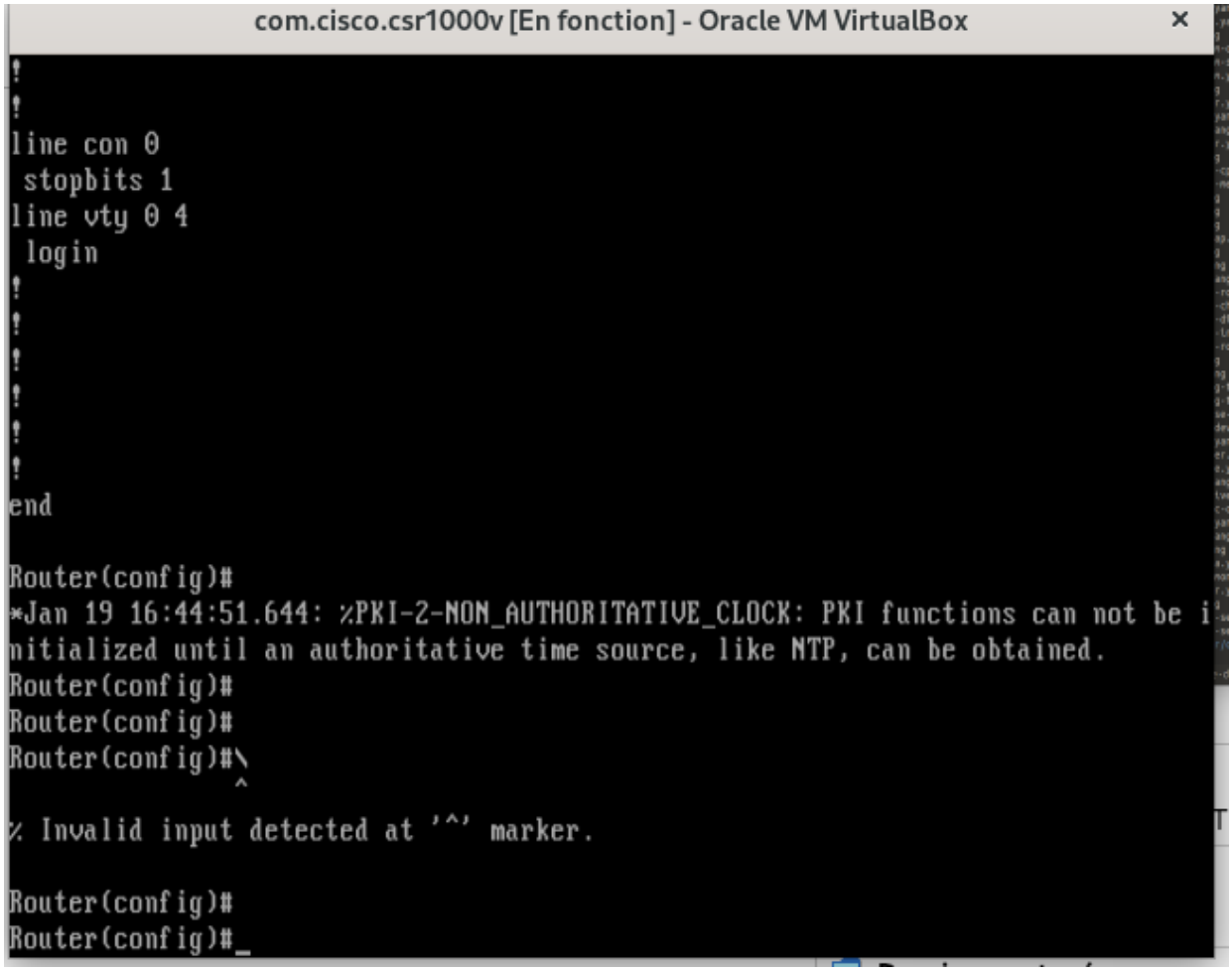
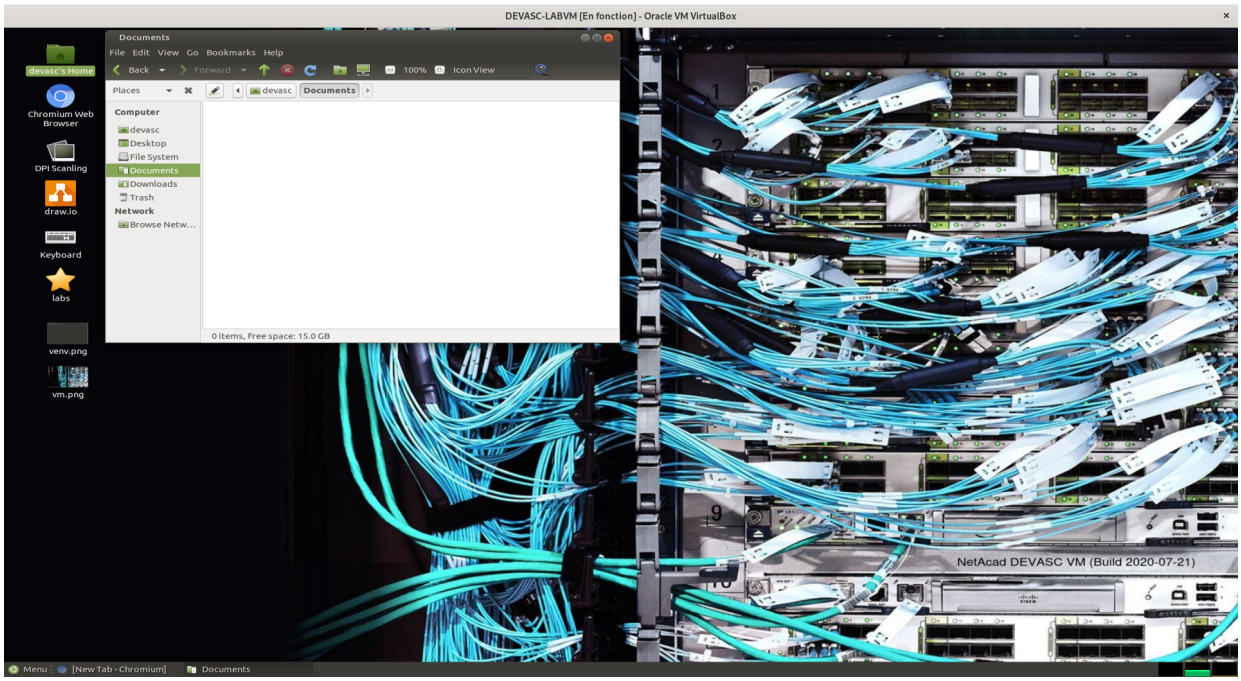
Install the CSR1000v VM

C1000 :





Tests automatisés utilisant PYATS et Genie

3. Utiliser la bibliothèque de tests PyATS

2. Verifying pyATS.

Lors de l'éxécution de la commande "pyats --help" nous avons un retour de l'utilisation de la commande pyats ce qui veut dire que pyats c'est bien installé.

3. Cloner et examiner les exemples de scripts PyATS de GitHub.

a)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ git clone
https://github.com/CiscoTestAutomation/examples
Cloning into 'examples'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 658 (delta 11), reused 18 (delta 4), pack-reused 623
Receiving objects: 100% (658/658), 1.00 MiB | 4.82 MiB/s, done.
Resolving deltas: 100% (338/338), done.
```

b)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ ls -l
total 24
drwxrwxr-x 2 devasc devasc 4096 May 31 16:07 bin
drwxrwxr-x 21 devasc devasc 4096 May 31 16:47 examples
drwxrwxr-x 2 devasc devasc 4096 May 31 16:07 include
drwxrwxr-x 3 devasc devasc 4096 May 31 16:07 lib
lrwxrwxrwx 1 devasc devasc 3 May 31 16:07 lib64 -> lib
-rw-rw-r-- 1 devasc devasc 69 May 31 16:07 pyvenv.cfg
drwxrwxr-x 3 devasc devasc 4096 May 31 16:07 share
```

c)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ ls -l examples
total 88
drwxrwxr-x 3 devasc devasc 4096 May 31 16:47 abstraction_example
drwxrwxr-x 2 devasc devasc 4096 May 31 16:47 basic
<output omitted>
drwxrwxr-x 2 devasc devasc 4096 May 31 16:47 uids
```

d)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ ls -l examples/basic
total 12
-rw-rw-r-- 1 devasc devasc 510 May 31 16:47 basic_example_job.py
-rwxrwxr-x 1 devasc devasc 4475 May 31 16:47 basic_example_script.py
```

4. Examinez les fichiers de script de base.

a)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ cat
examples/basic/basic_example_script.py | more
#!/usr/bin/env python
##############################################################################
```

```
# basic_example.py : A very simple test script example which include:
# common_setup
# Tescases
# common_cleanup
# The purpose of this sample test script is to show the "hello world"
# of aetest.
#############################################################################
# To get a logger for the script
import logging
# Needed for aetest script
from pyats import aetest
# Get your logger for your script
log = logging.getLogger(__name__)
#############################################################################
### COMMON SETUP SECTION ###
#############################################################################
# This is how to create a CommonSetup
# You can have one of no CommonSetup
# CommonSetup can be named whatever you want
class common_setup(aetest.CommonSetup):
""" Common Setup section """
# CommonSetup have subsection.
# You can have 1 to as many subsection as wanted
# here is an example of 2 subsections
# First subsection
@aetest.subsection
def sample_subsection_1 (self):
""" Common Setup subsection """
log.info("Aetest Common Setup ")
# If you want to get the name of current section,
# add section to the argument of the function.
# Second subsection
@aetest .sous-section
def sample_subsection_2(self, section):
""" Common Setup subsection """
log.info("Inside %s" % (section))
# And how to access the class itself ?
# self refers to the instance of that class, and remains consistent
# throughout the execution of that container.
log.info("Inside class %s" % (self.uid))
#############################################################################
### SECTION DE TESTCAS ###
#############################################################################
# This is how to create a testcase
# You can have 0 to as many testcase as wanted
# Testcase name : tc_one
class tc_one(aetest.Testcase):
""" This is user Testcases section """
# Testcases are divided into 3 sections
# Setup, Test and Cleanup.
# This is how to create a setup section
@aetest .setup
def prepare_testcase(self, section):
""" Testcase Setup section """
log.info("Preparing the test")
log.info (section)
# This is how to create a test section
# You can have 0 to as many test section as wanted
# First test section
@ aetest.test
def simple_test_1(self):
""" Sample test section. Only print """
log.info("First test section ")
```

```
# Second test section
@ aetest.test
def simple_test_2(self):
""" Sample test section. Only print """
log.info("Second test section ")
# This is how to create a cleanup section
@aetest .cleanup
def clean_testcase(self):
""" Testcase cleanup section """
log.info("Pass testcase cleanup")
# Testcase name : tc_two
class tc_two(aetest.Testcase):
""" This is user Testcases section """
@ aetest.test
def simple_test_1(self):
""" Sample test section. Only print """
log.info("First test section ")
self.failed('This is an intentional failure')
# Second test section
@ aetest.test
def simple_test_2(self):
""" Sample test section. Only print """
log.info("Second test section ")
# This is how to create a cleanup section
@aetest .cleanup
def clean_testcase(self):
""" Testcase cleanup section """
log.info("Pass testcase cleanup")
##############################################################################
#### COMMON CLEANUP SECTION ###
##############################################################################
# This is how to create a CommonCleanup
# You can have 0 , or 1 CommonCleanup.
# CommonCleanup can be named whatever you want :)
class common_cleanup (aetest.CommonCleanup):
""" Common Cleanup for Sample Test """
# CommonCleanup follow exactly the same rule as CommonSetup regarding
# subsection
# You can have 1 to as many subsection as wanted
# here is an example of 1 subsections
@aetest .sous-section
def clean_everything(self):
""" Common Cleanup Subsection """
log.info("Aetest Common Cleanup ")
if __name__ == '__main__': # pragma: no cover
aetest.main()
```

b)


```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ cat
examples/basic/basic_example_job.py
# To run the job:
# pyats run job basic_example_job.py
# Description: This example shows the basic functionality of pyats
# with few passing tests
import os
from pyats.easypy import run
# All run() must be inside a main function
def main():
# Find the location of the script in relation to the job file
test_path = os.path.dirname(os.path.abspath(__file__))
testscript = os.path.join(test_path, 'basic_example_script.py')
# Execute the testscript
run(testscript=testscript)
```

5. Exécutez PyATS manuellement pour appeler le cas de test de base.

a)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ pyats run job
examples/basic/basic_example_job.py
2020-05-31T17:10:17: %EASYPY-INFO: Starting job run: basic_example_job
2020-05-31T17:10:17: %EASYPY-INFO: Runinfo directory:
/home/devasc/.pyats/runinfo/basic_example_job.2020May31_17:10:16.735106
2020-05-31T17:10:17: %EASYPY-INFO: ----------------------------------------
-------
-----------------------------
2020-05-31T17:10:18: %EASYPY-INFO: Starting task execution: Task-1
2020-05-31T17:10:18: %EASYPY-INFO: test harness = pyats.aetest
2020-05-31T17:10:18: %EASYPY-INFO: testscript = /home/devasc/labs/devnet-
src/pyats/csr1kv/examples/basic/basic_example_script.py
2020-05-31T17:10:18: %AETEST-INFO: +--------------------------------------
-------
---------------------------+
2020-05-31T17:10:18: %AETEST-INFO: | Starting common setup |
<output omitted>
-------+
2020-05-31T17:10:18: %SCRIPT-INFO: First test section
2020-05-31T17:10:18: %AETEST-ERROR: Failed reason: This is an intentional
failure
2020-05-31T17:10:18: %AETEST-INFO: The result of section simple_test_1 is =>
FAILED
2020-05-31T17:10:18: %AETEST-INFO: +--------------------------------------
-------
---------------------------+
2020-05-31T17:10:18: %AETEST-INFO: | Starting section simple_test_2 |
<output omitted>
-------+
2020-05-31T17:10:20: %EASYPY-INFO: | Easypy Report |
2020-05-31T17:10:20: %EASYPY-INFO: +--------------------------------------
-------
---------------------------+
<output omitted>
2020-05-31T17:10:20: %EASYPY-INFO: Overall Stats
2020-05-31T17:10:20: %EASYPY-INFO: Passed : 3
2020-05-31T17:10:20: %EASYPY-INFO: Passx : 0
2020-05-31T17:10:20: %EASYPY-INFO: Failed : 1
2020-05-31T17:10:20: %EASYPY-INFO: Aborted : 0
```

```
2020-05-31T17:10:20: %EASYPY-INFO: Blocked : 0
2020-05-31T17:10:20: %EASYPY-INFO: Skipped : 0
2020-05-31T17:10:20: %EASYPY-INFO: Errored : 0
2020-05-31T17:10:20: %EASYPY-INFO:
2020-05-31T17:10:20: %EASYPY-INFO: TOTAL : 4
2020-05-31T17:10:20: %EASYPY-INFO:
2020-05-31T17:10:20: %EASYPY-INFO: Success Rate : 75.00 %
2020-05-31T17:10:20: %EASYPY-INFO:
2020-05-31T17:10:20: %EASYPY-INFO: +----------------------------------------
-------
---------------------------+
2020-05-31T17:10:20: %EASYPY-INFO: | Task Result Summary |
2020-05-31T17:10:20: %EASYPY-INFO: +----------------------------------------
-------
---------------------------+
2020-05-31T17:10:20: %EASYPY-INFO: Task-1: basic_example_script.common_setup
PASSED
2020-05-31T17:10:20: %EASYPY-INFO: Task-1: basic_example_script.tc_one PASSED
2020-05-31T17:10:20: %EASYPY-INFO: Task-1: basic_example_script.tc_two FAILED
2020-05-31T17:10:20: %EASYPY-INFO: Task-1: basic_example_script.common_cleanup
PASSED
2020-05-31T17:10:20: %EASYPY-INFO:
2020-05-31T17:10:20: %EASYPY-INFO: +----------------------------------------
-------
---------------------------+
2020-05-31T 17:10:20 :%EASYPY-INFO : | Détails du résultat de la tâche |
2020-05-31T17:10:20: %EASYPY-INFO: +----------------------------------------
-------
---------------------------+
2020-05-31T17:10:20: %EASYPY-INFO: Task-1: basic_example_script
2020-05-31T17:10:20: %EASYPY-INFO: |-- common_setup PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- sample_subsection_1 PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | `-- sample_subsection_2 PASSED
2020-05-31T17:10:20: %EASYPY-INFO: |-- tc_one PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- prepare_testcase PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- simple_test_1 PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- simple_test_2 PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | `-- clean_testcase PASSED
2020-05-31T17:10:20: %EASYPY-INFO: |-- tc_two FAILED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- simple_test_1 FAILED
2020-05-31T17:10:20: %EASYPY-INFO: | |-- simple_test_2 PASSED
2020-05-31T17:10:20: %EASYPY-INFO: | `-- clean_testcase PASSED
2020-05-31T17:10:20: %EASYPY-INFO: `-- common_cleanup PASSED
2020-05-31T17:10:20: %EASYPY-INFO: `-- clean_everything PASSED
2020-05-31T17:10:20: %EASYPY-INFO: Sending report email...
2020-05-31T17:10:20: %EASYPY-INFO: Missing SMTP server configuration, or failed
to
reach/authenticate/send mail. Result notification email failed to send.
2020-05-31T17:10:20: %EASYPY-INFO: Done!
Conseil d'expert
-------
Utilisez la commande suivante pour afficher vos journaux localement :
pyats logs view
```

4. Utiliser Genie pour analyser la sortie de la commande IOS

1. Créez un fichier YAML testbed.

```
csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ genie --help
Usage:
genie <command> [options]
Commands:
create Create Testbed, parser, triggers, ...
diff Command to diff two snapshots saved to file or directory
dnac Command to learn DNAC features and save to file (Prototype)
learn Command to learn device features and save to file
parse Command to parse show commands
run Run Genie triggers & verifications in pyATS runtime environment
shell enter Python shell, loading a pyATS testbed file and/or pickled data
General Options:
-h, --help Show help
Run 'genie <command> --help' for more information on a command.
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ genie create --help
Usage:
genie create <subcommand> [options]
  2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document pu
Subcommands:
parser create a new Genie parser from template
testbed create a testbed file automatically
trigger create a new Genie trigger from template
General Options:
-h, --help Show help
-v, --verbose Give more output, additive up to 3 times.
-q, --quiet Give less output, additive up to 3 times, corresponding to WARNING,
ERROR,
and CRITICAL logging levels
```

b)

```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ genie create testbed
interactive --output yaml/testbed.yml --encode-password
Start creating Testbed yaml file ...
Do all of the devices have the same username? [y/n] n
Do all of the devices have the same default password? [y/n] n
Do all of the devices have the same enable password? [y/n] n

Device hostname: CSR1kv
   IP (ip, or ip:port): 192.168.56.101
   Username: cisco
Default Password (leave blank if you want to enter on demand):
Enable Password (leave blank if you want to enter on demand):
   Protocol (ssh, telnet, ...): ssh -o KexAlgorithms=diffie-hellman-group14-
sha1   OS (iosxr, iosxe, ios, nxos, linux, ...): iosxe
More devices to add ? [y/n] n
Testbed file generated:
yaml/testbed.yml
```

c)


```
(csr1kv) devasc@labvm:~/labs/devnet-src/pyats/csr1kv$ cat yaml/testbed.yml
devices:
  CSR1kv:
    connections:
      cli:
        ip: 192.168.56.101
        protocol: ssh -o KexAlgorithms=diffie-hellman-group14-
    credentials:
      default:
        password: '%ASK{}'
        username: cisco
      enable:
        password: '%ASK{}'
    os: sha1iosxe
    type: sha1iosxe
```

2. Utilisez Génie pour analyser la sortie de la commande show ip interface brief dans JSON.

b)

```
Fri Jan 19 2024 16:56:28 GMT+0100 (heure normale d’Europe centrale)
==============================================================================
#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.6.255.224    YES NVRAM  up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
```

Explorer les modèles YANG

2. Explorer un modèle YANG sur GitHub

2. Copiez le modèle ietf-interfaces.yang dans un dossier sur votre machine virtuelle.

i)


```
wget
https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1693/ietf
interfaces.yang
--2024-01-19 16:23:22--
https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1693/ietf
interfaces.yang
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133
185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24248 (24K) [text/plain]
Saving to: ‘ietf-interfaces.yang’

ietf-interfaces.yang            100%
[=======================================================>]  23.68K  --.-KB/s    in
0.002s

2024-01-19 16:23:23 (10.6 MB/s) - ‘ietf-interfaces.yang’ saved [24248/24248]
```

