PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

МОДУЛЬ NTC-ANSIBLE

NTC-ANSIBLE

ntc-ansible - это модуль для работы с сетевым оборудованием, который не только выполняет команды на оборудовании, но и обрабатывает вывод команд и преобразует с помощью TextFSM

Этот модуль не входит в число core модулей Ansible, поэтому его нужно установить.

NTC-ANSIBLE

Но прежде нужно указать Ansible, где искать сторонние модули. Указывается путь в файле ansible.cfg:

```
[defaults]

inventory = ./myhosts

remote_user = cisco
ask_pass = True

library = ./library
```

NTC-ANSIBLE

После этого, нужно клонировать репозиторий ntc-ansible, находясь в каталоге library:

```
[~/pyneng_course/chapter15/library]
$ git clone https://github.com/networktocode/ntc-ansible --recursive
Cloning into 'ntc-ansible'...
remote: Counting objects: 2063, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 2063 (delta 1), reused 0 (delta 0), pack-reused 2058
Receiving objects: 100% (2063/2063), 332.15 KiB | 334.00 KiB/s, done.
Resolving deltas: 100% (1157/1157), done.
Checking connectivity... done.
Submodule 'ntc-templates' (https://github.com/networktocode/ntc-templates) registered for path 'ntc-template
Cloning into 'ntc-templates'...
remote: Counting objects: 902, done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 902 (delta 16), reused 0 (delta 0), pack-reused 868
Receiving objects: 100% (902/902), 161.11 KiB | 0 bytes/s, done.
Resolving deltas: 100% (362/362), done.
Checking connectivity... done.
Submodule path 'ntc-templates': checked out '89c57342b47c9990f0708226fb3f268c6b8c1549'
```

NTC-ANSIBLE

А затем установить зависимости модуля:

```
pip install ntc-ansible
```

NTC-ANSIBLE

Так как в текущей версии Ansible уже есть модули, которые работают с сетевым оборудованием и позволяют выполнять команды, из всех возможностей ntc-ansible, наиболее полезной будет отправка команд show и получение структурированного вывода. За это отвечает модуль ntc_show_command.

NTC_SHOW_COMMAND

Модуль использует netmiko для подключения к оборудованию (netmiko должен быть установлен) и, после выполнения команды, преобразует вывод команды show с помощью TextFSM в структурированный вывод (список словарей).

Преобразование будет выполняться в том случае, если в файле index была найдена команда и для команды был найден шаблон.

NTC_SHOW_COMMAND

Параметры для подключения:

- **connection** - тут возможны два варианта: ssh (подключение netmiko) или offline (чтение из файла для тестовых целей)
- **platform** - платформа, которая существует в index файле (library/ntc-ansible/ntc-templates/templates/index)
- **command** - команда, которую нужно выполнить на устройстве
- **host** - IP-адрес или имя устройства
- **username** - имя пользователя
- **password** - пароль
- **template_dir** - путь к каталогу с шаблонами (library/ntc-ansible/ntc-templates/templates

NTC_SHOW_COMMAND

Пример playbook 1_ntc_ansible.yml:

```
- name: Run show commands on router
  hosts: 192.168.100.1

  tasks:

    - name: Run sh ip int br
      ntc_show_command:
        connection: ssh
        platform: "cisco_ios"
        command: "sh ip int br"
        host: "{{ inventory_hostname }}"
        username: "cisco"
        password: "cisco"
        template_dir: "library/ntc-ansible/ntc-templates/templates"
      register: result

    - debug: var=result
```

NTC_SHOW_COMMAND

Результат выполнения playbook:

```
$ ansible-playbook 1_ntc-ansible.yml
```

```
SSH password:

PLAY [Run show commands on router] ********************************************

TASK [Run sh ip int br] *******************************************************
ok: [192.168.100.1]

TASK [debug] ******************************************************************
ok: [192.168.100.1] => {
    "result": {
        "changed": false,
        "response": [
            {
                "intf": "Ethernet0/0",
                "ipaddr": "192.168.100.1",
                "proto": "up",
                "status": "up"
            },
            {
                "intf": "Ethernet0/1",
                "ipaddr": "192.168.200.1",
                "proto": "up",
                "status": "up"
            },
            {
                "intf": "Ethernet0/2",
                "ipaddr": "unassigned",
                "proto": "down",
                "status": "administratively down"
            },
            {
                "intf": "Ethernet0/3",
                "ipaddr": "unassigned",
                "proto": "up",
                "status": "up"
            },
            {
                "intf": "Loopback0",
                "ipaddr": "10.1.1.1",
                "proto": "up",
                "status": "up"
            }
        ],
        "response_list": []
    }
}

PLAY RECAP ********************************************************************
192.168.100.1              : ok=2    changed=0    unreachable=0    failed=0
```

NTC_SHOW_COMMAND

В переменной response находится структурированный вывод в виде списка словарей. Ключи в словарях получены на основании переменных, которые описаны в шаблоне library/ntc-ansible/ntc-templates/templates/cisco_ios_show_ip_int_brief.template (единственное отличие - регистр):

```
Value INTF (\S+)
Value IPADDR (\S+)
Value STATUS (up|down|administratively down)
Value PROTO (up|down)

Start
  ^${INTF}\s+${IPADDR}\s+\w+\s+\w+\s+${STATUS}\s+${PROTO} -> Record
```

NTC_SHOW_COMMAND

Для того, чтобы получить вывод про первый интерфейс, можно поменять вывод модуля debug, таким образом:

```
    - debug: var=result.response[0]
```

Пример playbook 2_ntc_ansible_save.yml с сохранением результатов команды:

```
- name: Run show commands on routers
  hosts: cisco-routers

  tasks:

    - name: Run sh ip int br
      ntc_show_command:
        connection: ssh
        platform: "cisco_ios"
        command: "sh ip int br"
        host: "{{ inventory_hostname }}"
        username: "cisco"
        password: "cisco"
        template_dir: "library/ntc-ansible/ntc-templates/templates"
      register: result

    - name: Copy facts to files
      copy:
        content: "{{ result.response | to_nice_json }}"
        dest: "all_facts/{{inventory_hostname}}_sh_ip_int_br.json"
```

СОХРАНЕНИЕ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ КОМАНДЫ

Результат выполнения:

```
$ ansible-playbook 2_ntc-ansible_save.yml
```

СОХРАНЕНИЕ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ КОМАНДЫ

```
SSH password:

PLAY [Run show commands on routers] ********************************************

TASK [Run sh ip int br] ********************************************************
ok: [192.168.100.3]
ok: [192.168.100.1]
ok: [192.168.100.2]

TASK [Copy facts to files] *****************************************************
changed: [192.168.100.2]
changed: [192.168.100.1]
changed: [192.168.100.3]

PLAY RECAP *********************************************************************
192.168.100.1              : ok=2    changed=1    unreachable=0    failed=0
192.168.100.2              : ok=2    changed=1    unreachable=0    failed=0
192.168.100.3              : ok=2    changed=1    unreachable=0    failed=0
```

СОХРАНЕНИЕ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ КОМАНДЫ

В результате, в каталоге all_facts появляются соответствующие файлы для каждого маршрутизатора. Пример файла all_facts/192.168.100.1_sh_ip_int_br.json:


```
[
    {
        "intf": "Ethernet0/0",
        "ipaddr": "192.168.100.1",
        "proto": "up",
        "status": "up"
    },
    {
        "intf": "Ethernet0/1",
        "ipaddr": "192.168.200.1",
        "proto": "up",
        "status": "up"
    },
    {
        "intf": "Ethernet0/2",
        "ipaddr": "unassigned",
        "proto": "down",
        "status": "administratively down"
    },
...
```

ШАБЛОНЫ JINJA2

Для Cisco IOS в ntc-ansible есть такие шаблоны:

```
cisco_ios_dir.template
cisco_ios_show_access-list.template
cisco_ios_show_aliases.template
cisco_ios_show_archive.template
cisco_ios_show_capability_feature_routing.template
cisco_ios_show_cdp_neighbors_detail.template
cisco_ios_show_cdp_neighbors.template
cisco_ios_show_clock.template
...
```

ШАБЛОНЫ JINJA2

Список всех шаблонов можно посмотреть локально, если ntc-ansible установлен:

```
ls -ls library/ntc-ansible/ntc-templates/templates/
```

Или в репозитории проекта.

ШАБЛОНЫ JINJA2

Используя TextFSM можно самостоятельно создавать дополнительные шаблоны.

И, для того, чтобы ntc-ansible их использовал автоматически, добавить их в файл index (library/ntc-ansible/ntc-templates/templates/index)