**LAB 5**

Deployment aplikacji - kompletny scenariusz

Zawartość dokumentu:

[**Wstęp**](#_tb6qnnkzooba) **1**

[**Plan ćwiczenia**](#_6m59hr95si7n) **2**

[Tworzenie playbooka i instalacja paczek](#_10msqfiid1c8) 3

[Deployment aplikacji](#_vev3kqsodlao) 4

[TAGI](#_f7baywiu2yvk) 5

[Środowisko aplikacji - pip, virtualenv](#_3lcpvnj6v0ed) 6

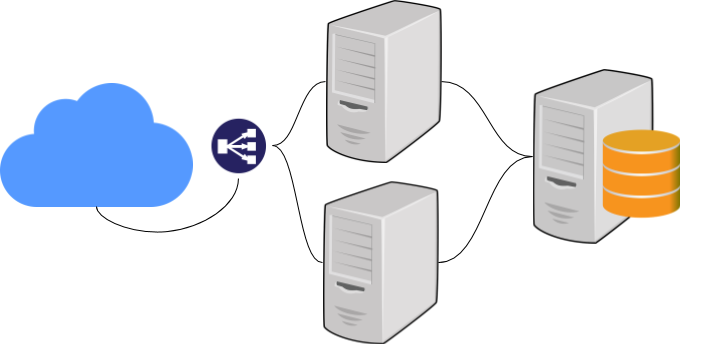
[Konfiguracja apache](#_f3aa1skmck72) 6

[Konfiguracja aplikacji](#_rgf8i8l7sml1) 8

[Restart apache](#_kw81wud5pbt3) 8

## Wstęp

W ciągu kolejnego cyklu ćwiczeń podejmiemy wyzwanie samodzielnej budowy platformy do hostingu aplikacji, z loadbalancerem oraz opcjonalnie bazą danych.



Ćwiczenie podzielone będzie na kilka etapów.

Wyobraźmy sobie scenariusz, gdzie w albo klient dostarcza aplikację, a naszym zadaniem jest uruchomić ją na wybranym środowisku, opcjonalnie w wielu instancjach ukrytych za loadbalancerem, oraz z możliwością dołożenia bazy danych.  
Aplikacja będzie dostarczana regularne, konieczne będą częste aktualizacje.

Aplikacja jest dostarczana poprzez repozytorium git, i posiada przykładową konfigurację.

Pierwszym etapem będzie uruchomienie aplikacji ręcznie na wybranym serwerze, tak aby zapoznać się z podstawowym działaniem.

Aplikacja docelowo ma być uruchomiona jako proces serwera web, wybrany został apache2 wraz z modułem wsgi pozwalającym uruchomienie aplikacji napisanej w jezyku Python w ramach procesu serwera www.

Należy zatem:

1. Zainstalować serwer apache2 oraz potrzebne moduły
2. Wgrać i sparametryzować konfigurację VirtualHost apache (przykładowa konfiguracja: naszaaplikacja.apache.conf) oraz wsgi (naszaaplikacja.wsgi)
3. W wybranym katalogu umieścić kod i konfigurację aplikacji, opcjonalnie dostosować przykładową konfigurację do środowiska docelowego
4. Utworzyć środowisko Virtualenv

Wykonanie tych czynności ręcznie dla osoby znającej proces (robiącej to wielokrotnie) to praca na kilka minut.

Ręczne techniki są dobre, jednak długoterminowo więcej korzyści daje opisanie danego procesu za pomocą skryptu.

Można do tego celu użyć język powłoki np bash, języka Python, czy bardziej wyszukanych rozwiązań, ale jak zapewne się domyślasz, wykorzystamy do tego celu Ansible.

Zobaczysz jak niewiele kodu wystarczy do wykonania tego zadania.

Podstawowy kod aplikacji:

|+.git/

|~aplikacja/

| |+static/

| |+templates/

| |-aplikacja.py

| |-config.cfg.example

| |-environment.sh

| |-naszaaplikacja.apache.conf

| |-naszaaplikacja.wsgi

| |-requirements.txt

|-.gitignore

`-README.md

## Plan ćwiczenia

1. Na serwerze mgmt3.muszynski.pro przejdź do katalogu ~/akademiaansible
   * cd ~/akademiaansible/playbooks/naszaaplikacja/  
     Znajdziesz tu przygotowany plik wgraj\_aplikacje.yml oraz przykładową konfigurację aplikacji i apache
2. Otwórz plik:
   * vi wgraj\_aplikacje.yml

Będziemy kolejno dokładać potrzebne elementy skryptu, tak aby można było wgrać naszą aplikację za pomocą jednego uruchomienia playbooka.

Znajdziesz tu proponowane etapy

* + instalacja apache2 libapache2-mod-wsgi-py3 python-dev
  + należy pobrać aplikację na serwer docelowy z repozytorium <https://github.com/sirkubax/naszaaplikacja.git>
  + następnie należy zainicjalizować środowisko virtualenv
  + sparametryzowanie i wgranie dostarczonego przez dewelopera pliku naszaaplikacja.apache.conf - przykładowy plik znajduje się w repozytorim aplikacji, jego kopia została umieszczona w repozytorium szkolenia w katalogu *naszaaplikacja* celem łatwiejszej parametryzacji
  + po wgraniu konfiguracji virtualhost, należy go aktywować
  + wgrać sparametryzowaną konfigurację naszaaplikacja.wsgi
  + utworzyć katalog na logi aplikacji
  + zrestartować apache

1. Kroki typu instalacja środowiska virtualenv mogą wydawać się zbędne - da się wykonać to na wiele sposobów - przyjmijmy, że takie są zalecenia dostawcy aplikacji, i naszym zadaniem jest opisanie wspomnianych wymagań za pomocą playbooka.  
   Informacje o ręcznym uruchomieniu aplikacji znajdziemy w pliku README.md

### Tworzenie playbooka i instalacja paczek

1. Pierwszym etapem naszego playbooka będzie instalacja aplikacji *apache2*.  
   System operacyjny to Ubuntu, najczęściej aplikacje instalujemy z ‘paczek’ przy pomocy programu apt.   
   Celem tego ćwiczenia jest nauka budowania rozwiązań.  
   Jeśli nie wiesz jakiego modułu użyć, proszę wykonaj kolejne kroki:
   * wyszukaj w Google frazę: “Ansible ubuntu install package”  
     pierwszym wynikiem powinno być  
     <https://docs.ansible.com/ansible/latest/modules/apt_module.html>
   * zapoznaj się z sekcją Examples - pierwszy wpis odpowiada na problem instalacji apache2 na serwerze

*Oczywiście pamiętaj - jeśli nie wiesz jakiego modułu użyć, nie masz aktualnie pomysłu, ale potrafisz wykonać ten kłopotliwy do opisania skryptem krok np zainstalować aplikację - ‘ręcznie’ przy użyciu konsoli - rozważ użycie modułu shell aby szybko wykonać całe zadanie, a dopiero gdy Twój playbook będzie kompletny, wróć do danego miejsca i poszukaj lepszego rozwiązania.  
 np: shell: apt-get install apache2*

1. Zanim wykonamy pierwszy playbook, zaktualizujemy nasze inventory.  
   Podany IP serwera w grupie *naszaaplikacja* jest przykładowy :) użyj właściwego   
   Znajdziesz go w pliku ~/hosts\_ansible - czyli w twoim katalogu domowym na serwerze - można tam przejść komendą cd  
   poszukaj IP swoich hostow  
   na podstawie tego pliku zmień hosty w grupie naszaaplikacja w inventory ~/akademiaansible/inventory/inventory\_akademiaansible)
2. Możesz wykonać pierwszy etap i uruchomić playbook
   * cd ~/akademiaansible/playbooks/naszaaplikacja/  
     ansible-playbook wgraj\_aplikacje.yml
3. Następnym krokiem będzie zainstalowanie pozostałych modułów *libapache2-mod-wsgi-py3   
   python-dev  
   apache2*
   * możemy powtórzyć zadanie z poprzedniego kroku o kolejne 2 wystąpienia, jednak lepszym pomysłem będzie przekazanie do modułu apt listy aplikacji do zainstalowania
   * moduł apt potrafi przyjąć jako argument listę paczek do zainstalowania, dodajmy *libapache2-mod-wsgi* do 2-elementowej listy
4. (Opcjonalnie) Chciałbym żebyśmy poznali pętlę: loop
   * W tym konkretnym przypadku dokumentacja mówi :  
     *When used with a loop: each package will be processed individually, it is much more efficient to pass the list directly to the name option.*  
     Użycie loop jest (w tym przypadku) mniej wydajnym rozwiązaniem, ponieważ moduł potrafi na wejściu przyjąć listę parametrów.  
     Jest jednak wiele sytuacji, gdzie będzie to najlepsze wyjście.  
       
     Przeróbmy (albo skopiujmy jako kolejna sekcja) zatem poprzednio utworzone zadanie, tak aby w jednym kroku zainstalować 3 paczki :  
      *libapache2-mod-wsgi-py3  
      python-dev  
      apache2*

przy pomocy modułu apt, oraz wykorzystując pętlę loop dokończmy proces instalacji wymaganych aplikacji.

* + Dokumentacja:  
    <https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html>

### Deployment aplikacji

1. Deweloperzy dostarczają aplikację poprzez publiczne repozytorium git <https://github.com/sirkubax/naszaaplikacja>   
   *To czy repozytorium jest publiczne czy prywatne nie ma dużego znaczenia dla naszego przykładu.*Musimy zatem pobrać aktualny kod aplikacji, i umieścić na serwerze.
   * Poszukajmy w wyszukiwarce frazy “Ansible clone git”  
     Pierwszy wynik powinien skierować nas do oficjalnej dokumentacji modułu *git*.   
     Nas będzie interesować możliwość sklonowania repozytorium przy pomocy protokołu https, do wskazanego katalogu  
     Link do klonowania: <https://github.com/sirkubax/naszaaplikacja.git>
   * Dobrą praktyką jest zdefiniowanie zmiennych typu katalog\_aplikacji czy log\_dir i szablonowanie zadań oraz konfiguracji.   
     Przykładowo jako miejsce do rozpakowania naszego repozytorium, możemy podać  
      dest: "{{ katalog\_glowny }}"  
     zamiast przekazywać statyczną wybraną ścieżkę

### TAGI - dodatek

1. Zapewne zauważyłeś, że część zadań - chociaż nie wprowadzają żadnych zmian - wykonuje się wielokrotnie.  
   Nie jest to dużym problemem, ale szczególnie w procesie tworzenia playbooka, potrafi to opóźnić proces.  
   W Ansible istnieje opcja selektywnego uruchamiania zadań (lub grup zadań - roli lub includowanych sekcji).  
   W tym celu wybrane zadania (lub grupy zadań) oznaczamy “Tagiem” (wyrazem)  
   Zadanie może mieć 0 lub więcej tagów - jest to lista.
   * Wyszukaj frazę “ansible tags”   
     pierwszy wynik powinien przekierować Cię do dokumentacji  
     <https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html>
   * Uruchomienie playbooka z konkretnym tagiem (tagami) wybieramy przez flagę --tags  
     przykładowo:

| ansible-playbook playbooks/naszaaplikacja/install\_flask.yml -D -vv --tags klonuj |
| --- |

* + zapoznaj się z dokumentacją, zwróć uwagę na tagi *always i never*   
    oraz flagi --tags, --skip-tags i --list-tags

1. “Otagujmy” zatem zadanie 'klonuj repozytorium aplikacji' tagiem *klonuj* i uruchommy playbook jak na powyższym przykładzie
   * wypróbuj tagowanie innych zadań np do zadania instalacji apache2 dodaj tag *systemowe* oraz *always* (lista 2 elementów)  
     Uruchom playbook jak w poprzednim punkcie, zobacz które zadania się wykonały
   * przetestuj --skip-tags
   * oraz playbook zawierający tagi, ale uruchamiany bez ich wskazywania w linii komend

### Środowisko aplikacji - pip, virtualenv

1. Gdy mamy już pobrany kod, oraz z pliku README.md wiemy, że aplikacja korzysta z biblioteki Flask napisanej w języku Python, należy ją zainstalować.  
   Otrzymaliśmy wytyczne, że aplikacja ma być uruchamiana z kontekstu virtualenv, należy więc go stworzyć bazując na informacjach od dewelopera.  
   Powinniśmy utworzyć virtualenv w katalogu *venv*oraz zainstalować zależności dostarczone w pliku *requirements.txt*
   * możemy poszukać frazy ‘ansible pip’ lub ‘ansible python virtualenv’ czy też ‘ansible pip requirements’  
     <https://docs.ansible.com/ansible/latest/modules/pip_module.html>
   * Potrzebujemy stworzyć virtualenv i użyć requirements  
     Czytając dokumentację natrafimy na te parametry.  
     Przykładowe szablony które można by tam ustawić:  
     {{ katalog\_aplikacji }}/venv  
     {{ katalog\_aplikacji }}/requirements.txt
   * Ciekawy przykład znajdziemy w sekcji Examples tej dokumentacji
   * ponownie możemy użyć tagów, aby sprawdzić dany kawałek playbooka

### Konfiguracja apache

1. Kolejnym krokiem będzie skonfigurowanie apache do serwowania naszej aplikacji jako moduł WSGI.  
   Deweloper dostarczył przykładową konfigurację VirtualHost apache *naszaaplikacja.apache.conf* oraz modułu WSGI: *naszaaplikacja.wsgi*Są to pliki statyczne - będziemy je chcieli przerobić na szablon, i wgrać z parametryzacją przy pomocy modułu *template*
   * Dokumentacja:  
     <https://docs.ansible.com/ansible/latest/modules/template_module.html>
   * Najprostszym sposobem będzie skopiowanie plików z repozytorium <https://github.com/sirkubax/naszaaplikacja> do katalogu akademiaansible/playbooks/naszaaplikacja/

Na początek będziemy chcieli umieścić plik naszaaplikacja.apache.conf w katalogu stron hostowanych przez apache2: /etc/apache2/sites-available/

* + Użyjmy modułu **template** aby wgrać plik naszaaplikacja.apache.conf  
    Po dopisaniu kodu obsługującego wspomniane akcje uruchommy:
  + ansible-playbook playbooks/naszaaplikacja/wgraj\_aplikacje.yml -D -vv
  + Możemy zalogować się na nasz serwer i sprawdzić stan środowiska  
    przykładowo:  
    ssh ubuntu@3.10.56.182 -i ~/.ssh/id\_rsa\_szkolenie

Jak zapewne zauważyłeś - wgraliśmy statyczny plik. Chcemy sparametryzować część konfiguracji aby ją łatwiej wgrywać i utrzymywać

* + otwórzmy teraz plik   
    vi akademiaansible/playbooks/naszaaplikacja/naszaaplikacja.apache.conf
  + Wymienimy wybrane statyczne sekcje konfiguracji na parametry uzupełniane w czasie tworzenia szablonu
    1. wymień wszystkie wystąpienia:  
        /var/www/szkolenie.pl/logs i /var/www/szkolenie/logs   
        na  
        {{ log\_dir }}
    2. /var/www/szkolenie.pl/aplikacja i /var/www/szkolenie/aplikacja  
       na  
        {{ katalog\_aplikacji }}
    3. port   
        5081  
       na {{ port }}
  + Zauważ że zmienne zostały wcześniej skonfigurowane w playbook w sekcji vars - tematem lokalizacji zmiennych zajmiemy się w kolejnych ćwiczeniach
  + Ponownie uruchamiamy playbook  
    ansible-playbook playbooks/naszaaplikacja/wgraj\_aplikacje.yml -D  
    Konfiguracja naszej aplikacji będzie wgrana zgodnie z naszymi parametrami
  + Podobnie musimy sparametryzować i **wgrać plik naszaaplikacja.wsgi**
  + **do {{ katalog\_aplikacji }}**

Dodajmy kolejną sekcję do naszego playbooka, i wykonajmy go ponownie.  
*Zamiast dodawać kolejną sekcję, można by skorzystać z jednego zadania template z pętlą loop i listą parametrów*

1. Po wgraniu naszego VirtualHost *naszaaplikacja.apache.conf* należy aktywować tą konfigurację poprzez wykonanie komendy *a2ensite*.  
   W tym przypadku nie istnieje dedykowany moduł Ansible - użyjemy zatem modułu *shell*  lub *command*.
   * jako parametr zadania (nie modułu) shell możemy przekazać   
     args:  
      creates: /etc/apache2/sites-enabled/{{ item }}  
     co pozwoli pominąć ten krok w przyszłości (sprawdzane jest czy istnieje dany plik, jeśli tak, zadanie jest pomijane).
2. Listen PORT  
   Jeśli chcemy żeby apache nasłuchiwał na porcie innym niż 80, musimy dodatkowo w pliku /etc/apache2/ports.conf zdefiniować właściwy parametr.
   * Wykorzystajmy do tego moduł lineinfile  
     prawdopodobnie jako argument modułu będziemy używać taką linię  
     line: "Listen {{ port }}"
   * możliwe byłoby też pobranie pliku na naszą maszynę, i użycie modułu template
3. Katalog na logi  
   Utwórz również modułem **file** katalog na logi, możesz wykorzystać do tego zmienną   
   {{ log\_dir }}

### Konfiguracja aplikacji

1. Kolejnym krokiem będzie przygotowanie i wgranie pliku z konfiguracją samej aplikacji.  
   Deweloper dostarczył plik *config.cfg.example*a nasza aplikacja oczekuje pliku *config.cfg*   
   wgranego w tym samym folderze co kod programu.  
   Należy zatem skopiować ten przykład, przerobić na template, i wgrać na serwer docelowy do właściwego katalogu.  
   Będziemy chcieli przekazać jako parametr zmienną *nazwa*, która jest potrzebna do działania aplikacji - niech będzie to nazwa Twojego użytkownika.

### Restart apache

1. Ostatnim krokiem będzie restart apache tak aby zaczął serwować serwis z nową konfiguracją.  
   Użyjemy do tego moduł *service*.
   * zapoznaj się z dokumentacja tego modułu
   * napisz zadanie restartu aplikacji *apache2*

**Gotowe - weryfikacja**

1. Poprawnie uruchomiony serwis będzie dostępny pod publicznym adresem IP naszego serwera, na wskazanym porcie.  
   Przykładowe adresy:
   * [http://3.10.56.182:5081/](http://3.10.56.182:81/)
   * http://3.10.56.182:5081/index

1. Jako ściągawka czy ‘odpowiedź’ w repozytorium jest plik
   * vi playbooks/naszaaplikacja/wgraj\_aplikacje\_odpowiedz.yml

Postaraj się jednak rozwiązać zadanie samodzielnie - celem ćwiczenia jest nauka procesu tworzenia.