Ansible

Marcin Słowikowski

1. Dokumentacja

Wspomnieć że dokumentacja nie gryzie https://docs.ansible.com/ansible/latest/

2. Instalacja

Sposoby instalacji, który wybrać i dlaczego

Dokumentacja

Instalację poprzez package manager wykonamy uruchamiając poniższe komendy na hoście Ansible:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Dodatkowo warto zainstalować argcomplete w celu dopełniania komend Ansible poprzez Tab

```
sudo apt install python3-argcomplete
sudo activate-global-python-argcomplete3
```

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu

3. Konfiguracja SSH

Ansible domyślnie wykorzystuje systemową konfigurację SSH

Stwórzmy użytkownika na maszynach node

```
sudo adduser ansible
```

Następnie wykorzystaj komendę visudo by dodać uprawnienia do uruchamiania komendy sudo bez hasła:

```
ansible ALL=(ALL) NOPASSWD:ALL
```

W celu łączenia się do zdalnych maszyn, dobrą praktyką jest wykorzystywanie kluczy SSH zamiast haseł. Obraz Ubuntu z którego korzysta vagrant domyślnie posiada wyłączony dostęp SSH poprzez hasło. W środowisku testowym możemy to zmienić w pliku /etc/ssh/sshd_config

```
PasswordAuthentication yes
```

I zrestartuj usługę SSH

```
sudo systemctl restart sshd
```

Utwórz parę kluczy na maszynie ansible a następnie dodaj klucz publiczny na serwerzach node:

```
ssh-keygen
ssh-copy-id ansible@10.0.0.11
```

Jeśli nie chcemy zmieniać ustawień usługi SSH na maszynie node, możesz utworzyć plik ~/.ssh/authorized_keys i ręcznie przenieść do niego zawartość klucza publicznego:

```
mkdir ~/.ssh
chmod 700 ~/.ssh
touch ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Przetestuj czy wykorzystując klucz prywatny możesz zalogować się z maszyny ansible do node

```
ssh ansible@10.0.0.11
```

Następnie przetestuj połączenie poprzez Ansible

```
ansible all --user ansible -i 10.0.0.12,10.0.0.11, -m ping
ansible all --user ansible -i 10.0.0.12,10.0.0.11, -m command -a "whoami"
ansible all --user ansible --become -i 10.0.0.12,10.0.0.11, -m command -a "whoami"
```

https://docs.ansible.com/ansible/latest/cli/ansible.html

https://docs.ansible.com/ansible/latest/user_guide/become.html

Wyjaśnij czym jest --become/-b

eval "$(ssh-agent -s)"

ssh-add /vagrant/.vagrant/machines/node3/virtualbox/private_key

4. Inventory

Ansible musi wiedzieć jak połączyć się do danego hosta i do tego wykorzystujemy inventory

Dokumentacja

Statyczne – plik tekstowy

Dynamicze – plugin/skrypt który wygeneruje wymagane informacje

Domyślnym plikiem inventory jest /etc/ansible/hosts. Możliwe jest również wskazanie pliku inventory poprzez opcję -i <inventory_path>

Dodaj do pliku inventory następującą treść:

```
ansible ansible_connection=local

[nodes]
node1 ansible_host=10.0.0.11
node2 ansible_host=10.0.0.12
10.0.0.[11:12]

[nodes:vars]
ansible_user=ansible
```

Komenda ansible-inventory wyświetla zawartość inventory w postaci w jakiej widzi go Ansible

```
ansible-inventory --host node1
ansible-inventory --graph
ansible-inventory --list
ansible-inventory -i /etc/ansible/hosts --list
```

Uruchom komendę która połączy się do hostów zdefiniowanych w inventory i zwróci hostname oraz adresy IPv4 każdej z maszyn

```
ansible all -m setup -a 'filter=*_ipv4_*,*hostname*'
ansible nodes -m setup -a 'filter=*_ipv4_*,*hostname*'
```

Postaraj się zrozumieć za co odpowiadają elementy które wykorzystaliśmy w inventory: „ansible_connection”, „[nodes]”, „ansible_host”, „[10:11]”, „[nodes:vars]”, „ansible_user”

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

https://docs.ansible.com/ansible/latest/cli/ansible-inventory.html

Omówmy każdą linię inventory.

Wspomnij o domyślnych grupach i domyślnym hoście.

5. Ad-hoc commands

Zapytaj kursantów czym są komendy ad-hoc i w jakim celu je wykorzystujemy

Wyjaśnij jeśli nie pamiętają

Komendy ad-hoc służą do wykonania pojedynczej komendy na wielu maszynach jednocześnie, np. restart maszyn, restart usługi, sprawdzenie wersji OS. Ad-hoc commands powinny być wykorzystywane jedynie doraźnie. Powinno się unikać wykonywania operacji złożonych z wielu komend.

Składnia:

ansible <pattern_grupy/hosta> -m <nazwa_modułu> -a <argumenty>

W poprzednim rozdziale zostały wykorzystane dwie komendy. Obie uruchamiały moduł setup (odpowiedzialny za zbieranie danych) filtrując wynik do elementów zawierających w nazwie „_ipv4_” oraz ”hostname”. Pierwsza wykonywana była dla wszystkich hostów jawnie zdefiniowanych w inventory, druga jedynie dla maszyn w grupie nodes.

Przetestuj podstawowe patterny wykorzystując moduł ping: all, node1, nodes, 10.0.0.*, 'nodes:!node1'

```
ansible all -m ping
```

Restart maszyny node1:

```
ansible node1 -m reboot -b
```

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/reboot_module.html

Na koniec usuń linię definiującą hosty przy pomocy adresu IP z inventory

https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html

https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html

6. Wyszukiwanie modułów

Czym są moduły i po co? Zapytać kursantów

Jak je wyszukiwać?

Przejdź przez dokumentację kilku przykładowych modułów
https://docs.ansible.com/ansible/latest/collections/all_plugins.html

Najprostszym sposobem na wyszukanie moduły jest skorzystanie z indeksu modułów z dokumentacji Ansible lub wykorzystanie Google'a.

Wykorzystaj moduł odpowiadający za zebranie informacji o usługach, by wyświetlić listę usług na node1, a następnie zwróć informacje o jednej z usług

```
ansible node1 -m service_facts
ansible node1 -m service -a "name=sshd"
```

https://docs.ansible.com/ansible/latest/collections/index_module.html

Przerwa

Praca w grupach: instalacja ansible i konfiguracja hostów

7. Playbook

Czym jest playbook?

Kiedy stosujemy playbooki

Przejdź przez dokumentację

Uruchomienie przykładowego playbooka

Stwórz plik install_webserver.yml z poniższą zawartością

```
---
- hosts: nodes
  become: true
  tasks:
    - name: Install Apache
      ansible.builtin.apt:
        name: apache2
        state: present
        update_cache: true

    - name: Change port
      ansible.builtin.lineinfile:
        path: /etc/apache2/ports.conf
        search_string: 'Listen 80'
        line: 'Listen 8080'
        state: present
      notify: Restart Apache

    - name: Create index.html
      ansible.builtin.copy:
        dest: /var/www/html/index.html
        content: "Server: {{ ansible_hostname }}\n"

  handlers:
    - name: Restart Apache
      ansible.builtin.service:
        name: apache2
        state: restarted
```

Uruchom playbook komendą:

```
ansible-playbook install_webserver.yml
```

```
---
- hosts: target1
  become: false
  gather_facts
  tasks:
    - name: Show facts
      debug:
        var: ansible_facts

Wyświetlanie facts

Register

---
- hosts: target1

  tasks:
    - name: Just list files
      shell: ls -a
      register: result

    - debug:
        var: result
```

https://docs.ansible.com/ansible/latest/network/getting_started/first_playbook.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_vars_facts.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html

7.1. Podstawowe elementy playbooka

Z czego składa się playbook?

Wyjaśnienie przykładowych podstawowych elementów playbooka

---	Rozpoczęcie dokumentu. Wynika ze specyfikacji YAML
hosts	Host/grupa/patter wskazujących na których maszynach zostaną wykonane działania
become	Zmienna oznaczająca czy zdefiniowane zadania mają zastać wykonane z podwyższonymi uprawnieniami
tasks	Lista zadań do wykonania
delegate_to	Uruchomienie zadania na określonej maszynie
gather_facts	Zmienna określająca czy moduł setup zostanie uruchomiony automatycznie w celu zebrania danych o maszynach

Więcej słów kluczowych możliwych do wykorzystania w playbookach:

https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html

7.2. Tryby uruchomienia

Ansible może uruchomić playbook w kilku trybach innych niż domyślne uruchomienie w celu wykonania zadań. Tryby te można uruchamiać między innymi przy pomocy opcji dodawanych do komendy ansible-playbook

--list-hosts	Wyświetla listę maszyn na których playbook zostałby uruchomiony, lecz nie wykonuje playbooka
--step	Wymaga potwierdzenia przed uruchomieniem z każdego z zadań
--syntax-check	Przeprowadź kontrolę składni w playbooku, bez wykonania playbooka
--check	Nie wykonuje zmian, lecz stara się przewidzieć zmiany które wystąpią podczas uruchomienia
--diff	Podczas zmiany plików wyświetla dokładne zmiany

https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html

7.3. Eskalacja uprawnień

Niektóre zadania wymagają wyższych uprawnień do wykonania. Np. instalacja paczki wymaga uprawnień użytkownika root. Do eskalacji (podniesienia) uprawnień służy opcja become.

Może zostać użyta w playbooka dla wszystkich zadań, dla konkretnego zadania lub z terminala podczas uruchamiania komendy ansible/ansible-playbook

```
ansible all --become -m command -a "whoami"
ansible-playbook all --become -m command -a "whoami"
```

Domyślnie eskalacja uprawnień wykonywana jest poprzez sudo, a domyślnie wykorzystywanym użytkownikiem do podniesienia uprawnień jest root

https://docs.ansible.com/ansible/latest/user_guide/become.html

7.4. Pętle

Ansible pozwala na wykonanie danego zadania wielokrotnie dla wielu podanych wartości.

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2
```

W podanym przykładzie Ansible zadanie utworzenia użytkowników wykona się dwa razy w celu utworzenia obu użytkowników z listy

```
    - name: Create index.html
      ansible.builtin.copy:
        dest: "/var/www/html/index{{ item }}.html"
        content: "File: {{ item }}\nServer: {{ ansible_hostname }}\n"
      tags: copy
      loop:
        - 1
        - 2
```

Output:

```
TASK [Create index.html] ********************************************************************
changed: [node2] => (item=1)
changed: [node1] => (item=1)
changed: [node2] => (item=2)
changed: [node1] => (item=2)
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

7.5. Warunki

W playbookach Ansible możemy definiować warunki których spełnienie spowoduje wykonanie zadania. W przeciwnym przypadku zadanie zostanie pominięte. Wykorzystywane jest do tego słowo kluczowe when, po którym występuje warunek

```
    - name: Install Apache
      ansible.builtin.apt:
        name: apache2
        state: present
        update_cache: true
      when: ansible_hostname == "node1"
```

Output:

```
TASK [Install Apache] ***********************************************************************
skipping: [node2]
ok: [node1]
```

```
tasks:
  - name: Shut down CentOS 6 systems
    ansible.builtin.command: /sbin/shutdown -t now
    when:
      - ansible_facts['distribution'] == "CentOS"
      - ansible_facts['distribution_major_version'] == "6"
```

```
tasks:
  - name: Shut down CentOS 6 and Debian 7 systems
    ansible.builtin.command: /sbin/shutdown -t now
    when: (ansible_facts['distribution'] == "CentOS" and
ansible_facts['distribution_major_version'] == "6") or
          (ansible_facts['distribution'] == "Debian" and
ansible_facts['distribution_major_version'] == "7")
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html

7.6. Bloki

W celu zastosowania konkretej dyrektywy do zioru zadań należy wykorzystać blok. Bloki pozwalają na grupowanie wielu zadań w logiczny zbiór. Przykładem praktycznego zastosowania jest wykorzystanie bloku by wykonać wiele zadań w konkretnym przypadku. Instalacja i konfiguracja oprogramowania w zależności od wersji OS.

```
tasks:
   - name: Install, configure, and start Apache
     block:
       - name: Install httpd and memcached
         ansible.builtin.yum:
           name:
           - httpd
           - memcached
           state: present

       - name: Apply the foo config template
         ansible.builtin.template:
           src: templates/src.j2
           dest: /etc/foo.conf

       - name: Start service bar and enable it
         ansible.builtin.service:
           name: bar
           state: started
           enabled: True
     when: ansible_facts['distribution'] == 'CentOS'
     become: true
     become_user: root
     ignore_errors: yes
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html

https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#block

7.7. Handlers

Czasami wymagane jest wykonanie danej operacji w przypadku gdy inne zadanie dokonało zmiany. Np. zmiana istotnego pliku konfiguracyjnego aplikacji wymaga jej zrestartowania, a nie jest oczekiwane by restartować aplikację przy każdym uruchomieniu playbooka. Taką możliwość zapewnia handler.

Handler jest blokiem zadań które zostaną wykonane gdy inne zadanie zawierające opcję notify z nazwą danego handlera wykona zmianę.

```
    - name: Change apache port
      ansible.builtin.lineinfile:
        path: /etc/apache2/ports.conf
        search_string: 'Listen 8080'
        line: 'Listen 80'
        state: present
      notify: Restart Apache

  handlers:
    - name: Restart Apache
      ansible.builtin.service:
        name: apache2
```

```
        state: restarted
```

Output:

```
TASK [Change apache port] ******************************************************************
changed: [node1]
changed: [node2]


RUNNING HANDLER [Restart Apache] ***********************************************************
changed: [node1]
changed: [node2]
```

Przy kolejnym uruchomieniu handler nie jest wykonywany.

https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html

7.8. Obsługa błędów

Bloki pozwalają także na stworzenie prostej obsługi błędów. W tym celu po bloku należy stworzyć blok rescue w którym znajdą się zadania w przypadku gdy zadania zdefiniowane w bloku się nie powiodą

```
tasks:
 - name: Handle the error
   block:
     - name: Print a message
       ansible.builtin.debug:
         msg: 'I execute normally'

     - name: Force a failure
       ansible.builtin.command: /bin/false

     - name: Never print this
       ansible.builtin.debug:
         msg: 'I never execute, due to the above task failing, :-('
   rescue:
     - name: Print when errors
       ansible.builtin.debug:
         msg: 'I caught an error, can do stuff here to fix it, :-)'
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html

7.9. Ponowne użycie

Możliwe jest pisanie bardzo dużych playbooków, lecz dzielenie ich na mniejsze elementy przechowywane w osobnych plikach ułatwia ich utrzymywanie a także pozwala na ponowne wykorzystanie w innych podobnych sytuacjach.

Ansible pozwala na dynamiczne dołączanie poprzez słowo include_tasks oraz statyczne import_task. Include zapewnia procesowanie dołączonego pliku w trakcie działania playbooka, więc wynik dołączonych zadań mogą wpływać na pozostałe zadania z głównego playbooka. Zadania zaimportowane przez import_task są przeprocesowywane przed uruchomieniem jakiegokolwiek zadania z playbooka.

```
- include_tasks: install_with_apt.yml
  when: not install_from_source

- include_tasks: compile_and_install_from_source.yml
  when: install_from_source
```

```
- include_tasks: "setup-{{ ansible_os_family }}.yml"
```

https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse.html

	Include_*	Import_*
Type of re-use	Dynamic	Static
When processed	At runtime, when encountered	Pre-processed during playbook parsing
Task or play	All includes are tasks	`import_playbook` cannot be a task
Task options	Apply only to include task itself	Apply to all child tasks in import
Calling from loops	Executed once for each loop item	Cannot be used in a loop
Using `--list-tags`	Tags within includes not listed	All tags appear with `--list-tags`
Using `--list-tasks`	Tasks within includes not listed	All tasks appear with `--list-tasks`
Notifying handlers	Cannot trigger handlers within includes	Can trigger individual imported handlers
Using `--start-at-task`	Cannot start at tasks within includes	Can start at imported tasks
Using inventory variables	Can `include_*: {{ inventory_var }}`	Cannot `import_*: {{ inventory_var }}`
With playbooks	No `include_playbook`	Can import full playbooks
With variables files	Can include variables files	Use `vars_files:` to import variables

7.10. Tagi

Możliwe jest uruchomienie wybranych zadań z playbooka lub wykluczenie konkretnych. Stosuje się do tego tagi.

```
- name: Template a file to /var/www/html/index.html
  ansible.builtin.template:
    src: index.j2
    dest: /var/www/html/index.html
  tags: template

- name: Create index.html
  ansible.builtin.copy:
    dest: "/var/www/html/index{{ item }}.html"
    content: "File: {{ item }}\nServer: {{ ansible_hostname }}\n"
  tags: copy
```

Uruchomienie lub wykluczenie zadania z danym tagiem odbywa się poprzez opcję --tags lub --skip-tags

```
ansible-playbook install_webserver.yml --skip-tags template
ansible-playbook '/home/vagrant/install_webserver.yml' --tags copy
```

Pierwsza komenda uruchomi wszystkie zadania z pominięciem tych oznaczonych tagiem template. Druga uruchomi jedynie oznaczone tagiem copy.

https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html

8. Praca z wrażliwymi danymi

Narzędzie ansible-vault służy do szyfrowania wrażliwych danych. Można nim zaszyfrować pliki ze zmiennymi, pojedyncze hasło lub connection string lub całe playbooki. Do szyfrowania można wykorzystać hasło które będzie statycznie podawane podczas każdego uruchomienia lub plik zawierający hasło. Mimo wykorzystania ansible-vault należy także pamiętać że zabezpiecza jedynie „dane w spoczynku”, nadal będzie możliwe ujawnienie wrażliwych danych podczas działania Ansible, dlatego istotne jest by wyłączyć logowanie tasków korzystających z haseł. Niektóre moduły także mogą doprowadzić do np. zapisania haseł w pliku, z tego powodu należy mieć świadomość zmian dokonywanych w playbooku.

Przykład zaszyfrowania pliku ze zmiennymi

```
ansible-vault create variables.yml
New Vault password:
Confirm New Vault password:
```

Plik w postaci zaszyfrowanej

```
$ANSIBLE_VAULT;1.1;AES256
35643639633736613166323833653664363835643464303039643535663131393332653962386362
37616135666437376535663630626462303232613535333300a626362393463613333383162653336
63356234343236316536623062363830616139393339336232363135333965323239363165376534
34383932313337363 60a6362363135663465636663333062303632376663653965343161383464 33
65333430366132313239633633663663337663434636366393937353866613338613 1
```

Podgląd zaszyfrowanego pliku po podaniu hasła

```
ansible-vault view variables.yml
Vault password:
password: secret123!
```

W przypadku uruchomienia playbooka korzystającego z zaszyfrowanego pliku należy wykorzystać opcję --ask-vault-pass lub --vault-pass-file w zależności czy do zaszyfrowania został wykorzystany plik czy bezpośrednio wprowadzone hasło

Możliwe jest również wprowadzenie zaszyfrowanej wartości zmiennej w podanej postaci

```
---
- hosts: localhost
  vars:
    var1: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          30656532343063313966636364623165366637336263666336323232386134383865616466313235
          62326233303066616335343738653335626362313934303330a343464326237633238623763343639
          65363130626430626531363739363739376264353266346534303066626531646235353636666330
          34323730616263346 40a6139323732326330643231613332646261633564656434343835326166 33
          3961
  tasks:
…
```

https://docs.ansible.com/ansible/latest/user_guide/vault.html

https://www.redhat.com/sysadmin/ansible-playbooks-secrets

9. Ansible Galaxy

Ansible Galaxy jest platformą gdzie można udostępniać role oraz kolekcje. Korzystanie z gotowych zasobów może znacznie przyspieszać pracę.

https://galaxy.ansible.com/home

Pobierz kolekcję nginx_core. Jest to kolekcja zawierające role których zadaniem jest wdrożyć web serwer nginx

```
ansible-galaxy collection install nginxinc.nginx_core
```

https://galaxy.ansible.com/nginxinc/nginx_core

10. Role

Stwórz playbook który wykorzysta rolę nginx z pobranej kolekcji

```
---
        - hosts: all
          become: true
          collections:
            - nginxinc.nginx_core
          tasks:
            - name: Install NGINX
              ansible.builtin.include_role:
                name: nginx
```

Uruchom playbook

```
ansible-playbook nginx_role.yml

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

TASK [Gathering Facts] *********************************************************************
```

```
ok: [localhost]

TASK [Install NGINX] ***********************************************************************

TASK [nginxinc.nginx_core.nginx : Check whether you are using a supported NGINX
distribution] *****************************************
ok: [localhost] => {
    "changed": false,
    "msg": "Your OS, Ubuntu is supported by NGINX Open Source"
}

TASK [nginxinc.nginx_core.nginx : Check that NGINX setup is an allowed value] *************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [nginxinc.nginx_core.nginx : Set up prerequisites] ********************************
included:
/home/vagrant/.ansible/collections/ansible_collections/nginxinc/nginx_core/roles/nginx/task
s/prerequisites/prerequisites.yml for localhost

TASK [nginxinc.nginx_core.nginx : Install dependencies] ********************************
included:
/home/vagrant/.ansible/collections/ansible_collections/nginxinc/nginx_core/roles/nginx/task
s/prerequisites/install-dependencies.yml for localhost

TASK [nginxinc.nginx_core.nginx : (Alpine Linux) Install dependencies] *******************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : (Debian/Ubuntu) Install dependencies] *******************
ok: [localhost]

TASK [nginxinc.nginx_core.nginx : (Amazon Linux/CentOS/Oracle Linux/RHEL) Install
dependencies] *****************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : (SLES) Install dependencies] ****************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : (FreeBSD) Install dependencies using package(s)] ********
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : (FreeBSD) Install dependencies using port(s)] ***********
skipping: [localhost] => (item=security/ca_root_nss)

TASK [nginxinc.nginx_core.nginx : Check if SELinux is enabled] ****************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Configure SELinux] ***************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Set up signing keys] *************************************
included:
/home/vagrant/.ansible/collections/ansible_collections/nginxinc/nginx_core/roles/nginx/task
s/keys/setup-keys.yml for localhost

TASK [nginxinc.nginx_core.nginx : (Alpine Linux) Set up NGINX signing key URL]
*************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : (Alpine Linux) Download NGINX signing key]
*************************************************************
skipping: [localhost]
```

```
TASK [nginxinc.nginx_core.nginx : (Debian/Red Hat/SLES OSs) Set up NGINX signing key URL]
*******************************************
ok: [localhost]

TASK [nginxinc.nginx_core.nginx : (Debian/Ubuntu) Add NGINX signing key]
*************************************************************
changed: [localhost]

TASK [nginxinc.nginx_core.nginx : (Amazon Linux/CentOS/Oracle Linux/RHEL/SLES) Add NGINX
signing key] ***********************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX Open Source]
*****************************************************************************
included:
/home/vagrant/.ansible/collections/ansible_collections/nginxinc/nginx_core/roles/nginx/task
s/opensource/install-oss.yml for localhost

TASK [nginxinc.nginx_core.nginx : Install NGINX from repository]
*************************************************************************
included:
/home/vagrant/.ansible/collections/ansible_collections/nginxinc/nginx_core/roles/nginx/task
s/opensource/install-debian.yml for localhost

TASK [nginxinc.nginx_core.nginx : (Debian/Ubuntu) Configure NGINX repository]
*************************************************
changed: [localhost] => (item=deb [arch=amd64 signed-by=/usr/share/keyrings/nginx-archive-
keyring.gpg] https://nginx.org/packages/mainline/ubuntu/ focal nginx)
changed: [localhost] => (item=deb-src [signed-by=/usr/share/keyrings/nginx-archive-
keyring.gpg] https://nginx.org/packages/mainline/ubuntu/ focal nginx)

TASK [nginxinc.nginx_core.nginx : (Debian/Ubuntu) Pin NGINX repository]
*************************************************************
changed: [localhost]

TASK [nginxinc.nginx_core.nginx : (Debian/Ubuntu) Install NGINX]
*************************************************************************
changed: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX from source]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX from package]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX in Unix systems]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Set up NGINX Plus license]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX Plus]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX dynamic modules]
*************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Remove NGINX Plus license]
*************************************************************************
skipping: [localhost]
```

```
TASK [nginxinc.nginx_core.nginx : Modify systemd parameters]
*********************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Trigger handlers if necessary]
*********************************************************************

RUNNING HANDLER [nginxinc.nginx_core.nginx : (Handler) Start/reload NGINX]
************************************************************
changed: [localhost]

RUNNING HANDLER [nginxinc.nginx_core.nginx : (Handler) Check NGINX]
*************************************************************
ok: [localhost]

RUNNING HANDLER [nginxinc.nginx_core.nginx : (Handler) Print NGINX error if syntax check
fails] ***************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Debug NGINX output]
*********************************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Configure logrotate for NGINX]
*********************************************************************
skipping: [localhost]

TASK [nginxinc.nginx_core.nginx : Install NGINX Amplify]
*********************************************************************************
skipping: [localhost]

PLAY RECAP ********************************************************************************
localhost                  : ok=16   changed=5    unreachable=0    failed=0    skipped=22
rescued=0    ignored=0
```

Sprawdź komendą curl localhost czy web serwer został zainstalowany.

https://github.com/nginxinc/ansible-collection-nginx#usage

11. Zarządzanie chmurą *

Ansible pozwala na konfigurację środowisk chmurowych. Dostępne są kolekcje pozwalające na zarządzaniem między innymi chmurą Azure oraz AWS.

W przypadku Azure aby skonfigurować poświadczenia można wykorzystać zmienne środowiskowe, zapisać wymagane zmienne w pliku lub przekazać jako parametry. Do połączenia można wykorzystać service principal, czyli rodzaj tożsamości wykorzystywany przez aplikacje, automatyzacje lub własne konto. Wykorzystanie własnego konta użytkownika jest szybszym sposobem. Wystarczy do tego zainstalować az cli oraz uruchomić komendę odpowiadającą za logowanie

```
sudo apt install azure-cli
az login
```

Kolejnym krokiem jest pobranie kolekcji Azure oraz zainstalowanie zależności

```
ansible-galaxy collection install azure.azcollection
pip install -r ~/.ansible/collections/ansible_collections/azure/azcollection/requirements-
azure.txt
```

Po wykonaniu komendy az login powinno już być możliwe uruchamianie playbooków przygotowujących konfigurację w chmurze.

```
---
- name: Create Azure VM
  hosts: localhost
  connection: local
  vars_files:
    - vars_azure_vm.yml
  tasks:
    - name: Create resource group
      azure_rm_resourcegroup:
        name: "{{ name_prefix }}-rg"
        location: "{{ location }}"

    - name: Create virtual network
      azure_rm_virtualnetwork:
        resource_group: "{{ name_prefix }}-rg"
        name: "{{ name_prefix }}-vnet"
        address_prefixes: "{{ ip_range }}"

    - name: Create Network Security Group that allows SSH
      azure_rm_securitygroup:
        resource_group: "{{ name_prefix }}-rg"
        name: "{{ name_prefix }}-nsg"
        rules:
          - name: SSH
            protocol: Tcp
            destination_port_range: 22
            access: Allow
            priority: 1001
            direction: Inbound

    - name: Add subnet
      azure_rm_subnet:
        resource_group: "{{ name_prefix }}-rg"
        name: "{{ name_prefix }}-subnet"
        address_prefix: "{{ ip_range }}"
        virtual_network: "{{ name_prefix }}-vnet"
        security_group_name: "{{ name_prefix }}-nsg"

    - name: Create public IP address
      azure_rm_publicipaddress:
        resource_group: "{{ name_prefix }}-rg"
        allocation_method: Static
        name: "{{ item }}-pip"
      loop: "{{ vm_list }}"
      register: output_ip_address

    - name: Public IP of VM
      ansible.builtin.debug:
        msg: "The public IP is {{ item.state.ip_address }}."
      loop: "{{ output_ip_address.results }}"
      loop_control:
        label: "{{ item.item }}"

    - name: Create virtual network interface card
      azure_rm_networkinterface:
        resource_group: "{{ name_prefix }}-rg"
        name: "{{ item.item }}-nic"
        virtual_network: "{{ name_prefix }}-vnet"
        subnet: "{{ name_prefix }}-subnet"
        create_with_security_group: false
        ip_configurations:
          - name: ipconfig
            public_ip_address_name: "{{ item.state.name }}"
      loop: "{{ output_ip_address.results }}"
      loop_control:
        label: "{{ item.item }}"
```

```
    - name: Create VM
      azure_rm_virtualmachine:
        resource_group: "{{ name_prefix }}-rg"
        name: "{{ item }}"
        vm_size: Standard_B1s
        admin_username: ansible
        ssh_password_enabled: false
        ssh_public_keys:
          - path: /home/ansible/.ssh/authorized_keys
            key_data: "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCeNcdBZWHvgVtoFhdZRRnQrOrCB9lgr0Pq6XBNyAUfcET0JTOV7aOLrb2nRfg
fKnTp4WhT1qVO1HxblEDRLupuGA1PJRSzZbLX+RdmZOkse7vxMbNKVAkHTSuIrfAAoxsmfImqokfN2/1R8a9AVn3Kr9
JXIwf0UnJu/pCOSV4Y6bnTXqvNVl76dNMlhx9I+PoGAVgLkiz4x0Jf+AmPq+pPudkfpoW1iGpmCxYuWRPqtXc5vESVD
gRyjHctLkW/jVuXsLk4p/P5S4pXR/5vohpwXOYutsTRK5sBv8NRJ71Mv6QasEw3k1/fXCGeLTx33FDyzLbHMEniOi9v
OKBx68o0fIhpliXM6FYnE1/wTCOpj/7wYcPu2hz6ptAxD2zob3XYM7/I/rop9DecPnkgA3YUKD8ga7slpkr/XutG/zc
wBnBHr7FzPxTzl2gr+I/NPLHZU6/c3ub5IWoVwQAjiPh/hnKQUk0o5t5a3Q3RRAtCs3RxB4ixqnAUJfXfCNX6O/s=
vagrant@ansible"
        network_interfaces: "{{ item }}-nic"
        image:
          offer: 0001-com-ubuntu-server-focal
          publisher: Canonical
          sku: 20_04-lts-gen2
          version: latest
        managed_disk_type: Standard_LRS
      loop: "{{ vm_list }}"
```

vars_azure_vm.yml:

```
name_prefix: training
location: westeurope
ip_range: 10.0.0.0/24
vm_list:
  - vm1
  - vm2
```

Wynikiem uruchomienia powinny być dwie działające maszyny wirtualne ze skonfigurowanym dostępem po SSH.

https://galaxy.ansible.com/azure/azcollection

https://docs.microsoft.com/en-us/azure/developer/ansible/install-on-linux-vm?tabs=azure-cli

https://docs.ansible.com/ansible/latest/collections/azure/azcollection/index.html#plugins-in-azure-azcollection

12. Dynamic inventory *

Dynamic inventory służy do dynamicznego tworzenia listy hostów. Jest możliwe tworzenie własnych dynamic inventory, lecz zanim podejmie się takiego działania warto sprawdzić w dokumentacji Ansible czy już ktoś ich kiedyś nie stworzył.

https://docs.ansible.com/ansible/latest/collections/index_inventory.html

Często dynamic inventory zwane także inventory plugins dystrybuowane są w ramach kolekcji dotyczącej danej technologii/systemu.

W przypadku wygenerowania listy utworzonych maszyn w Azure, wymagane jest stworzenie pliku o nazwie kończącej się na azure_rm.(yml|yaml) z ustawieniami pluginu. Przykład poniżej

```
plugin: azure.azcollection.azure_rm
```

```
auth_source: auto
plain_host_names: yes
```

A następnie podczas uruchomienia należy wskazać plik z konfiguracją jako inventory

```
ansible-inventory -i azure_rm.yml --graph
@all:
  |--@ungrouped:
  |  |--vm1
  |  |--vm2
```

Plugin azure_rm zapewnia możliwości filtrowania, grupowania, uwzględniania, wykluczania, dodawania warunków, modyfikacji dużej grupy zmiennych dostarczanych domyślnie itp. dlatego warto zapoznać się z dokumentacją danego pluginu.

https://docs.ansible.com/ansible/latest/collections/azure/azcollection/azure_rm_inventory.html