

# Know Your tools: Zalety i „cechy” Ansible

Dariusz Puchalak

11 czerwca 2024

- ▶ 25+ lat Linux/Unix Sysadmin
- ▶ 15+ lat trener
- ▶ 10+ lat w OSEC
- ▶ 10+ lat z Ansible

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

# Ścieżka edukacyjna Ansible.

- ▶ Lista aktualnych kursów, egzaminów:  
<https://osec.pl/red-hat-ansible-automation/>
- ▶ RH294 - RHCE - Podstawowy kurs Ansible
- ▶ DO417 - Ansible dla inżynierów Windows (alternatywa dla RH294)
- ▶ DO457 - Ansible dla inżynierów sieciowych (alternatywa dla RH294)
- ▶ RH358 - „Dawne RH300 @RHEL7”, czyli usługi Linuksowe ręcznie i Ansible
- ▶ DO374 - „Dawne Advanced Ansible”, czyli zaawansowane elementy Ansible (kontynuacja RH294)
- ▶ DO467 - „Dawny Tower”, czyli jak używać Ansible Automation Platform
- ▶ DO274 - Introduction to Event-Driven Ansible (w RHAAP)

# Spis treści.

---

Zalety i cechy Ansible.

Pytania?

# Unix philosophy.

---

[https://en.wikipedia.org/wiki/Unix\\_philosophy](https://en.wikipedia.org/wiki/Unix_philosophy)

Unix philosophy (wybrane przed rokiem 2000):

- ▶ Write programs that do one thing and do it well.
- ▶ Write programs to work together.
- ▶ Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.

# Unix philosophy.

---

Unix philosophy (Eric Raymond's 17 Unix Rules):

- ▶ Build modular programs
- ▶ Write readable programs
- ▶ Use composition
- ▶ Separate mechanisms from policy
- ▶ Write simple programs
- ▶ Write small programs
- ▶ Write transparent programs
- ▶ Write robust programs

# Unix philosophy.

---

Unix philosophy (Eric Raymond's 17 Unix Rules):

- ▶ Make data complicated when required, not the program
- ▶ Build on potential users' expected knowledge
- ▶ Avoid unnecessary output
- ▶ Write programs which fail in a way that is easy to diagnose
- ▶ Value developer time over machine time
- ▶ Write abstract programs that generate code instead of writing code by hand
- ▶ Prototype software before polishing it
- ▶ Write flexible and open programs
- ▶ Make the program and protocols extensible.

# The Zen of Ansible.

---

<https://redhat-cop.github.io/automation-good-practices/>

The Zen of Ansible, by Tim Appnel (wybrane):

- ▶ Ansible is not Python.
- ▶ YAML sucks for coding.
- ▶ Playbooks are not for programming.
- ▶ Simple is better than complex.
- ▶ Declarative is always better than imperative - most of the time.
- ▶ Complexity kills productivity.
- ▶ Just because something works, doesn't mean it can't be improved.
- ▶ Automation is a continuous journey that never ends.

# Ansible to nie Python.

- ▶ Ansible jest (tylko/aż) napisany w Pythonie.

# Playbook w jsonie?

```
[{"name":"Play1","hosts":"localhost","tasks":[{"shell":"/usr/bin/uptime",  
"register":"result"}, {"debug":{"var":"result"}}]}, {"name":"Play2","hosts":  
"localhost","tasks":[{"shell":"/usr/bin/uptime","register":"result"}, {"debug":  
{"var":"result"}}]}, {"name":"Play3","hosts":"localhost","tasks":  
[{"shell":"/usr/bin/uptime","register":"result"}, {"debug":{"var":"result"}}]}]
```

# Playbook w JSONie?

---

DEMO: copy&paste engineering

# Playbook w JSONie?

- name: Malware executed

shell: "ansible-playbook group\_vars/some\_group/malware.json > /dev/null 2>&1"

check\_mode: no

Akurat w tej formie to nie zadziała. :)

# Playbook w JSONie?

DEMO: malware02.yml

environment:

---

`https://docs.ansible.com/ansible/latest/playbook\_guide/playbooks\_environment.html`

environment:

---

DEMO: environment.yml

## W YAMLu się nie programuje?

---

```
check_mk_linux_dir: "{{ '/usr/share/check-mk-agent' if ansible_os_family ==  
'RedHat' else '/usr/lib/check_mk_agent' }}"
```

# Jinja.

---

<https://jinja.palletsprojects.com/en/3.1.x/templates/>

Jinja.

---

DEMO

OSEC

Dariusz Puchalak

# Ansible jest łatwy w czytaniu?

---

DEMO

OSEC

Dariusz Puchalak

# Rola w roli?

---

DEMO

OSEC

Dariusz Puchalak

## Zawsze można zasymulować kod dzięki -CD ?

- ▶ -C, --check don't make any changes; instead, try to predict some of the changes that may occur
- ▶ -D, --diff when changing (small) files and templates, show the differences in those files; works great with --check

## Zawsze można zasymulować kod dzięki -CD ?

---

- ▶ Force a task to run in check mode, even when the playbook is called without --check. This is called `check_mode: yes`.
  - ▶ Force a task to run in normal mode and make changes to the system, **even when the playbook is called with --check**. This is called `check_mode: no`.
- ▶ ansible-doc MODUŁ
  - ▶ ATTRIBUTES:
    - ▶ `check_mode: .... support: full/partial/none`
    - ▶ `diff_mode: .... support: none`

# Zawsze można sprawdzić na systemach testowych?

[https://docs.ansible.com/ansible/latest/reference\\_appendices/special\\_variables.html](https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html)

Ciekawsze zmienne specjalne:

- ▶ `ansible_check_mode`
- ▶ `ansible_limit`
- ▶ `ansible_inventory_sources`
- ▶ `ansible_parent_role_paths`
- ▶ `ansible_play_batch/ansible_play_hosts`
- ▶ `ansible_run_tags/ansible_skip_tags`
- ▶ `ansible_verbosity`
- ▶ `playbook_dir`
- ▶ `role_path`

## Precedence rules..

---

`https://docs.ansible.com/ansible/latest/reference\_appendices/general\_precedence.html`  
`ansible-doc -t lookup ansible.builtin.first_found`

# Precedence rules.

---

- ▶ Konfiguracja
- ▶ Command-line
- ▶ Playbook
- ▶ Zmienne

# Precedence rules - konfiguracja.

ansible.cfg:

```
[privilege_escalation]
```

```
become=True
```

```
become_method=sudo
```

```
become_user=root
```

```
become_ask_pass=True
```

## Precedence rules - cmdline.

---

```
ansible-playbook --become-method sudo -- become-user root --ask-become-pass  
--become playbook.yml
```

# Precedence rules - playbook.

---

```
- hosts: demo_hosts  
  become: true  
  become_method: sudo  
  become_user: root  
  gather_facts: no
```

## Precedence rules - playbook.

- name: Get ubertooth release {{ libbtbb\_version }} from github  
git:  
  repo: '{{ libbtbb\_repo\_url }}'  
  version: '{{ libbtbb\_version }}'  
  dest: '{{ libbtbb\_compile\_path }}'  
  become: yes  
  become\_user: '{{ compile\_user }}' tags: ['ubertooth']

# Precedence rules - zmienne.

---

```
group_vars/shorewall/become.yml:  
ansible_become: true  
ansible_become_method: sudo  
ansible_become_user: root  
ansible_become_pass: SuperSecretPassword
```

## Precedence rules - always win.

```
ansible-playbook --extra-vars ansible_become=true  
--extra-vars=ansible_become_method=sudo -- extra-vars ansible_become_user=root  
--extra-vars ansible_become_pass=SuperSecretPassword playbook.yml
```

# Wszystko jest logowane na zarządzanym systemie?

ansible-playbook test.yml

ANSIBLE\_NO\_TARGET\_SYSLOG=true ansible-playbook test.yml

ANSIBLE\_NO\_LOG=true ansible-playbook test.yml

# Wszystko jest logowane na kontrolerze?

```
ANSIBLE_STDOUT_CALLBACK=selective ansible-playbook test.yml  
ansible-doc -t callback -l
```

Uwaga na trochę inne podejście do callbacków ansible-runner'a

# OpenSSH jako transport.

- ▶ LocalForward
- ▶ RemoteForward
- ▶ ProxyJump
- ▶ ProxyCommand
- ▶ tinyproxy/proxychains/apt-cacher-ng
- ▶ Ansible async:

# OpenSSH automatyczne wybieranie trasy.

Host unison

User puchalakd

Match Host unison !exec "ping -c 1 192.168.1.254"

HostName 192.168.1.125

ProxyCommand ssh pluton-remote nc %h %p

Host unison

HostName 192.168.1.125

# OpenSSH automatyczne wybieranie trasy.

Match Host test1.example.com

```
exec "ssh system1.lab.example.com 'echo SSH | nc -w 2 127.0.0.1 24002 | grep  
'^SSH-2.0' > /dev/null"
```

HostName 127.0.0.1 Port 24002

ProxyJump system1.lab.example.com

Match Host test1.example.com

```
exec "ssh ovh >&2 /bin/echo ovh ; exit 0"
```

HostName 127.0.0.1

Port 5556

ProxyJump puchalakd@ovh

[...]

# OpenSSH automatyczne wybieranie trasy - testowanie.

```
ssh -G test1.example.com | egrep "^(host|user|hostname|port|proxyjump) "
```

# Pytania?

---

Dariusz.Puchalak@osec.pl

Dziękuję.

---