# Автоматизация 6. Napalm

Приветствую всех читателей цикла, мы продолжаем знакомиться с инструментами по автоматизации конфигурирования сетевой инфраструктуры.

Предыдущие статьи:

1. Краткая теория SDN и предпосылок автоматизации
2. Практика с TelnetLib
3. Практика с Paramiko
4. Практика с Netmiko. Часть 1
5. Практика с Netmiko. Часть 2

# Теория по Napalm

Прежде чем рассказывать о новой библиотеке, позволим себе небольшое лирическое отступление.

После того, как мы достаточно подробно изучили Netmiko, казалось бы все наши потребности будут удовлетворены и мы можем выполнять любые задачи. Но как известно - аппетит растет во время еды, и вот уже у нас рождаются новые  "хотелки" в виде более глубокой работы с конфигурацией оборудования: версионирование изменений, откат к предыдущей конфигурации и пр. Подобные задачи уже не решить с помощью Netmiko, который отлично справляется с базовым взаимодействием с сетевым оборудованием (сбор и отправка конфигурации, установка соединения и т.д).

Библиотека Napalm (Network Automation and Programmability Abstraction Layer with Multivendor support), опенсорсный проект написан на языке Python, содержит различные функции для взаимодействия с различными сетевыми ОС за счет единого API.

По своей архитектуре, Napalm требует определенного драйвера для работы с тем или иным вендором. В официальный список - "Napalm Core Developers" (нативная поддержка взаимодействия с ОС) входят:

| – | EOS | Junos | IOS-XR | NX-OS | NX-OS SSH | IOS |
|---|---|---|---|---|---|---|
| Driver Name | eos | junos | iosxr | nxos | nxos_ssh | ios |
| Structured data | Yes | Yes | No | Yes | No | No |
| Minimum version | 4.15.0F | 12.1 | 5.1.0 | 6.1 [1] | | 12.4(20) |
| Backend library | pyeapi | junos-eznc | pyIOSXR | pynxos | netmiko | netmiko |
| Caveats | EOS | | | NXOS | NXOS | IOS |

Как вы можете заметить из  таблицы, представленной выше, далеко не все системы поддерживаются, поскольку для них нет драйвера. Кроме того, есть ограничения по версии ОС и поддерживаемым функциям. Например, IOS не имеет своего API для отправки команд, поэтому внутри Napalm используется библиотека Netmiko.

В свою очередь, на Github есть репозиторий "Community Drivers", где отдельные разработчики-энтузиасты предлагают драйвера для неподдерживаемых изначально сетевых ОС (ссылка).  За счет сообщества, вы сможете опробовать Napalm с такими системами как VyOS, Dlink, FortiOS и др.

Кроме ограничений, связанных с наличием соответствующего драйвера, также есть матрица поддержки функций Napalm:

| _ | EOS | Junos | IOS-XR | NX-OS | IOS |
|---|---|---|---|---|---|
| Config. replace | Yes | Yes | Yes | Yes | Yes |
| Config. merge | Yes | Yes | Yes | Yes | Yes |
| Compare config | Yes | Yes | Yes [2] | Yes [4] | Yes |
| Atomic Changes | Yes | Yes | Yes | Yes/No [5] | Yes/No [5] |
| Rollback | Yes [3] | Yes | Yes | Yes/No [5] | Yes |

В целом почти все опции работают:

- замена конфигурации;
- добавление конфигурации;
- сравнение конфигурации;
- откат изменений.

Укажем расшифровку примечаний:

[2] - Реализовано за счет API Napalm, по скольку не поддерживается ОС изначально.
[3] - Не поддерживается, но эмулируется.
[4] - Для слияний, часть возможностей по сравнению - отсутствует.
[5] - Не поддерживается, но эмулируется.

С более подробным описанием функций возможно ознакомиться в официальном документе по Napalm, крайне рекомендовано понимать их назначение, прежде чем писать какой-либо скрипт.

В качестве сетевого протокола по доставке команд используются:

| _ | EOS | NXOS | IOS |
|---|---|---|---|
| Default | `https` | `https` | `ssh` |
| Supported | `http`, `https` | `http`, `https` | `telnet`, `ssh` |

# Установка

Для начала потребуется установить библиотеку Napalm на ваш хост управления, стабильно пакет работает в Linux, поэтому мы остановимся на одном из его дистрибутивов (Ubuntu).

```
apt-get update
apt-get install python -y
apt-get install build-essential libssl-dev libffi-dev -y
apt-get install python3-pip -y
pip3 install -U cryptography
pip3 install -U netmiko
pip3 install -U napalm
pip3 install -U simplejson
```

Кроме установки пакета самой библиотеки Napalm, также понадобится: python3, pip, cryptography, netmiko, simplejson.

После успешной установки, начинаем знакомство с базовым синтаксисом библиотеки и ее возможностями, для установки соединения с сетевым оборудованием потребуются строки кода:

```
"> from napalm import get_network_driver
"> driver = get_network_driver("eos")
"> device = driver("192.168.76.10", "admin", "pass")
"> device.open()
```

1. импортируется общая библиотека драйверов;
2. указывается драйвер устройства в переменной "driver";
3. объявляется переменная "device", которая содержит необходимые параметры для подключения;
4. устанавливается соединение.

# Первые скрипты

После первичной инициализации подключения, становятся доступны команды по работе с конфигурацией:

```
from napalm import get_network_driver # импорт драйвера
driver = get_network_driver("ios") # выбор типа OS
iosvl2 = driver("192.168.122.72", "david", "cisco") # параметры
iosvl2.open()
ios_output = iosvl2.get_facts()# сбор фактов
print (ios_output)
```

```
root@Ubuntu-1:~# python napalm1.py
{u'os_version': u'vios_l2 Software (vios_l2-ADVENTERPRISEK9-M), Version 15.2(4.0
.55)E, TEST ENGINEERING ESTG_WEEKLY BUILD, synced to  END_OF_FLO_ISP', u'uptime'
: 1740, u'interface_list': [u'GigabitEthernet0/0', u'GigabitEthernet0/1', u'Giga
bitEthernet0/2', u'GigabitEthernet0/3', u'GigabitEthernet1/0', u'GigabitEthernet
1/1', u'GigabitEthernet1/2', u'GigabitEthernet1/3', u'GigabitEthernet2/0', u'Gig
abitEthernet2/1', u'GigabitEthernet2/2', u'GigabitEthernet2/3', u'GigabitEtherne
t3/0', u'GigabitEthernet3/1', u'GigabitEthernet3/2', u'GigabitEthernet3/3', u'Vl
an1', u'Vlan2', u'Vlan3'], u'vendor': u'Cisco', u'serial_number': u'9H7G1KOBYQL'
, u'model': u'IOSv', u'hostname': u'S1', u'fqdn': u'S1.cciepython.com'}
root@Ubuntu-1:~#
```

Как видим, после выполнения скрипта, на экране пользователя вывелись факты об устройстве, согласно методу - "get_facts". Информации достаточно много, пусть и не в самом удобно читаемом виде.

Чтобы решить проблему с выводом, мы используем библиотеку json. Формат JSON является структурированным текстовым типом данных, достаточно удобным для чтения человеком. Добавим следующие строки в предыдущий скрипт, как первую и последнюю строчку (вместо обычного print) соответственно:

```
import json
print json.dumps(ios_output, indent=4)
```

После изменений получим вывод:

```
root@Ubuntu-1:~# python napalm1.py
{
    "os_version": "vios_l2 Software (vios_l2-ADVENTERPRISEK9-M), Version 15.2(4.
0.55)E, TEST ENGINEERING ESTG_WEEKLY BUILD, synced to  END_OF_FLO_ISP",
    "uptime": 2340,
    "interface_list": [
        "GigabitEthernet0/0",
        "GigabitEthernet0/1",
        "GigabitEthernet0/2",
        "GigabitEthernet0/3",
        "GigabitEthernet1/0",
        "GigabitEthernet1/1",
        "GigabitEthernet1/2",
        "GigabitEthernet1/3",
        "GigabitEthernet2/0",
        "GigabitEthernet2/1",
        "GigabitEthernet3/3",
        "Vlan1",
        "Vlan2",
        "Vlan3"
    ],
    "vendor": "Cisco",
    "serial_number": "9H7G1KOBYQL",
    "model": "IOSv",
    "hostname": "S1",
    "fqdn": "S1.cciepython.com"
}
```

Соответственно, получили данные от сетевого оборудования Cisco, которые представлены в формате JSON, что в разы повышает их читаемость и удобство для дальнейшей работы.

## Сравнение конфигурации

Прежде чем отправлять какие-либо изменения в конфигурацию вашего сетевого устройства, с помощью Napalm появляется возможность сравнить настройки (доступно при поддержке ОС).

Считаем файл-кандидат с новыми настройками и вызовем метод сравнения настроек:

```
"> device.load_replace_candidate(filename="test/new_good.conf")
"> print(device.compare_config())

+ hostname pyeos-unittest-changed
- hostname pyeos-unittest
router bgp 65000
   vrf test
     + neighbor 1.1.1.2 maximum-routes 12000
     + neighbor 1.1.1.2 remote-as 1
     - neighbor 1.1.1.1 remote-as 1
     - neighbor 1.1.1.1 maximum-routes 12000
   vrf test2
     + neighbor 2.2.2.3 remote-as 2
     + neighbor 2.2.2.3 maximum-routes 12000
     - neighbor 2.2.2.2 remote-as 2
     - neighbor 2.2.2.2 maximum-routes 12000
interface Ethernet2
+ description ble
- description bla
```

Библиотека наглядно выводит ожидаемые изменения на экран пользователя, где знаками "+" и "-" отмечены новые строки или те, которые будут удалены, соответственно.

Для того чтобы принять изменения, посылается команда:

```
"> device.commit_config()
```

Если вы изменили свое решение и не хотите применять новые настройки, то:

```
"> device.discard_config()
```

### Слияние конфигураций

В целом логика та же, что и в предыдущем сценарии отправки новой конфигурации, только необходимо передавать новый файл при вызове метода слияния:

```
"> device.load_merge_candidate(config="test\description bla")
"> print(device.compare_config())
configure
hostname test
interface Ethernet2
description bla
end
```

Для того чтобы принять изменения, посылается команда:

```
"> device.commit_config()
```

Если вы изменили свое решение и не хотите применять новые настройки, то:

```
"> device.discard_config()
```

### Откат принятой конфигурации

Может возникнуть ситуация, когда после операции device.commit_config()необходимо будет откатить изменения, библиотека поддерживает эту функцию:

```
"> device.rollback()
```

### Окончание сессии

Наконец, для завершения соединения с оборудованием достаточно команды:

```
"> device.close()
```

## Вместо заключения

Сегодня мы познакомились с библиотекой Napalm, способной гибко взаимодействовать с

конфигурацией сетевого оборудования, позволяя учитывать примененную версию настроек, откатывать изменения и т.д. Далее планируется еще одна статья по Napalm, где уже  будут разобраны исключительно готовые скрипты для решения различных практических кейсов, на сегодня всё, до встречи на NAG.ru.