PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

Модули для работы с сетевым оборудованием, можно разделить на две части:

- модули для оборудования с поддержкой API
- модули для оборудования, которое работает только через CLI

Если оборудование поддерживает API, как например, NXOS, то для него создано большое количество модулей, которые выполняют конкретные действия, по настройке функционала (например, для NXOS создано более 60 модулей).

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

Для оборудования, которое работает только через CLI, Ansible поддерживает такие три типа модулей:

- os_command - выполняет команды show
- os_facts - собирает факты об устройствах
- os_config - выполняет команды конфигурации

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

Соответственно, для разных операционных систем, будут разные модули. Например, для Cisco IOS, модули будут называться:

- ios_command
- ios_config
- ios_facts

МОДУЛИ ДЛЯ РАБОТЫ С СЕТЕВЫМ ОБОРУДОВАНИЕМ

Аналогичные три модуля доступны для таких ОС:

- Dellos10
- Dellos6
- Dellos9
- EOS
- IOS
- IOS XR
- JUNOS
- SR OS
- VyOS

ОСОБЕННОСТИ ПОДКЛЮЧЕНИЯ К СЕТЕВОМУ ОБОРУДОВАНИЮ

При работе с сетевым оборудованием надо указать, что должно использоваться подключение типа network_cli. Это можно указывать в инвентарном файле, файлах с переменными и т.д.

Пример настройки для сценария (play):

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

ОСОБЕННОСТИ ПОДКЛЮЧЕНИЯ К СЕТЕВОМУ ОБОРУДОВАНИЮ

В Ansible переменные можно указывать в разных местах, поэтому те же настройки можно указать по-другому.

Например, в инвентарном файле:

```
[cisco-routers]
192.168.100.1
192.168.100.2
192.168.100.3

[cisco-switches]
192.168.100.100

[cisco-routers:vars]
ansible_connection=network_cli
```

ОСОБЕННОСТИ ПОДКЛЮЧЕНИЯ К СЕТЕВОМУ ОБОРУДОВАНИЮ

Такой вариант подходит в том случае, когда Ansible используется больше для подключение к сетевым устройствам (или, локальные playbook используются для подключения к сетевому оборудованию).

В таком случае, нужно будет наоборот явно включать сбор фактов, если он нужен.

ОСОБЕННОСТИ ПОДКЛЮЧЕНИЯ К СЕТЕВОМУ ОБОРУДОВАНИЮ

Указать, что нужно использовать локальное подключение, также можно по-разному.

В инвентарном файле:

```
[cisco-routers]
192.168.100.1
192.168.100.2
192.168.100.3

[cisco-switches]
192.168.100.100

[cisco-routers:vars]
ansible_connection=network_cli
```

ОСОБЕННОСТИ ПОДКЛЮЧЕНИЯ К СЕТЕВОМУ ОБОРУДОВАНИЮ

Или в файлах переменных, например, в group_vars/all.yml:

```
ansible_connection: network_cli
```

Модули, которые используются для работы с сетевым оборудованием, требуют задания нескольких параметров.

- ansible_network_os - например, ios, eos
- ansible_user - имя пользователя
- ansible_password - пароль
- ansible_become - нужно ли переходить в привилегированный режим (enable, для Cisco)
- ansible_become_method - каким образом надод переходить в привилегированный режим
- ansible_become_pass - пароль для привилегированного режима

Пример указания всех параметров в group_vars/all.yml:

```
ansible_connection: network_cli
ansible_network_os: ios
ansible_user: cisco
ansible_password: cisco
ansible_become: yes
ansible_become_method: enable
ansible_become_pass: cisco
```

ПОДГОТОВКА К РАБОТЕ С СЕТЕВЫМИ МОДУЛЯМИ

Инвентарный файл myhosts:

```
[cisco-routers]
192.168.100.1
192.168.100.2
192.168.100.3

[cisco-switches]
192.168.100.100
```

ПОДГОТОВКА К РАБОТЕ С СЕТЕВЫМИ МОДУЛЯМИ

Конфигурационный файл ansible.cfg:

```
[defaults]

inventory = ./myhosts
```

ПОДГОТОВКА К РАБОТЕ С СЕТЕВЫМИ МОДУЛЯМИ

В файле group_vars/all.yml надо создать параметры для подключения к оборудованию:

```
ansible_connection: network_cli
ansible_network_os: ios
ansible_user: cisco
ansible_password: cisco
ansible_become: yes
ansible_become_method: enable
ansible_become_pass: cisco
```

МОДУЛЬ IOS_COMMAND

МОДУЛЬ IOS_COMMAND

Модуль **ios_command** - отправляет команду show на устройство под управлением IOS и возвращает результат выполнения команды.

Модуль ios_command не поддерживает отправку команд в конфигурационном режиме. Для этого используется отдельный модуль - ios_config.

МОДУЛЬ IOS_COMMAND

Модуль ios_command поддерживает такие параметры:

- commands - список команд, которые надо отправить на устройство
- wait_for (или waitfor) - список условий на которые надо проверить вывод команды. Задача ожидает выполнения всех условий. Если после указанного количества попыток выполнения команды условия не выполняются, будет считаться, что задача выполнена неудачно.

МОДУЛЬ IOS_COMMAND

Модуль ios_command поддерживает такие параметры:

- match - этот параметр используется вместе с wait_for для указания политики совпадения. Если параметр match установлен в all, должны выполниться все условия в wait_for. Если параметр равен any, достаточно чтобы выполнилось одно из условий.
- retries - указывает количество попыток выполнить команду, прежде чем она будет считаться невыполненной. По умолчанию - 10 попыток.
- interval - интервал в секундах между повторными попытками выполнить команду. По умолчанию - 1 секунда.

МОДУЛЬ IOS_COMMAND

Перед отправкой самой команды, модуль:

- выполняет аутентификацию по SSH,
- переходит в режим enable
- выполняет команду `terminal length 0`, чтобы вывод команд show отражался полностью, а не постранично.
- выполняет команду `terminal width 512`

МОДУЛЬ IOS_COMMAND

Пример использования модуля ios_command (playbook 1_ios_command.yml):

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

  tasks:

    - name: run sh ip int br
      ios_command:
        commands: show ip int br
      register: sh_ip_int_br_result

    - name: Debug registered var
      debug: var=sh_ip_int_br_result.stdout_lines
```

МОДУЛЬ IOS_COMMAND

Модуль ios_command ожидает параметры:

- commands - список команд, которые нужно отправить на устройство
  - в нашем случае, он указан в файле group_vars/all.yml

Обратите внимание, что параметр register находится на одном уровне с именем задачи и модулем, а не на уровне параметров модуля ios_command.

Результат выполнения playbook:

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

```
SSH password:

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

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

TASK [Debug registered var] ************************************************
ok: [192.168.100.1] => {
    "sh_ip_int_br_result.stdout_lines": [
        [
            "Interface                  IP-Address      OK? Method Status                Protocol",
            "Ethernet0/0                192.168.100.1   YES NVRAM  up                    up      ",
            "Ethernet0/1                192.168.200.1   YES NVRAM  up                    up      ",
            "Ethernet0/2                unassigned      YES manual administratively down down    ",
            "Ethernet0/3                unassigned      YES manual up                    up      ",
            "Loopback0                  10.1.1.1        YES manual up                    up      "
        ]
    ]
}
ok: [192.168.100.2] => {
    "sh_ip_int_br_result.stdout_lines": [
        [
            "Interface                  IP-Address      OK? Method Status                Protocol",
            "Ethernet0/0                192.168.100.2   YES manual up                    up      ",
            "Ethernet0/1                unassigned      YES unset  administratively down down    ",
            "Ethernet0/2                192.168.200.1   YES manual administratively down down    ",
            "Ethernet0/3                unassigned      YES manual up                    up      ",
            "Loopback0                  10.1.1.1        YES manual up                    up      "
        ]
    ]
}
ok: [192.168.100.3] => {
    "sh_ip_int_br_result.stdout_lines": [
        [
            "Interface                  IP-Address      OK? Method Status                Protocol",
            "Ethernet0/0                192.168.100.3   YES manual up                    up      ",
            "Ethernet0/1                unassigned      YES unset  administratively down down    ",
            "Ethernet0/2                192.168.200.1   YES manual administratively down down    ",
            "Ethernet0/3                unassigned      YES manual up                    up      ",
            "Loopback0                  10.1.1.1        YES manual up                    up      ",
            "Loopback10                 10.255.3.3      YES manual up                    up      "
        ]
    ]
}

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

ВЫПОЛНЕНИЕ НЕСКОЛЬКИХ КОМАНД

Playbook 2_ios_command.yml выполняет несколько команд и получает их вывод:

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

  tasks:

    - name: run show commands
      ios_command:
        commands:
          - show ip int br
          - sh ip route
      register: show_result

    - name: Debug registered var
      debug: var=show_result.stdout_lines
```

ВЫПОЛНЕНИЕ НЕСКОЛЬКИХ КОМАНД

Результат выполнения playbook (вывод сокращен):

```
$ ansible-playbook 2_ios_command.yml
```

```
SSH password:

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

TASK [run show commands] *******************************************************
ok: [192.168.100.3]
ok: [192.168.100.1]
ok: [192.168.100.2]

TASK [Debug registered var] ****************************************************
ok: [192.168.100.1] => {
    "show_result.stdout_lines": [
        [
            "Interface                  IP-Address      OK? Method Status                Protocol",
            "Ethernet0/0                192.168.100.1   YES NVRAM  up                    up      ",
            "Ethernet0/1                192.168.200.1   YES NVRAM  up                    up      ",
            "Ethernet0/2                unassigned      YES manual administratively down down    ",
            "Ethernet0/3                unassigned      YES manual up                    up      ",
            "Loopback0                  10.1.1.1        YES manual up                    up      "
        ],
        [
            "Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP",
            "       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area ",
            "       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2",
            "       E1 - OSPF external type 1, E2 - OSPF external type 2",
            "       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2",
            "       ia - IS-IS inter area, * - candidate default, U - per-user static route",
            "       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP",
            "       + - replicated route, % - next hop override",
            "",
            "Gateway of last resort is not set",
            "",
            "      10.0.0.0/32 is subnetted, 2 subnets",
            "C        10.1.1.1 is directly connected, Loopback0",
            "D        10.255.3.3 [90/409600] via 192.168.100.3, 02:04:51, Ethernet0/0",
            "      192.168.100.0/24 is variably subnetted, 2 subnets, 2 masks",
            "C        192.168.100.0/24 is directly connected, Ethernet0/0",
            "L        192.168.100.1/32 is directly connected, Ethernet0/0",
            "      192.168.200.0/24 is variably subnetted, 2 subnets, 2 masks",
            "C        192.168.200.0/24 is directly connected, Ethernet0/1",
            "L        192.168.200.1/32 is directly connected, Ethernet0/1"
        ]
    ]
}
```

ВЫПОЛНЕНИЕ НЕСКОЛЬКИХ КОМАНД

Если модулю передаются несколько команд, результат выполнения команд находится в переменных stdout и stdout_lines в списке. Вывод будет в том порядке, в котором команды описаны в задаче.

Засчет этого, например, можно вывести результат выполнения первой команды, указав:

```
    - name: Debug registered var
      debug: var=show_result.stdout_lines[0]
```

ОБРАБОТКА ОШИБОК

В модуле встроено распознание ошибок. Поэтому, если команда выполнена с ошибкой, модуль отобразит, что возникла ошибка.

Например, если сделать ошибку в команде, и запустить playbook еще раз

```
$ ansible-playbook 2_ios_command.yml
```

ОБРАБОТКА ОШИБОК

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

TASK [run show commands] *******************************************************
fatal: [192.168.100.2]: FAILED! => {"changed": false, "failed": true, "msg": "shw ip in
t br\r\n     ^\r\n% Invalid input detected at '^' marker.\r\n\r\nR2#", "rc": 1}
fatal: [192.168.100.3]: FAILED! => {"changed": false, "failed": true, "msg": "shw ip in
t br\r\n     ^\r\n% Invalid input detected at '^' marker.\r\n\r\nR3#", "rc": 1}
fatal: [192.168.100.1]: FAILED! => {"changed": false, "failed": true, "msg": "shw ip in
t br\r\n     ^\r\n% Invalid input detected at '^' marker.\r\n\r\nR1#", "rc": 1}
        to retry, use: --limit @/home/vagrant/repos/pyneng-online-jun-jul-2017/examples
/15_ansible/3_network_modules/ios_command/2_ios_command.retry

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

ОБРАБОТКА ОШИБОК

Ansible обнаружил ошибку и возвращает сообщение ошибки. В данном случае - 'Invalid input'.

Аналогичным образом модуль обнаруживает ошибки:

- Ambiguous command
- Incomplete command

WAIT_FOR

Пример playbook (файл 3_ios_command_wait_for.yml):

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

  tasks:

    - name: run show commands
      ios_command:
        commands: ping 192.168.100.100
        wait_for:
          - result[0] contains 'Success rate is 100 percent'
```

WAIT_FOR

В playbook всего одна задача, которая отправляет команду ping 192.168.100.100 и проверяет есть ли в выводе команды фраза 'Success rate is 100 percent'.

Если в выводе команды содержится эта фраза, задача считается корректно выполненой.

Запуск playbook:

```
$ ansible-playbook 3_ios_command_wait_for.yml -v
```

WAIT_FOR