User Manual

Studienarbeit FS-2020

25. Mai 2020

Autoren:

Mike Schmid
mike.schmid@hsr.ch

Janik Schlatter
janik.schlatter@hsr.ch

Supervisors:

Prof. Stettler Beat
beat.stettler@hsr.ch

Baumann Urs
urs.baumann@hsr.ch

Inhaltsverzeichnis

1 Installation

Es wird keine spezielle Installation benötigt. Das Repository auf GitHub muss lokal geklont werden. Danach muss man noch das requirements File installieren mit dem Befehl: 'pip install requirements.txt'.

1.1 Ausführen

Das Programm kan regulär in einer Programmierumgebung wie zum Beispiel PyCharm ausgeführt werden. Um es in einer Konsole zu starten muss zum Ordner NUTS2.0 navigiert werden. Man kann es mit dem Befehl: 'python -m nuts' starten. Wenn man das GUI auslassen möchte und direkt alle Tests ausführen möchte kann man mit dem Befehl: 'python -m nuts -r' starten.

1.1.1 Konfiguration

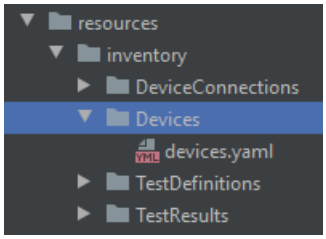
Im File 'Config.yaml' können die Pfäde aller Ordner geändert werden. Zusätzlich kann noch bestimmt werden, ob das GUI Per default übersprungen werden soll.

2 Inventar

Um Netzwerktests auszuführen braucht man zuerst ein Inventar mit Devices und Device Connections. Die Devices sind die Netzwerkgeräte wie zum Beispiel Router oder Switches. Die Device Connections sind die Verbindungen zwischen den Devices.

2.1 Devices

Die Definitionen der Devices sind unter Resources/Inventory/Devices/Devices.yaml:



Um Devices zu erfassen müssen folgende Informationen im yaml eingegeben werden:

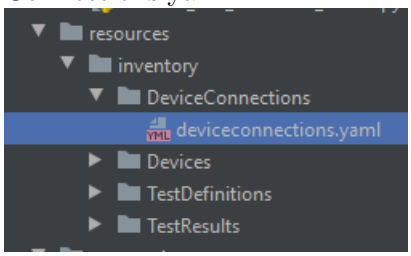
Attribut	Beschreibung
device_id	Eine eindeutige ID für das Device
platform	Das OS welches das Device benutzt
username	Der username welches das Device für das Login benutzt
password	Das Passwort welches das Device für das Login benutzt
hostname	Die Ip Adresse über welche das Device angesprochen werden kann

Diese Informationen sollten wie folgt dargestellt werden:

```
router01:
  - router01
  - cisco_ios
  - Cisco
  - cisco
  - 10.20.0.31
  - 172.16.255.1
```

2.2 Device Connections

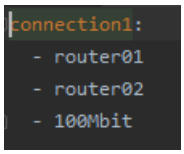
Die Definitionen der Device Connections sind unter Resources/Inventory/DeviceConnections/Device-Connections.yaml:



Um Device Connections zu erfassen müssen folgende Informationen im yaml eingegeben werden:

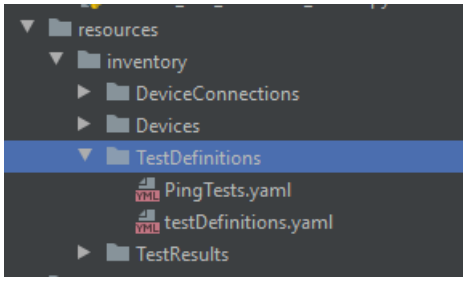
Attribut	Beschreibung
device a	Die ID des ersten Devices
device b	Die ID des zweiten Devices
connection speed	Die Übertragungsrate der Verbindung

Diese Informationen sollten wie folgt dargestellt werden:



3 Netzwerktests

Die Netzwerktests sind die Tests, welche effektiv auf dem zu testenden Netzwerk ausgeführt werden. Die Testdefinitionen befinden sich unter Resources/Inventory/TestDefinitions:



Es können in diesem Ordner beliebig viele YAML Files abgelegt werden und es werden vonn allen Files die Tests erfasst.

Um die Tests zu erfassen müssen folgende Informationen im yaml eingegeben werden:

Attribut	Beschreibung
test_id	Eine eindeutige ID für den Test
command	Ein Command um den Test zu bestimmen
test_device	Die ID des Devices auf welchem der Test ausgeführt werden soll
target	Das Ziel (Zum Beispiel eine IP im Falle eines Pings)
expected_result	Das erwartete Resultat (Zum Beispiel Success im Falle eines Pings)
test_group	Ein Gruppenname um später die Tests zu Kategorisieren

Diese Informationen sollten wie folgt dargestellt werden:

```
test01:
  - PingRouter1Loopback0
  - Ping
  - router01
  - 172.16.255.1
  - Success
  - connectivity
```

3.1 Commands

Folgende Commands sind bereits implementiert und können mit den jeweiligen expected_results verwendet werden:

3.1.1 Ping

Als erwartetes Resultat kann 'Success' oder 'Failure' verwendet werden.

3.1.2 Show Interfaces

Dies benötigt kein Target also kann bei der definition einfach 'No Target' eingegeben werden. Als erwartetes Resultat wird ein Dictionary mit key:'Interfacename' und value:'True' oder 'False' verwendet werden. Dies sollte in folgender Form dargestellt werden:

```
- {'GigabitEthernet1': True, 'GigabitEthernet2': True, 'GigabitEthernet3': True, 'GigabitEthernet4': True, 'Loopback0': True}
```

3.1.3 Traceroute

Als erwartetes Resultat wird ein Array von IP Adressen angegeben. Dies sollte in folgeder Form dargestellt werden:

```
- ["172.16.13.1", "172.16.14.4"]
```

3.1.4 Arp Table

Dies benötigt kein Target also kann bei der definition einfach 'No Target' eingegeben werden. Als erwartetes Resultat wird ein Array mit Dictionaries erwartet. In den Dictionaries werden folgende Informationen benötigt: 'interface' : 'interfacename', 'mac' : 'macadresse', 'ip' : 'ipadresse'
Dies sollte in folgeder Form dargestellt werden:

```
- [
    {'interface': 'GigabitEthernet3', 'mac': '52:54:00:67:A5:39', 'ip': '172.16.12.1'},
    {'interface': 'GigabitEthernet3', 'mac': '52:54:00:28:F0:9D', 'ip': '172.16.12.2'},
    {'interface': 'GigabitEthernet4', 'mac': '52:54:00:16:91:60', 'ip': '172.16.13.1'},
    {'interface': 'GigabitEthernet4', 'mac': '52:54:00:12:E0:4C', 'ip': '172.16.13.3'},
    {'interface': 'GigabitEthernet2', 'mac': '52:54:00:63:CD:6C', 'ip': '172.16.14.1'},
    {'interface': 'GigabitEthernet2', 'mac': '52:54:00:5D:8E:C7', 'ip': '172.16.14.4'}
  ]
```

3.1.5 Ospf Neighbor

Dies benötigt kein Target also kann bei der definition einfach 'No Target' eingegeben werden. Als erwartetes Resultat wird ein Array mit Dictionaries erwartet. In den Dictionaries werden folgende Informationen benötigt: 'Neighbor-ID' : 'ipadresse', 'Priority' : 'Nummer', 'State' : 'Den Status', 'Address' : 'ipadresse', 'Interface' : 'Interfacename'
Dies sollte in folgeder Form dargestellt werden:

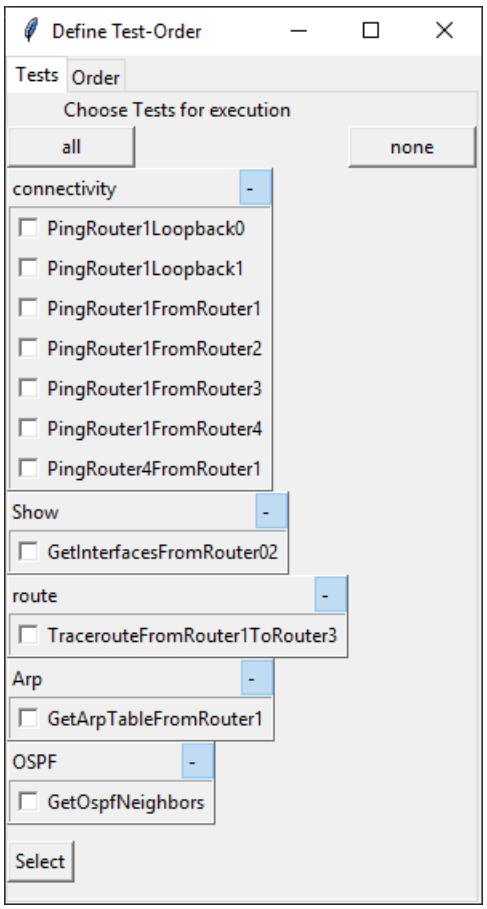
```
- [
    {'Neighbor-ID': '172.16.255.3', 'Priority': '1', 'State': 'FULL/BDR', 'Address': '172.16.13.3', 'Interface': 'GigabitEthernet4'},
    {'Neighbor-ID': '172.16.255.2', 'Priority': '1', 'State': 'FULL/BDR', 'Address': '172.16.12.2', 'Interface': 'GigabitEthernet3'},
    {'Neighbor-ID': '172.16.255.4', 'Priority': '1', 'State': 'FULL/BDR', 'Address': '172.16.14.4', 'Interface': 'GigabitEthernet2'}
  ]
```

4 Durchführung

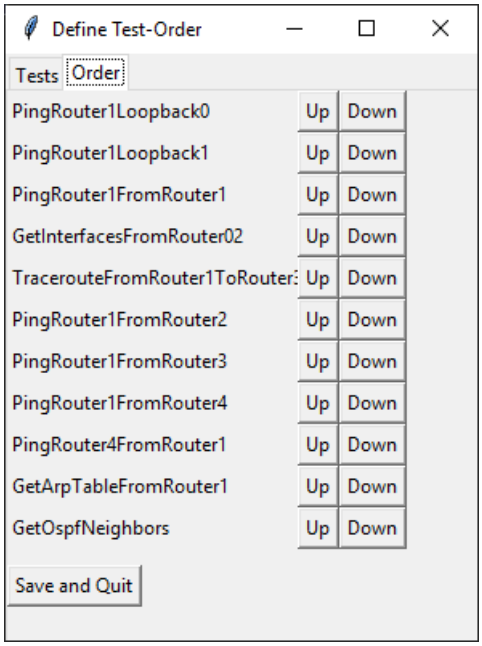
Nachdem dass Inventar erstellt und die Testdefinitionen erfasst wurden kann man das Programm starten. Falls die Option Skip-GUI aktiviert wurde werden alle Tests durchgeführt. Falls dies nicht aktiviert wurde öffnet sich ein GUI.

4.1 GUI

Das GUI für die Definition der Test Reihenfolge besteht aus zweit Tabs:



Im ersten Tab werden alle Tests nach Gruppen sortiert angezeigt. Hier kann der Benutzer auswählen welche Tests er ausführen möchte. Nachdem die Tests ausgewählt wurden weden mit dem Button 'Select' alle ausgewählten Tests in den zweiten Tab gezogen.



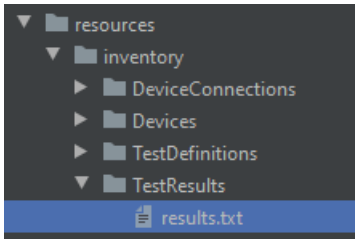
Im zweiten Tab werden alle ausgewählten Tests angezeigt und der Benutzer kann mit den jeweiligen Buttons die Reihenfolge bestimmen. Nachdem der Benutzer mit der Reihenfolge zufrieden ist kann mit dem Button 'Save and Quit' das GUI beendet werden.

4.2 Test Resultate

Die Resultate der jeweiligen Durchführungen werden in der Konsole angezeigt und zusätzlich noch in einem File gespeichert. In der Konsole sehen die Resultate wie folgt aus:

```
    _   ___  ____________    ___    ____
   / | / / / /_  __/ ___/   |__ \  / __ \
  /  |/ / / / / / /  \__ \   __/ / / / / /
 / /|  / /_/ / / /  ___/ /  / __/_/ /_/ /
/_/ |_/\____/ /_/  /____/  /____(_)____/


+-----------------------------------------------------------------------------------------------+
| INITIALIZING TEST SUITE                                                                       |
+-----------------------------------------------------------------------------------------------+
 Create Device Objects from YAML: 100%|████████████████████████████| 4/4 [00:00<00:00, 4009.85it/s]
 Read objects from PingTests.yaml: 100%|████████████████████████████| 2/2 [00:00<00:00, 2004.93it/s]
 Read objects from testDefinitions.yaml: 100%|████████████████████████| 9/9 [00:00<00:00, 8934.61it/s]
 Create runnable tests: 100%|████████████████████████████████████| 11/11 [00:00<00:00, 479.54it/s]
+-----------------------------------------------------------------------------------------------+
| RUN ALL TESTS                                                                                 |
+-----------------------------------------------------------------------------------------------+
 Execute tests: 100%|████████████████████████████████████████| 11/11 [01:30<00:00,  8.19s/it]
+-----------------------------------------------------------------------------------------------+
| TEST RESULTS                                                                                  |
+-----------------------------------------------------------------------------------------------+
Passed Tests
|   |-- Test: PingRouter1Loopback0 has PASSED
|   |-- Test: PingRouter1Loopback1 has PASSED
|   |-- Test: PingRouter1FromRouter1 has PASSED
|   |-- Test: GetInterfacesFromRouter02 has PASSED
|   |-- Test: TracerouteFromRouter1ToRouter3 has PASSED
|   |-- Test: PingRouter1FromRouter2 has PASSED
|   |-- Test: PingRouter1FromRouter3 has PASSED
|   |-- Test: PingRouter1FromRouter4 has PASSED
|   |-- Test: PingRouter4FromRouter1 has PASSED
|   |-- Test: GetArpTableFromRouter1 has PASSED
|   |-- Test: GetOspfNeighbors has PASSED
-------------------------------------------------------------------------------------------------
No tests failed
```

Das File befindet sich unter: 'Resources/Inventory/TestResults/results.txt'.



In dem File wird zuerst die Zeit des TestRuns angegeben, danach werden zuerst die Passed Tests und danach die Failed Tests angezeigt. Bei den Failed Tests werden zusätzlich noch das Erwartete und das

Effektive Resultat angezeigt.

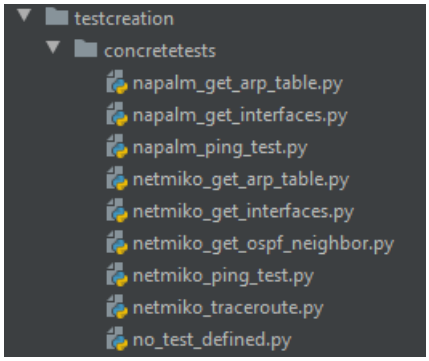
```
New Test Run on 2020-05-13 10:10:54.465216:
  Passed Tests:
    Test: PingRouter1Loopback0 has PASSED
    Test: PingRouter1Loopback1 has PASSED
    Test: PingRouter1FromRouter1 has PASSED
    Test: GetInterfacesFromRouter02 has PASSED
    Test: TracerouteFromRouter1ToRouter3 has PASSED
    Test: PingRouter1FromRouter2 has PASSED
    Test: PingRouter1FromRouter3 has PASSED
    Test: PingRouter1FromRouter4 has PASSED
    Test: PingRouter4FromRouter1 has PASSED

  Failed Tests:
    Test: GetArpTableFromRouter1 has FAILED
      Expected: [{'interface': 'GigabitEthernet3', 'mac': '52:54:00:67:A5:39', 'ip': '172.16.12.1'},
                 {'interface': 'GigabitEthernet3', 'mac': '52:54:00:28:F0:9D', 'ip': '172.16.12.2'},
                 {'interface': 'GigabitEthernet4', 'mac': '52:54:00:16:91:60', 'ip': '172.16.13.1'},
                 {'interface': 'GigabitEthernet4', 'mac': '52:54:00:12:E0:4C', 'ip': '172.16.13.3'},
                 {'interface': 'GigabitEthernet2', 'mac': '52:54:00:63:CD:6C', 'ip': '172.16.14.1'},
                 {'interface': 'GigabitEthernet2', 'mac': '52:54:00:5D:8E:C7', 'ip': '172.16.14.1'}]

      Actual:   [{'interface': 'GigabitEthernet3', 'mac': '52:54:00:67:A5:39', 'ip': '172.16.12.1'},
                 {'interface': 'GigabitEthernet3', 'mac': '52:54:00:28:F0:9D', 'ip': '172.16.12.2'},
                 {'interface': 'GigabitEthernet4', 'mac': '52:54:00:16:91:60', 'ip': '172.16.13.1'},
                 {'interface': 'GigabitEthernet4', 'mac': '52:54:00:12:E0:4C', 'ip': '172.16.13.3'},
                 {'interface': 'GigabitEthernet2', 'mac': '52:54:00:63:CD:6C', 'ip': '172.16.14.1'},
                 {'interface': 'GigabitEthernet2', 'mac': '52:54:00:5D:8E:C7', 'ip': '172.16.14.4'}]

End of Test Run
```

5 Zusätzliche Tests hinzufügen

Falls der Benutzer eigene Tests hinzufügen möchte müssen an folgenden Orten änderungen vorgenommen werden:

5.1 Conctrete Tests

In diesem Ordner muss ein neues File angelegt werden mit dem Test, der implementiert werden möchte. Bei der Strucktur des Files sollte man sich an die bisherigen Files halten.



5.2 Network Test Strategy Factory

In dieser Klasse ist die Test Map aufgrund welcher entschieden wird, welcher Test verwendet wird für welche Test Definition. Hier muss in der Test Map der neue Test vermerkt werden.

```
def __init__(self):
    self.test_map = {
        "Ping": {
            "Napalm": NapalmPingTest,
            "Netmiko": NetmikoPingTest,
        },
        "Show Interfaces": {
            "Napalm": NapalmShowInterfaces,
            "Netmiko": NetmikoShowInterfaces
        },
        "Traceroute": {
            "Napalm": None,
            "Netmiko": NetmikoTraceroute
        },
        "Arp Table": {
            "Napalm": NapalmShowArpTables,
            "Netmiko": NetmikoShowArpTables
        },
        "Ospf Neighbor": {
            "Napalm": None,
            "Netmiko": NetmikoShowOspfNeighbor
        }

        # Add more Tests as "Testcommand: {connection_dictionary}
    }
```