PyNSXv-examples (CLI + Scripts)

1. **CLI examples**
2. Python script examples

vmworld®

Use PyNSXv as "CLI commands" – Find PyNSXv CLI commands

- Find all available CLI commands
  - `PyNSXv -h`

```
C:\pynsxv>pynsxv.exe -h
usage: pynsxv-script.py [-h] [-i INI] [-v] [-d]
                        {lswitch,dlr,esg,dfw,usage} ...

PyNSXv Command Line Client for NSX for vSphere

positional arguments:
  {lswitch,dlr,esg,dfw,usage}
    lswitch             Functions for logical switches
    dlr                 Functions for distributed logical routers
    esg                 Functions for edge services gateways
    dfw                 Functions for distributed firewall
    usage               Functions to retrieve NSX-v usage statistics

optional arguments:
  -h, --help            show this help message and exit
  -i INI, --ini INI     nsx configuration file
  -v, --verbose         increase output verbosity
  -d, --debug           print low level debug of http transactions
```

Windows

```
dimi@ubuntu-python:~$ pynsxv -h
usage: pynsxv [-h] [-i INI] [-v] [-d] {lswitch,dlr,esg,dfw,usage} ...

PyNSXv Command Line Client for NSX for vSphere

positional arguments:
  {lswitch,dlr,esg,dfw,usage}
    lswitch             Functions for logical switches
    dlr                 Functions for distributed logical routers
    esg                 Functions for edge services gateways
    dfw                 Functions for distributed firewall
    usage               Functions to retrieve NSX-v usage statistics

optional arguments:
  -h, --help            show this help message and exit
  -i INI, --ini INI     nsx configuration file
  -v, --verbose         increase output verbosity
  -d, --debug           print low level debug of http transactions
```

Linux

Use PyNSXv as "CLI commands" – Find PyNSXv CLI commands

- Find PyNSXv parameters for a specific command
  - PyNSXv [lswitch|dlr|esg|dfw|usage] -h

```
dimi@ubuntu-python:~$ pynsxv dlr -h
usage: pynsxv dlr [-h] [-n NAME] [-p DLRPASSWORD] [-s DLRSIZE] [--ha_ls HA_LS]
                  [--uplink_ls UPLINK_LS] [--uplink_ip UPLINK_IP]
                  [--uplink_subnet UPLINK_SUBNET] [--uplink_dgw UPLINK_DGW]
                  [--interface_ls INTERFACE_LS] [--interface_ip INTERFACE_IP]
                  [--interface_subnet INTERFACE_SUBNET]
                  command

nsxv function for dlr 'pynsxv dlr @params.conf'.

positional arguments:
  command
                            create:         create a new dlr
                            read:           return the id of a dlr
                            delete:         delete a dlr
                            list:           return a list of all dlr
                            dgw_set:        set dlr default gateway ip addr
                            dgw_del:        delete dlr default gateway ip a
                            add_interface:  add interface in dlr
                            del_interface:  delete interface of dlr
                            list_interfaces:list all interfaces of dlr
```

```
optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  dlr name
  -p DLRPASSWORD, --dlrpassword DLRPASSWORD
                        dlr admin password
  -s DLRSIZE, --dlrsize DLRSIZE
                        dlr size (compact, large, quadlarge, xlarge)
  --ha_ls HA_LS         dlr ha LS name
  --uplink_ls UPLINK_LS
                        dlr uplink logical switch name
  --uplink_ip UPLINK_IP
                        dlr uplink ip address
  --uplink_subnet UPLINK_SUBNET
                        dlr uplink subnet
  --uplink_dgw UPLINK_DGW
                        dlr uplink default gateway
  --interface_ls INTERFACE_LS
                        interface logical switch in dlr
  --interface_ip INTERFACE_IP
                        interface ip address in dlr
  --interface_subnet INTERFACE_SUBNET
                        interface subnet in dlr
```

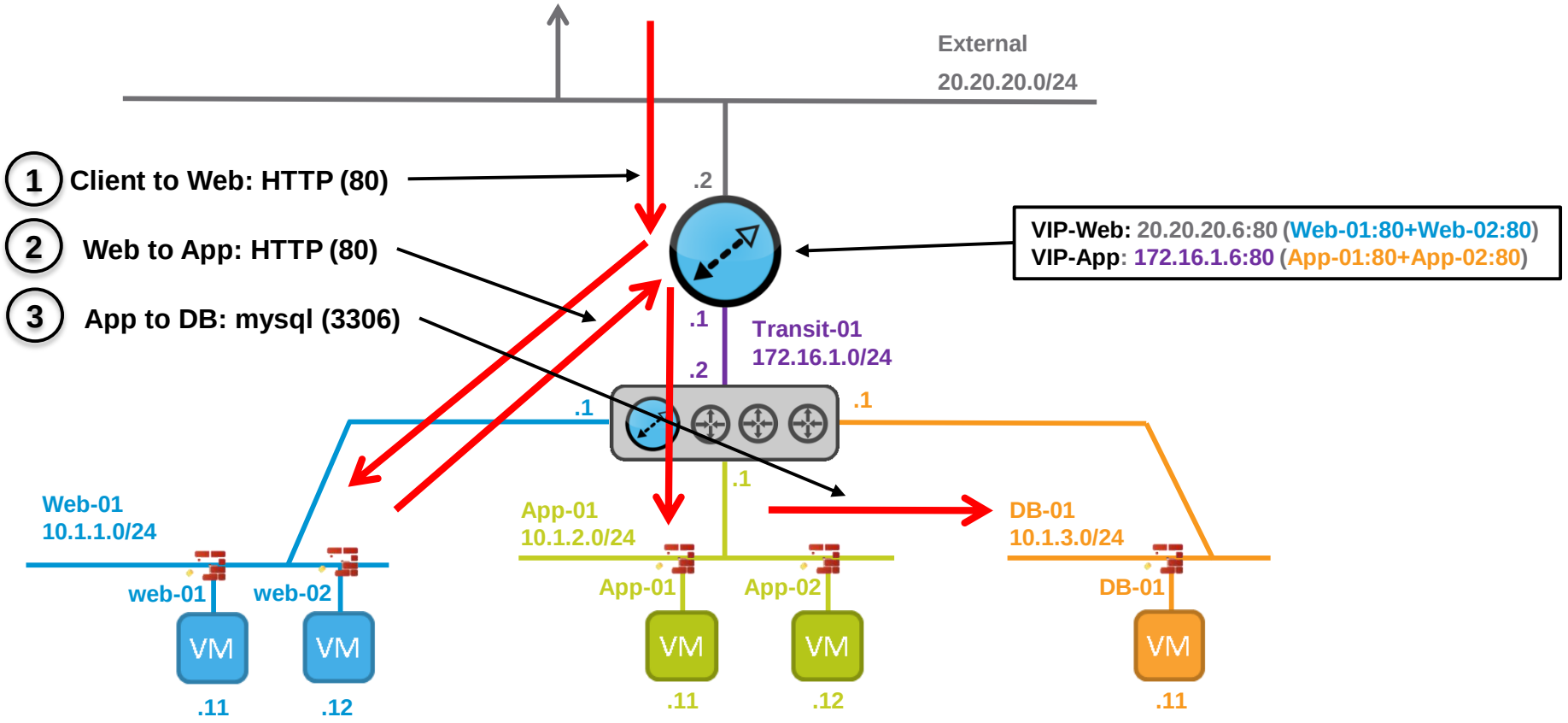
Use PyNSXv as "CLI commands" – Basic CLI example

- Let's create a logical switch

```
dimi@ubuntu-python:~$ pynsxv lswitch create -n python_cli-LS1
```

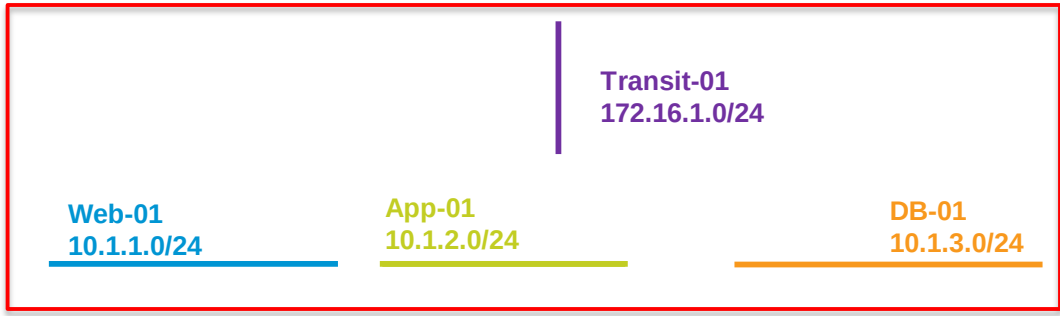
Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Topology



Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Creation of subnets

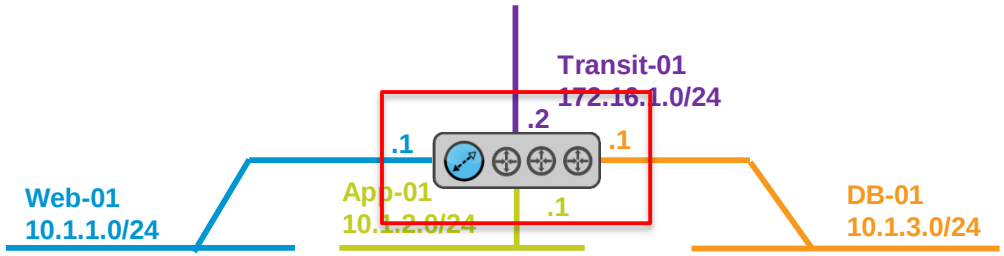


Create logical switches

```
dimi@ubuntu-python:~$ pynsxv lswitch create -n python_cli-Web-01
dimi@ubuntu-python:~$ pynsxv lswitch create -n python_cli-App-01
dimi@ubuntu-python:~$ pynsxv lswitch create -n python_cli-DB-01
dimi@ubuntu-python:~$ pynsxv lswitch create -n python_cli-Transit-01
```

Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Creation of distributed logical router



Create logical router

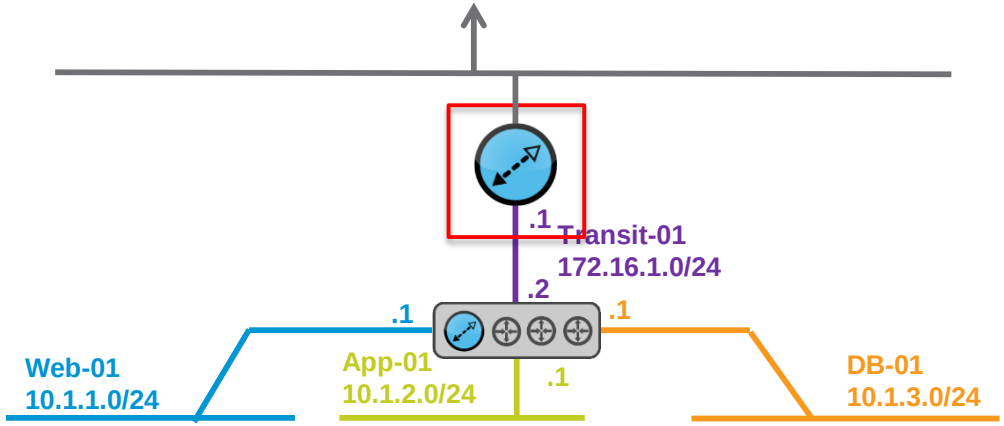
```
dimi@ubuntu-python:~$ pynsxv dlr create -n python_cli-dlr-01 --uplink_ls python_cli-Transit-01 --uplink_ip 172.16.1.2 --uplink_subnet 255.255.255.0 --uplink_dgw 172.16.1.1 --ha_ls=vDS-Mgt_PG
```

Add logical router internal interfaces

```
dimi@ubuntu-python:~$ pynsxv dlr add_interface -n python_cli-dlr-01 --interface_ls python_cli-Web-01 --interface_ip 10.1.1.1 --interface_subnet 255.255.255.0
dimi@ubuntu-python:~$ pynsxv dlr add_interface -n python_cli-dlr-01 --interface_ls python_cli-App-01 --interface_ip 10.1.2.1 --interface_subnet 255.255.255.0
dimi@ubuntu-python:~$ pynsxv dlr add_interface -n python_cli-dlr-01 --interface_ls python_cli-DB-01 --interface_ip 10.1.3.1 --interface_subnet 255.255.255.0
```

Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Creation of Edge logical router



Create edge router

```
dimi@ubuntu-python:~$ pynsxv esg create -n python_cli-edge-01 --esg_remote_access True --portgroup vDS-External_PG
```

Configure uplink + internal edge router

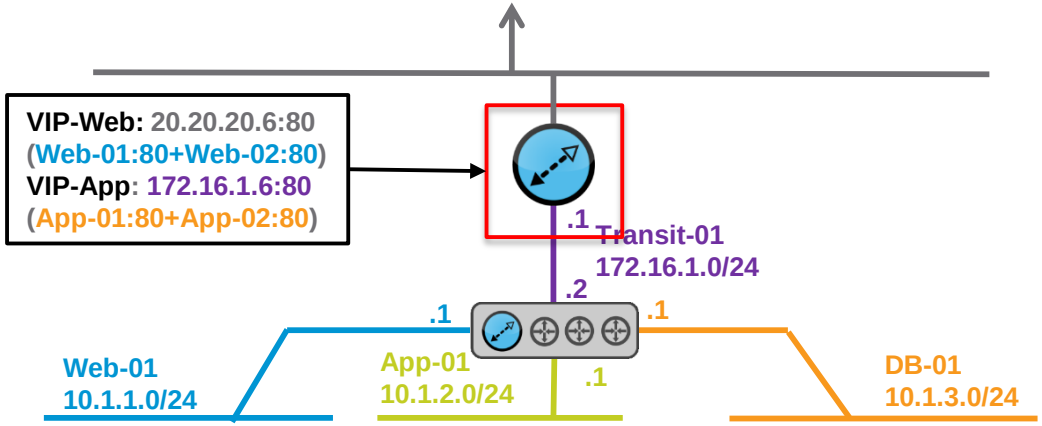
```
dimi@ubuntu-python:~$ pynsxv esg cfg_interface -n python_cli-edge-01 --portgroup vDS-External_PG --vnic_index 0 --vnic_type uplink --vnic_name External --vnic_ip 20.20.20.2 --vnic_mask 255.255.255.0
dimi@ubuntu-python:~$ pynsxv esg cfg_interface -n python_cli-edge-01 --logical_switch python_cli-Transit-01 --vnic_index 1 --vnic_type internal --vnic_name Transit --vnic_ip 172.16.1.1 --vnic_mask 255.255.255.0
```

Configure firewall + default gateway + static route

```
dimi@ubuntu-python:~$ pynsxv esg set_fw_status -n python_cli-edge-01 --fw_default accept
dimi@ubuntu-python:~$ pynsxv esg set_dgw -n python_cli-edge-01 --next_hop 20.20.20.1
dimi@ubuntu-python:~$ pynsxv esg add_route -n python_cli-edge-01 --route_net 10.1.0.0/22 --next_hop 172.16.1.2
```

Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Creation of LB



Enable load balancing

```
dimi@ubuntu-python:~$ pynsxv lb enable_lb -n python_cli-edge-01
```

Create LB Application Profile

```
dimi@ubuntu-python:~$ pynsxv lb add_profile -n python_cli-edge-01 --profile_name python_cli-lb-appprofile --
protocol HTTP
```

Create LB Pool

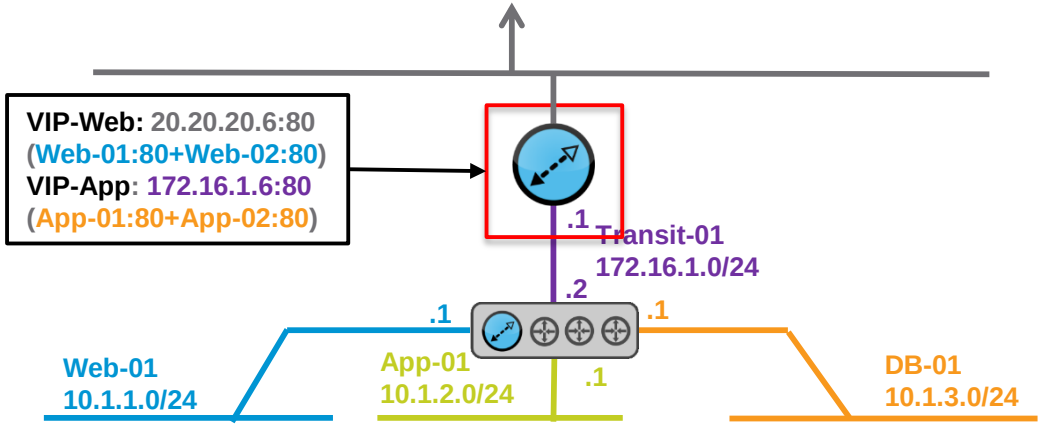
```
dimi@ubuntu-python:~$ pynsxv lb add_pool -n python_cli-edge-01 --pool_name python_cli-pool_web --transparent true
--monitor default_tcp_monitor
dimi@ubuntu-python:~$ pynsxv lb add_member -n python_cli-edge-01 --pool_name python_cli-pool_web --member_name
web01 --member 10.1.1.11 --port 80
dimi@ubuntu-python:~$ pynsxv lb add_member -n python_cli-edge-01 --pool_name python_cli-pool_web --member_name
web02 --member 10.1.1.12 --port 80
```

Create LB VIP

```
dimi@ubuntu-python:~$ pynsxv esg cfg_interface -n python_cli-edge-01 --vnic_index 0 --vnic_ip 20.20.20.2 --
vnic_mask 255.255.255.0 --vnic_secondary_ips 20.20.20.6
dimi@ubuntu-python:~$ pynsxv lb add_vip -n python_cli-edge-01 --vip_name python_cli-vip_web --pool_name
python_cli-pool_web --profile_name python_cli-lb-appprofile --vip_ip 20.20.20.6 --protocol HTTP --port 80
```

Use PyNSXv as "CLI commands" – Advanced CLI example

- Let's create a 3-Tier network + security topology
  - Creation of LB - cont



Create LB Pool

```
dimi@ubuntu-python:~$ pynsxv lb add_pool -n python_cli-edge-01 --pool_name python_cli-pool_app --monitor default_tcp_monitor
dimi@ubuntu-python:~$ pynsxv lb add_member -n python_cli-edge-01 --pool_name python_cli-pool_app --member_name app01 --member 10.1.2.11 --port 80
dimi@ubuntu-python:~$ pynsxv lb add_member -n python_cli-edge-01 --pool_name python_cli-pool_app --member_name app02 --member 10.1.2.12 --port 80
```

Create LB VIP

```
dimi@ubuntu-python:~$ pynsxv esg cfg_interface -n python_cli-edge-01 --vnic_index 1 --vnic_ip 172.16.1.1 --vnic_mask 255.255.255.0 --vnic_secondary_ips 172.16.1.6
dimi@ubuntu-python:~$ pynsxv lb add_vip -n python_cli-edge-01 --vip_name python_cli-vip_app --pool_name python_cli-pool_app --profile_name python_cli-lb-appprofile --vip_ip 172.16.1.6 --protocol HTTP --port 80
```

PyNSXv-examples (CLI + Scripts)

1. CLI examples
2. **Python script examples**

Use PyNSXv as "script" – Install pre-requirements

1. Install NSX-v RAML (RESTful API Modeling Language)
   a. Installation of git

https://git-scm.com/download/win

Windows

```
sudo apt-get install git
```

```
dimi@ubuntu-python:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.7.4-0ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 93 not upgraded.
```

Linux

Use PyNSXv as "script" – Install pre-requirements

1. Install NSX-v RAML (RESTful API Modeling Language)
   b. Install of NSX-v RAML

```
git clone https://github.com/vmware/nsxraml
```

```
C:\Windows\System32>cd \

C:\>git clone https://github.com/vmware/nsxraml
Cloning into 'nsxraml'...
remote: Counting objects: 831, done.
remote: Total 831 (delta 0), reused 0 (delta 0), pack-reused 831 eceiving objects: 100% (831/831), 2.3
Receiving objects: 100% (831/831), 2.66 MiB | 1.12 MiB/s, done.

Resolving deltas: 100% (462/462), done.
Checking connectivity... done.
```

Windows

```
dimi@ubuntu-python:~$ git clone https://github.com/vmware/nsxraml.git
Cloning into 'nsxraml'...
remote: Counting objects: 831, done.
remote: Total 831 (delta 0), reused 0 (delta 0), pack-reused 831
Receiving objects: 100% (831/831), 2.66 MiB | 555.00 KiB/s, done.
Resolving deltas: 100% (462/462), done.
Checking connectivity... done.
```

Linux

Use PyNSXv as "script" – Install pre-requirements

2. (optional) Install ipython (interactive python shell to test script)

```
pip install ipython
```

Administrator: C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>pip install ipython
Collecting ipython
  Downloading ipython-5.0.0-py2.py3-none-any.whl (743kB)
    100% |################################| 747kB 409kB/s
Collecting win-unicode-console>=0.5; sys_platform == "win32" (from ipython)
  Downloading win_unicode_console-0.5.zip
Collecting traitlets>=4.2 (from ipython)
  Downloading traitlets-4.2.2-py2.py3-none-any.whl (68kB)
    100% |################################| 71kB 114kB/s
```

Windows

```
sudo apt-get install ipython
```

```
dimi@ubuntu-python:~$ apt-get install ipython
E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root?
dimi@ubuntu-python:~$ sudo apt-get install ipython
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python-decorator python-pexpect python-ptyprocess python-simplegeneric
Suggested packages:
  ipython-doc ipython-notebook ipython-qtconsole python-matplotlib python-numpy python-zmq python-pexpect-doc
The following NEW packages will be installed:
  ipython python-decorator python-pexpect python-ptyprocess python-simplegeneric
0 upgraded, 5 newly installed, 0 to remove and 93 not upgraded.
Need to get 685 kB of archives.
After this operation, 3,741 kB of additional disk space will be used.
```

Linux

Use PyNSXv as "script" – Basic script example

- Create logical switches
  1. Find the PyNSXv python library for logical switch creation

     Go on PyNSXv github library page: https://github.com/vmware/pynsxv/tree/master/pynsxv/library

     Note: It's also on your server "<python_folder>/dist-packages/pynsxv/library"

api_spec	[#30] Bundle latest RAML spec with the installer and default to it	18 days ago
__init__.py	First Project Commit	3 months ago
libutils.py	Fixed bug in libutils check_for_parameters	10 days ago
nsx_dfw.py	[#4] Distributed Firewall functionality	10 days ago
nsx_dhcp.py	[#5] DHCP Server functionality	10 days ago
nsx_dlr.py	Change DLR because of breaking change in libutils	10 days ago
nsx_esg.py	Change ESG because of breaking change in libutils	10 days ago
nsx_logical_switch.py	[#30] Bundle latest RAML spec with the installer and default to it	18 days ago
nsx_usage.py	[#30] Bundle latest RAML spec with the installer and default to it	18 days ago

Use PyNSXv as "script" – Basic script example

- Create one logical switch
  1. Find the PyNSXv python library for logical switch creation

     Open the library

     and look at available functions

Code | Issues 25 | Pull requests 2 | Wiki | Pulse | Graphs

Branch: master ▾ **pynsxv** / pynsxv / library / nsx_logical_switch.py Find file Copy path

yfauser Modified LICENSE to include the right MIT License format 2e208fc on May 29

1 contributor

224 lines (187 sloc) | 9.78 KB Raw Blame History

```
def logical_switch_create(client_session, transport_zone, logical_switch_name, control_plane_mode=None):
    """
    This function will create a new logical switch in NSX
    :param client_session: An instance of an NsxClient Session
    :param transport_zone: The name of the Scope (Transport Zone)
    :param logical_switch_name: The name that will be assigned to the new logical switch
    :param control_plane_mode: (Optional) Control Plane Mode, uses the Transport Zone default if not specified
    :return: returns a tuple, the first item is the logical switch ID in NSX as string, the second is string
             containing the logical switch URL location as returned from the API
    """
    vdn_scope_id, vdn_scope = get_scope(client_session, transport_zone)
    assert vdn_scope_id, 'The Transport Zone you defined could not be found'
    if not control_plane_mode:
        control_plane_mode = vdn_scope['controlPlaneMode']

    # get a template dict for the lswitch create
```

Use PyNSXv as "script" – Basic script example

- Create one logical switch
  2. Build the script

```
import ConfigParser
from nsxramlclient.client import NsxClient
from pynsxv.library.nsx_logical_switch import *

# Path for Windows
# nsxraml_file = 'c:\\nsxraml\\nsxvapi.raml'
# Path for Linux
nsxraml_file = 'nsxraml/nsxvapi.raml'

# Variables for script
nsx_manager = "192.168.10.5"
nsx_username = "admin"
nsx_password = "vmware"
vcenter = "192.168.10.4"
vcenter_user = "administrator@vsphere.local"
vcenter_passwd  = "VMware1!"
transport_zone = "TZ1"
datacenter_name = "Lab1"
edge_datastore = "NFS_Lab1"
edge_cluster = "Cluster-MgtEdge"

# Connection to NSX-v Manager
client_session = NsxClient(nsxraml_file, nsxmanager, nsx_username,nsx_password, debug=False)
```

Libraries used in that script

raml file path info (select Windows / Linux format)

Specific lab settings

Connection to NSX-v Manager

Use PyNSXv as "script" – Basic script example

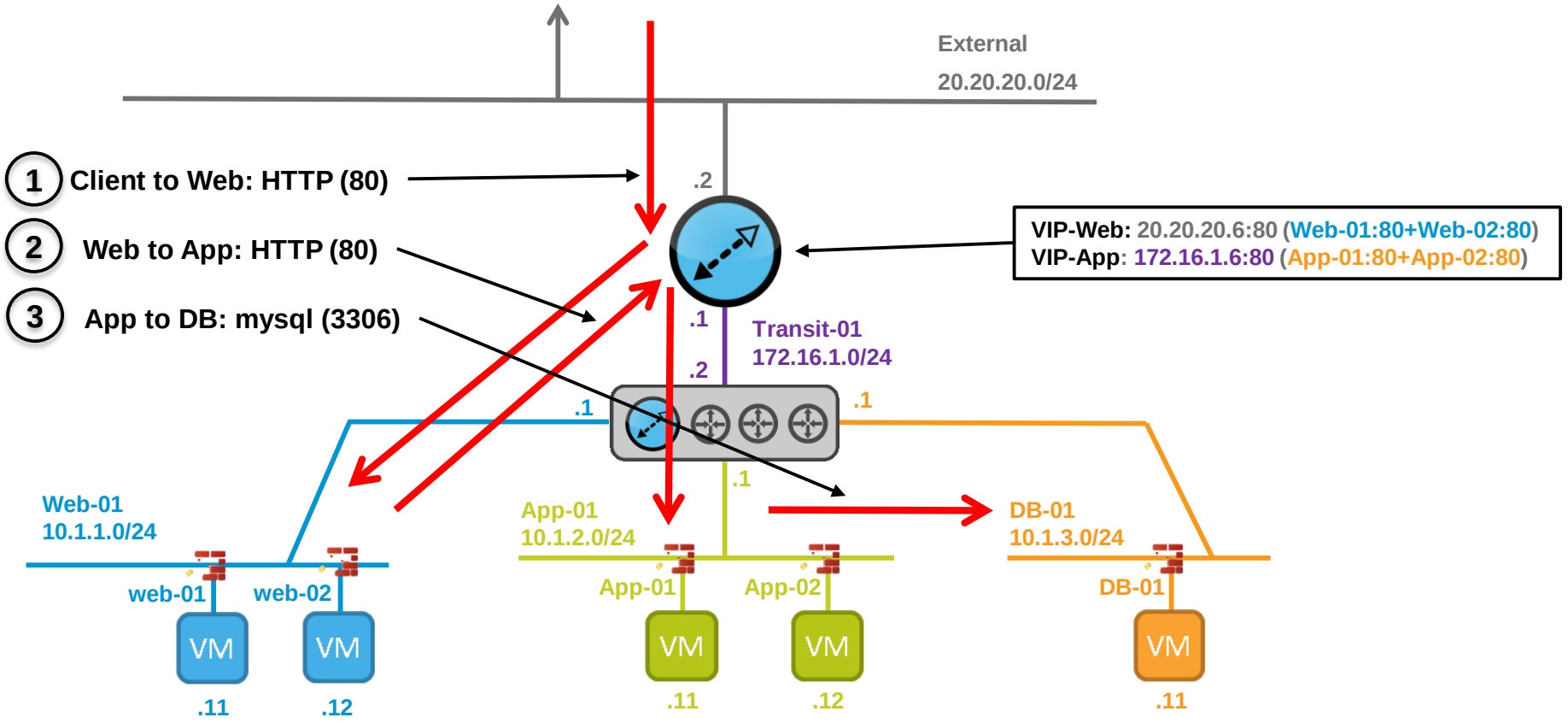
- Create one logical switch
  2. Build the script

```
# Create Logical Switch
new_ls_name = 'python-ls1'
logical_switch_create (client_session, transport_zone, new_ls_name)
print ('Logical Switch created')
```

Create logical switch

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology



Note: The whole PyNSXv python script is in the Notes.

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  1. Common section

```
import ConfigParser
from nsxramlclient.client import NsxClient
from pynsxv.library.nsx_logical_switch import *
from pynsxv.library.nsx_dlr import *
from pynsxv.library.nsx_esg import *
from pynsxv.library.nsx_dfw import *
from pynsxv.library.nsx_lb import *
from pynsxv.library.libutils import *
```

Libraries used in that script

raml file path info (select Windows / Linux format)

```
# Path for Windows
# nsxraml_file = 'c:\\nsxraml\\nsxvapi.raml'
# Path for Linux
nsxraml_file = 'nsxraml/nsxvapi.raml'
```

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  1. Common section – cont.

```
# Read nsx.ini file
config = ConfigParser.ConfigParser()
config.read("nsx.ini")
nsxmanager = config.get('nsxv', 'nsx_manager'); nsx_username = config.get('nsxv', 'nsx_username'); nsx_password
= config.get('nsxv', 'nsx_password')
vcenter = config.get('vcenter', 'vcenter'); vcenter_user = config.get('vcenter', 'vcenter_user');
vcenter_passwd = config.get('vcenter', 'vcenter_passwd')
transport_zone = config.get('defaults', 'transport_zone')
datacenter_name = config.get('defaults', 'datacenter_name'); edge_datastore = config.get('defaults',
'edge_datastore'); edge_cluster = config.get('defaults', 'edge_cluster')
# Collect vCenter MoID information
vccontent = connect_to_vc(vcenter, vcenter_user, vcenter_passwd)
datacentermoid = get_datacentermoid(vccontent, datacenter_name)
datastoremoid = get_datastoremoid(vccontent, edge_datastore)
resourcepoolid = get_edgeresourcepoolmoid(vccontent, edge_cluster)

# Connection to NSX-v Manager
client_session = NsxClient(nsxraml_file, nsxmanager, nsx_username,nsx_password, debug=False)
print ('Connection to NSX-v Manager')
```

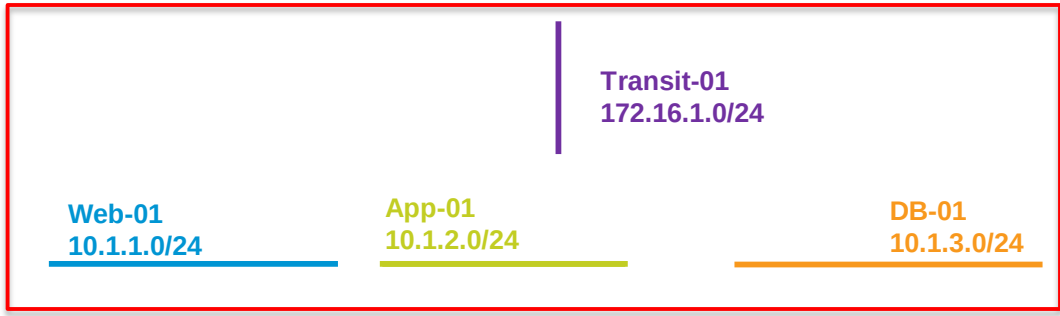
Specific lab settings (NSX-v and vCenter)

Connection to NSX-v Manager

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  2. Network and security creation: LS

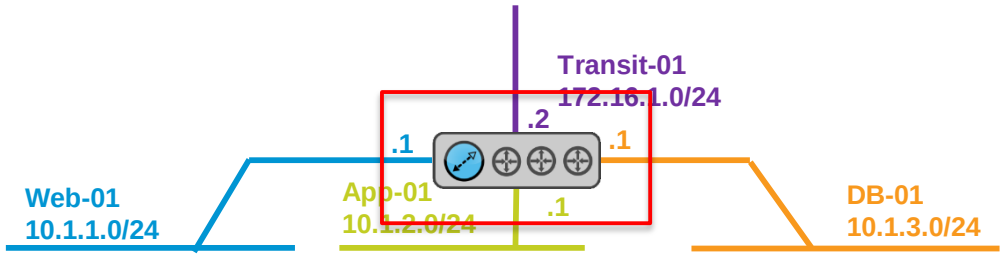
```
# Create 3 Tier-Switches
new_ls_name = 'python-web01'
logical_switch_create (client_session, transport_zone, new_ls_name)
new_ls_name = 'python-app01'
logical_switch_create (client_session, transport_zone, new_ls_name)
new_ls_name = 'python-db01'
logical_switch_create (client_session, transport_zone, new_ls_name)
new_ls_name = 'python-transit01'
logical_switch_create (client_session, transport_zone, new_ls_name)
print ('4 Logical Switches created')
```



Create logical switches

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  2. Network and security creation: DLR



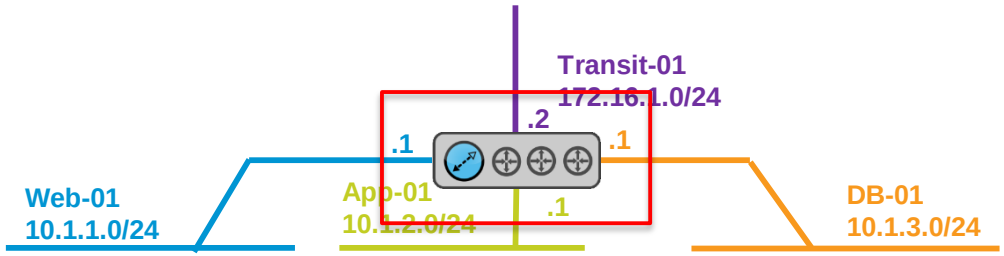

```
# Create Dist Router
### DLR settings
new_dlr_name = "python-dlr01"
new_dlr_pwd = "VMware1!VMware1!"
new_dlr_size = "compact"
ha_ls_name = "vDS-Mgt_PG"
uplink_ls_name = "python-transit01"
uplink_ip = "172.16.1.2"
uplink_subnet = "255.255.255.0"
uplink_dgw = "172.16.1.1"

ha_ls_id = get_vdsportgroupid(vccontent, ha_ls_name)
uplink_ls_id,null = get_logical_switch(client_session, uplink_ls_name)

### Create the DLR
dlr_create (client_session, new_dlr_name, new_dlr_pwd, new_dlr_size,
               datacentermoid, datastoremoid, resourcepoolid,
               ha_ls_id, uplink_ls_id, uplink_ip, uplink_subnet, uplink_dgw)
```

Enter DLR settings

Create logical router

Use PyNSXv as "script" – Advanced script example

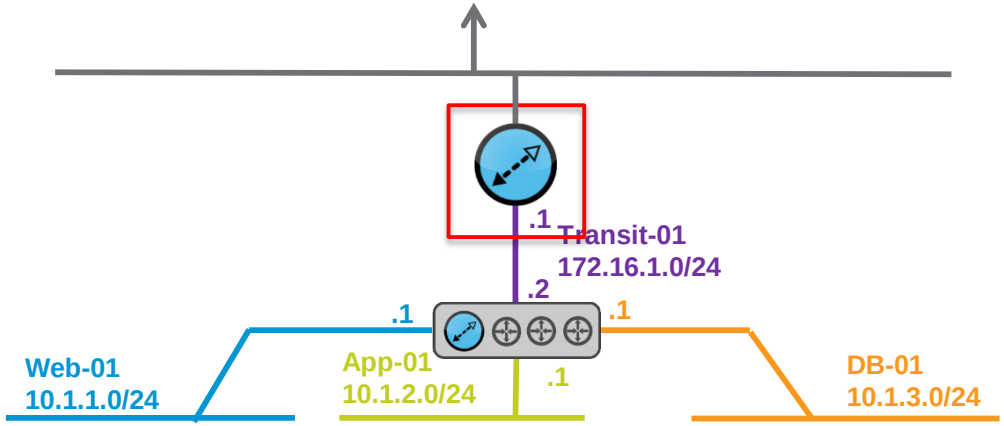
- Create a 3-Tier network + security topology
  2. Network and security creation: DLR – cont.




```
### Add DLR internal interface1
interface_ls_name = "python-web01"
interface_ip = "10.1.1.1"
interface_subnet = "255.255.255.0"
dlr_id, null = dlr_read(client_session, new_dlr_name)
interface_ls_id,null = get_logical_switch(client_session, interface_ls_name)
dlr_add_interface(client_session, dlr_id, interface_ls_id, interface_ip, interface_subnet)

### Add DLR internal interface2
interface_ls_name = "python-app01"
interface_ip = "10.1.2.1"
interface_subnet = "255.255.255.0"
dlr_id, null = dlr_read(client_session, new_dlr_name)
interface_ls_id,null = get_logical_switch(client_session, interface_ls_name)
dlr_add_interface(client_session, dlr_id, interface_ls_id, interface_ip, interface_subnet)

### Add DLR internal interface3
interface_ls_name = "python-db01"
interface_ip = "10.1.3.1"
interface_subnet = "255.255.255.0"
dlr_id, null = dlr_read(client_session, new_dlr_name)
interface_ls_id,null = get_logical_switch(client_session, interface_ls_name)
dlr_add_interface(client_session, dlr_id, interface_ls_id, interface_ip, interface_subnet)
print ('DLR created')
```

Add logical router internal interfaces

Use PyNSXv as "script" – Advanced script example



- Create a 3-Tier network + security topology
  3. Network and security creation: Edge

```
# Create Edge Service Router
### Edge settings
esg_name = "python-edge01"
esg_username = "admin"
esg_pwd = "VMware1!VMware1!"
esg_size = "compact"
default_pg = "vDS-External_PG"
esg_remote_access = "True"

default_pg_id = get_vdsportgroupid(vccontent, default_pg)

### Create the Edge
esg_create(client_session, esg_name, esg_pwd, esg_size, datacentermoid, datastoremoid, resourcepoolid,
               default_pg_id, esg_username, esg_remote_access)

### Configure Edge uplink interface
ifindex = "0"
ipaddr = "20.20.20.21"
netmask = "255.255.255.0"
vnic_type = "uplink"
esg_cfg_interface(client_session, esg_name, ifindex, ipaddr, netmask, vnic_type=vnic_type)
```

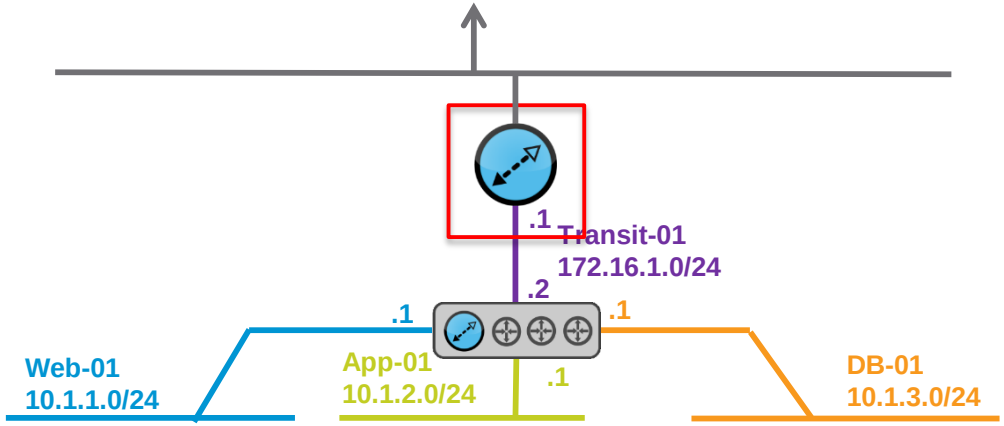
Enter edge settings

Create edge router

Configure uplink edge router

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  3. Network and security creation: Edge – cont.




```
### Add Edge internal interface1
ifindex = "1"
ipaddr = "172.16.1.1"
netmask = "255.255.255.0"
vnic_type = "internal"
interface_ls_name = "python-transit01"
interface_ls_id,null = get_logical_switch(client_session, interface_ls_name)
esg_cfg_interface(client_session, esg_name, ifindex, ipaddr, netmask, is_connected=True,
portgroup_id=interface_ls_id, vnic_type=vnic_type)

### Configure Edge Firewall default rule "Accept"
esg_fw_default_set(client_session, esg_name, "accept", logging_enabled=None)

### Configure Edge default gateway
dgw_ip = "20.20.20.1"
esg_dgw_set(client_session, esg_name, dgw_ip, "0")

### Configure Edge static route
subnet = "10.1.0.0/22"
next_hop = "172.16.1.2"
esg_route_add(client_session, esg_name, subnet, next_hop, "1")

print ('Edge created')
```

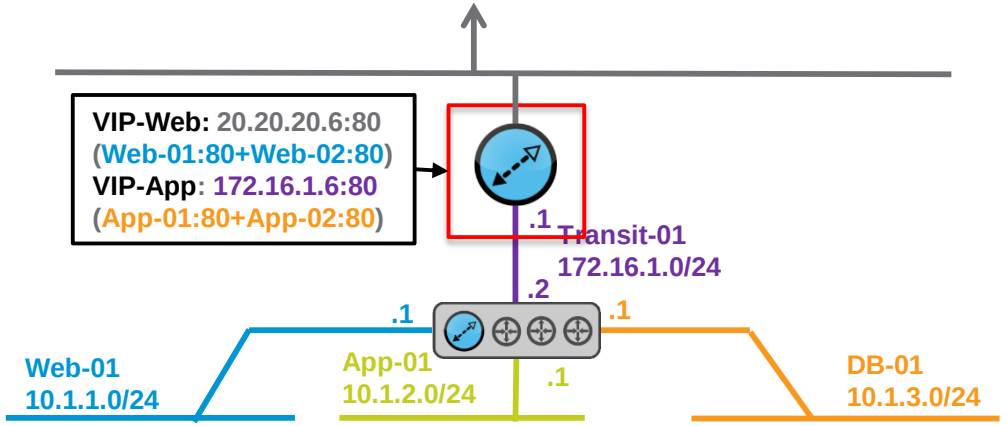
Add logical router internal interfaces

Configure Edge default FW "Accept"

Configure Edge default gateway

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  4. Network and security creation: Load Balancing




```
# Configure Load Balancing
### Enable Load Balancing on the Edge
load_balancer(client_session, esg_name, enabled=True)

### Create the LB Application Profile
lb_app_profile = "python-ap_http"
lb_proto = "HTTP"
add_app_profile(client_session, esg_name, lb_app_profile, lb_proto)

### Create the LB Web Pool
lb_pool_name = "python-pool_web"
lb_monitor = "default_tcp_monitor"
add_pool(client_session, esg_name, lb_pool_name, monitor=lb_monitor)
add_member(client_session, esg_name, lb_pool_name, "web01", "10.1.1.11", port="80")
add_member(client_session, esg_name, lb_pool_name, "web02", "10.1.1.12", port="80")

### Create the LB Web VIP
lb_vip_name = "python-vip_web"
lb_vip_ip = "20.20.20.6"
lb_vip_port = "80"
esg_cfg_interface(client_session, esg_name, "0", "20.20.20.2", "255.255.255.0", secondary_ips=lb_vip_ip)
add_vip(client_session, esg_name, lb_vip_name, lb_app_profile, lb_vip_ip, lb_proto, lb_vip_port, lb_pool_name)
print 'VIP {} created with {}:{}'.format(lb_vip_name, lb_vip_ip, lb_vip_port)
```

Enable Load Balancing

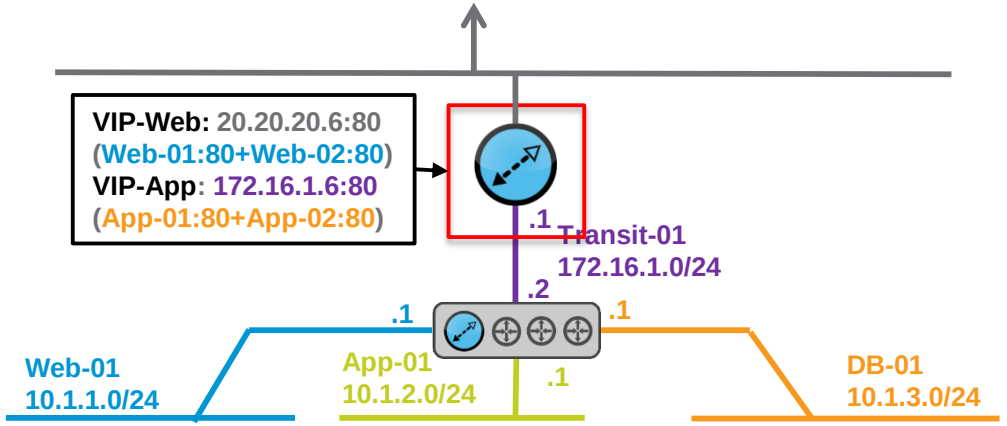
Create LB Application Profile

Create LB Web-Pool

Create LB Web-VIP

Use PyNSXv as "script" – Advanced script example

- Create a 3-Tier network + security topology
  4. Network and security creation: Load Balancing – cont



Create LB App-Pool

```
### Create the LB App Pool
lb_pool_name = "python-pool_app"
lb_monitor = "default_tcp_monitor"
add_pool(client_session, esg_name, lb_pool_name, monitor=lb_monitor)
add_member(client_session, esg_name, lb_pool_name, "app01", "10.1.2.11", port="80")
add_member(client_session, esg_name, lb_pool_name, "app02", "10.1.2.12", port="80")
```

Create LB App-VIP

```
### Create the LB Web VIP
lb_vip_name = "python-vip_app"
lb_vip_ip = "172.16.1.6"
lb_vip_port = "80"
esg_cfg_interface(client_session, esg_name, "1", "172.16.1.1", "255.255.255.0", secondary_ips=lb_vip_ip)
add_vip(client_session, esg_name, lb_vip_name, lb_app_profile, lb_vip_ip, lb_proto, lb_vip_port, lb_pool_name)
print 'VIP {} created with {}:{}'.format(lb_vip_name, lb_vip_ip, lb_vip_port)
```