OSEC

Know your tools: Ansible Networking

Dariusz Puchalak

Dariusz Puchalak

- 20+ lat Linux/Unix Sysadmin
- 10+ lat trener
- 4+ lat w OSEC
- 4+ lat z Ansible

http://www.osec.pl

- Od 2009 na rynku
- doświadczona kadra (ACNI, RHCA)
- specjalizacja open-source
- subskrypcje, szkolenia, wdrożenia, konsultacje

Dlaczego Ansible?

Ansible

- silnik do automatyzacji systemów IT.
- Linux
- Windows
- Clouds
- Security
- Networks

Ansible Network Automation

- Proste
- Brak agentów
- Natywny sposób komunikacji z zarządzanymi urządzeniami
- Wspiera SSH
- Wspiera podnoszenie uprawnień

Czy automatyzacja to „tylko” zarządzanie konfiguracją?

- Czy można zlecić wymianę routera prawie dowolnej osobie?
- Upgrade firmwareu?
- Wgranie właściwej konfiguracji?
- Dodanie „niewspieranych” elementów konfiguracji?
- Fizyczną wymianę sprzętu?

Ansible i urządzenia sieciowe kiedyś.

Jak brakowało modułów.

```
edgemax_run_configure_command: |
   source /opt/vyatta/etc/functions/script-template
    configure


edgemax_run_command: /opt/vyatta/bin/vyatta-op-cmd-wrapper


shell: '{{ edgemax_run_command }}  show version | grep "Version:" | egrep -o
"[.0-9]+"'


shell: 'yes | {{ edgemax_run_command }} add system image /tmp/
{{ ubiquity_edgemax_firmware_file }}'
```

ubiquity-edgemax-initialize

Ansible i urządzenia sieciowe dzisiaj.

ios_ping

```
---
- name: a reachability test
  hosts: rtr1
  vars:
    destination: 192.168.17.18
  tasks:
  - name: "test reachability to {{ destination }}"
    ios_ping:
      dest: "{{ destination }}"
```

```
---
- name: back up config and looks at device health indicators on ios devices
  hosts: ios

  tasks:
  - name: backup the device configuration
    ios_config:
      backup: yes

  - name: look at device health indicators
    ios_command:
      commands:
        # this provides hostname and uptime
        - sh ver | include uptime
        - sh ip domain
        - sh clock
        - sh ip name-server
        - sh proc mem | include Total
    register: results

  - name: show results
    debug:
      msg: "{{ item }}"
    loop: "{{ results.stdout_lines }}"
...
```

Różnice pomiędzy automatyzacją Linux/Windows a urządzeń sieciowych.

Wykonywanie kodu.

- Linux – Python
- Windows – Python/PowerShell
- Urządzenia sieciowe – Python

Ale w przypadku urządzeń sieciowych całość uruchamiania jest po stronie stacji zarządzającej.

Wykonywanie kodu.

Ale w przypadku urządzeń sieciowych całość uruchamiania jest po stronie stacji zarządzającej!

Połączenia.

- Linux – SSH
- Windows – WinRM
- Urządzenia sieciowe – SSH
- Inne – API danego rozwiązania

Połączenia dla urządzeń sieciowych.

- network_cli
- netconf
- httpapi
- local

Połączenia dla urządzeń sieciowych.

- network_cli – większość nowych modułów
- netconf – niektóre moduły
- httpapi – niektóre moduły – dla sprzętu udostępniającego API
- local – stara metoda (nie rekomendowana min. ze względu na brak stałego (ang. persistent) połączenia

Podnoszenie uprawnień.

- become: yes
- become_method: enable
- authorize: yes (dla starych wersji)

Fakty.

- Faktów brak. ;]

Fakty.

- Fakty można zebrać przy pomocy:
  - cnos_facts
  - edgeos_facts
  - enos_facts
  - eos_facts
  - exos_facts
  - ios_facts
  - junos_facts
  - nos_facts
  - nxos_facts
  - slxos_facts
  - vultr_os_facts
  - vyos_facts

ansible-doc ios_facts

```
# hardware
ansible_net_filesystems:
  description: All file system names available on the device
  returned: when hardware is configured
  type: list
ansible_net_filesystems_info:
  description: A hash of all file systems containing info about each file system (e.g. free and
  returned: when hardware is configured
  type: dict
ansible_net_memfree_mb:
  description: The available free memory on the remote device in Mb
  returned: when hardware is configured
  type: int
ansible_net_memtotal_mb:
  description: The total memory on the remote device in Mb
  returned: when hardware is configured
  type: int
```

Moduły.

Każda platforma ma osobne moduły:

- Arista: eos_
- Cisco: ios_, iosxr_, nxos_
- EdgeOS: edgeos_
- Juniper: junos_
- VyOS: vyos_

Moduły.

- ansible-doc -l | grep edgeos_
- ansible-doc -l | grep ^ios_
- ansible-doc -l | grep ^vyos_

Podstawowe moduły.

- *os_command
- *os_config
- *os_facts
- *os_interface
- *os_l3_interface
- *os_logging
- *os_ping
- *os_static_route
- *os_system
- *os_user
- *os_vlan

Ale....

- Nie wszystkie są dostępne na każdej platformie. :(

Przykłady.

traceroute z routera

```
---
# run ansible-playbook tracert.yml -e dest=192.168.1.1
- name: traceroute to {{ dest }}
  hosts: core
  tasks:
  - name: traceroute to dest
    ios_command:
      commands:
        - traceroute {{ dest }} probe 2 timeout 2
    register: result

  - name: show result
    debug:
      var: result.stdout_lines
...
```

ping

```
---
- name: A reachability test
  hosts: core

  vars_files:
    - vars/myvars.yml

  tasks:
    - name: "Test reachability from {{ inventory_hostname }} to interesting destinations"
    ios_ping:
      dest: "{{ item }}"
    loop: "{{ interesting_destinations }}"
...
```

another ping

```
---
- name: A reachability test
  hosts: core

  vars_files:
  - vars/myvars.yml

  tasks:
  - name: "test reachability from target to 172.25.250"
    ios_ping:
      dest: "{{ item }}"
    loop: "{{ ipv4_addresses | select('match', '^172\\.25\\.250\\..*') | list }}"
    when: ansible_network_os == 'ios'
...
```

Multivendor playbook?

Multivendor playbook 1/2

```
---
- name: back up config and record device health indicators
  hosts: network

  tasks:
  - name: backup vyos config
    vyos_config:
      backup: yes
    when: ansible_network_os == "vyos"

  - name: look at vyos device health indicators
    vyos_command:
      commands:
        - sh host name
        - sh system uptime
        - sh host domain
        - sh host date
        - sh host os
        - sh sys mem
    register: vyos_result
    when: ansible_network_os == "vyos"
```

Multivendor playbook 2/2

```
- name: backup ios config
  ios_config:
    backup: yes
  when: ansible_network_os == "ios"

- name: look at ios device health indicators
  ios_command:
    commands:
      - sh ver | include uptime
      - sh ip domain
      - sh clock
      - sh ip name-server
      - sh proc mem platform | include System memory
  register: ios_result
  when: ansible_network_os == "ios"

- name: show results
  debug:
    msg: "{{ item }}"
  loop: "{{ (vyos_result | combine(ios_result)).stdout_lines }}"

...
```

Multivendor lepiej (Ansible 2.7+).

- cli_command - Run a cli command on cli-based network devices
- cli_config - Push text based configuration to network devices over network_cli

  Uwaga: dotyczy tylko modułów wspierających połączenia poprzez network_cli

Filtry

```
{{ "http://user:password@www.acme.com:9000/dir/index.html?
query=term#fragment" | urlsplit }}
# =>
#   {
#       "fragment": "fragment",
#       "hostname": "www.acme.com",
#       "netloc": "user:password@www.acme.com:9000",
#       "password": "password",
#       "path": "/dir/index.html",
#       "port": 9000,
#       "query": "query=term",
#       "scheme": "http",
#       "username": "user"
#   }
```

output to json

```
{{ output | parse_cli('path/to/spec') }}


---
vars:
  vlan:
    vlan_id: "{{ item.vlan_id }}"
    name: "{{ item.name }}"
    enabled: "{{ item.state != 'act/lshut' }}"
    state: "{{ item.state }}"

keys:
  vlans:
    value: "{{ vlan }}"
    items: "^(?P<vlan_id>\\d+)\\s+(?P<name>\\w+)\\s+(?P<state>active|act/lshut|suspended)"
  state_static:
    value: present
```

Troubleshooting

Ansible debugger.

```
puchalakd@neptune:~/Ansible-demo$ ansible-playbook playbook-debugger.yml

PLAY [localhost] ***************************************************************

TASK [wrong variable] **********************************************************
fatal: [localhost]: FAILED! => {"msg": "The task includes an option with an undefined variable
. The error was: 'wrong_var' is undefined\n\nThe error appears to have been in '/home/puchalak
d/Ansible-demo/playbook-debugger.yml': line 7, column 7, but may\nbe elsewhere in the file dep
ending on the exact syntax problem.\n\nThe offending line appears to be:\n\n  tasks:\n    - na
me: wrong variable\n      ^ here\n"}
[localhost] TASK: wrong variable (debug)>
```

Best practice.

Role.

Roles

- Budujcie role
- Role powinny robić tylko jedną rzecz
- Uwaga na konflikt nazw zmiennych
- Podwójna uwaga na handlery
- Rola może i powinna być uniwersalna (ale nie zawsze wiele dystrybucji, Linux, Windows, urządzenia sieciowe ogarnięte w jeden roli mają sens. Lepiej użyć zależności lub include_role/import_role).

Tagi

Tags

- Używać zawsze i wszędzie :)
- Testować w trybie dry-run
- Testować poprzez uruchomienie tylko z danym tagiem (–tags testowany)
- Pamiętać o dobrym otagowaniu zadań z modułami non-idempotent (np.. shell, raw, command). Oraz jeśli nic nie zmieniają to dodaniu check_mode: no .
- Min 2-3 tagi:
- Ogólny do zadania
- Szczegółowy dot. danego zadania

Łączyć kod dla systemów Linux.

```
- include_vars: "{{ item }}"
  with_first_found:
    - "../vars/{{ ansible_distribution }}-{{ ansible_distribution_major_version | int}}.yml"
    - "../vars/{{ ansible_distribution }}.yml"
    - "../vars/{{ ansible_os_family }}.yml"
    - "../vars/package_default.yml"
  when: package_prerequisites is not defined
  tags: ['ubertooth' ]

- name: Install ubertooth prerequisites
  package:
    name: "{{ item }}"
    state: latest
  with_items: "{{ package_prerequisites }}"
  tags: ['ubertooth' ]
```

Osobno traktować Linuksy i Windows.

```
- name: gather os specific variables
  include_vars: "{{ item }}"
  with_first_found:
    - files:
        - "{{ ansible_distribution }}-{{ ansible_distribution_major_version}}.yml"
        - "{{ ansible_distribution }}.yml"
        - "{{ ansible_os_family }}.yml"
      skip: true # Skip if no var files found
  tags: [ vars, 'check_mk-agent', 'check_mk-server' ]


- { include: check-mk-agent-linux.yml, when: ansible_system is defined and ansible_system == "Linux" }
- { include: check-mk-agent-windows.yml, when: ansible_system is defined and ansible_system == "Win32NT" }
- { include: check-mk-agent-generate-config.yml , when: ansible_system is defined and ansible_system == "Linux" or ansible_system == "Win32NT" }
```

Osobno traktować Linuksy i Windows.

```
puchalakd@neptune:~/Ansible-demo$ ansible -m setup w2k12 | grep reboot
        "ansible_reboot_pending": false,
puchalakd@neptune:~/Ansible-demo$ ansible -m setup localhost | grep reboot
puchalakd@neptune:~/Ansible-demo$

root@pluton:~# needrestart -rl -pk ; echo $?
CRIT - Kernel: 4.9.0-7-amd64!=4.9.0-8-amd64 (!)|Kernel=2;0;;0;2
2
```

Urządzenia sieciowe.

- Każda rodzina osobno. :(
- Ale można używać cli_command i cli_config :)
- Po naszej stronie leży znajomość danych urządzeń i obługiwanych przez nich poleceń/funkcji.

Pluginy

Przeglądać dokumentacje pluginów. Można znaleźć coś ciekawego np:

https://docs.ansible.com/ansible/2.5/plugins/callback/selective.html

This callback only prints tasks that have been tagged with print_action or that have failed. This allows operators to focus on the tasks that provide value only.

Pluginy -

```
ANSIBLE_STDOUT_CALLBACK=actionable
ansible-playbook production.yml --user rootdp
--skip-tags strongswan,bareos -CD
```

Jinja filters.

```
# SSH access for admins
SSH/ACCEPT:$LOG         any:{{ admin_ips | join(",") }}   fw
# SSH access for monitoring
SSH/ACCEPT:$LOG         any:{{ check_mk_servers | join(",") }}    fw
# Proxy for apt-get/yum to access repositories
ACCEPT:debug    fw      any:{{ proxy_repo | urlsplit('hostname') }}       tcp     {{ proxy_repo | urlsplit('port') }}
# Syslog remote
ACCEPT:debug    fw      any:{{ rsyslog_server }}        tcp     514
# system mails
ACCEPT:debug    fw      any:{{ mail_relay }}        tcp     25
# NTP
ACCEPT:debug    fw      any:{{ ntp_servers | join(",") }} udp ntp

                                                                                                1,1           All
```

Jinja filters.

```
- name: Create remote users for share Umowy
  win_user:
    account_disabled: no
    account_locked: no
    name: "{{item.name}}"
    fullname: "{{item.fullname if item.fullname is defined else ''}}"
    description: "{{item.description if item.description is defined else ''}}"
    groups: "{{item.groups | join(',')}}"
    password: "{{item.password if item.password is defined else default_password}}"
    groups_action: add
    password_expired: no # Don't require user to change password on first login
    password_never_expires: yes
    state: present
    update_password: on_create # If it exists do not update password
    user_cannot_change_password: yes
  with_items: "{{ users }}"
  tags: [users, config, windows]
-- INSERT --                                                    1,44
```

Jinja filters.

```
##### data from host_vars/system.dns.name/local_users.yaml
default_password: Pa$$w0rd

- name: Delete old users
  win_user:
    name: "{{item.name}}"
    state: absent
  with_items: "{{ users_deleted }}"
  tags: [users, config, windows]
```

```
users:
  - fullname: Joe Test
    name: test1
    groups: test1, test2
    description: Test user 1
  - fullname: Joe Doe
    name: test2
    groups: test2, admins
    description: Test user 2

users_deleted:
  - name: Jan Testowy
```

```
-- INSERT --                                                    34,1          65%
```

Pytania?

Dariusz.Puchalak@osec.pl