Evolving to a CRUD+V NSO package

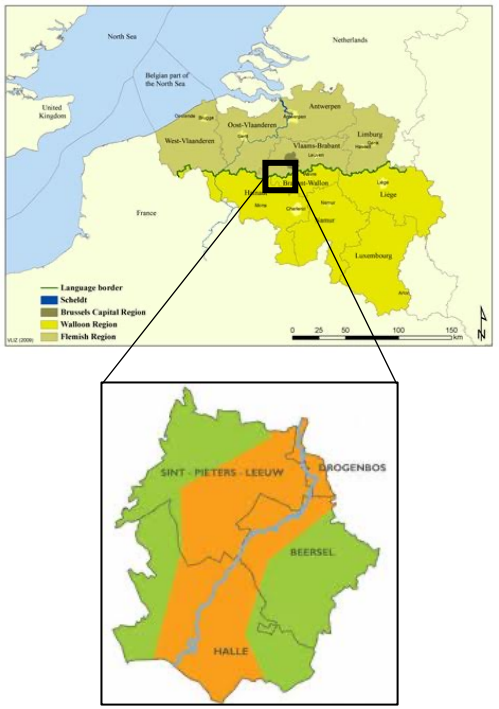
in search of a perfect marriage

dstny

05/2023

Alain Pieters
alain.pieters@dstny.com

Introduction



https://en.wikipedia.org/wiki/Gueuze
https://boon.be/
https://www.horal.be/

oude geuze

black label
(special edition)

horal
(megablend)

Oude Kriek

CML2

aka VIRL

NSO

NSO in docker

pyATS
print(network.profile)

Genie
Python Test Automation

Used seperately

dstny

```
11:53:36-apieters@as8368-netdevops-vm:/opt/docker/nso/builds/nso-service-dev/ipsec_ikev1_l2l (master)$ echo $NSO_VERSION
```

Files

A perfect marriage

testbed setup

CRUD + V

testbed teardown

```
robot -L trace -e update -d ~/robot/QA robot/Test_Cases/
==============================================================================
Test Cases
==============================================================================
06-04-2023 02:06
starting the NSO docker container (This could take a few minutes...)
   06-04-2023 02:10
Starting virtual lab topology on CML2 server (this could take several minutes...)
   06-04-2023 02:13
NSO staging: configuring devices on NSO and syncing devices (no sleep as only IOS-XE devices...)
Test Cases.Functional Tests
==============================================================================
Test Cases.Functional Tests.NSO crud operations :: NSO API test suites
==============================================================================
[VERIFY] if NSO is responding to REST API calls                       | PASS |
------------------------------------------------------------------------------
[CREATE] NSO service instance                                         | PASS |
------------------------------------------------------------------------------
[VERIFY] NSO service instance                                         | PASS |
------------------------------------------------------------------------------
[DELETE] NSO service instance                                         | PASS |
------------------------------------------------------------------------------
Test Cases.Functional Tests.NSO crud operations :: NSO API test su... | PASS |
4 tests, 4 passed, 0 failed
==============================================================================
Test Cases.Functional Tests                                           | PASS |
4 tests, 4 passed, 0 failed
==============================================================================
06-04-2023 02:17
Stopping the NSO docker container
06-04-2023 02:17
Stopping virtual lab topology on CML2 server
Test Cases                                                            | PASS |
4 tests, 4 passed, 0 failed
==============================================================================
Output:  /home/apieters/robot/QA/output.xml
Log:     /home/apieters/robot/QA/log.html
Report:  /home/apieters/robot/QA/report.html
Stop Log - 02:22:05
```

Idempotent

NSO in Docker Framework

$DEFAULT_TESTENV=cml2

1. make testenv-start
2. make testenv-start-cml2
3. make testenv-nso-device-sync

REST API via Northbound API of NSO +pyATS action in NSO service

NSO in Docker Framework

$DEFAULT_TESTENV=cml2

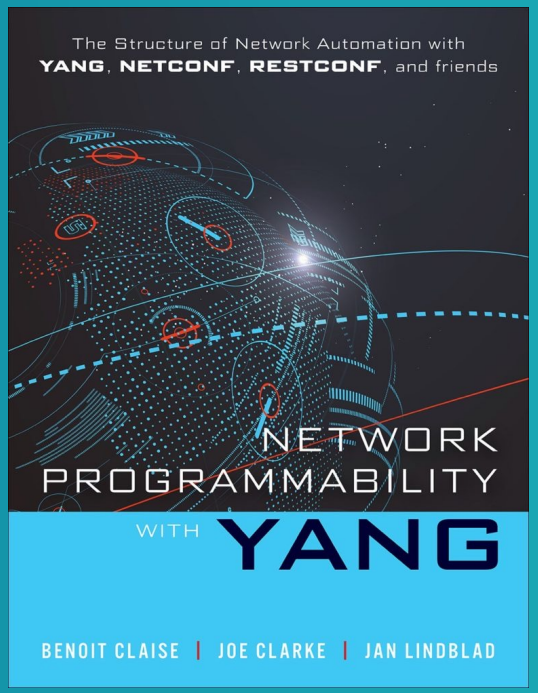
1. make testenv-stop
2. make testenv-stop-cml2

dstny

Agenda

- Intro - Demo
- #01 – Networking automation is as good as…
- #02 – Service models are more then CRUD only
- #03 - Test automation: Bringing it all into a test automation pipeline
- Wrapping Up

#01 Networking automation is as good as...



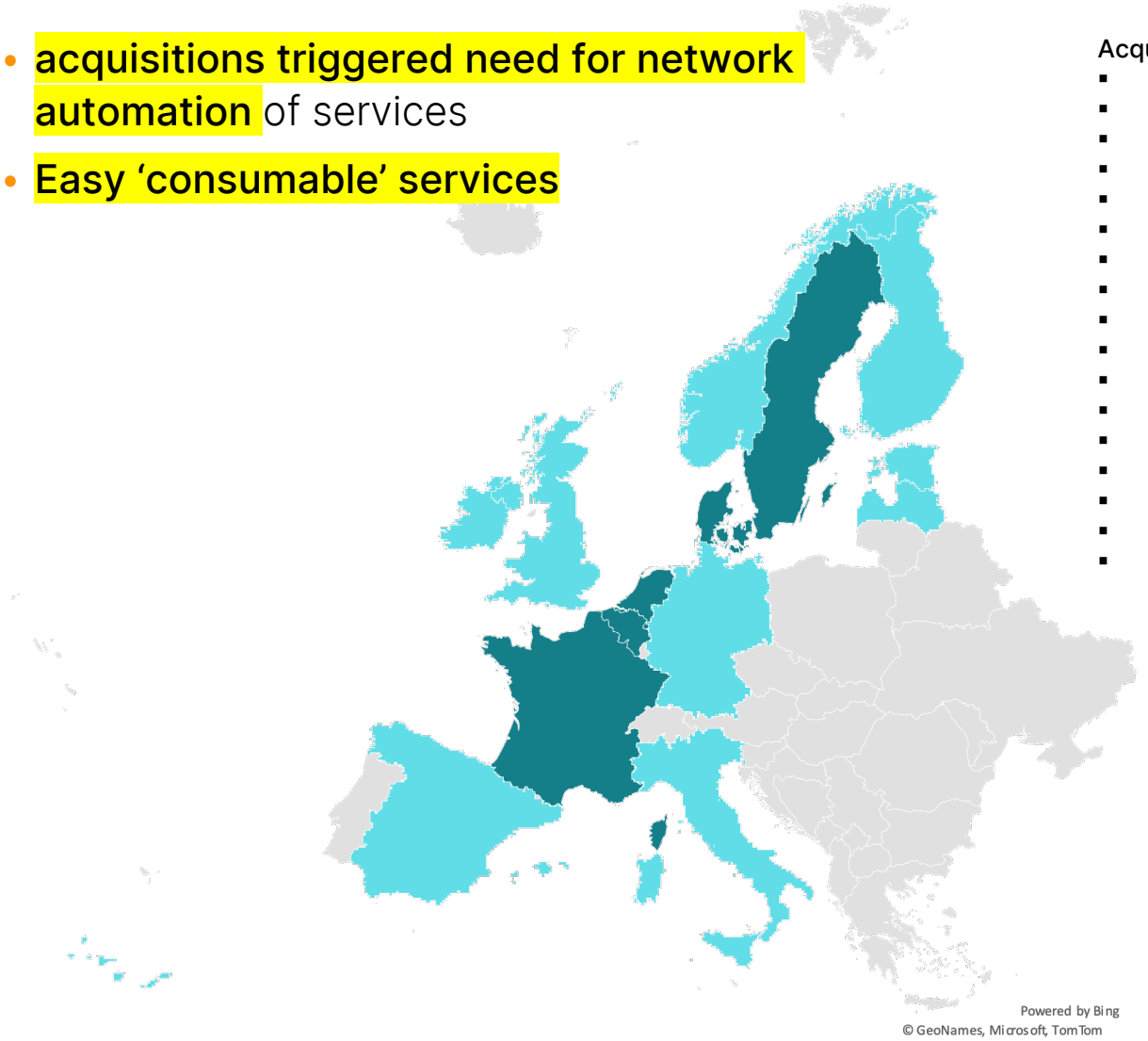
https://www.claise.be/automation-is-as-good-as/

dstny

about

dstny

- acquisitions triggered need for network automation of services
- Easy ‘consumable’ services



Acquisitions over the last 5 years:

- Flexfone (DK)
- Easybell (D)
- Tactful AI (UK)
- Soluno (SW)
- Lynes (SW)
- Fuzer (B)
- Escaux (B)
- Ulysse (B)
- Belgium Telecom (B)
- Voips (NL)
- Motto (NL)
- Panas (NL)
- Ozmo (NL)
- DSD (NL)
- OpenIP (FR)
- IPLine (FR)
- …

NETWORK AUTOMATION via NSO

NSO local / system install

- First steps into network automation (2019)
- No big monolethic network service model
- service deployment via API | GUI | CLI
- drawback -> snowflakes everywhere (local install/service coding, etc...)

NSO in docker

https://github.com/NSO-developer/nso-docker

- NSO version upgrades
- **Skeletons**
  - Base / NED / **package** / system
  - **Testenvs**
- integrated in GITLAB:
  - ✓ CI/CD pipelines
  - ✓ Container registry

dstny

NSO Package Skeleton
“Everything” is easier on a small thing

```
├── extra-files
├── includes
├── nid
├── packages
│   └── ipsec_ikev1_l2l
│       ├── python
│       ├── src
│       ├── templates
│       └── test
├── robot
│   ├── Library
│   ├── Resources
│   ├── Results
│   ├── Test_Cases
│   │   └── Functional_Tests
├── testenvs
│   ├── cml2
│   └── quick
└── test-packages
```

```
├── includes
│   ├── bgp
│   ├── bgp-redist-static
│   └── vrf
│
│   ├── cisco-ios-cli-6.72
│   ├── cisco-iosxr-cli-7.18
│   ├── nokia-sros-nc-21.10.r1
│   └── oracle-sbc
```

more includes/cisco-iosxr-cli-7.18
**${NED_PATH}**cisco-iosxr-cli/**package**-cisco-iosxr-cli-7.18:$NSO_VERSION

more includes/vrf
**${PKG_PATH}**vrf/**package**:$NSO_VERSION

nidvars.mk
export NSO_IMAGE_PATH ?= *<my_registry>*/...../nso-base/
export PKG_PATH ?= *<my_registry>*/..../service_packages/
export NED_PATH ?= *<my_registry>*/..../neds/

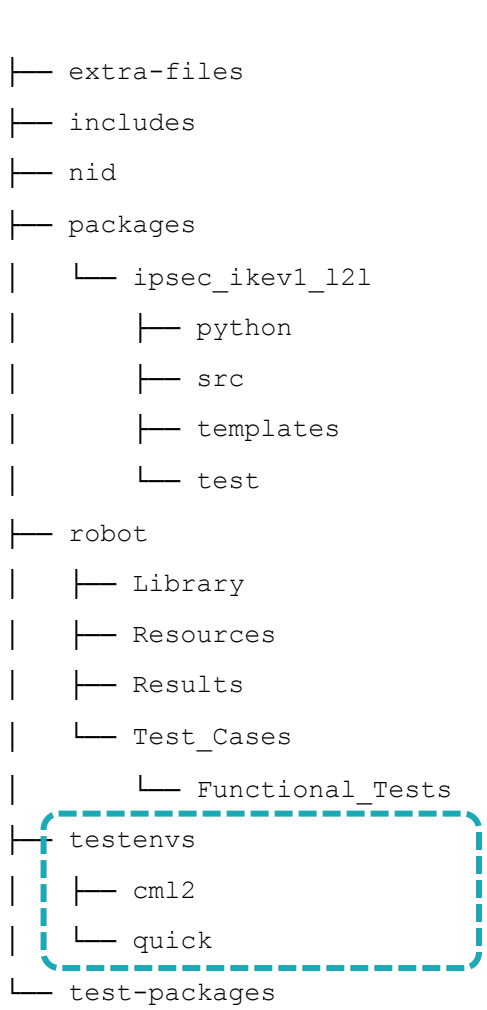
README Package Skeleton:
https://github.com/NSO-developer/nso-docker/blob/master/skeletons/package/README.nid-package.org

development improvement!
- always latest code changes
lasted container image will be pulled from the registry when building the container image for the package

see addendum #01

dstny

NSO in Docker Testenvs



$NSO_VERSION

$DEFAULT_TESTENV

- quick (CRUD only)
  - targets:
    - start/stop
    - shell / dev-shell
    - cli / runcmdC / runcmdJ
    - rebuild / clean-rebuild
- cml2
  - identical environment as quick - added **3 extra targets**
    - **start-cml2 / stop-cml2**
    - **nso-staging**

dstny

Testenv CML2

```
start-cml2:
	@echo "\n== start up CML2 lab environment via terraform"
	@echo "\n== env vars loaded in shell $TF_VAR_cml2_username"
	terraform init
	terraform plan
	terraform apply --auto-approve

stop-cml2:
	@echo "\n== destroy CML2 lab environment"
	terraform destroy --auto-approve

nso-staging:
	@echo "\n== sync devices within NSO - sleep timer used ASR9000v devices"
	sleep 1000
	$(MAKE) loadconf FILE="devices.xml"
	$(MAKE) runcmdC CMD="config\ndevices fetch-ssh-host-keys\nexit\n"
	$(MAKE) runcmdC CMD="config\ndevices sync-from\nexit\n"
	$(MAKE) updatepyats (*_*)
```

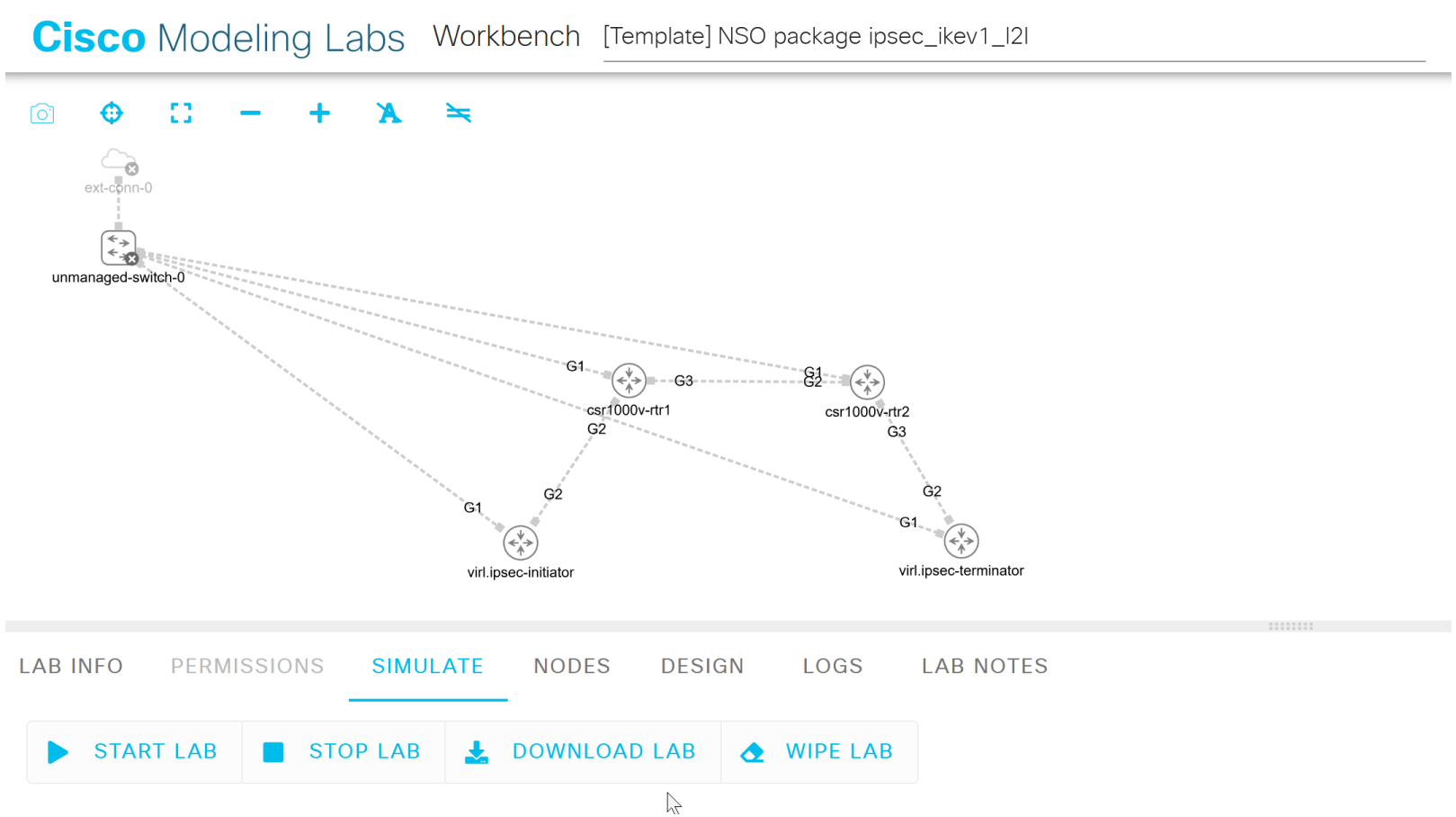
Terraform CML2 provider plugin

```
.../testenvs/cml2 (master)$ tree
.
├── authgroup.xml
├── cml2_lab_topology.yaml
├── cml_testbed.yaml
├── devices.xml
├── main.tf
├── Makefile
├── nso_local_user.xml
├── terraform.tfstate
└── terraform.tfstate.backup
```

References:

- https://blogs.cisco.com/learning/get-started-with-terraform-and-cisco-modeling-labs
- https://github.com/CiscoDevNet/terraform-provider-cml2

CML2 Lab Export
Tweak it to your needs!



cml2_lab_topology.yaml

default configuration settings of devices have to be countered in the yaml
i.e.

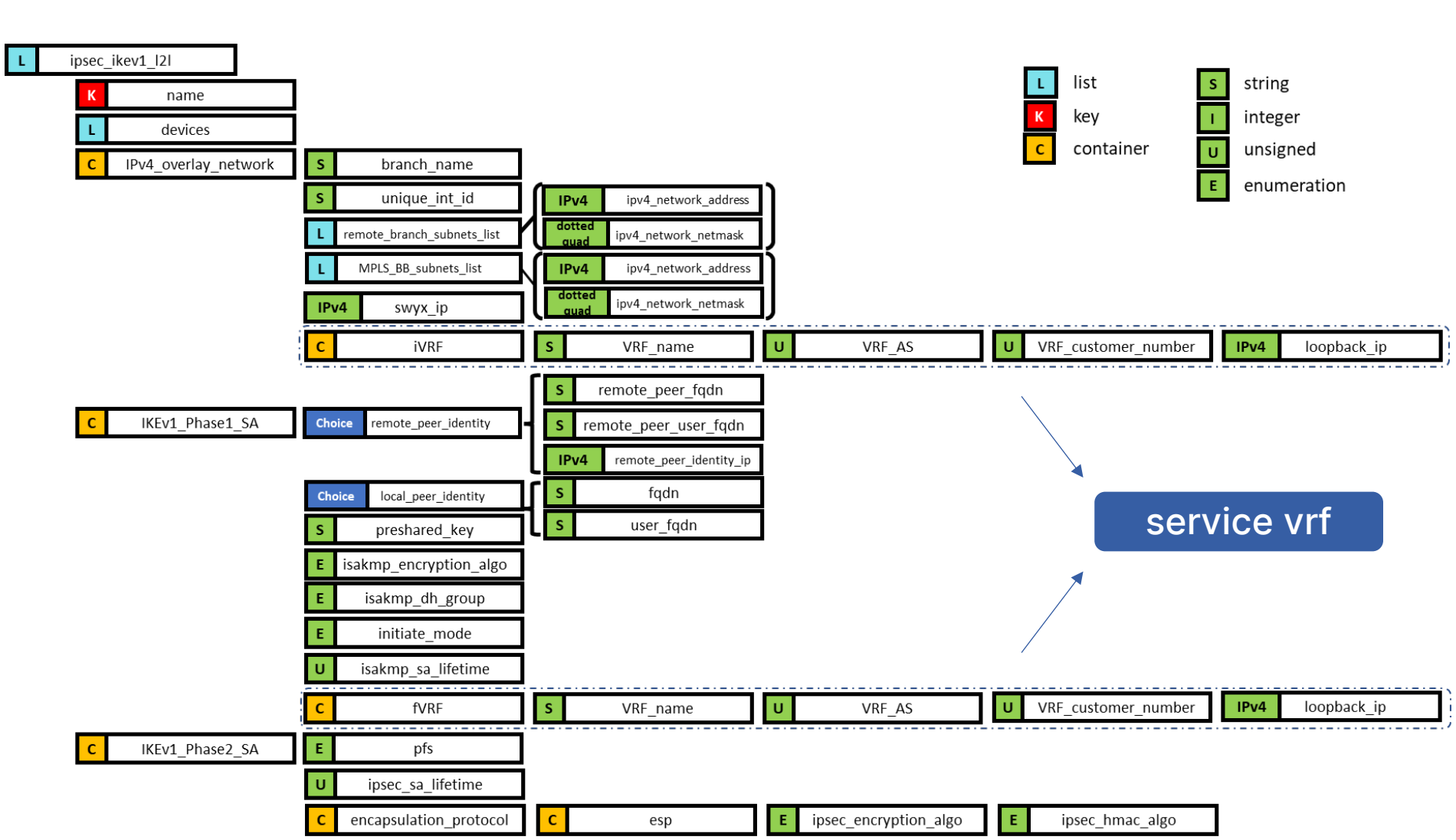
- default configuration settings of an interface is in shutdown state
- crypto keys for ssh
  - in IOS-XE, just add crypto key generate rsa to it

dstny

#02
Service models are more then CRUD only

dstny

CRUD is business as usual

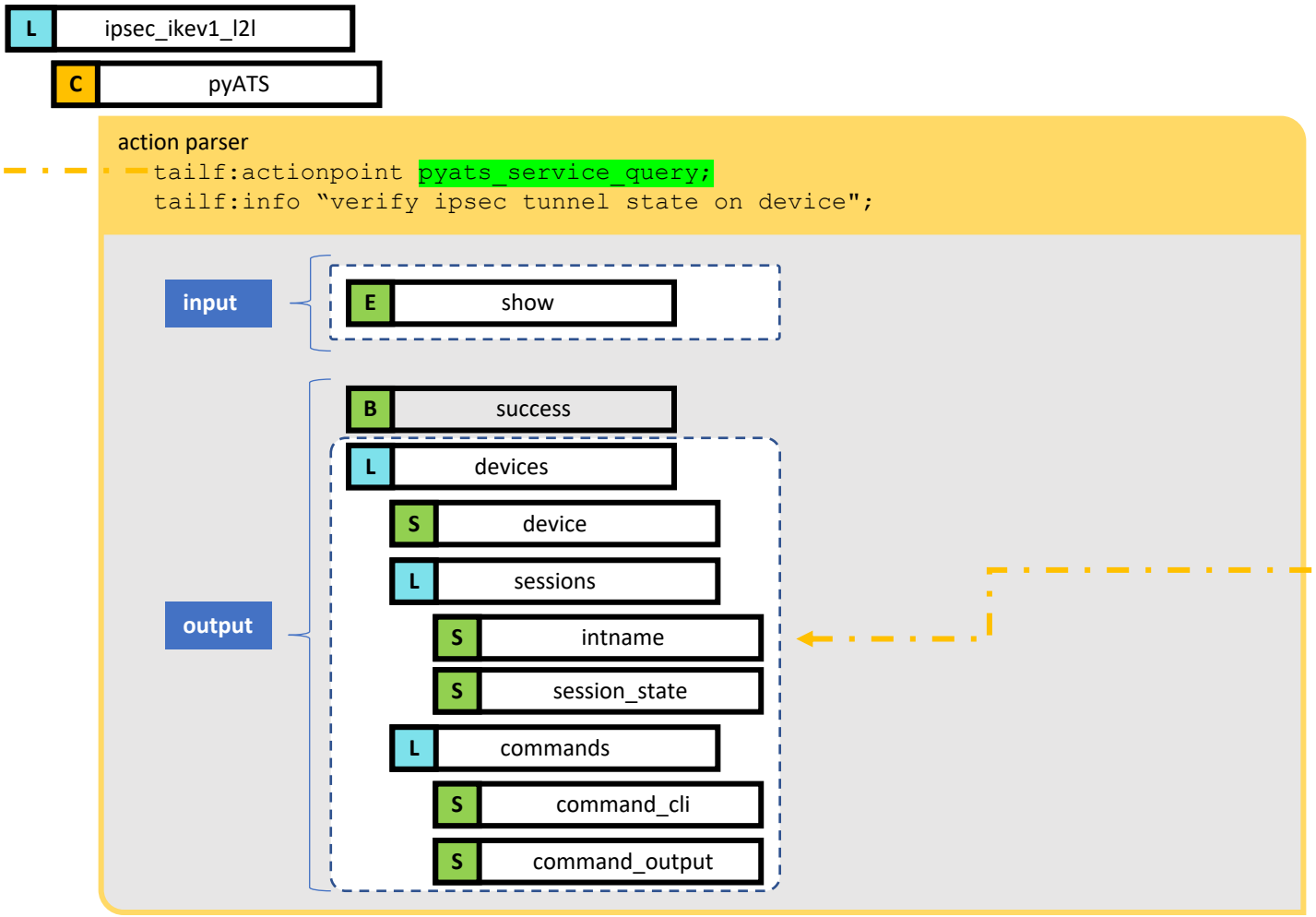


YANG service model for ipsec_ikev1_l2l service

see addendum #02 for more information

dstny

extend YANG model with a Verify action



YANG element type:

- L list
- K key
- C container
- S string
- I integer
- U unsigned
- E enumeration
- B boolean

About action:

- **NSO documentation**:
  Most common way to implement non-configuration automation in NSO
- **RFC7950** – section **7.15**:
  used to define operations tied to data nodes
  input and **output** are substatements of this YANG type

main.py

```
from .pyATS_parser import pyATSParserClass

class Main(ncs.application.Application):
    def setup(self):
        self.log.info('Main RUNNING')

        self.register_service('ipsec_ikev1_l2l-servicepoint', ServiceCallbacks)
        self.register_action('pyats_service_query', pyATSParserClass)
```

pyATS_parser.py

```
from crypt import crypt
import ncs
import json
from ncs.dp import Action
import subprocess

class pyATSParserClass(Action):

    @Action.action
    def cb_action(self, uinfo, name,
    kp, input, output, trans):

        #output is linked to yang model
        pyATS parser output!
```

Doing Verification inside NSO
Different ways

- NSO Device live-status -> unparsed (see demo 07:45)
- Incorporate pyATS into NSO -> parsed
    1. using a python library

       Reference: https://github.com/rtrjl/nso_live_status
        - integrated with NSO devices CDB!
    2. extending Genie parser or pyATS (constructor)

       separate python venv for pyATS under nso-base docker image (see addendum)

       Advantage:
        - separate environments that can be managed independently from each other
        - extend parser or add constructor

       Disadvantage:
        - pyATS device yaml file to be generated -> reason: see (*_*) later

NSO Action
pyATS parser Class – working with YANG input

From within the action, a python script is called within the pyATS env

```
venv_python = '/pyats/bin/python'
pyats_nso_main_script='/<pyats_script_folder>/NSO_CMD_Launcher.py'
```

Script needs a list of devices and a list of commands as input

```
nso_node = ncs.maagic.get_node(trans, kp)
qservice = nso_node._parent
```

```
device_dict=root.ipsec_ikev1_l2l[qservice.name].device
```

```
command_set={
  'all':
     [f'show crypto session ivrf
      {root.ipsec_ikev1_l2l[qservice.name].IPv4_overlay_network.iVRF.VRF_name} detail',
      f'show ip route vrf
      {root.ipsec_ikev1_l2l[qservice.name].IPv4_overlay_network.iVRF.VRF_name}
      {root.ipsec_ikev1_l2l[qservice.name].IPv4_overlay_network.swyx_ip}'
     ]
}
```

enum entries for each key in command_set directory

input

E show

```
leaf show {
  type enumeration {
    enum "all";
}
```

```
spawncc=subprocess.run(
    [
      venv_python,
      pyats_nso_main_script,
      '--device_list',
      device_list,
      '--command_list',
      command_list
    ],
    capture_output=True
)
```

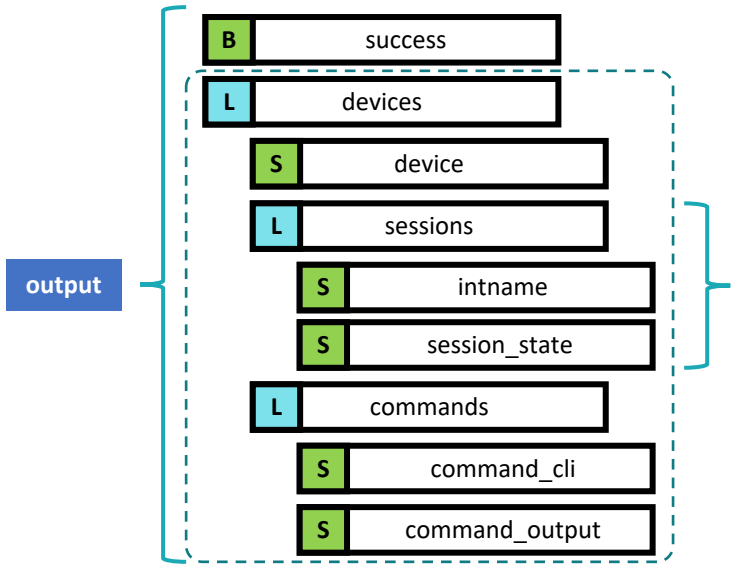
```
python NSO_CMD_Launcher.py --help
usage: NSO_CMD_Launcher.py [-h] -d DEVICE_LIST -c COMMAND_LIST

optional arguments:
  -h, --help            show this help message and exit
  -d DEVICE_LIST, --device_list DEVICE_LIST
                        list of remote devices that should be queried
  -c COMMAND_LIST, --command_list COMMAND_LIST
                        command list to be parsed
```


dstny

NSO Action
pyATS parser Class – working with YANG output

```
if spawncc.returncode != 0:
 output.success=False
 raise Exception (
   "pyATS python script execution return code was not 0!\n"
   f"Output: {str(spawncc.stderr)}"
 )
else:
 output.success=True
```

```
# JSON.loads:Serialising output spawncc.stdout from pyats script into python dict
pyats_parser_load.update(json.loads(spawncc.stdout))
```




```
if each_command.startswith("show crypto session"):
  for each_int in pyats_parser_load[each_device][each_command]['interface']:
    output.devices[str(each_device)].sessions.create(each_int)
     output.devices[str(each_device)].sessions[each_int].session_state=
     pyats_parser_load[each_device][each_command]['interface'][each_int]['session_status']
```

Operations can pull this data via API too

dstny

NSO Feature Request

- Parsing power within NSO
  - use_genie flag like within netmiko in live status?

```
def send_command(self, command_string: str, expect_string: Optional[str] = None,
                 read_timeout: float = 10.0, delay_factor: Optional[float] = None,
                 max_loops: Optional[int] = None, auto_find_prompt: bool = True,
                 strip_prompt: bool = True, strip_command: bool = True,
                 normalize: bool = True, use_textfsm: bool = False,
                 textfsm_template: Optional[str] = None, use_ttp: bool = False,
                 ttp_template: Optional[str] = None, use_genie: bool = False,
                 cmd_verify: bool = True) -> Union[str, List[Any], Dict[str, Any]]
```

  - control on what commands are send to what devices
  - link output with yang service model (pyATS>JSON <> NSO> YANG)
  - Allow Development on pyATS framework

dstny

#03
Test automation: Bringing it all into a test automation pipeline

dstny

Test Automation under package skeleton

ROBOT FRAMEWORK

```
├── extra-files
├── includes
├── nid
├── packages
├── robot/
│       ├── Library
│       ├── Resources
│       │       └── testbed.resource
│       ├── Results
│       └── Test_Cases
│               ├── Functional_Tests
│               │       ├── NSO_crud_operations.robot
│               │       ├── service_create.json
│               │       ├── service_update.json
│               │       └── service_verify.json
│               └── __init__.robot
├── testenvs
│   ├── cml2
│   └── quick
└── test-packages
```

- Hierachical Test Suite Structure
  - **Suite initialization** with **__init__.robot**
  - Test case files are organized into **directories**, and these directories create higher-level test suites
    - [allows to skip suite initialisation]

__init__.robot

```
Resource            ../Resources/testbed.resource
Suite Setup         Start Testenv
Suite Teardown      Stop Testenv

*** Keywords ***
#run via package skeleton Makefile
Start Testenv
    ${rc_nso_start}    ${stdout}    Run And Return Rc And Output        make testenv-start
    Should Not Contain  ${stdout}       ${SPACE}Error      ignore_case=True

    ${rc_nso_start}    ${stdout}    Run And Return Rc And Output        make testenv-start-cml2
    Should Not Contain  ${stdout}       ${SPACE}Error      ignore_case=True

    ${rc_nso_start}    ${stdout}    Run And Return Rc And Output        make testenv-nso-device-sync
    Should Not Contain  ${stdout}       ${SPACE}Failed      ignore_case=True

Stop Testenv
    ${rc_nso_start}    ${stdout}    Run And Return Rc And Output        make testenv-stop
    Should Not Contain  ${stdout}       Makefile      ignore_case=True
    ${rc_nso_stop}    ${stdout}    Run And Return Rc And Output        make testenv-stop-cml2
    Should Not Contain  ${stdout}       Error       ignore_case=True
```

Test Automation under package skeleton testbed.resource

- Robot Libraries
  - Standard:
    - Selenium
    - OperatingSystem
    - Collections
    - String
    - DateTime
  - Others:
    - RequestLibrary
    - JSONLibrary

- Initialisation of internal variables

```
*** Settings ***
Documentation   NSO API testsuite -  ipsec_ikev1_L2L package
Library         SeleniumLibrary
Library         OperatingSystem
Library         RequestsLibrary
Library         JSONLibrary
Library         Collections
Library         String
Library         DateTime

*** Variables ***
#NSO_ENDPOINT           ---
${AUTH}                 /auth
#HEADERS                ---
${CHARSET}              utf-8
${CONTENT_TYPE}         application/yang-data+json
${ACCEPT}               application/yang-data+json
&{HEADERS}=             Content-Type=${CONTENT_TYPE}
...                     User-Agent=RobotFramework
...                     charset=${CHARSET}
...                     Accept=${ACCEPT}

# NSO package skeleton variable
# FROM BASH env variable
${default_testenv}      %{DEFAULT_TESTENV}
${nso_version}          %{NSO_VERSION}
# FOR AUTH on NSO       ---
${NSO_USERNAME}         %{NSO_USERNAME}
${NSO_PASSWD}           %{NSO_PASSWD}
```

Test Automation under package skeleton REST API's come for free

robot framework tests will trigger northbound API for all CRUD+V operations

NSO URI:

```
{{PROTOCOL}}://{{NSO_IP}}:{{NSO_HTTP_PORT}}/restconf/data/{{URI_END}}
and {{NSO_SERVICE}}==ipsec_ikev1_l2l:ipsec_ikev1_l2l
```

	HTTP Method	{{URI_END}}	JSON BODY (*)
CREATE	POST		requires {{NSO_SERVICE}} + Non-Existing Key Value -> name
READ	GET	{{NSO_SERVICE}}=<service key value>	/
UPDATE	PATCH		Must contain NSO service name {{NSO_SERVICE}} + existing Key Value name
DELETE	DELETE	{{NSO_SERVICE}}=<service key value>	/
VERIFY	POST	{{NSO_SERVICE}}=<service key value>/pyATS/parser	{ "input": {"show":"all"} }

(*) see addendum on how to create the JSON Body for a service

dstny

Robot CRUD Operations – Test Init Test Setup / Test Teardown

```
*** Settings ***
Documentation   NSO API test suites
Library  Process
Library  OperatingSystem

Resource    ../../Resources/testbed.resource

Test Setup            Find docker IP
Test Teardown         Teardown NSO API Session
```

Docker inspect command will return the IP address of the testNSO container used in the test

```
*** Keywords***
Find docker IP

    ${cwd}=     Split String From Right  ${EXECDIR}     /
    ${host_ip}=        Run     docker inspect -f {{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}} testenv-${cwd}[-1]-${default_testenv}-
                               ${nso_version}-${current_user}-nso

    IF  "${host_ip}"!=""

        Set Suite Variable   ${HTTPS_LOCAL_SERVER}    https://${host_ip}

    ELSE
        FATAL ERROR   Fatal Error: value of API port is None - no API call can be send to NSO!
    END

Teardown NSO API Session
    Delete All Sessions
```


dstny

Robot CRUD+V Operations -1
NSO reponding to API? + Create Service

```
*** Test Cases ***
[VERIFY] if NSO is responding to REST API calls
    [Tags]  get

    #open TCP session to NSO docker instance
    ${auth}=  Create List  ${NSO_USERNAME}  ${NSO_PASSWD}
    Create Session  nso_session  ${HTTPS_LOCAL_SERVER}  auth=${auth}  headers=${HEADERS}  verify=False  disable_warnings=1

    ${resp}=  GET On Session  nso_session  url=/restconf/data  params=depth=1  expected_status=200
```

refer to tag in robot command line option: -i <tag> or –e <tag>

```
[CREATE] NSO service instance
    [Tags]  create

    #API POST construct where json body is loading service_create.json
    File Should Exist  ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_create.json  msg="json service file not found!"
    ${body}=  Get File  ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_create.json

    ${auth}=  Create List  ${NSO_USERNAME}  ${NSO_PASSWD}
    Create Session  nso_session  ${HTTPS_LOCAL_SERVER}  auth=${auth}  headers=${HEADERS}  verify=False  disable_warnings=1

    ${resp}=  Post On Session  nso_session  url=/restconf/data/  data=${body}  expected_status=201

    sleep  1m
```

see addendum

dstny

Robot CRUD+V Operations - 2
UPDATE + DELETE Service

```
[UPDATE] NSO service instance
    [Tags]  update
    File Should Exist   ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_update.json      msg="json file not found!"
    ${body}=          Get File  ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_update.json

    ${auth}=  Create List  ${NSO_USERNAME}   ${NSO_PASSWD}
    Create Session  nso_session  ${HTTPS_LOCAL_SERVER}   auth=${auth}  headers=${HEADERS}   verify=False    disable_warnings=1

    ${resp}=    Patch On Session  nso_session     url=/restconf/data/  data=${body}
```

```
[DELETE] NSO service instance
    [Tags]  delete
    ${auth}=  Create List  ${NSO_USERNAME}   ${NSO_PASSWD}
    Create Session  nso_session  ${HTTPS_LOCAL_SERVER}   auth=${auth}  headers=${HEADERS}   verify=False    disable_warnings=1

    ${resp}=    Delete On Session  nso_session     url=/restconf/data/ipsec_ikev1_l2l:ipsec_ikev1_l2l=AGG_TUNNEL_IP
```


dstny

Robot CRUD+V Operations - 3
Verify Service and Fail if session_state !="UP-ACTIVE"

```
[VERIFY] NSO service instance
    [Tags]  verify

    File Should Exist    ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_verify.json      msg="json file not found!"
    ${body}=          Get File  ${EXECDIR}/robot/Test_Cases/Functional_Tests/service_verify.json

    ${auth}=  Create List  ${NSO_USERNAME}    ${NSO_PASSWD}
    Create Session  nso_session  ${HTTPS_LOCAL_SERVER}    auth=${auth}  headers=${HEADERS}    verify=False    disable_warnings=1

    ${response}=    Post On Session  nso_session     url=/restconf/data/ipsec_ikev1_l2l:ipsec_ikev1_l2l=AGG_TUNNEL_IP/pyATS/parser    data=${body}

    # NOT valid json format when parsed in URL https://jsonpathfinder.com/
    # response.json() -> it is a python dict!!

    ${session_state list}=    Get Value From Json    ${response.json()}     $..devices.[0].sessions.[0].session_state     fail_on_empty=${True}
    ${session_state} =        Convert to String      ${session_state list}[0]

    # alternatively -> use response.content
    # valid json format when parsed in URL https://jsonpathfinder.com/

    # ${dict}=          Evaluate      json.loads('''${response.content}''')          json
    # ${session_state}=       Get Value From Json     ${dict}      $..devices.[0].sessions.[0].session_state       fail_on_empty=${True}


    Should Be Equal As Strings         ${session_state}        UP-ACTIVE
```

see addendum

dstny

Wrapping Up

Closing Comments Summary

- What do you think about this blend of pyATS/NID/CML2?
  - A perfect Marriage?
  - adding more flavours to it (terraform plugin for CML2/robot).
- Automation is as good as your data models, their related metadata, your toolchain, *and* what you can do with them
  - First time getting all things right is hard
  - easy reusable for other NSO service packages (add inside skeleton package).
- Sharing Responsibilities
  - package development (NetDev) / robot test cases (Ops)

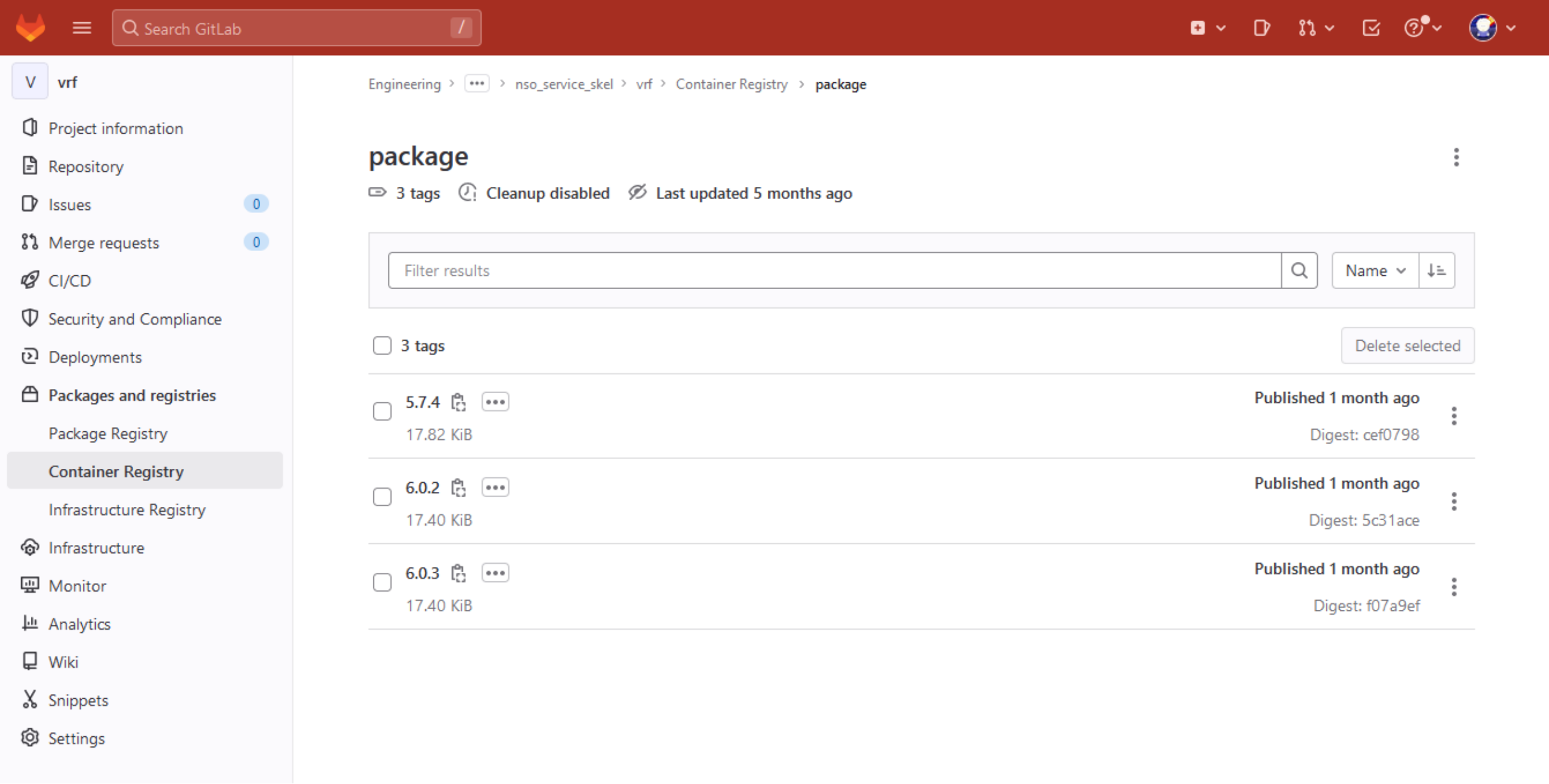
Thank you.

Alain Pieters

alain.pieters@dstny.com
www.linkedin.com/in/alainpieters/

demo
Addendum

#01
Addendum

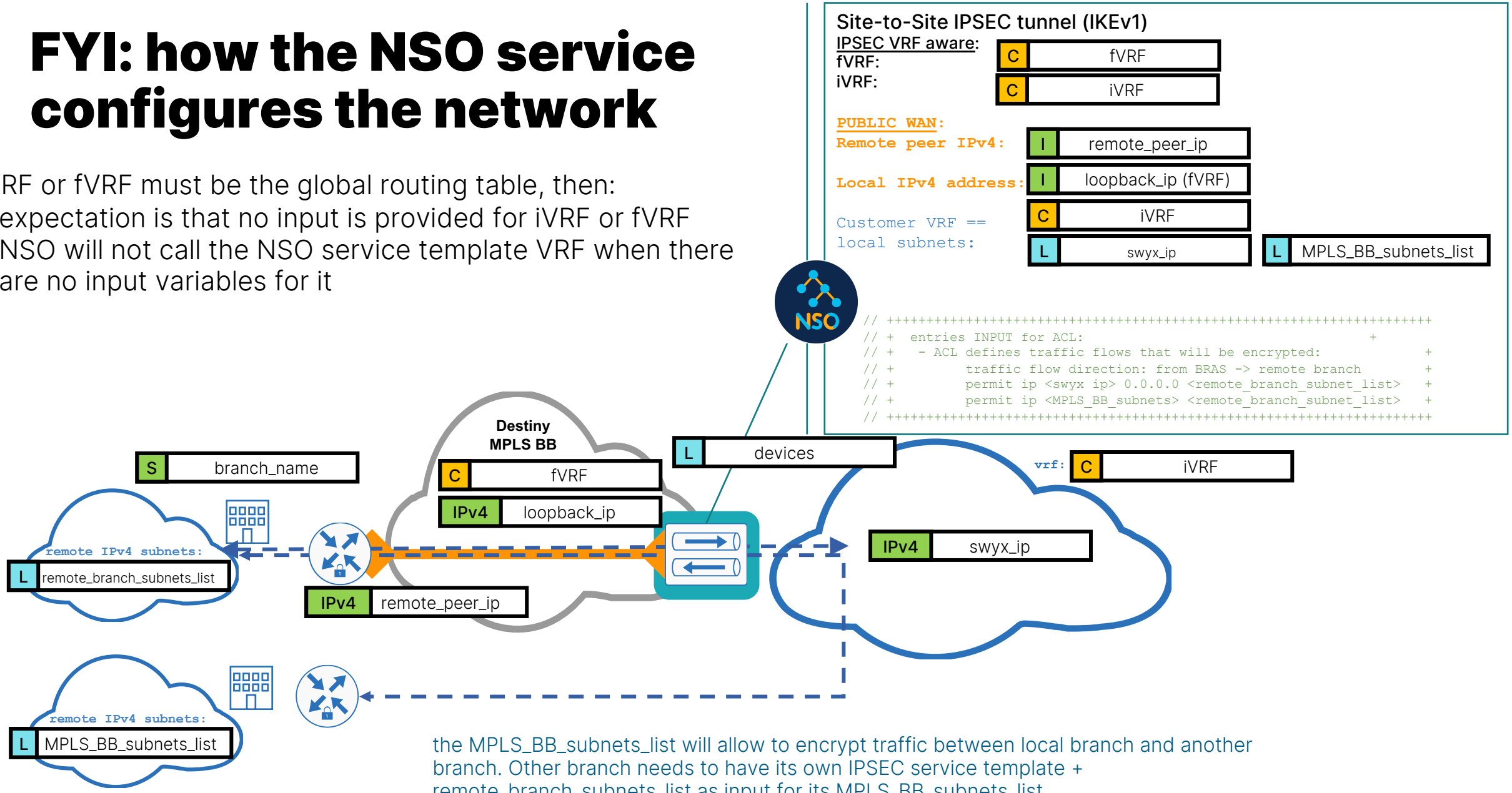

Search GitLab /
V vrf
Project information
Repository
Issues 0
Merge requests 0
CI/CD
Security and Compliance
Deployments
Packages and registries
Package Registry
Container Registry
Infrastructure Registry
Infrastructure
Monitor
Analytics
Wiki
Snippets
Settings
Engineering > ••• > nso_service_skel > vrf > Container Registry > package
package
3 tags
Cleanup disabled
Last updated 5 months ago
Filter results
Name
3 tags
Delete selected
5.7.4
17.82 KiB
Published 1 month ago
Digest: cef0798
6.0.2
17.40 KiB
Published 1 month ago
Digest: 5c31ace
6.0.3
17.40 KiB
Published 1 month ago
Digest: f07a9ef

#02
Addendum

FYI: how the NSO service configures the network

if iVRF or fVRF must be the global routing table, then:

1. expectation is that no input is provided for iVRF or fVRF
2. NSO will not call the NSO service template VRF when there are no input variables for it



the MPLS_BB_subnets_list will allow to encrypt traffic between local branch and another branch. Other branch needs to have its own IPSEC service template + remote_branch_subnets_list as input for its MPLS_BB_subnets_list
Traffic between branches is hairpinned over the IPSEC aggregator.

dstny

NSO pyATS Service Verification
adding pyATS / Genie into NSO base container

docker-images/Dockerfile

```
FROM debian:buster AS nso_install
...
# pyats workspace location
# PYATS python venv install example
# https://github.com/CiscoTestAutomation/pyats-docker/release/Dockerfile
ARG WORKSPACE
ENV WORKSPACE ${WORKSPACE:-/pyats}
# copy wheel files into this container
COPY pyats /src
# create virtualenv and install pyats packages
RUN apt-get update
RUN apt-get install -y --no-install-recommends iputils-ping telnet curl build-
essential python3-pip git
RUN pip3 install --upgrade --no-cache-dir setuptools pip virtualenv
RUN virtualenv ${WORKSPACE}
RUN ${WORKSPACE}/bin/pip install --no-cache-dir --upgrade pip setuptools wheel
RUN ${WORKSPACE}/bin/pip install -r /src/requirements.txt
...

FROM debian:buster AS deb_base
#adding pyats framework underneath /pyats as a python venv
COPY --from=nso_install /pyats /pyats
...
FROM debian:buster AS nso_install
#adding pyats framework underneath /pyats as a python venv
COPY --from=nso_install /pyats /pyats
```

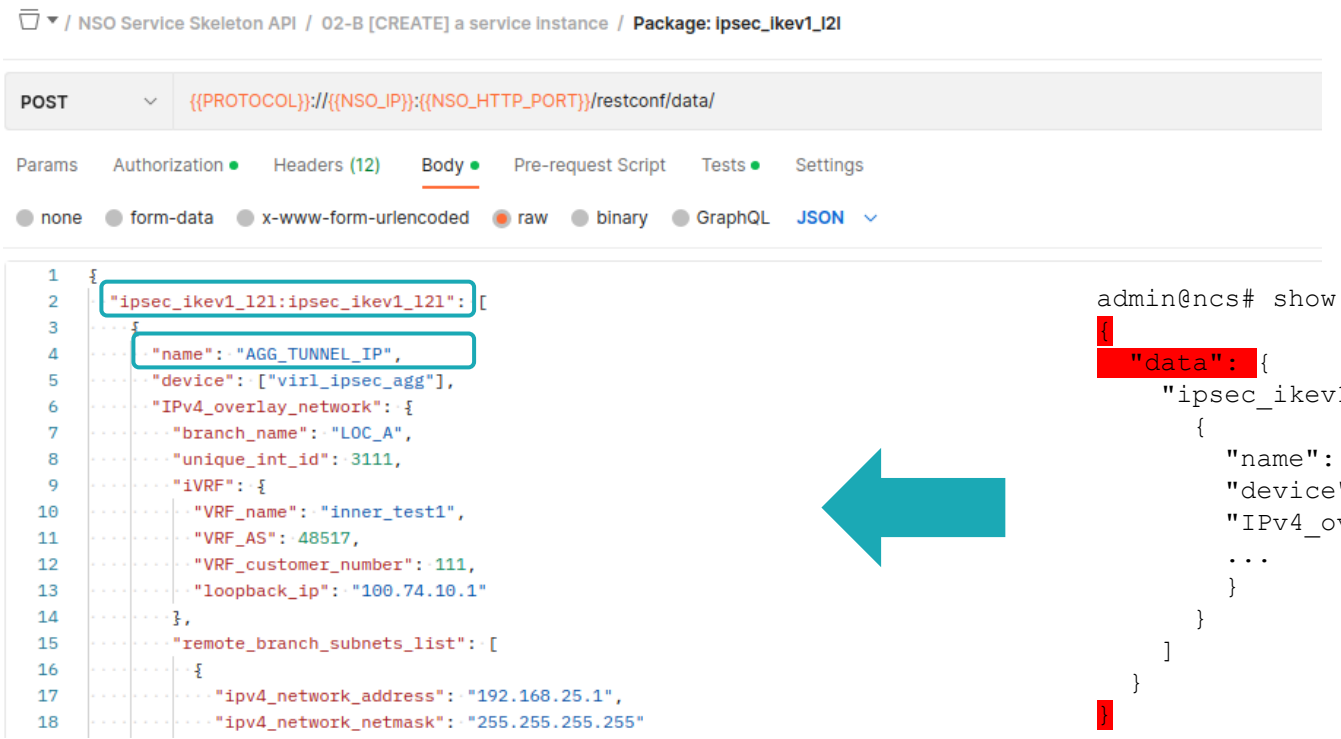
requirements.txt points to project directory on your remote git repo:

```
-e git+https:// project_y_bot:<secret>
@<your_gitlab_server>/<parent_dir>/
genieparser.git@master#egg=genie.libs.parser

-e git+https:// project_y_bot:<secret>@ <your_gitlab_server>/
<parent_dir>/ unicon.plugins.git@master#egg=unicon.plugins
```


dstny

#03
Addendum

Test Automation under package skeleton Collecting JSON Body from NSO




```
admin@ncs# show running-config ipsec_ikev1_l2l AGG_TUNNEL_IP | display json
{
  "data": {
    "ipsec_ikev1_l2l:ipsec_ikev1_l2l": [
      {
        "name": "AGG_TUNNEL_IP",
        "device": ["virl_ipsec_agg"],
        "IPv4_overlay_network": {
        ...
        }
      }
    ]
  }
}
```

https://jsonpathfinder.com/

JSON Path Finder

Sample | Beautify | Minify

```
{
  "ipsec_ikev1_l2l:output": {
    "success": true,
    "devices": [
      {
        "device": "virl_ipsec_agg",
        "sessions": [
          {
            "intname": "Tunnel3111",
            "session_state": "UP-ACTIVE"
          }
        ],
        "commands": [
          {
            "command_cli": "show crypto session ivrf inner_test1 detail",
            "command_output": "{'interface': {'Tunnel3111': {'profile': 'ISAKMP-inner_test1-LOC_A',
          },
          {
            "command_cli": "show ip route vrf inner_test1 100.74.10.1",
            "command_output": "{'entry': {'100.74.10.1/32': {'ip': '100.74.10.1', 'mask': '32', 'kr
          }
        ]
      }
    ]
  }
}
```

Path: `x["ipsec_ikev1_l2l:output"].devices[0].sessions[0]` Copy

- ipsec_ikev1_l2l:output:
  - success: true
  - devices:
    - 0:
      - device: virl_ipsec_agg
      - sessions:
        - 0:
          - intname: Tunnel3111
          - session_state: UP-ACTIVE
        - commands:

Created by Joe Beach

dstny