profile': {'xff_enabled': True}}}
[2021-03-15 14:24:30,386] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:fastL4-cmd with patch {'match_name':
'.*', 'patch': {'http_profile': {'xff_enabled': True}}}
[2021-03-15 14:24:30,386] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:dns-cmd with patch {'match_name': '.*',
'patch': {'http_profile': {'xff_enabled': True}}}
[2021-03-15 14:24:30,386] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:f5-default-profile-http-cmd with patch
{'match_name': '.*', 'patch': {'http_profile':
{'xff_enabled': True}}}
[2021-03-15 14:24:30,386] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:Bigip-support-Profile-FastL4-cmd with
patch {'match_name': '.*', 'patch': {'http_profile':
{'xff_enabled': True}}}
[2021-03-15 14:24:30,386] DEBUG
[config_patch.apply_obj_patch:182] patching
ApplicationProfile:sharedHttpPolicy-cmd with patch
{'match_name': '.*', 'patch': {'http_profile':
{'xff_enabled': True}}}
Progress
|################################################| 100.0%
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output#
tree
.
|-- app-profile-patch.yaml
|-- avi_config_create_object.yml
|-- avi_config_delete.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log

0 directories, 8 files
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# python3
/usr/local/bin/avi_config_to_ansible.py -c bigip_v11-
Output.json.patched

root@avitools:/opt/avi/output# tree
.
|-- app-profile-patch.yaml
|-- avi_config.yml
|-- avi_config_create_object.yml
|-- avi_config_delete.yml
|-- avi_config_delete_object.yml
|-- bigip_v11-ConversionStatus.xlsx
|-- bigip_v11-Output.json
|-- bigip_v11-Output.json.patched
|-- converter.log

0 directories, 9 files
root@avitools:/opt/avi/output#
root@avitools:/opt/avi/output# cat avi_config.yml | grep
xff
        xff_alternate_name: null
        xff_enabled: true
        xff_alternate_name: null
        xff_enabled: true
        xff_alternate_name: null
        xff_enabled: true
        xff_enabled: true
        xff_enabled: true
        xff_alternate_name: null
        xff_enabled: true
        xff_enabled: true
        xff_alternate_name: null
        xff_enabled: true
```

Return to patch section

Item No: vmw-wp-temp-uslet-word-2021 8/21