ANSIBLE - zeszyt ćwiczeń

Materiały dydaktyczne

ELA-099

Wersja: 1.0

Altkom Akademia S.A.
00-867 Warszawa, ul. Chłodna 51
Tel. (+48 22) 460 99 99
Fax. (+48 22) 460 99 90

Spis treści:

Wstęp

Informacje wstępne

Niniejszy zeszyt ćwiczeń jest zaplanowany jako materiał wspomagający do szkoleń autorskich i konsultacji dotyczących automatyzacji z wykorzystaniem technologii Ansible. Skrypt nie zawiera pełnego kompendium wiedzy, jest natomiast zestawem ćwiczeń z odpowiedziami, przykładowymi kodami źródłowymi i różnymi sugestiami.

Rozdziały zawierają różnego typu treści:

- Ćwiczenia typu STEP-BY-STEP - gdzie proces edukacyjny realizowany jest poprzez realizację ćwiczenia krok po kroku z pomocą niniejszego skryptu i analizowanie efektów które uzyskujemy wykonując kolejne kroki. Podczas robienia tych ćwiczeń mamy uczyć się i poznawać nowe elementy.
- LABy - to jest zazwyczaj duże ćwiczenie na koniec rozdziału, które zawiera w sobie elementy z ćwiczeń step-by-step. Skrypt nie zawiera w sobie listingu rozwiązania w formie każdego kroku, ponieważ to ćwiczenie zaleca się by było realizowane samodzielnie. Pełne rozwiązania w formie np. całościowego kodu playbooka są dostępne i zaleca się przeglądanie ich jako porównania i weryfikacji własnego rozwiązania lub jako podpowiedź. LABy traktujemy jako ugruntowanie nowo zdobytej wiedzy.
- QUIZY - w niektórych rozdziałach pojawią się pytania teoretyczne

Środowisko pracy

Każdy z uczestników powinien dysponować 3 hostami. Jeden z hostów będzie pełnił funkcję maszyny zarządczej, na której będą uruchamiane polecania typu ad-hoc lub pełne playbooki, a zadania definiowane przez ansibla będą wykonywane na pozostałych dwóch maszynach.

W skrypcie zakładamy, że maszyny nazywają się: warszawa, krakow, gdynia i są w domenie domainX.local. Poniższa tabelka pokazuje konfigurację wykorzystywanych hostów. W Twoim środowisku pracy adresy lub nazwy
mogą się różnić!

Lp.	Host	FQDN	IP	Rola
1	warszawa	warszawa.domainX.local	192.168.122.200	Ansible console
2	gdynia	gdynia.domainX.local	192.168.122.201	Managed host
3	krakow	krakow.domainX.local	192.168.122.202	Managed host

Pamiętaj, że jeśli będziesz chciał testować pewne rzeczy kopiując je bezpośrednio z tego podręcznika mogą potrzebne być zmiany adresów lub nazw domenowych, tak aby pasowały do środowiska na którym wykonujesz ćwiczenia.

Wersje oprogramowania na którym wykonywane są te ćwiczenia to:
Dystrybucja Linux: CentOS 7.7 - 1908.
Ansible: 2.8.4

Wersja Twojego systemu i oprogramowania Ansible mogą się lekko różnić, ale dopóki będą to wersje zbliżone, wszystko powinno działać prawidłowo ze względu na kompatybilności między wersjami.

Oprogramowanie Ansible powinno być zainstalowane na w/w hostach, jeśli nie jest należy je zainstalować.

Na każdym z hostów powinniśmy posiadać dostęp do użytkownika, który ma możliwość wykonywania komend z uprawnieniami administratora systemu z wykorzystaniem sudo bez podawania hasła.
W prezentowanym środowisku będzie to użytkownik: **user**

```
[user@warszawa ~]$ ip a show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP group default qlen 1000
        link/ether 52:54:00:a1:4e:3c brd ff:ff:ff:ff:ff:ff
        inet 192.168.122.200/24 brd 192.168.122.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::8491:75f4:f498:f37/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@warszawa ~]$ ping -c1 gdynia
PING gdynia (192.168.122.201) 56(84) bytes of data.
64 bytes from gdynia (192.168.122.201): icmp_seq=1 ttl=64 time=0.496 ms
```

```
--- gdynia ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.496/0.496/0.496/0.000 ms
[user@warszawa ~]$ ping -c1 krakow
PING krakow (192.168.122.202) 56(84) bytes of data.
64 bytes from krakow (192.168.122.202): icmp_seq=1 ttl=64 time=0.424 ms

--- krakow ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.424/0.424/0.424/0.000 ms
[user@warszawa ~]$ ansible --version
ansible 2.8.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/user/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red
Hat 4.8.5-39)]
[user@warszawa ~]$ sudo -i
[root@warszawa ~]#
```

Na maszynach **gdynia** i **krakow** istnieje użytkownik **devops** który będzie wykorzystywany jako konto dostępowe do systemu dla Ansible. Konto to z wykorzystaniem sudo bez podawania hasła ma możliwość wykonywania komend z przywilejami administratora.

Jeśli nie jest wspomniane inaczej wszystkie kroki ćwiczeń wykonujemy na maszynie **warszawa**, na koncie **user**.
W listingach komend zawsze możesz zobaczyć na jakim hoście i z jakiego konta wydawana jest komenda. Na przykład jeśli coś należy zrobić z konta root na maszynie warszawa będzie to zaznaczone tak:

```
[root@warszawa ~]# przykladowa_komenda
```

Rozdział 1 - Ansible - deploying

Ćwiczenie 1 - Instalacja i konfiguracja

Kroki wstępne:	Maszyny wirtualne i sieć działają prawidłowo
Cel:	Weryfikacja środowiska
Opis:	Instalacja i konfiguracja wstępna została wykonana. W ćwiczeniu tym weryfikujemy wszystkie składowe konfiguracji: - Użytkownik **user** w systemie **warszawa** posiada wygenerowane klucze SSH - Użytkownik **user** z systemu **warszawa** może dostać się z wykorzystaniem kluczy SSH na użytkownika devops - Użytkownik **devops** na **gdynia** i **krakow** ma skonfigurowane sudo za pomocą którego może wykonywać komendy z przywilejami administratora, bez podawania hasła. - Ansible jest zainstalowany na maszynie **warszawa**

1. Sprawdź czy użytkownik **user** posiada wygenerowane klucze ssh: **~/.ssh/id_rsa** i **~/.ssh/id_rsa.pub**

```
[user@warszawa ~]$ ls -al ~/.ssh/id_rsa*
-rw-------. 1 user user 1675 Sep 28 20:17 /home/user/.ssh/id_rsa
-rw-r--r--. 1 user user  409 Sep 28 20:17 /home/user/.ssh/id_rsa.pub
```

2. Zweryfikuj czy użytkownik **devops** bez podawania hasła (uwierzytelnienie z wykorzystaniem kluczy SSH) możesz na maszynach **gdynia** i **krakow** wykonać przykładowe polecenie np. hostname.

```
[user@warszawa ~]$ ssh devops@gdynia hostname
gdynia.domainx.local
[user@warszawa ~]$ ssh devops@krakow hostname
krakow.domainx.local
```

3. Na serwerze: **gdynia** obejrzyj konfigurację sudo dla konta **devops** i przetestuj czy działa prawidłowo. Następnie wykonaj te same kroki na maszynie **krakow**.

```
[root@gdynia ~]# grep devops /etc/sudoers
devops ALL=(ALL) NOPASSWD: ALL
[root@gdynia ~]# su - devops
Last login: Sat Sep 28 20:38:18 CEST 2019 on pts/0
[devops@gdynia ~]$ whoami
devops
[devops@gdynia ~]$ sudo whoami
root
```

4. Sprawdź czy na maszynie **warszawa** jest zainstalowany ansible

```
[root@warszawa ~]# rpm -qa ansible
ansible-2.8.4-1.el7.noarch
[root@warszawa ~]# ansible --version
ansible 2.8.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /bin/ansible
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red
Hat 4.8.5-39)]
```

Ćwiczenie 2 - Tworzenie pliku inventory

Kroki wstępne:	Brak
Cel:	Testowanie różnych konfiguracji pliku inventory
Opis:	W ćwiczeniu będziemy tworzyć różnego rodzaju konfiguracje pliku inventory i testować będziemy czy ansible w danej konfiguracji będzie w stanie zarządzać definiowanymi hostami/grupami.

1. Stwórz katalog **~/ansible-inventory** który będzie katalogiem roboczym.

```
[user@warszawa ~]$ mkdir ~/ansible-inventory
[user@warszawa ~]$ cd ~/ansible-inventory/
[user@warszawa ansible-inventory]$
```

2. Sprawdź jakimi hostami jest w stanie zarządzać teraz ansible. W związku z tym, że nie ma aktualnie żadnego pliku inventory z prawidłowo skonfigurowanymi hostami zobaczysz tylko i wyłącznie informację o localhost.

```
[user@warszawa ansible-inventory]$ ansible --list-hosts all
 [WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

  hosts (0):
```

3. Stwórz plik **inventory** który będzie zawierała jedną linię z napisem:
   gdynia.domainx.local
   Następnie korzystając z opcji **-i** wskaż ten plik jako inventory w komendzie ansible.
   Zrób różnego rodzaju zapytania z opcją --list-hosts i zobacz jakimi hostami mógłbyś aktualnie zarządzać. Zwróć uwagę co dzieje się gdy nie jest podany dokładny wpis FQDN tylko sam host oraz co dzieje się gdy zostanie podany adres IP.

```
[user@warszawa ansible-inventory]$ echo gdynia.domainx.local > inventory
```

```
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts all
  hosts (1):
          gdynia.domainx.local
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts \
gdynia.domainx.local
  hosts (1):
          gdynia.domainx.local
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts \
krakow.domainx.local
 [WARNING]: Could not match supplied host pattern, ignoring: krakow.domainx.local

 [WARNING]: No hosts matched, nothing to do

  hosts (0):
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts gdynia
 [WARNING]: Could not match supplied host pattern, ignoring: gdynia

 [WARNING]: No hosts matched, nothing to do

  hosts (0):
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts 192.168.122.201
 [WARNING]: Could not match supplied host pattern, ignoring: 192.168.122.201

 [WARNING]: No hosts matched, nothing to do

  hosts (0):
```

4. Zmień zawartość pliku inventory na podaną niżej:

```
gdynia.domainx.local
gdynia
192.168.122.[201:202]

[webservers]
gdynia
krakow

[prod]
gdynia

[dev]
krakow
```

```
[servers:children]
prod
dev
```

5. Przetestuj jakie hosty są dostępne i co jest dostępne w poszczególnych grupach definiowanych i wbudowanych.

```
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts gdynia
  hosts (1):
        gdynia
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts krakow
  hosts (1):
        krakow
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts
krakow.domainx.local
 [WARNING]: Could not match supplied host pattern, ignoring: krakow.domainx.local

 [WARNING]: No hosts matched, nothing to do

  hosts (0):
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts 192.168.122.200
 [WARNING]: Could not match supplied host pattern, ignoring: 192.168.122.200

 [WARNING]: No hosts matched, nothing to do

  hosts (0):
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts 192.168.122.201
  hosts (1):
        192.168.122.201
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts *.domainx.local
  hosts (1):
        gdynia.domainx.local
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts ungrouped
  hosts (3):
        gdynia.domainx.local
        192.168.122.201
        192.168.122.202
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts all
  hosts (5):
        gdynia.domainx.local
        192.168.122.201
```

```
        192.168.122.202
        gdynia
        krakow
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts localhost
  hosts (1):
        localhost
[user@warszawa ansible-inventory]$ ansible -i ./inventory --list-hosts servers
  hosts (2):
        gdynia
        krakow
```

Ćwiczenie 3 - Plik konfiguracyjny

Kroki wstępne:	Brak
Cel:	Poznanie pliku konfiguracyjnego ansible
Opis:	W ćwiczeniu analizowane są różne możliwe lokalizacje pliku konfiguracyjnego, jak również kilka dyrektyw konfiguracyjnych: **remote_user** i **become**. W ćwiczeniu zostanie użyta komenda typu **ad-hoc**, która szczegółowo zostanie omówiona w następnym rozdziale: **ansible gdynia -m shell -a whoami** Komenda ta na zdalnym hoście **gdynia** próbuje wykonać z wykorzystaniem powłoki polecenie **whoami**.

1. Stwórz katalog **~/ansible-config** który będzie katalogiem roboczym.

```
[user@warszawa ~]$ mkdir ~/ansible-config
[user@warszawa ~]$ cd ~/ansible-config/
[user@warszawa ansible-config]$
```

2. Sprawdź jaki plik konfiguracyjny używany jest aktualnie przez komendę ansible

```
[user@warszawa ansible-config]$ ansible --version  | grep "config file"
  config file = /etc/ansible/ansible.cfg
```

3. Utwórz pusty plik w katalogu domowym ~/.ansible.cfg i wykonaj sprawdzenie ponownie.

```
[user@warszawa ansible-config]$ touch ~/.ansible.cfg
[user@warszawa ansible-config]$ ansible --version  | grep "config file"
  config file = /home/user/.ansible.cfg
```

4. W katalogu w którym jest stwórz plik ansible.cfg i wykonaj sprawdzenie ponownie.

```
[user@warszawa ansible-config]$ touch ansible.cfg
[user@warszawa ansible-config]$ ansible --version  | grep "config file"
  config file = /home/user/ansible-config/ansible.cfg
```

5. Stwórz plik **inventory** z jednym hostem **gdynia** i przetestuj z wykorzystaniem opcji **--list-hosts**

```
[user@warszawa ansible-config]$ echo gdynia > inventory
[user@warszawa ansible-config]$ ansible --list-hosts gdynia
(...)

  hosts (0):
[user@warszawa ansible-config]$ ansible --list-hosts -i ./inventory gdynia
  hosts (1):
        gdynia
```

6. Wyedytuj plik konfiguracyjny ansible w katalogu bieżącym i wskaż domyślną lokalizację pliku **inventory**. Uzupełnij plik **ansible.cfg** tak jak poniżej:

```
[defaults]
inventory = ./inventory
```

7. Zobacz czy **--list-hosts** działa bez podawania opcji **-i**

```
[user@warszawa ansible-config]$ ansible --list-hosts gdynia
  hosts (1):
        gdynia
```

8. Wykonaj komende **ad-hoc** by przetestować połączenia do zdalnego hosta

```
[user@warszawa ansible-config]$ ansible gdynia -m shell -a whoami
gdynia | UNREACHABLE! => {
        "changed": false,
        "msg": "Failed to connect to the host via ssh: Permission denied
(publickey,gssapi-keyex,gssapi-with-mic,password).",
        "unreachable": true
}
```

9. Dodaj do pliku **ansible.cfg** w sekcji **defaults** deklarację zdalnego użytkownika, który będzie wykorzystywany przez ansible.

```
remote_user = devops
```

10. Wykonaj komendę **ad-hoc** by przetestować połączenia do zdalnego hosta. Dodatkowo możesz użyć specjalnej opcji **--become**, która powoduje przejście przez ansible na konto uprzywilejowane.

```
[user@warszawa ansible-config]$ ansible gdynia -m shell -a whoami
gdynia | CHANGED | rc=0 >>
devops

[user@warszawa ansible-config]$ ansible gdynia -b -m shell -a whoami
gdynia | CHANGED | rc=0 >>
root
```

LAB: Ansible deploying

Kroki wstępne:	Brak
Cel:	Stworzenie środowiska pracy pod kątem uruchamiania zadań automatyzacji ansible.
Opis:	Stwórz roboczy katalog zawierający odpowiednie pliki **inventory** i **ansible.cfg**. Katalog ten będzie wykorzystywany jako baza **w następnych rozdziałach**.

1. Stwórz katalog **~/ansible-base** i utwórz w nim lokalna konfigurację zgodną z poniżą specyfikacją:
2. Plik inventory powinien:
    a. Zawierać niezgrupowane wpisy serwera **gdynia i krakow** w formie krótkiej nazwy, pełnego FQDN oraz adresów IP
    b. W tworzonych grupach hosty powinny być definiowane tylko krótkimi nazwami
    c. Powinna istnieć grupa **pomorskie** z hostem **gdynia**
    d. Grupa **malopolskie** powinna zawierac hosta **krakow**
    e. Grupa **polska** powinna zawierać hosty z grup: **pomorskie** i **malopolskie**
    f. Grupa **prod** powinna zawierać hosta **gdynia**
    g. Grupa **dev** powinna zawierać hosta **krakow**
    h. Grupa **servers** powinna zawierać grupy: **prod** i **dev**
    i. Grupa **www** powinna zawierać hosta **krakow**
    j. Grupa **db** powinna zawierać hosty: **gdynia** i **krakow**

Lp.	Host	prod	dev	www	db	servers	polska	pom.	mal.
1.	gdynia	**X**			**X**	**X**	**X**	**X**	
2.	krakow		**X**	**X**	**X**	**X**	**X**		**X**

3. Domyślna konfiguracja ansible powinna:
    a. Wykorzystywać plik **inventory** z bieżącego katalogu

   b. Przechodzić domyślnie na konto uprzywilejowane
   c. Mieć zdefiniowane konto uprzywilejowane jako **root**
   d. Wykorzystywać metodę podnoszenia uprawnień: **sudo**
4. Na koniec sprawdź połączenie do zdalnych hostów i zweryfikuj przynależność hostów do grup

LAB: Ansible deploying - rozwiązanie

5. Stwórz katalog **~/ansible-base** i przejdź do niego.

```
[user@warszawa ~]$ mkdir ~/ansible-base
[user@warszawa ~]$ cd ~/ansible-base/
```

6. Stwórz plik **inventory** z poniższą zawartością:

```
gdynia
krakow
gdynia.domainx.local
krakow.domainx.local
192.168.122.201
192.168.122.202

[pomorskie]
gdynia

[malopolskie]
krakow

[polska:children]
pomorskie
malopolskie

[prod]
gdynia

[dev]
krakow

[servers:children]
prod
dev

[www]
krakow

[db]
gdynia
```

```
krakow
```

7. Stwórz plik **ansible.cfg** z poniższą zawartością:

```
[defaults]
inventory=./inventory
remote_user=devops

[privilege_escalation]
become=True
become_method=sudo
become_user=root
```

8. Przetestuj przynależność hostów do grup:

```
[user@warszawa ansible-base]$ ansible --list-hosts prod
  hosts (1):
        gdynia
[user@warszawa ansible-base]$ ansible --list-hosts dev
  hosts (1):
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts www
  hosts (1):
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts db
  hosts (2):
        gdynia
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts servers
  hosts (2):
        gdynia
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts polska
  hosts (2):
        gdynia
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts pomorskie
  hosts (1):
        gdynia
[user@warszawa ansible-base]$ ansible --list-hosts malopolskie
  hosts (1):
```

```
        krakow
[user@warszawa ansible-base]$ ansible --list-hosts ungrouped
  hosts (4):
        gdynia.domainx.local
        krakow.domainx.local
        192.168.122.201
        192.168.122.202
```

9. Przetestuj połączenia do zdalnych hostów:

```
[user@warszawa ansible-base]$ ansible -m shell -a whoami gdynia
gdynia | CHANGED | rc=0 >>
root

[user@warszawa ansible-base]$ ansible -m shell -a whoami krakow
krakow | CHANGED | rc=0 >>
root
```

10. **WAŻNE** - katalog **ansible-base** będzie używany jako bazowy katalog roboczy w większości kolejnych ćwiczeń. Należy kopiować go w nienaruszonej formie do nowych folderów. Informacje o tym znajdować się będą w **Kroki wstępne** definiujące dane ćwiczenie. W listingach nie będą uwazglądniane za każdym razem komendy do stworzenia katalogu robocze i przejścia do niego. Poniżej ogólna instrukcja:

```
[user@warszawa ~]$ cp -r ansible-base/ ansible-cwiczenieX
[user@warszawa ~]$ cd ansible-cwiczenieX/
[user@warszawa ansible-cwiczenieX]$
```

Rozdział 2 - Komendy ad-hoc

Ćwiczenie 1 - Podstawowe moduły

Kroki wstępne:	Katalog roboczy: ~/ansible-adhoc
Cel:	Poznanie modułów: **shell, command, ping**
Opis:	W ćwiczeniu tym sprawdzisz dostępność hostów z wykorzystaniem modułu **ping** oraz będziesz wykonywać komendy na zdalnych serwerach, poznając różnice pomiędzy **shell** i **command**.

1. Wykonaj moduł ansible **ping** na grupie hostów: **all**. Zwróć uwagę jak ansible podchodzi do zdefiniowanych hostów z inventory, kiedy jeden host wpisany jest pod kilkoma nazwami - sprawdź ile odpowiedzi pong zostanie wygenerowanych.

```
[user@warszawa ansible-adhoc]$ ansible all -m ping
192.168.122.202 | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
krakow.domainx.local | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
gdynia.domainx.local | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
```

```
gdynia | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
192.168.122.201 | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
krakow | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
```

2. Użyj modułu **ping** tak na grupie hostów: **servers**

```
[user@warszawa ansible-adhoc]$ ansible servers -m ping
gdynia | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
krakow | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": false,
        "ping": "pong"
}
```

3. Użyj modułu **command** i **shell** aby wykonać komendę hostname na grupie **servers**, sprawdź co zmienia dodanie opcji **-o** w generowanym wyniku:

```
[user@warszawa ansible-adhoc]$ ansible servers -m shell -a hostname
krakow | CHANGED | rc=0 >>
krakow.domainx.local

gdynia | CHANGED | rc=0 >>
gdynia.domainx.local

[user@warszawa ansible-adhoc]$ ansible servers -m command -a hostname
gdynia | CHANGED | rc=0 >>
gdynia.domainx.local

krakow | CHANGED | rc=0 >>
krakow.domainx.local

[user@warszawa ansible-adhoc]$ ansible servers -o -m command -a hostname
krakow | CHANGED | rc=0 | (stdout) krakow.domainx.local
gdynia | CHANGED | rc=0 | (stdout) gdynia.domainx.local
```

4. Używając modułu **shell** i **command** sprawdź jak zachowują się komendy zawierające przekierowania.

```
[user@warszawa ansible-adhoc]$ ansible gdynia -m shell -a "echo TEST >
/tmp/shell.txt"
gdynia | CHANGED | rc=0 >>

[user@warszawa ansible-adhoc]$ ansible gdynia -m command -a "echo TEST >
/tmp/command.txt"
gdynia | CHANGED | rc=0 >>
TEST > /tmp/command.txt
```

5. Sprawdź czy pliki utworzyły się z wykorzystaniem obu modułów:

```
[user@warszawa ansible-adhoc]$ ansible gdynia -m shell -a "cat /tmp/shell.txt"
gdynia | CHANGED | rc=0 >>
TEST

[user@warszawa ansible-adhoc]$ ansible gdynia -m shell -a "cat /tmp/command.txt"
gdynia | FAILED | rc=1 >>
cat: /tmp/command.txt: No such file or directorynon-zero return code
```

6. Zobacz jak zachowuje się wykonanie kilku komend oddzielonych znakiem “;” w modułach **shell** i **command**

```
[user@warszawa ansible-adhoc]$ ansible gdynia -m shell -a "hostname;date"
gdynia | CHANGED | rc=0 >>
gdynia.domainx.local
Sun Sep 29 13:37:06 CEST 2019

[user@warszawa ansible-adhoc]$ ansible gdynia -m command -a "hostname;date"
gdynia | FAILED | rc=2 >>
[Errno 2] No such file or directory
```

7. Wyświetl zawartość zmiennej UID na zdalnym hoście z wykorzystaniem modułów **shell** i **command**.

```
[user@warszawa ansible-adhoc]$ ansible gdynia -m shell -a 'echo $UID'
gdynia | CHANGED | rc=0 >>
0

[user@warszawa ansible-adhoc]$ ansible gdynia -m command -a 'echo $UID'
gdynia | CHANGED | rc=0 >>
$UID
```

Ćwiczenie 2 - Moduły ansible i ich dokumentacja

Kroki wstępne:	Katalog roboczy: ~/ansible-adhoc2
Cel:	Użycie kilku przykładowych modułów ansible.
Opis:	Korzystając z ansible, powinniśmy unikać wykorzystywania modułów typu **shell** i **command**. Za chwilę przetestujesz kilka dedykowanych modułów, którymi możesz wpływać na konfigurację serwerów. Dodatkowo zobaczysz jakie moduły są dostępne i jak korzystać z dokumentacji do nich. Aktywności wykonywanie będą na hoście: **gdynia** jeśli nie wskazano inaczej w danym punkcie.

1. Sprawdź czy istnieje rekord kowalski w pliku /etc/passwd z wykorzystaniem modułu shell, następnie korzystając z modułu **user** stwórz takiego użytkownika. Użyj komendy ad-hoc do tworzenia użytkownika dwa razy i zobacz jak zachowa się ansible gdy użytkownik już będzie (zwróć uwagę na wpisy **CHANGED** i **SUCCESS**.

```
[user@warszawa ansible-adhoc2]$ ansible gdynia -m shell -a "grep kowalski
/etc/passwd"
gdynia | FAILED | rc=1 >>
non-zero return code

[user@warszawa ansible-adhoc2]$ ansible gdynia -m user -a 'name=kowalski
comment=Janek shell=/sbin/nologin'
gdynia | CHANGED => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
        "comment": "Janek",
        "create_home": true,
        "group": 1002,
        "home": "/home/kowalski",
        "name": "kowalski",
        "shell": "/sbin/nologin",
        "state": "present",
```

```
        "system": false,
        "uid": 1002
}
[user@warszawa ansible-adhoc2]$ ansible gdynia -m user -a 'name=kowalski
comment=Janek shell=/sbin/nologin'
gdynia | SUCCESS => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "append": false,
        "changed": false,
        "comment": "Janek",
        "group": 1002,
        "home": "/home/kowalski",
        "move_home": false,
        "name": "kowalski",
        "shell": "/sbin/nologin",
        "state": "present",
        "uid": 1002
}
```

2. Zobacz jakie inne argumenty dostępne są dla modułu **user**. Poszukaj jakie argumenty powinny być użyte by skasować użytkownika i jego katalog domowy.

```
[user@warszawa ansible-adhoc2]$ ansible-doc user
```

3. Sprawdź czy użytkownik kowalski istnieje w systemie, następnie z wykorzystaniem modułu **user** usuń go i zweryfikuj.

```
[user@warszawa ansible-adhoc2]$ ansible gdynia -m shell -a "grep kowalski
/etc/passwd"
gdynia | CHANGED | rc=0 >>
kowalski:x:1002:1002:Janek:/home/kowalski:/sbin/nologin

[user@warszawa ansible-adhoc2]$ ansible gdynia -m user -a 'name=kowalski
state=absent remove=yes'
gdynia | CHANGED => {
        "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
        },
        "changed": true,
```

```
        "force": false,
        "name": "kowalski",
        "remove": true,
        "state": "absent"
}
[user@warszawa ansible-adhoc2]$ ansible gdynia -m shell -a "grep kowalski
/etc/passwd"
gdynia | FAILED | rc=1 >>
non-zero return code
```

4. Wyświetl jakie są dostępne moduły w ansible

```
[user@warszawa ansible-adhoc2]$ ansible-doc -l
```

5. Stwórz lokalnie plik test.txt z napisem “Ansible copy test” i z wykorzystaniem modułu **copy** wgraj go do katalogu /tmp na wszystkie serwery z grupy **servers**. Zweryfikuj.

```
[user@warszawa ansible-adhoc2]$ echo "Ansible copy test" > test.txt
[user@warszawa ansible-adhoc2]$ ansible servers -m copy -a "src=./test.txt
dest=/tmp"
[user@warszawa ansible-adhoc2]$ ansible servers -m shell -a "cat /tmp/test.txt" -o
krakow | CHANGED | rc=0 | (stdout) Ansible copy test
gdynia | CHANGED | rc=0 | (stdout) Ansible copy test
```

6. Z wykorzystaniem modułu **copy** stwórz plik /tmp/hello.txt o zawartości “Hello World” na serwerze **gdynia**.

```
[user@warszawa ansible-adhoc2]$ ansible gdynia -m copy -a 'content="Hello World"
dest=/tmp/hello.txt'
[user@warszawa ansible-adhoc2]$ ansible gdynia -m shell -a "cat /tmp/hello.txt" -o
gdynia | CHANGED | rc=0 | (stdout) Hello World
```

7. Z wykorzystaniem modułu **yum** zainstaluj na **localhost** pakiet **tree**

```
[user@warszawa ansible-adhoc2]$ ansible localhost -m yum -a name=tree
```

8. Przegraj ze wszystkich serwerów z grupy **servers** plik /etc/passwd do katalogu lokalnego **backups_passwd.** Użyj modułu **fetch**. Następnie zobacz jak wygląda struktura utworzonego katalogu.

```
[user@warszawa ansible-adhoc2]$ ansible servers -m fetch -m fetch -a \
'src=/etc/passwd dest=./backup_passwd'

[user@warszawa ansible-adhoc2]$ tree ./backup_passwd/
./backup_passwd/
├── gdynia
│   └── etc
│       └── passwd
└── krakow
        └── etc
        └── passwd
```

LAB: Komendy ad-hoc

Kroki wstępne:	Katalog roboczy: ~/ansible-adhoc-lab
Cel:	Zaimplementuj serwer HTTPD z wykorzystaniem komend ad-hoc na serwerze **gdynia** i **krakow**
Opis:	Korzystając z ansible zadbaj o to, żeby na serwerach **gdynia** i **krakow** działała usługa sieciowa HTTPD zgodnie ze zdefiniowaną specyfikacja.

1. Specyfikacja usługi:
    a. Usługa httpd powinna być w najnowszej możliwej wersji
    b. Usługa httpd powinna działać i uruchamiać się domyślnie podczas startu systemu
    c. Port 80/tcp powinien być otwarty
    d. Domyślna strona powinna wyświetlać napis “Hello Apache”
2. Zweryfikuj działanie usług na obu serwerach z maszyny **warszawa.**

LAB: Komendy ad-hoc - rozwiązanie

1. Implementacja:

```
[user@warszawa ansible-adhoc-lab]$ ansible servers -m yum -a 'name=httpd
state=latest'
[user@warszawa ansible-adhoc-lab]$ ansible servers -m copy -a 'content="Hello
Apache\n" dest=/var/www/html/index.html'
[user@warszawa ansible-adhoc-lab]$ ansible servers -m firewalld -a 'port=80/tcp
permanent=yes state=enabled immediate=true'
[user@warszawa ansible-adhoc-lab]$ ansible servers -m service -a 'name=httpd
state=started enabled=yes'
```

2. Test:

```
[user@warszawa ansible-adhoc-lab]$ curl http://gdynia
Hello Apache
[user@warszawa ansible-adhoc-lab]$ curl http://krakow
Hello Apache
```

Rozdział 3 - Playbooks

Ćwiczenie 1 - Pierwszy playbook

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-playbook1
Cel:	Napisz playbooka, który implementować będzie prosty serwer WWW.
Opis:	Playbook powinien implementować usługę WWW zgodną ze specyfikacją na serwerach z grupy **servers**. Specyfikacja: a. Usługa httpd powinna być w najnowszej możliwej wersji b. Usługa httpd powinna działać i uruchamiać się domyślnie podczas startu systemu c. Port 80/tcp powinien być otwarty d. Domyślna strona powinna wyświetlać napis “Hello Apache” Specyfikacja jest identyczna jak w LABie z rozdziału 2. Tym razem nie będziesz wykorzystywać komend ad-hoc, lecz napiszesz playbooka, który będzie mógł być w łatwy sposób modyfikowany i uruchamiany na różnych środowiskach w celu identycznej implementacji zdefiniowanych zadań.

1. Stwórz plik **www.yml** w którym będzie definicja play’a z nazwą, listą hostów i początkiem definiowania zadań.

```
---
- name: Implementacja serwera WWW
  hosts: servers
  tasks:
```

2. Dodaj zadanie instalujące pakiet httpd.

```
- name: instalacja pakietu httpd
        yum:
        name: httpd
        state: latest
```

3. Dodaj tworzenie pliku index.html

```
- name: stworzenie pliku index.html
        copy:
        content: "Hello Apache\n"
        dest: /var/www/html/index.html
```

4. Zadbaj o autostart i uruchomienie usługi

```
 - name: uruchomienie i autostart uslugi
        service:
        name: httpd
        state: started
        enabled: true
```

5. Odblokuj port na firewallu

```
- name: firewall dla portu 80/tcp
        firewalld:
        port: 80/tcp
        permanent: yes
        state: enabled
        immediate: true
```

6. Cały playbook powinien wyglądać tak:

```
---
- name: Implementacja serwera WWW
  hosts: servers
  tasks:

  - name: instalacja pakietu httpd
```

```
        yum:
        name: httpd
        state: latest

  - name: stworzenie pliku index.html
        copy:
        content: "Hello Apache\n"
        dest: /var/www/html/index.html

  - name: uruchomienie i autostart uslugi
        service:
        name: httpd
        state: started
        enabled: true

  - name: firewall dla portu 80/tcp
        firewalld:
        port: 80/tcp
        permanent: yes
        state: enabled
        immediate: true
```

7. Zweryfikuj czy plik nie ma posiada błędów składni:

```
[user@warszawa ansible-playbook1]$ ansible-playbook --syntax-check ./www.yml

playbook: ./www.yml
```

8. Uruchom playbook:

```
[user@warszawa ansible-playbook1]$ ansible-playbook ./www.yml
```

9. Zweryfikuj czy na serwerach: **gdynia** i **krakow** działa usługa WWW:

```
[user@warszawa ansible-playbook1]$ curl http://gdynia
Hello Apache
[user@warszawa ansible-playbook1]$ curl http://krakow
Hello Apache
```

Ćwiczenie 2 - Multiple plays

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-playbook-multi
Cel:	Napisz playbooka, który implementować będzie prosty serwer WWW na serwerze **gdynia** i **krakow**, ale oddzielnymi **playami**. Serwer **gdynia**, w przeciwieństwie do serwera **krakow** ma mieć otwarty port 80/tcp. Zweryfikuj samym ansiblem czy usługi są dostępne.
Opis:	Playbook powinien implementować usługę WWW zgodną ze specyfikacją na serwerach z grupy **servers**. Specyfikacja: a. Usługa httpd powinna być w najnowszej możliwej wersji b. Usługa httpd powinna działać i uruchamiać się domyślnie podczas startu systemu c. Port 80/tcp powinien być otwarty (**TYLKO na serwerze gdynia)** d. Domyślna strona powinna wyświetlać napis “Hello Apache” Dodatkowo na maszynie która uruchamia playbooka powinny zostać wykonane testy. Zadanie powinno zostać zrealizowane trzema różnymi play’ami w jednym playbooku.

1. Stwórz playbook’a **www-multi.yml** w którym będzie definicja pierwszego play’a dotyczącego konfiguracji serwera **gdynia**.

```
---
- name: Implementacja serwera WWW na serwerze gdynia
  hosts: gdynia
  tasks:

    - name: instalacja pakietu httpd
      yum:
        name: httpd
```

```
      state: latest

  - name: stworzenie pliku index.html
    copy:
      content: "Hello Apache\n"
      dest: /var/www/html/index.html

  - name: uruchomienie i autostart uslugi
    service:
      name: httpd
      state: started
      enabled: true

  - name: firewall dla portu 80/tcp
    firewalld:
      port: 80/tcp
      permanent: yes
      state: enabled
      immediate: true
```

2. Dodaj drugiego play'a związanego z serwerem **krakow**. Pamiętaj, że tu należy zadbać aby reguła firewalla blokowała dostęp do portu 80/tcp.

```
- name: Implementacja serwera WWW na serwerze krakow
  hosts: krakow
  tasks:
    - name: instalacja pakietu httpd
      yum:
        name: httpd
        state: latest

    - name: stworzenie pliku index.html
      copy:
        content: "Hello Apache\n"
        dest: /var/www/html/index.html

    - name: uruchomienie i autostart uslugi
      service:
        name: httpd
        state: started
        enabled: true
```

```
  - name: firewall dla portu 80/tcp - WYLACZENIE
    firewalld:
      port: 80/tcp
      permanent: yes
      state: disabled
      immediate: true
```

3. Dodaj trzeciego play'a, który będzie uruchamiał się na **localhost** i działać będzie bez podnoszenia uprawnień do roota. Ma on zweryfikować czy serwery **krakow** i **gdynia** zwracają właściwy http status code: 200.

```
- name: Test uslug
  hosts: localhost
  become: no
  tasks:
    - name: test polaczenia z gdynia
      uri:
        url: http://gdynia
        return_content: yes
        status_code: 200
    - name: test polaczenia z krakow
      uri:
        url: http://krakow
        return_content: yes
        status_code: 200
```

4. Cały playbook powinien wyglądać następująco:

```
---
- name: Implementacja serwera WWW na serwerze gdynia
  hosts: gdynia
  tasks:

    - name: instalacja pakietu httpd
      yum:
        name: httpd
        state: latest

    - name: stworzenie pliku index.html
      copy:
```

```
        content: "Hello Apache\n"
        dest: /var/www/html/index.html

    - name: uruchomienie i autostart uslugi
      service:
        name: httpd
        state: started
        enabled: true

    - name: firewall dla portu 80/tcp
      firewalld:
        port: 80/tcp
        permanent: yes
        state: enabled
        immediate: true

- name: Implementacja serwera WWW na serwerze krakow
  hosts: krakow
  tasks:
    - name: instalacja pakietu httpd
      yum:
        name: httpd
        state: latest

    - name: stworzenie pliku index.html
      copy:
        content: "Hello Apache\n"
        dest: /var/www/html/index.html
   - name: uruchomienie i autostart uslugi
      service:
        name: httpd
        state: started
        enabled: true

    - name: firewall dla portu 80/tcp - WYLACZENIE
      firewalld:
        port: 80/tcp
        permanent: yes
        state: disabled
        immediate: true

- name: Test uslug
  hosts: localhost
```

```
  become: no
  tasks:
    - name: test polaczenia z gdynia
      uri:
        url: http://gdynia
        return_content: yes
        status_code: 200
    - name: test polaczenia z krakow
      uri:
        url: http://krakow
        return_content: yes
        status_code: 200
```

5. Uruchom playbooka z opcją verbose i zwróć uwagę na wyniki zadań Playa: test uslug.

```
[user@warszawa ansible-playbook-multi]$ ansible-playbook ./www-multi.yml  -v
(...)
PLAY [Test uslug]
*********************************************************************************************************
*********************************************************************

TASK [Gathering Facts]
*********************************************************************************************************
*********************************************************************
ok: [localhost]

TASK [test polaczenia z gdynia]
*********************************************************************************************************
*********************************************************
ok: [localhost] => {"accept_ranges": "bytes", "changed": false, "connection": "close",
"content": "Hello Apache\n", "content_length": "13", "content_type": "text/html;
charset=UTF-8", "cookies": {}, "cookies_string": "", "date": "Sun, 29 Sep 2019 16:33:37
GMT", "elapsed": 0, "etag": "\"d-593b2768c6ec1\"", "last_modified": "Sun, 29 Sep 2019
15:06:25 GMT", "msg": "OK (13 bytes)", "redirected": false, "server": "Apache/2.4.6
(CentOS)", "status": 200, "url": "http://gdynia"}

TASK [test polaczenia z krakow]
*********************************************************************************************************
*********************************************************
fatal: [localhost]: FAILED! => {"changed": false, "content": "", "elapsed": 0, "msg":
"Status code was -1 and not [200]: Request failed: <urlopen error [Errno 113] No
```

route to host>", "redirected": false, "status": -1, **"url": "http://krakow"}**

LAB: Playbooks

Kroki wstępne:	Zresetuj maszyny: **gdynia, krakow** Katalog roboczy: ~/ansible-playbooks-lab
Cel:	Zaimplementuj serwer VSFTPD na serwerach w grupie **dev.** Na maszynie **warszawa**, z której będzie uruchamiany musi być zainstalowany program **ftp**, w celu wykonania testu.
Opis:	Napisz playbook, który zainstaluje i uruchomi na domyślnej konfiguracji serwer VSFTPD. Usługa ma być aktywna, dostępna z sieci i musi uruchamiać się po restarcie. Na **localhost** (warszawa) musi istnieć aplikacja **ftp**.

1. Specyfikacja usługi:
    a. Usługa vsftpd powinna być w najnowszej możliwej wersji
    b. Usługa vsftpd powinna działać i uruchamiać się domyślnie podczas startu systemu
    c. Port 21/tcp powinien być otwarty

2. Na komputerze **warszawa** musi być zainstalowany pakiet.

3. Zweryfikuj działanie usługi korzystając z aplikacji ftp na komputerze **warszawa**

LAB: Playbooks - rozwiązanie

1. Stwórz plik ftp.yml z poniższą zawartością:

```
---
- name: Implementacja serwera vsftpd na serwerze gdynia
  hosts: dev
  tasks:

    - name: instalacja pakietu vsftpd
      yum:
        name: vsftpd
        state: latest

    - name: uruchomienie i autostart uslugi
      service:
        name: vsftpd
        state: started
        enabled: true

    - name: firewall dla uslugi ftp
      firewalld:
        service: ftp
        permanent: yes
        state: enabled
        immediate: true

- name: Instalacja pakietu ftp na localhost
  hosts: localhost
  tasks:
    - name: Instalacja pakietu ftp
      yum:
        name: ftp
        state: latest
```

2. Uruchom playbook:

```
[user@warszawa ansible-playbooks-lab]$ ansible-playbook ./ftp.yml
```

3. Wykonaj test:

```
[user@warszawa ansible-playbooks-lab]$ ftp krakow
Connected to krakow (192.168.122.202).
220 (vsFTPd 3.0.2)
Name (krakow:user):
```

Rozdział 4 - Zmienne i fakty

Ćwiczenie 1 - Wykorzystywanie zmiennych

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-vars1
Cel:	Napisz playbooka który zaimplementuje na serwerze **krakow** usługę FTP.
Opis:	Playbook powinien implementować usługę VSFTP zgodną ze specyfikacją na serwerze **krakow**. Specyfikacja: a. Usługa vsftpd powinna być w najnowszej możliwej wersji b. Usługa vsftpd powinna działać i uruchamiać się domyślnie podczas startu systemu c. Usługa ftp powinna być odblokowana na firewallu Nazwa pakietu, nazwa usługi systemd i nazwa usługi firewalld powinny zostać zaimplementowane jako zmienne w play'a. Poznasz trzy techniki dołączania zmiennych: - **vars** - **vars_files** - **include_vars**

1. Stwórz plik **ftp-vars.yml** w którym będzie definicja play'a z nazwą, listą hostów:

```
---
- name: Implementacja serwera vsftpd na serwerze gdynia
  hosts: krakow
```

2. Dopisz sekcję z definicją zmiennych:

```
  vars:
        ftp_package: vsftpd
        ftp_firewall: ftp
```

```
    ftp_service: vsftpd
```

3. Dodaj definicje zadań do wykonania:

```
tasks:

  - name: instalacja pakietu {{ ftp_package }}
    yum:
      name: "{{ ftp_package }}"
      state: latest

  - name: uruchomienie i autostart uslugi {{ ftp_service }}
    service:
      name: "{{ ftp_service }}"
      state: started
      enabled: true

  - name: firewall dla uslugi {{ ftp_firewall }}
    firewalld:
      service: "{{ ftp_firewall }}"
      permanent: yes
      state: enabled
      immediate: true
```

4. Pełen plik **ftp-vars.yml** powinien wyglądać w sposób następujący:

```
---
- name: Implementacja serwera vsftpd na serwerze gdynia
  hosts: krakow
  vars:
    ftp_package: vsftpd
    ftp_firewall: ftp
    ftp_service: vsftpd

  tasks:

    - name: instalacja pakietu {{ ftp_package }}
      yum:
        name: "{{ ftp_package }}"
```

```
      state: latest

  - name: uruchomienie i autostart uslugi {{ ftp_service }}
    service:
      name: "{{ ftp_service }}"
      state: started
      enabled: true

  - name: firewall dla uslugi {{ ftp_firewall }}
    firewalld:
      service: "{{ ftp_firewall }}"
      permanent: yes
      state: enabled
      immediate: true
```

5. Przetestuj działanie playbook'a:

```
[user@warszawa ansible-vars1]$ ansible-playbook ./ftp-vars.yml
[user@warszawa ansible-vars1]$ ftp krakow
Connected to krakow (192.168.122.202).
220 (vsFTPd 3.0.2)
Name (krakow:user):
```

6. Skopiuj plik **ftp-vars.yml** do pliku **ftp-vars-gdynia.yml** i:
    a. Usuń sekcję **vars** definiującą zmienne
    b. Dopisz w definicji play'a dołączenie zewnętrznego pliku ze zmiennymi: **vars.yml** - dyrektywa **vars_files**
    c. Zmień hosta z **krakow** na **gdynia**

   Plik powinien wyglądać w sposób następujący:

```
---
- name: Implementacja serwera vsftpd na serwerze gdynia
  hosts: gdynia
  vars_files: vars.yml
  tasks:
    - name: instalacja pakietu {{ ftp_package }}
      yum:
        name: "{{ ftp_package }}"
        state: latest
```

```
  - name: uruchomienie i autostart uslugi {{ ftp_service }}
    service:
      name: "{{ ftp_service }}"
      state: started
      enabled: true

  - name: firewall dla uslugi {{ ftp_firewall }}
    firewalld:
      service: "{{ ftp_firewall }}"
      permanent: yes
      state: enabled
      immediate: true
```

7. Stwórz plik **vars.yml** o następującej zawartości:

```
ftp_package: vsftpd
ftp_firewall: ftp
ftp_service: vsftpd
```

8. Sprawdź czy playbook działa:

```
[user@warszawa ansible-vars1]$ ansible-playbook ./ftp-vars-gdynia.yml
```

9. Przetestuj kolejną formę dołączania zmiennych do play'a - bazując na pliku **ftp-vars-gdynia.yml** stwórz **ftp-vars-gdynia2.yml** modyfikując:
    a. Usuń z definicji play'a dyrektywę: **vars_files: vars.yml**
    b. Dodaj jako pierwszy task użycie modułu: **include_vars**

    Plik **ftp-vars-gdynia2.yml** powinien wyglądać w sposób następujący:

```
---
- name: Implementacja serwera vsftpd na serwerze gdynia
  hosts: gdynia
  tasks:
    - name: Ustaw zmienne
      include_vars: vars.yml

    - name: instalacja pakietu {{ ftp_package }}
      yum:
        name: "{{ ftp_package }}"
```

```
    state: latest

- name: uruchomienie i autostart uslugi {{ ftp_service }}
  service:
    name: "{{ ftp_service }}"
    state: started
    enabled: true

- name: firewall dla uslugi {{ ftp_firewall }}
  firewalld:
    service: "{{ ftp_firewall }}"
    permanent: yes
    state: enabled
    immediate: true
```

10. Sprawdź czy playbook działa:

```
[user@warszawa ansible-vars1]$ ansible-playbook ./ftp-vars-gdynia2.yml
```

Ćwiczenie 2 - Wykorzystywanie zmiennych z projektu

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-vars2
Cel:	Definiowanie zmiennych typu host_vars lub group_vars
Opis:	Stwórz odpowiednią strukturę do przechowywania zmiennych dla grup i hostów w projekcie. Sprawdź jak zachowuje się nadpisywanie zmiennych definiowanych w różnych miejscach.

1. Stwórz odpowiednią strukturę katalogów:

```
[user@warszawa ansible-vars2]$ mkdir -p group_vars/dev
[user@warszawa ansible-vars2]$ mkdir -p host_vars/gdynia
```

2. Stwórz pliki definiujące zmienne zgodnie z wytycznymi poniżej:

```
[user@warszawa ansible-vars2]$ cat ./group_vars/dev/1.yml
zmienna: "Zmienna zdefiniowana dla wszystkich z grupy dev"

[user@warszawa ansible-vars2]$ cat ./host_vars/gdynia/1.yml
zmienna: "Zmienna zdefiniowana jako zmienna hostvars dla gdyni"
```

3. Struktura katalogów powinna wyglądać jak poniżej:

```
[user@warszawa ansible-vars2]$ tree ./
./
├── ansible.cfg
├── group_vars
│   └── dev
│       └── 1.yml
├── host_vars
│   └── gdynia
│       └── 1.yml
└── inventory
```

4. Z wykorzystaniem modułu **debug** wyświetl zmienną o nazwie **zmienna** na serwerach z grupy **servers**.

```
[user@warszawa ansible-vars2]$ ansible servers -m debug -a "var=zmienna"
gdynia | SUCCESS => {
        "zmienna": "Zmienna zdefiniowana jako zmienna hostvars dla gdyni"
}
krakow | SUCCESS => {
        "zmienna": "Zmienna zdefiniowana dla wszystkich z grupy dev"
}
```

Ćwiczenie 3 - Wykorzystywanie faktów

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-facts1
Cel:	Korzystanie z faktów wbudowanych i definiowanych
Opis:	Celem ćwiczenia jest: - Stworzenie playbooka który zwróci informacje o FQDN, IP i ilości RAMu na zdalnych hostach - Definiowanie własnych faktów na zdalnych serwerach - faktem będzie nazwa pakietu, który instalowany będzie poprzez moduł ansible. - Analiza dostępnych Magics Vars

1. Sprawdź za pomocą modułu **setup** jakie dostępne są fakty na maszynie **gdynia**

```
[user@warszawa ansible-facts1]$ ansible gdynia -m setup
gdynia | SUCCESS => {
        "ansible_facts": {
        "ansible_all_ipv4_addresses": [
        "192.168.122.201"
        ],
        "ansible_all_ipv6_addresses": [
        "fe80::8491:75f4:f498:f37"
        ],
        "ansible_apparmor": {
        "status": "disabled"
        },
        "ansible_architecture": "x86_64",
(...)
```

2. Napisz playbook'a który z wykorzystaniem modułu **debug** wyświetli następujące informacje o serwerach z grupy **servers**
    a. FQDN
    b. Adres IP
    c. Ilość RAMu

    Playbook powinien wyglądać tak:

```
---
- name: Wyswietlanie faktow
  hosts: servers
  tasks:
    - name: Wyswietl fakty
      debug:
        msg: "{{ ansible_fqdn }} {{ ansible_default_ipv4.address }} {{
ansible_memory_mb.real.total }}"
```

3. Uruchom playbook i sprawdź wyniki:

```
[user@warszawa ansible-facts1]$ ansible-playbook ./facts.yml

PLAY [Wyswietlanie faktow]
*********************************************************************************************
*****************************************************************

TASK [Gathering Facts]
*********************************************************************************************
*****************************************************************
ok: [krakow]
ok: [gdynia]

TASK [Wyswietl fakty]
*********************************************************************************************
*****************************************************************************
ok: [gdynia] => {
        "msg": "gdynia.domainx.local 192.168.122.201 990"
}
ok: [krakow] => {
        "msg": "krakow.domainx.local 192.168.122.202 990"
}
```

4. Na serwerze **gdynia** stwórz definicje własnych faktów. Fakt będzie nazwą pakietu, który ma zostać zainstalowany. Krok wykonaj na serwerze **gdynia** z konta **root**.
    a. Stwórz katalog **/etc/ansible/facts.d**

```
[root@gdynia ~]# mkdir -p /etc/ansible/facts.d
```

b. Stwórz plik **/etc/ansible/facts.d/custom.fact** o następującej zawartości:

```
[requests]
package = screen
state = present
```

5. Napisz playbook **gdynia-facts.yml**, który używając powyższych faktów wykona instalację lub deinstalację określonego pakietu.

```
---
- name: Instalacja lub dezinstalacja requestowanego pakietu ze zdalnego hosta
  hosts: gdynia
  tasks:
    - name: Zainstaluj requestowany pakiet
      yum:
        name: "{{ ansible_local['custom']['requests']['package'] }}"
        state: "{{ ansible_local['custom']['requests']['state'] }}"
```

6. Uruchom playbook:

```
[user@warszawa ansible-facts1]$ ansible-playbook ./gdynia-facts.yml  -v
```

7. Sprawdź czy na hoście gdynia jest zainstalowany pakiet screen:

```
[user@warszawa ansible-facts1]$ ansible gdynia -m shell -a "rpm -q screen"
(...)

gdynia | CHANGED | rc=0 >>
screen-4.1.0-0.25.20120314git3c2946.el7.x86_64
```

8. Ostatnim krokiem tego ćwiczenia jest zapoznanie się z dostępnymi wbudowanymi faktami typu Magic Variables - wyświetl wszystkie dostępne dla localhost:

```
[user@warszawa ansible-facts1]$ ansible localhost -m debug -a \
'var=hostvars["localhost"]'
```

9. Zobacz jak możesz za pomocą ansible korzystać ze zmiennych takich jak lista wszystkich hostów lub hosty z danej grupy:

```
[user@warszawa ansible-facts1]$ ansible localhost -m debug -a \
'var=hostvars["localhost"]["groups"]["all"]'
localhost | SUCCESS => {
        "hostvars[\"localhost\"][\"groups\"][\"all\"]": [
        "gdynia.domainx.local",
        "krakow.domainx.local",
        "192.168.122.201",
        "192.168.122.202",
        "gdynia",
        "krakow"
        ]
}

[user@warszawa ansible-facts1]$ ansible localhost -m debug -a \
'var=hostvars["localhost"]["groups"]["prod"]'
localhost | SUCCESS => {
        "hostvars[\"localhost\"][\"groups\"][\"prod\"]": [
        "gdynia"
        ]
}
```

Ćwiczenie 3 - Sekrety

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-secrets
Cel:	Definiowanie i używanie sekretów
Opis:	Celem ćwiczenia jest: - Stworzenie playbooka który zwróci informacje o FQDN, IP i ilości RAMu na zdalnych hostach - Definiowanie własnych faktów na zdalnych serwerach - faktem będzie nazwa pakietu, który instalowany będzie poprzez moduł ansible. - Analiza dostępnych Magics Vars

1. Wygeneruj zaszyfrowane hasło “altkom” tworząc tymczasowego użytkownika. Zrób to korzystając z konta **root** na maszynie **warszawa**.

```
[user@warszawa ansible-secrets]$ sudo -i

[root@warszawa ~]# useradd test

[root@warszawa ~]# echo altkom | passwd --stdin test
Changing password for user test.
passwd: all authentication tokens updated successfully.

[root@warszawa ~]# tail -n1 /etc/shadow| cut -f2 -d:
$6$AnCZzn6D$xMtHEfms6YKI6WBsCoaQp6uEW.oHONwC9cbp7.VijeAaNsLPUEBV
WpMNNtciWXXk2fGsWKI7PGv.ehA4gJVHY1

[root@warszawa ~]# tail -n1 /etc/shadow| cut -f2 -d: > /tmp/password

[root@warszawa ~]# userdel -r test
```

2. Stwórz plik **password.yml** który będzie zaszyfrowany i zawierać będzie zmienną pass o zawartości zaszyfrowanego hasła wygenerowane przed chwilą.
    a. Stwórz plik, podczas tworzenia **ansible-vault** zapytania się jakim hasłem chcesz zabezpieczyć plik - użyj hasła **altkom**.

```
[user@warszawa ansible-secrets]$ ansible-vault create password.yml
New Vault password:
Confirm New Vault password:
```

b. Zawartość pliku powinna wyglądać jak poniżej. **Uwaga**: jest to jedna linia.

```
pw:
'$6$AnCZzn6D$xMtHEfms6YKI6WBsCoaQp6uEW.oHONwC9cbp7.VijeAaNsLPUEBV
WpMNNtciWXXk2fGsWKI7PGv.ehA4gJVHY1'
```

3. Sprawdź, że plik jest zaszyfrowany:

```
[user@warszawa ansible-secrets]$ cat password.yml
$ANSIBLE_VAULT;1.1;AES256
396461323665613765636563363139323637613034646531386333333438666630316
36464656635
356536636535613664346533323234306539623634666363 0a3339653363393964343
46630656538
636430303238323261383232386431396465343331613838616461326563643064663
93535306339
643364383039323337 0a373065626337643234383363383130386165636464656165 3
56262616164
643930636663373366313635346464303061326262613536333165313830663137316
13266386362
383937326463313765373231346664613163363932323634643161316338383161373
53437643037
393439303038646232363539336533303264613736343065333934356466373231383
13837653933
356639393961663362303933646333326436356231353030653563653964333133643
36335393061
333463653964326135643336373631336464383038373332386430343837626164 32
```

4. Zweryfikuj odkodowaną zawartość pliku:

```
[user@warszawa ansible-secrets]$ ansible-vault view ./password.yml
Vault password:
pw:
'$6$AnCZzn6D$xMtHEfms6YKI6WBsCoaQp6uEW.oHONwC9cbp7.VijeAaNsLPUEBV
WpMNNtciWXXk2fGsWKI7PGv.ehA4gJVHY1'
```

5. Stwórz playbook, który na serwerach z grupy **servers** będzie tworzyć z wykorzystaniem modułu **user** użytkownika **secretuser** a hasło będzie pobierane ze zmiennej **pw** zdefiniowanej w zaszyfrowanym pliku **password.yml**

```
---
- name: Stworzenie usera z haslem z sekretow
  hosts: servers
  vars_files:
    - password.yml
  tasks:
    - name: Stworz usera
      user:
        name: secretuser
        password: "{{ pw }}"
```

6. Uruchom playbooka z opcją **--ask-vault-pass** i wpisz hasło gdy zostaniesz o nie poproszony:

```
[user@warszawa ansible-secrets]$ ansible-playbook ./user.yml --ask-vault-pass
Vault password:

PLAY [Stworzenie usera z haslem z sekretow]
*********************************************************************************************************
*********************************************

TASK [Gathering Facts]
*********************************************************************************************************
*****************************************************************
ok: [gdynia]
ok: [krakow]

TASK [Stworz usera]
*********************************************************************************************************
*********************************************************************
changed: [gdynia]
changed: [krakow]

PLAY RECAP
*********************************************************************************************************
*************************************************************************
gdynia                     : ok=2 changed=1    unreachable=0        failed=0        skipped=0
```

```
        rescued=0    ignored=0
krakow                  : ok=2 changed=1    unreachable=0       failed=0        skipped=0
        rescued=0    ignored=0
```

7. Zweryfikuj nowo utworzonych użytkowników

```
[user@warszawa ~]$ ssh secretuser@gdynia
[secretuser@gdynia ~]$ exit
logout
Connection to gdynia closed.

[user@warszawa ~]$ ssh secretuser@krakow
secretuser@krakow's password:
[secretuser@krakow ~]$ exit
logout
Connection to krakow closed.
```

Rozdział 5 - Templates

Ćwiczenie 1 - Wykorzystywanie szablonów

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-templates1
Cel:	Generowanie zawartości z wykorzystaniem templates.
Opis:	Z wykorzystaniem templates zaimplementuj właściwe powitanie serwera (/etc/issue) na serwerach z grupy **servers**.

1. Stwórz plik **vars.yml** ze zmiennymi, które będą wykorzystywane w template:

```
nazwa_firmy: ACME Sp. z o.o.
tech_email: 31337@remotehost
admin_name: Super admin
```

2. Stwórz plik **issue.j2** który będzie template'em dla /etc/issue

```
Witaj w {{ ansible_hostname }}
Dostep do tego systemu moga miec tylko pracownicy {{ nazwa_firmy }}
Jesli masz jakiekolwiek problemy techniczne - skontaktuj sie z:
{{ admin_name }} - {{ tech_email }}
```

3. Stwórz plik **vars.yml** ze zmiennymi, które będą wykorzystywane w template:

```
nazwa_firmy: ACME Sp. z o.o.
tech_email: 31337@remotehost
admin_name: Super admin
```

4. Napisz playbook'a korzystającego z modułu **template** aby zaimplementować plik /etc/issue na serwerach z grupy **servers**

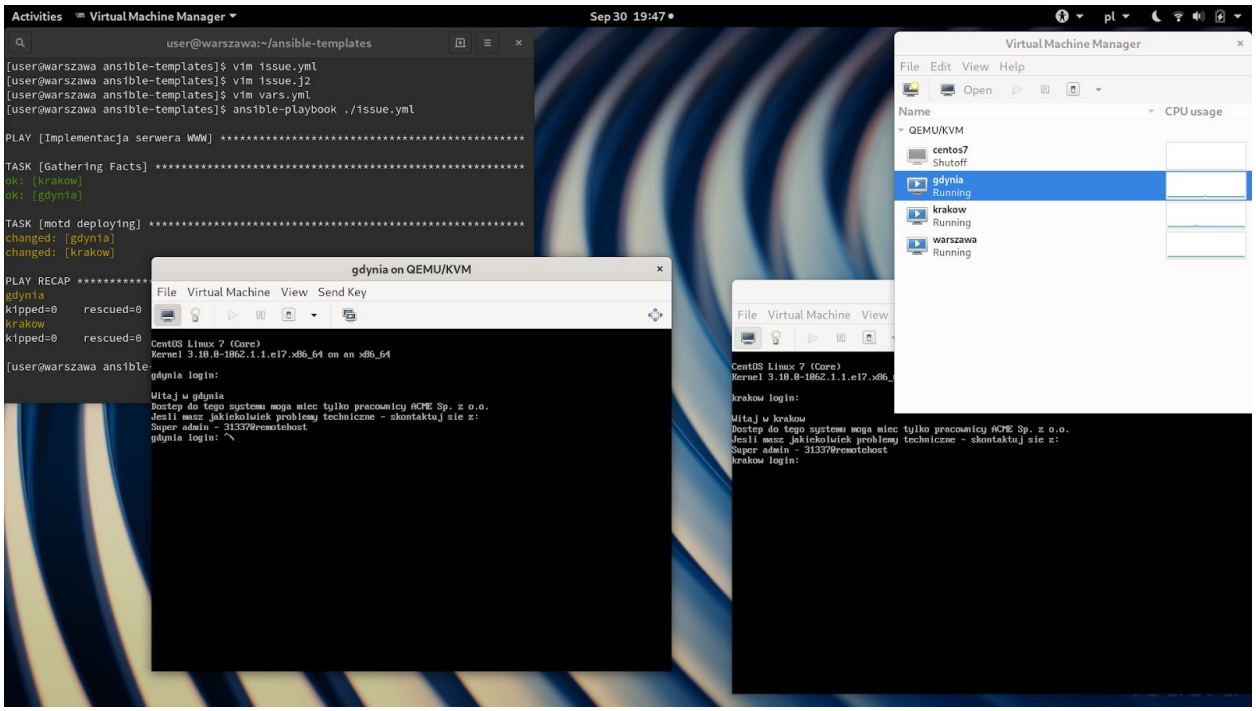
```
---
```

```
- name: Implementacja serwera WWW
  hosts: servers
  vars_files: vars.yml
  tasks:

  - name: motd deploying
    template:
      src: issue.j2
      dest: /etc/issue
      owner: root
      group: root
      mode: 06444
```

5. Uruchom playbook'a i sprawdź poprzez konsole dostępową do maszyn wirtualnych czy plik issue został prawidłowo zaimplementowany.

```
[user@warszawa ansible-templates]$ ansible-playbook ./issue.yml
```



LAB: Templates

Kroki wstępne:	Zresetuj maszyny: **gdynia, krakow** Katalog roboczy: ~/ansible-templates-lab
Cel:	Wykorzystywanie templates w tworzeniu VirtualHostów w usłudze HTTPD.
Opis:	Napisz playbook, który zainstaluje i uruchomi serwer HTTPD, zgodnie ze specyfikacją.

1. Specyfikacja usługi (na serwerach z grupy **servers**):
    a. Usługa httpd powinna być w najnowszej możliwej wersji
    b. Usługa httpd powinna działać i uruchamiać się domyślnie podczas startu systemu
    c. Port 80/tcp powinien być otwarty
    d. Powinny istnieć wirtualny host dla krótkiej nazwy hosta.
    e. VirtualnyHost powinien wyświetlać napis: “Jestem wirtualnym hostem dla krótkiej nazwy <hostname>”

LAB: Templates - rozwiązanie

1. Stwórz template dla VirtualnegoHosta - **virtualhost.j2**

```
<VirtualHost *:80>
    ServerName {{ ansible_hostname }}
    DocumentRoot /var/www/{{ ansible_hostname }}
</VirtualHost>
```

2. Stwórz playbook'a **virtualhost-template.yml**

```
---
- name: Implementacja serwera WWW
  hosts: servers
  tasks:

  - name: instalacja pakietu httpd
    yum:
      name: httpd
      state: latest

  - name: Stworzenie katalogu dla VirtualnegoHosta
    file:
      name: "/var/www/{{ ansible_hostname }}"
      state: directory

  - name: stworzenie pliku index.html
    copy:
      content: "Jestem wirtualnym hostem dla krotkiej nazwy {{ ansible_hostname }}\n"
      dest: "/var/www/{{ ansible_hostname }}/index.html"

  - name: firewall dla portu 80/tcp
    firewalld:
      port: 80/tcp
      permanent: yes
      state: enabled
      immediate: true

  - name: stworzenie wirtualnego hosta
    template:
```

```
    src: virtualhost.j2
    dest: "/etc/httpd/conf.d/{{ ansible_hostname }}.conf"

- name: restart uslugi
  service:
    name: httpd
    state: restarted
    enabled: true
```

3. Uruchom playbook:

```
[user@warszawa ansible-templates-lab]$ ansible-playbook ./virtualhost-template.yml
```

4. Dokonaj sprawdzenia na jednym z hostów.

```
[root@gdynia ~]# httpd -S
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
using gdynia.domainx.local. Set the 'ServerName' directive globally to suppress this
message
VirtualHost configuration:
*:80                   gdynia (/etc/httpd/conf.d/gdynia.conf:1)
ServerRoot: "/etc/httpd"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/etc/httpd/logs/error_log"
Mutex rewrite-map: using_defaults
Mutex authdigest-client: using_defaults
Mutex proxy: using_defaults
Mutex authn-socache: using_defaults
Mutex default: dir="/run/httpd/" mechanism=default
Mutex mpm-accept: using_defaults
Mutex authdigest-opaque: using_defaults
Mutex proxy-balancer-shm: using_defaults
PidFile: "/run/httpd/httpd.pid"
Define: _RH_HAS_HTTPPROTOCOLOPTIONS
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="apache" id=48
Group: name="apache" id=48

[root@gdynia ~]# cat /etc/httpd/conf.d/gdynia.conf
<VirtualHost *:80>
```

```
    ServerName gdynia
    DocumentRoot /var/www/gdynia
</VirtualHost>
[root@gdynia ~]# ls /var/www/gdynia/
index.html
[root@gdynia ~]# curl http://gdynia
Jestem wirtualnym hostem dla krotkiej nazwy gdynia
[root@gdynia ~]#
```

Rozdział 6 - Zarządzanie zadaniami

Ćwiczenie 1 - Instrukcja when

Kroki wstępne:	Katalog roboczy: ~/ansible-when
Cel:	Wykonywanie zadań tylko i wyłącznie gdy spełniony jest określony warunek
Opis:	Baza mariadb będzie instalowana tylko i wyłącznie na hostach, które spełnią dwa warunki: - Host posiada wystarczającą ilość RAMu - Host oparty jest na dystrybucji, który jest supportowana

1. Napisz moduł, który będzie zawierać:
    a. Listę supportowanych dystrybucji - Debian, RedHat
    b. Określoną minimalną ilość RAMu - 20000MB
    c. Zadanie korzystające z modułu **yum** i instrukcji **when** określającej czy pakiet **mariadb** może zostać zainstalowany
    d. Jeśli któryś z warunków nie jest spełniony ansible powinien o tym napisać korzystając z modułu **debug**

```
---
- name: Instalacja serwera MariaDB na hostach spelniajacych warunki
  hosts: servers
  vars:
    min_ram_mb: 20000
    supported_distros:
      - Debian
      - RedHat

  tasks:
    - name: instalacja pakietu mariadb
      yum:
        name: mariadb
        state: latest
      when:
        - ansible_memtotal_mb >= min_ram_mb
```

```
      - ansible_distribution in supported_distros

   - name: informacje o RAM
     debug:
       msg: "{{ inventory_hostname }} posiada za malo RAMu - {{ ansible_memtotal_mb
}}MB a powinno byc minimum {{ min_ram_mb }}MB"
     when:
       - ansible_memtotal_mb < min_ram_mb

   - name: informacje o distro
     debug:
       msg: "{{ inventory_hostname }} jest {{ ansible_distribution }} ktory nie jest
supportowany a supportowane sa tylko {{ supported_distros }}"
     when:
       - ansible_distribution not in supported_distros
```

2. Uruchom playbook - powinieneś zobaczyć dwie informacje o niespełnionych warunkach

```
[user@warszawa ansible-when]$ ansible-playbook ./mariadb-when.yml
(...)

TASK [instalacja pakietu mariadb]
*********************************************************************************************************
**********************************************************
skipping: [gdynia]
skipping: [krakow]

TASK [informacje o RAM]
*********************************************************************************************************
*********************************************************************
ok: [gdynia] => {
        "msg": "gdynia posiada za malo RAMu - 990MB a powinno byc minimum
20000MB"
}
ok: [krakow] => {
        "msg": "krakow posiada za malo RAMu - 990MB a powinno byc minimum
20000MB"
}

TASK [informacje o distro]
*********************************************************************************************************
```

```
*****************************************************************
ok: [gdynia] => {
        "msg": "gdynia jest CentOS ktory nie jest supportowany a supportowane
sa tylko [u'Debian', u'RedHat']"
}
ok: [krakow] => {
        "msg": "krakow jest CentOS ktory nie jest supportowany a supportowane
sa tylko [u'Debian', u'RedHat']"
}

PLAY RECAP
*********************************************************************************************************
*************************************************************************
gdynia                     : ok=3 changed=0    unreachable=0       failed=0       skipped=1
        rescued=0    ignored=0
krakow                     : ok=3 changed=0    unreachable=0       failed=0       skipped=1
        rescued=0    ignored=0
```

3. Dodaj w pliku playbook'a w definicji supportowanych dystrybucji rekord: **CentOS**

```
        supported_distros:
        - Debian
        - RedHat
         - CentOS
```

4. Uruchom playbook jeszcze raz. Tym razem informacje o niewspieranej dystrybucji nie powinna się pojawić.

```
[user@warszawa ansible-when]$ ansible-playbook ./mariadb-when.yml
(...)
```

5. Zmodyfikuj ponownie playbook i zmień definicje **min_ram_mb** na taką którą spełnią hosty w ćwiczeniu np. 200mb. Początek play'a powinien wyglądać tak:

```
---
- name: Instalacja serwera MariaDB na hostach spelniajacych warunki
  hosts: servers
  vars:
```

```
    min_ram_mb: 200
    supported_distros:
    - Debian
    - RedHat
    - CentOS
```

6. Uruchom playbook ponownie, mariadb powinna zainstalować się na serwerach z grupy **servers**

```
[user@warszawa ansible-when]$ ansible-playbook ./mariadb-when.yml

PLAY [Instalacja serwera MariaDB na hostach spelniajacych warunki] ***********************************************************************

TASK [Gathering Facts] ***********************************************************************
ok: [krakow]
ok: [gdynia]

TASK [instalacja pakietu mariadb] ***********************************************************************
changed: [gdynia]
changed: [krakow]

TASK [informacje o RAM] ***********************************************************************
skipping: [gdynia]
skipping: [krakow]

TASK [informacje o distro] ***********************************************************************
skipping: [gdynia]
skipping: [krakow]

PLAY RECAP ***********************************************************************
```

```
gdynia                     : ok=2 changed=1    unreachable=0    failed=0    skipped=2
        rescued=0    ignored=0
krakow                     : ok=2 changed=1    unreachable=0    failed=0    skipped=2
        rescued=0    ignored=0
```

Ćwiczenie 2 - Pętle

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-loop
Cel:	Korzystanie z dyrektywy **loop**
Opis:	Celem ćwiczenia jest: - Stwórz playbooka, który stworzy użytkowników ze zdefiniowanej listy

1. Stwórz playbooka, który będzie definiować listę użytkowników i z wykorzystaniem pętli **loop** doda ich na serwerach z grupy **servers.** Hasło każdego użytkownika będzie brzmieć tak jak jego login z “123” dodanym na końcu.

```
---
- name: Tworzenie uzytkownikow
  hosts: servers
  vars:
    users:
      - lolek
      - bolek
      - krzysiek
  tasks:
    - name: Informacja
      debug:
        msg: "Uzytkownicy ktorzy beda dostepni w systemie to: {{ users | join(', ') }}"
#         msg: "Uzytkownicy ktorzy beda dostepni w systemie to: {{ users }}"

    - name: Tworzenie uzytkownikow
      user:
        name: "{{ item }}"
        state: present
        password: "{{ ( item + '123' ) | password_hash('sha512') }}"
      loop: "{{ users }}"
```

2. Uruchom playbooka i przetestuj logując się na konto **krzysiek** z hasłem **krzysiek123** na host **gdynia**

```
[user@warszawa ansible-loop]$ ansible-playbook ./users.yml

[user@warszawa ansible-loop]$ ssh krzysiek@gdynia
krzysiek@gdynia's password:
Last login: Mon Sep 30 22:43:07 2019 from gateway
[krzysiek@gdynia ~]$
```

CIEKAWOSTKA - Został użyty tutaj filtr: {{ users **| join(‘, ‘)** }}, który powoduje zamiane domyślnego wyświetlania listy elementów na bardziej przyjazną formę z separatorem w formie przecinka - różnica outputu jest taka:

Bez filtra:
"msg": "Uzytkownicy ktorzy beda dostepni w systemie to: [u'lolek', u'bolek', u'krzysiek']"

Z filtrem:
"msg": "Uzytkownicy ktorzy beda dostepni w systemie to: lolek, bolek, krzysiek"

CIEKAWOSTKA - Został użyty tutaj filtr:{{ (item + '123') **| password_hash('sha512')** }}, który generuje zaszyfrowane hasło. Dodatkowo widzimy w jaki sposób można połączyć dwa ciągi znaków w jeden - w tym wypadku zawartość zmiennej **item + ‘123’**

Ćwiczenie 3 - Handlers

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-handlers Na hostach: **gdynia** i **krakow** musi być zainstalowany pakiet **policycoreutils-python** w celu poprawnego działania modułu dotyczącego SELinux - **seport**
Cel:	Korzystanie z **handlerów**. Dodatkowo zostaną wykorzystane moduły: **lineinfile** oraz **seport**.
Opis:	Spraw, że domyślnym portem usługi HTTPD będzie 12345/tcp. Jeśli doszło do zmiany w pliku konfiguracyjnym konieczne jest zrestartowanie usługi.

1. Zainstaluj na serwerach z grupy **servers** z wykorzystaniem komendy **ad-hoc** pakiet: policycoreutils-python

```
[user@warszawa ansible-handlers]$ ansible servers -m yum -a \
'name=policycoreutils-python'
```

2. Napisz playbooka, który uruchomi serwer HTTPD na zdefiniowanym w zmiennej porcie. Pamiętaj, że jeśli modyfikowana jest reguła SELinuxa lub sam plik konfiguracyjny, to usługa musi zostać zrestartowana.

```
---
- name: Uruchom httpd na niestandardowym porcie
  hosts: servers
  vars:
    http_port: 12345
  tasks:
    - name: httpd - instalacja
      yum:
        name: httpd
        state: latest

    - name: httpd - autostart i start
      service:
```

```
      name: httpd
      state: started
      enabled: yes

  - name: Dodaj port {{ http_port }} do httpd.conf
    lineinfile:
      path: /etc/httpd/conf/httpd.conf
      regexp: '^Listen '
      insertafter: '^#Listen '
      line: "Listen {{ http_port }}"
    notify:
      - restart apache

  - name: Dodaj port {{ http_port }} konfiguracji SELinux
    seport:
      ports: "{{ http_port }}"
      proto: tcp
      setype: http_port_t
      state: present
    notify:
      - restart apache

  - name: Dodaj port {{ http_port }} do firewall
    firewalld:
      port: "{{ http_port }}/tcp"
      state: enabled
      permanent: yes
      immediate: true

handlers:
  - name: restart apache
    service:
      name: httpd
      state: restarted
```

3. Uruchom playbook i przetestuj połączenie:

```
[user@warszawa ansible-handlers]$ ansible-playbook ./apache-handlers.yml
(...)
[user@warszawa ansible-handlers]$ curl http://gdynia:12345
```

4. Uruchom playbook ponownie - każde z zadań powinno kończyć się statusem OK - kolor zielony (czyli bez wprowadzania zmian)

```
[user@warszawa ansible-handlers]$ ansible-playbook ./apache-handlers.yml

PLAY [Uruchom httpd na niestandardowym porcie] *********************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [gdynia]
ok: [krakow]

TASK [httpd - instalacja] ******************************************************************************************
ok: [gdynia]
ok: [krakow]

TASK [httpd - autostart i start] ***********************************************************************************
ok: [gdynia]
ok: [krakow]

TASK [Dodaj port 12345 do httpd.conf] ******************************************************************************
ok: [krakow]
ok: [gdynia]

TASK [Dodaj port 12345 konfiguracji SELinux] ***********************************************************************
ok: [krakow]
ok: [gdynia]

TASK [Dodaj port 12345 do firewall] ********************************************************************************
ok: [gdynia]
```

```
ok: [krakow]

PLAY RECAP
*********************************************************************************************************
**********************************************
gdynia                     : ok=6    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
krakow                     : ok=6    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

5. Na serwerze **gdynia** z konta **root** zmodyfikuj ręcznie linijkę “Listen 12345” na “Listen 80” i zrestartuj usługę httpd.

```
[user@warszawa ansible-handlers]$ ssh root@gdynia
root@gdynia's password:
[root@gdynia ~]# vim /etc/httpd/conf/httpd.conf
[root@gdynia ~]# systemctl restart httpd
[root@gdynia ~]# logout
```

6. Uruchom playbook jeszcze raz i zobacz czy są wywoływane handlery - kolor żółty na listingu:

```
[user@warszawa ansible-handlers]$ ansible-playbook ./apache-handlers.yml

PLAY [Uruchom httpd na niestandardowym porcie]
*********************************************************************************************************
*************

TASK [Gathering Facts]
*********************************************************************************************************
************************************
ok: [gdynia]
ok: [krakow]

TASK [httpd - instalacja]
*********************************************************************************************************
*********************************
ok: [gdynia]
ok: [krakow]
```

```
TASK [httpd - autostart i start]
*********************************************************************************************************
**************************
ok: [gdynia]
ok: [krakow]

TASK [Dodaj port 12345 do httpd.conf]
*********************************************************************************************************
**********************
ok: [krakow]
changed: [gdynia]

TASK [Dodaj port 12345 konfiguracji SELinux]
*********************************************************************************************************
***************
ok: [gdynia]
ok: [krakow]

TASK [Dodaj port 12345 do firewall]
*********************************************************************************************************
************************
ok: [krakow]
ok: [gdynia]

RUNNING HANDLER [restart apache]
*********************************************************************************************************
**************************
changed: [gdynia]

PLAY RECAP
*********************************************************************************************************
************************************************
gdynia                     : ok=7   changed=2   unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
krakow                     : ok=6   changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

Ćwiczenie 4 - Statusy zadań

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-tasks-status
Cel:	Przegląd opcji ansible pod kątem obsługi statusów zadań..
Opis:	Stwórz zadania, które będą zwracać różne statusy (failed=1, changed=1) i zobacz jak działają takie opcje jak: **Ignore_errors, changed_when, failed_when**

1. Stwórz playbooka, zawierającego dwa zadania realizowane na maszynie **gdynia** - instalacja pakietu screen i wyświetlenie jakiegokolwiek napisu:

```
---
- name: Testowanie taskow
  hosts: gdynia
  vars:
        package: screen

  tasks:
        - name: instalacja pakietu {{ package }}
        yum:
        name: "{{ package }}"
        state: latest

        - name: kolejne zadanie
        debug:
        msg: "Kolejne zadanie zostalo uruchomione"
```

2. Uruchom i sprawdź czy oba zadania zostały wykonane:

```
[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status.yml

PLAY [Testowanie taskow]
*****************************************************************************************************
*********************************************************************

TASK [Gathering Facts]
```

```
**********************************************************************************************************
**************************************************************
ok: [gdynia]

TASK [instalacja pakietu screen]
**********************************************************************************************************
**********************************************************
ok: [gdynia]

TASK [kolejne zadanie]
**********************************************************************************************************
**********************************************************************
ok: [gdynia] => {
        "msg": "Kolejne zadanie zostalo uruchomione"
}

PLAY RECAP
**********************************************************************************************************
**************************************************************************************
gdynia                     : ok=3 changed=0    unreachable=0        failed=0        skipped=0
        rescued=0    ignored=0
```

3. Zmodyfikuj nazwę pakietu z “screen” na “scren” i przetestuj działanie playbooka ponownie. Taki pakiet nie istnieje i zadanie zostanie wykonane z błędem. Kolejne zadanie (wyświetlające napis) nie zostanie wykonane. Play zakończy działanie.

```
(...)
  vars:
        package: scren
(...)


[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status.yml

PLAY [Testowanie taskow]
**********************************************************************************************************
**************************************************************

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

```
ok: [gdynia]

TASK [instalacja pakietu scren]
*****************************************************************************************************
*************************************************************
fatal: [gdynia]: FAILED! => {"changed": false, "msg": "No package matching 'scren'
found available, installed or updated", "rc": 126, "results": ["No package matching
'scren' found available, installed or updated"]}

PLAY RECAP
*****************************************************************************************************
*************************************************************************
gdynia                     : ok=1 changed=0    unreachable=0        failed=1        skipped=0
        rescued=0    ignored=0
```

4. Dodaj do zadania instalującego pakiet, opcję **ignore_errors**:

```
 tasks:
        - name: instalacja pakietu {{ package }}
        yum:
        name: "{{ package }}"
        state: latest
        ignore_errors: true
```

5. Przetestuj ponownie - mimo błędu play nie powinien przerywać wykonywania kolejnych zadań.

```
[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status.yml

PLAY [Testowanie taskow]
*****************************************************************************************************
*************************************************************

TASK [Gathering Facts]
*****************************************************************************************************
*************************************************************
ok: [gdynia]

TASK [instalacja pakietu scren]
```

```
**********************************************************************************************************
**********************************************************
fatal: [gdynia]: FAILED! => {"changed": false, "msg": "No package matching 'scren' found available, installed or updated", "rc": 126, "results": ["No package matching 'scren' found available, installed or updated"]}
...ignoring

TASK [kolejne zadanie]
**********************************************************************************************************
*****************************************************************
ok: [gdynia] => {
    "msg": "Kolejne zadanie zostalo uruchomione"
}

PLAY RECAP
**********************************************************************************************************
*************************************************************************
gdynia                     : ok=3   changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=1
```

6. Stwórz nowego playbooka, który na hoście gdynia wykonuje komendę “whoami”. Po uruchomieniu zweryfikuj, że status jest **changed=1** (moduł shell gdy z sukcesem wykona zadanie zawsze zwraca status changed=1)

```
---
- name: Testowanie taskow
  hosts: gdynia

  tasks:
          - name: Sprawdz kim jestes
          shell: "whoami"
```

```
[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status2.yml

PLAY [Testowanie taskow]
**********************************************************************************************************
*****************************************************************

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

```
ok: [gdynia]

TASK [Sprawdz kim jestes]
*********************************************************************************************
*************************************************************
changed: [gdynia]

PLAY RECAP
*********************************************************************************************
*************************************************************************
gdynia                     : ok=2   changed=1   unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

7. Dodaj warunek **changed_when** i przetestuj ponownie. Tym razem status changed powinien być ustawiony na 0

```
[user@warszawa ansible-tasks-status]$ cat tasks-status2.yml
---
- name: Testowanie taskow
  hosts: gdynia

  tasks:
    - name: Sprawdz kim jestes
      shell: "whoami"
      changed_when: false
[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status2.yml

PLAY [Testowanie taskow]
*********************************************************************************************
*************************************************************

TASK [Gathering Facts]
*********************************************************************************************
*************************************************************
ok: [gdynia]

TASK [Sprawdz kim jestes]
*********************************************************************************************
*************************************************************
ok: [gdynia]

PLAY RECAP
```

```
**********************************************************************************************
**************************************************************************
gdynia                     : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

8. Zamień warunek **changed_when** na **failed_when** i przetestuj ponownie:

```
[user@warszawa ansible-tasks-status]$ cat tasks-status2.yml
---
- name: Testowanie taskow
  hosts: gdynia

  tasks:
    - name: Sprawdz kim jestes
      shell: "whoami"
      failed_when: true
[user@warszawa ansible-tasks-status]$ ansible-playbook ./tasks-status2.yml

PLAY [Testowanie taskow]
**********************************************************************************************
*****************************************************************

TASK [Gathering Facts]
**********************************************************************************************
*****************************************************************
ok: [gdynia]

TASK [Sprawdz kim jestes]
**********************************************************************************************
*****************************************************************
fatal: [gdynia]: FAILED! => {"changed": true, "cmd": "whoami", "delta":
"0:00:00.002823", "end": "2019-10-11 10:16:03.273232", "failed_when_result": true,
"rc": 0, "start": "2019-10-11 10:16:03.270409", "stderr": "", "stderr_lines": [], "stdout":
"root", "stdout_lines": ["root"]}

PLAY RECAP
**********************************************************************************************
**************************************************************************
gdynia                     : ok=1    changed=0    unreachable=0    failed=1    skipped=0
```

```
rescued=0    ignored=0
```

Ćwiczenie 5 - Bloki w zadaniach

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-tasks-block
Cel:	Tworzenie bloków ratunkowych (rescue) w zadaniach
Opis:	Stworzenie play'a który uruchamia zadania w zależności od statusu wcześniejszego zadania lub grupy zadań lub bez względu na ten status.

1. Stwórz playbooka, zawierającego bloki. Zadanie główne instalacja pakietu **screen** (literówka celowa) oraz sekcje **rescue** i **always.** Zobacz jakie zadania się wykonają i czy playbook przerwie wykonywanie działań czy będzie procesował się dalej. Play będzie procesowany, gdy zadanie z sekcji **rescue** wykona się prawidłowo.

```
---
- name: Testowanie taskow
  hosts: gdynia
  vars:
    package: scren

  tasks:
    - block:
        - name: instalacja pakietu
          yum:
            name: "{{ package }}"
            state: latest

      rescue:
        - name: informacja1
          debug:
            msg: "Uruchamiam sie tylko ratunkowo, jak wczesniejsze zadanie z bloku
zakonczy sie bledem"

      always:
        - name: informacja2
          debug:
            msg: "A ja uruchamiam sie zawsze!"
```

```
  - name: kolejne zadanie
    debug:
      msg: "Kolejne zadanie zostalo uruchomione"
```

2. Przetestuj:

```
[user@warszawa ansible-tasks-block]$ ansible-playbook ./tasks-block.yml

PLAY [Testowanie taskow]
**********************************************************************************************
*****************************************************************

TASK [Gathering Facts]
**********************************************************************************************
*****************************************************************
ok: [gdynia]

TASK [instalacja pakietu]
**********************************************************************************************
*****************************************************************
fatal: [gdynia]: FAILED! => {"changed": false, "msg": "No package matching 'scren'
found available, installed or updated", "rc": 126, "results": ["No package matching
'scren' found available, installed or updated"]}

TASK [informacja1]
**********************************************************************************************
*****************************************************************
ok: [gdynia] => {
    "msg": "Uruchamiam sie tylko ratunkowo, jak wczesniejsze zadanie z bloku
zakonczy sie bledem"
}

TASK [informacja2]
**********************************************************************************************
*****************************************************************
ok: [gdynia] => {
    "msg": "A ja uruchamiam sie zawsze!"
}

TASK [kolejne zadanie]
**********************************************************************************************
```

```
*********************************************************************
ok: [gdynia] => {
    "msg": "Kolejne zadanie zostalo uruchomione"
}

PLAY RECAP
*********************************************************************************************************
*********************************************************************************
gdynia                     : ok=4    changed=0    unreachable=0    failed=0    skipped=0
rescued=1    ignored=0
```

Rozdział 7 - Role

Ćwiczenie 1 - Tworzenie roli

Kroki wstępne:	Zresetuj maszyny **gdynia** i **krakow**. Katalog roboczy: ~/ansible-roles
Cel:	Tworzenie roli i zastosowanie jej. Użycie galaxy-init.
Opis:	Stworzenie i użycie roli implementującej serwer WWW. W roli będą zdefiniowane: - Różne pliki z taskami - Handlery - Zmienne - Templates

1. Stwórz szkielet katalogów dla nowej roli o nazwie **server_www**

```
[user@warszawa ansible-roles]$ ansible-galaxy init server_www
- server_www was created successfully
[user@warszawa ansible-roles]$ tree ./server_www/
./server_www/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

```
8 directories, 8 files
```

2. Zdefiniuj zmienną **napis**, która będzie używana jako tytuł implementowanej strony WWW: vars/main.yml i nadaj jej wartość domyślną

```
[user@warszawa ansible-roles]$ cat server_www/vars/main.yml
---
# vars file for server_www
napis: "Hello World"
```

3. Stwórz template strony WWW

```
[user@warszawa ansible-roles]$ cat server_www/templates/index.html.j2
<!doctype html>
<html>
 <head>
  <title>{{ napis }}</title>
 </head>
 <body>
    <p>Sprawdzmy czy dziala PHP:</p>
    <?php phpinfo(); ?>
 </body>
</html>
```

4. Stwórz handler na potrzeby restartowania usługi httpd

```
[user@warszawa ansible-roles]$ cat server_www/handlers/main.yml
---
# handlers file for server_www

- name: httpd restart
  service:
          name: httpd
          state: restarted
```

5. Stwórz plik z taskami do instalacji:

```
[user@warszawa ansible-roles]$ cat server_www/tasks/install.yml
---
- name: install httpd package
```

```
  yum:
        name: httpd
        state: latest

- name: install php package
  yum:
        name: mod_php
        state: latest
  notify: httpd restart
```

6. Stwórz plik z taskami do konfiguracji:

```
[user@warszawa ansible-roles]$ cat server_www/tasks/configure.yml
---
- name: index.php
  template:
        src: index.html.j2
        dest: /var/www/html/index.html

- name: httpd config to use php inside html files
  copy:
        content: "AddHandler application/x-httpd-php .html"
        dest: /etc/httpd/conf.d/custom.conf
  notify: httpd restart
```

7. Stwórz plik z taskami na potrzeby startu usługi i dostępu z sieci

```
[user@warszawa ansible-roles]$ cat server_www/tasks/service.yml
---
- name: httpd autostart and start
  service:
        name: httpd
        state: started
        enabled: true

- name: firewalld for httpd
  firewalld:
        service: http
        state: enabled
```

```
      permanent: yes
```

8. Stwórz główny plik z taskami, który includować będzie zadania z wcześniejszych plików

```
[user@warszawa ansible-roles]$ cat server_www/tasks/main.yml
---
# tasks file for server_www
- include_tasks: install.yml
- include_tasks: configure.yml
- include_tasks: service.yml
```

9. Wygląda drzewa katalogu, po stworzeniu dodatkowych plików powinien wyglądać tak:

```
[user@warszawa ansible-roles]$ tree ./server_www/
./server_www/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   ├── configure.yml
│   ├── install.yml
│   ├── main.yml
│   └── service.yml
├── templates
│   └── index.html.j2
├── tests
│   ├── inventory
│   └── test.yml
└── vars
        └── main.yml
```

10. Spraw, że rola będzie dostępna dla ansible

```
[user@warszawa ansible-roles]$ mkdir -p ~/.ansible/roles

[user@warszawa ansible-roles]$ ln -s /home/user/ansible-roles/server_www/
~/.ansible/roles/server_www
```

11. Stwórz playbooka, który użyje roli **server_www** na serwerze **gdynia** z parametrem **napis**

```
[user@warszawa ansible-roles]$ cat gdynia_server_www.yml
---
- name: Testowanie roli
  hosts: gdynia

  roles:
        - role: server_www
        napis: "Niedomyslne hello world"
```

12. Uruchom i przetestuj czy strona PHP działa prawidłowo

```
[user@warszawa ansible-roles]$ ansible-playbook ./gdynia_server_www.yml
(...)
[user@warszawa ansible-roles]$ curl -s http://gdynia | grep "PHP Version"
<a href="http://www.php.net/"><img border="0"
src="/index.html?=PHPE9568F34-D428-11d2-A769-00AA001ACF42" alt="PHP Logo"
/></a><h1 class="p">PHP Version 5.4.16</h1>
<tr><td class="e">PHP Version </td><td class="v">5.4.16 </td></tr>
```

Ćwiczenie 2 - QUIZ - Ansible-galaxy

PYTANIE 1:

Wydajesz komendę zdefiniowaną jak poniżej. Jak musi wyglądać zawartość pliku, aby rola zainstalowała się z pliku /test.tar?

```
ansible-galaxy install -r requirements.yml -p roles
```

Odpowiedź A:

```
- src: /test.tar
  name: test
```

Odpowiedź B:

```
- src: file:///test.tar
  name: test
```

Odpowiedź C:

```
- role:
   name: test
   src: /test.tar
```

Odpowiedź D:

```
- role:
  name: test
  src: file:///test.tar
```

PYTANIE 2:

Czasami poniższa komenda nie działa gdy nie ma dostępu do Internetu, jakiej opcji użyć w takim przypadku?

```
ansible-galaxy init rola
```

Odpowiedź A:

```
--without-refresh
```

Odpowiedź B:

```
--offline
```

Odpowiedź C:

```
--no-updates
```

Odpowiedź D:

```
--no-login
```

PYTANIE 3:

Jak nazywa się moduł, który pozwala używać roli w play'u?

Odpowiedź A:

role

Odpowiedź B:

include_role

Odpowiedź C:

with_role

Odpowiedź D:

roles

Odpowiedzi: 1B, 2B, 3B

Rozdział 8 - Dodatkowe funkcjonalności i troubleshooting

Ćwiczenie 1 - Dynamic inventory

Kroki wstępne:	Katalog roboczy: ~/ansible-dynamic-inventory
Cel:	Przykład działania dynamicznych inventory
Opis:	Wykorzystanie skryptu napisanego w PHP do generowania dynamicznych inventory. Kod skryptu oparty na: https://www.jeffgeerling.com/blog/creating-custom-dynamic-inventories-ansible

1. Stwórz katalog **inventory** (istniejący plik inventory nazwij inventory_static i umieść w tym katalogu). Dodatkowo sprawdź, że statyczne inventory cały czas działa np. odpytaj się o host **gdynia**

```
[user@warszawa ansible-dynamic-inventory]$ mv inventory inventory_static
[user@warszawa ansible-dynamic-inventory]$ mkdir inventory
[user@warszawa ansible-dynamic-inventory]$ mv inventory_static inventory
[user@warszawa ansible-dynamic-inventory]$ ansible --list-hosts gdynia
  hosts (1):
        gdynia
```

2. Stwórz plik w katalogu **inventory** o nazwie **dynamic.php** z następującą zawartością:

```
#!/usr/bin/php
<?php

function example_inventory() {
```

```
  return [
        'group' => [
        'hosts' => ['swinoujscie', 'pacanow'],
        'vars' => [
        'ansible_ssh_user' => 'devops',
        'example_variable' => 'value',
        ],
        ],
        '_meta' => [
        'hostvars' => [
        'swinoujscie' => [
        'host_specific_var' => 'foo',
        ],
        'pacanow' => [
        'host_specific_var' => 'bar',
        ],
        ],
        ],
  ];
}

function empty_inventory() {
  return ['_meta' => ['hostvars' => new stdClass()]];
}

function get_inventory($argv = []) {
  $inventory = new stdClass();

  // Called with `--list`.
  if (!empty($argv[1]) && $argv[1] == '--list') {
        $inventory = example_inventory();
  }
  // Called with `--host [hostname]`.
  elseif ((!empty($argv[1]) && $argv[1] == '--host') && !empty($argv[2])) {
        // Not implemented, since we return _meta info `--list`.
        $inventory = empty_inventory();
  }
  // If no groups or vars are present, return an empty inventory.
  else {
        $inventory = empty_inventory();
  }

  print json_encode($inventory);
```

```
}

// Get the inventory.
get_inventory($_SERVER['argv']);

?>
```

3. Spraw, że skrypt będzie się prawidłowo uruchamiał - zainstaluj pakiet php i nadaj na skrypcie bit wykonywalności

```
[user@warszawa ansible-dynamic-inventory]$ chmod +x inventory/dynamic.php
[user@warszawa ansible-dynamic-inventory]$ sudo yum install php
```

4. Uruchom skrypt z poniższymi opcją **--list**

```
[user@warszawa ansible-dynamic-inventory]$ ./inventory/dynamic.php --list
{"group":{"hosts":["swinoujscie","pacanow"],"vars":{"ansible_ssh_user":"devops","exam
ple_variable":"value"}},"_meta":{"hostvars":{"swinoujscie":{"host_specific_var":"foo"},"p
acanow":{"host_specific_var":"bar"}}}}

[user@warszawa ansible-dynamic-inventory]$ ./inventory/dynamic.php --list |
python -m json.tool
{
        "_meta": {
        "hostvars": {
        "pacanow": {
                "host_specific_var": "bar"
        },
        "swinoujscie": {
                "host_specific_var": "foo"
        }
        }
        },
        "group": {
        "hosts": [
        "swinoujscie",
        "pacanow"
        ],
        "vars": {
        "ansible_ssh_user": "devops",
        "example_variable": "value"
```

```
        }
        }
}
```

5. Sprawdź czy Ansible byłby wstanie wykonywać operacje na hoście **pacanow,** mimo braku takiego statycznym inventory

```
[user@warszawa ansible-dynamic-inventory]$ ansible --list-hosts pacanow
  hosts (1):
        pacanow
```

Ćwiczenie 2 - Ansible parallelism

Kroki wstępne:	Katalog roboczy: ~/ansible-parallelism Zresetuj maszyny: **gdynia** i **krakow**
Cel:	Testowanie wykonywania działań na hostach równolegle z różnymi ustawieniami zmiennej **forks**.
Opis:	Wykonaj testy uruchamiania zadań zajmujących zdefiniowany czas (10 sekund) na różnych konfiguracjach Ansible / Playbooka.

1. Stwórz playbooka, który na hoście: **all** wykona zadanie trwające 10 sekund. Pamiętaj, że pod hasłem **all** zdefiniowane są wszystkie rekordy z inventory - w tym przypadku jest to 6 hostów (mimo, że realnie 2) - krótkie nazwy: **krakow gdynia**, pełne nazwy FQDN **krakow.domainx.local** i **gdynia.domainx.local** oraz adresy IP **192.168.122.201** i **192.168.122.202**

```
---
- name: Sekund dziesiec
  hosts: all
  tasks:
        - name: Dziesiec sekund
        shell: sleep 10
```

2. Uruchom play'a mierząc czas (można to zrobić używając komendy time)

```
[user@warszawa ansible-parallelism]$ time ansible-playbook ./forks.yml

PLAY [Sekund dziesiec]
*********************************************************************************************************
*********************************************************************

TASK [Gathering Facts]
*********************************************************************************************************
*********************************************************************
ok: [krakow.domainx.local]
ok: [192.168.122.202]
ok: [gdynia]
```

```
ok: [192.168.122.201]
ok: [gdynia.domainx.local]
ok: [krakow]

TASK [Dziesiec sekund]
*********************************************************************************************************
*****************************************************************
changed: [krakow.domainx.local]
changed: [192.168.122.202]
changed: [gdynia]
changed: [gdynia.domainx.local]
changed: [192.168.122.201]
changed: [krakow]

PLAY RECAP
*********************************************************************************************************
*********************************************************************************
192.168.122.201            : ok=2 changed=1    unreachable=0       failed=0
skipped=0    rescued=0    ignored=0
192.168.122.202            : ok=2 changed=1    unreachable=0       failed=0
skipped=0    rescued=0    ignored=0
gdynia                     : ok=2 changed=1    unreachable=0       failed=0       skipped=0
        rescued=0    ignored=0
gdynia.domainx.local       : ok=2 changed=1    unreachable=0       failed=0
skipped=0    rescued=0    ignored=0
krakow                     : ok=2 changed=1    unreachable=0       failed=0       skipped=0
        rescued=0    ignored=0
krakow.domainx.local       : ok=2 changed=1    unreachable=0       failed=0
skipped=0    rescued=0    ignored=0


real    0m23.854s
user    0m2.715s
sys     0m0.980s
```

3. Sprawdź, że domyślna liczba hostów na których wykonywane są zadania wynosi: 5. To znaczy, że 6 hostów na których działamy w tym ćwiczeniu realizowana jest w dwóch turach: pierwsza tura - 5 hostów; 2 tura - 1 host. Wystarczyło by zmienić domyślne ustawienie max. Ilości hostów na których ansible działa równolegle na 6 i już wtedy powinniśmy zauważyć poprawę w czasie wykonania całości play'a.

```
[user@warszawa ansible-parallelism]$ ansible-config dump | grep -i fork
DEFAULT_FORKS(default) = 5

[user@warszawa ansible-parallelism]$ ansible-config list
(...)
DEFAULT_FORKS:
  default: 5
  description: Maximum number of forks Ansible will use to execute tasks on targ
          hosts.
  env:
  - {name: ANSIBLE_FORKS}
  ini:
  - {key: forks, section: defaults}
  name: Number of task forks
  type: integer
(...)
```

4. Zmień wartość opcji **forks** z 5 na 6. Wyedytuj plik z ansible.cfg w katalogu roboczym i w sekcji [defaults] dodaj **forks=6**

```
[user@warszawa ansible-parallelism]$ cat ansible.cfg
[defaults]
inventory=./inventory
remote_user=devops
forks=6

[privilege_escalation]
become=True
become_method=sudo
become_user=root
```

5. Przetestuj ile teraz będzie trwało wykonanie play'a - wykonaj test tak jak wcześniej w punkcie 2.

```
[user@warszawa ansible-parallelism]$ time ansible-playbook ./forks.yml

PLAY [Sekund dziesiec] ***********************************************************

TASK [Gathering Facts] ***********************************************************
ok: [krakow.domainx.local]
ok: [krakow]
```

```
ok: [192.168.122.202]
ok: [192.168.122.201]
ok: [gdynia]
ok: [gdynia.domainx.local]

TASK [Dziesiec sekund] ***********************************************************
changed: [192.168.122.201]
changed: [krakow.domainx.local]
changed: [krakow]
changed: [192.168.122.202]
changed: [gdynia]
changed: [gdynia.domainx.local]

PLAY RECAP ***********************************************************************
192.168.122.201            : ok=2 changed=1    unreachable=0        failed=0
skipped=0    rescued=0    ignored=0
192.168.122.202            : ok=2 changed=1    unreachable=0        failed=0
skipped=0    rescued=0    ignored=0
gdynia                     : ok=2 changed=1    unreachable=0        failed=0        skipped=0
        rescued=0    ignored=0
gdynia.domainx.local       : ok=2 changed=1    unreachable=0        failed=0
skipped=0    rescued=0    ignored=0
krakow                     : ok=2 changed=1    unreachable=0        failed=0        skipped=0
        rescued=0    ignored=0
krakow.domainx.local       : ok=2 changed=1    unreachable=0        failed=0
skipped=0    rescued=0    ignored=0


real    0m13.314s
user    0m2.073s
sys     0m0.542s
```

6. Ponownie zmień wartość **forks**, tym razem na **1** i przetestuj czasy ponownie

```
[defaults]
inventory=./inventory
remote_user=devops
forks=1

[privilege_escalation]
become=True
become_method=sudo
```

```
become_user=root


become_user=root
```

7. Zmień wartość z 5 na 6. Wyedytuj plik z ansible.cfg w katalogu roboczym i w sekcji [defaults] dodaj forks=6

```
[user@warszawa ansible-parallelism]$ cat ansible.cfg
[defaults]
inventory=./inventory
remote_user=devops
forks=6

[privilege_escalation]
become=True
become_method=sudo
become_user=root

[user@warszawa ansible-parallelism]$ time ansible-playbook ./forks.yml

PLAY [Sekund dziesiec]
**********************************************************************************************
**********************************************************************

TASK [Gathering Facts]
**********************************************************************************************
**********************************************************************
ok: [gdynia.domainx.local]
ok: [krakow.domainx.local]
ok: [192.168.122.201]
ok: [192.168.122.202]
ok: [gdynia]
ok: [krakow]

TASK [Dziesiec sekund]
**********************************************************************************************
**********************************************************************
changed: [gdynia.domainx.local]
changed: [krakow.domainx.local]
changed: [192.168.122.201]
changed: [192.168.122.202]
changed: [gdynia]
```

```
changed: [krakow]

PLAY RECAP
*********************************************************************************************************
*********************************************************************************
192.168.122.201            : ok=2 changed=1    unreachable=0        failed=0
skipped=0     rescued=0     ignored=0
192.168.122.202            : ok=2 changed=1    unreachable=0        failed=0
skipped=0     rescued=0     ignored=0
gdynia                     : ok=2 changed=1    unreachable=0        failed=0        skipped=0
          rescued=0     ignored=0
gdynia.domainx.local       : ok=2 changed=1    unreachable=0        failed=0
skipped=0     rescued=0     ignored=0
krakow                     : ok=2 changed=1    unreachable=0        failed=0        skipped=0
          rescued=0     ignored=0
krakow.domainx.local       : ok=2 changed=1    unreachable=0        failed=0
skipped=0     rescued=0     ignored=0


real    1m6.469s
user    0m3.080s
sys     0m1.961s
```

Ćwiczenie 3 - Ansible troubleshooting

Kroki wstępne:	Katalog roboczy: ~/ansible-troubleshooting
Cel:	Poznaj jak wykorzystywać moduł **debug**, używać **zmiennych rejestrowanych (register)** oraz wykonywać zadania krok po kroku (**step**).
Opis:	Play wykonywać będzie dwa zadania: wykonanie komendy **date** oraz instalacja pakietu **scren** (literówka celowa). Oba taski będą rejestrować swoje stany i w zależności od uruchomienia zadania typu **debug** będą je wyświetlać.

1. Stwórz playbooka realizującego zadanie **shell** i **yum** z dodatkowymi zadaniami typu **debug**:

```
[user@warszawa ansible-troubleshooting]$ cat ansible-troubleshooting.yml
---
- name: Korzystanie z modulu debug i opcji register
  hosts: gdynia
  tasks:
    - name: Data
      shell: /usr/bin/date
      register: data_results

    - name: Debug dla data
      debug:
        msg: "{{ data_results }}"
        verbosity: 2

    - name: instalacja pakietu ktorego nie ma
      yum:
        name: scren
      register: yum_register
      ignore_errors: true

    - name: Debug dla yum
      debug:
        msg: "{{ yum_register }}"
        verbosity: 2
```

2. Uruchom playbooka i zauważ, że zadania z modułów **debug** nie wykonały się.

```
[user@warszawa ansible-troubleshooting]$ ansible-playbook
./ansible-troubleshooting.yml

PLAY [Korzystanie z modulu debug i opcji register]
**********************************************************************************************
****************************************

TASK [Gathering Facts]
**********************************************************************************************
*****************************************************************
ok: [gdynia]

TASK [Data]
**********************************************************************************************
***************************************************************************
changed: [gdynia]

TASK [Debug dla data]
**********************************************************************************************
*********************************************************************
skipping: [gdynia]

TASK [instalacja pakietu ktorego nie ma]
**********************************************************************************************
************************************************
fatal: [gdynia]: FAILED! => {"changed": false, "msg": "No package matching 'scren'
found available, installed or updated", "rc": 126, "results": ["No package matching
'scren' found available, installed or updated"]}
...ignoring

TASK [Debug dla yum]
**********************************************************************************************
*********************************************************************
skipping: [gdynia]

PLAY RECAP
**********************************************************************************************
***************************************************************************
gdynia                     : ok=3   changed=1   unreachable=0    failed=0    skipped=2
rescued=0    ignored=1
```

3. Uruchom playbooka z opcjami --step oraz -vv i zaobserwuj różnice

```
[user@warszawa ansible-troubleshooting]$ ansible-playbook --step -vv
./ansible-troubleshooting.yml
ansible-playbook 2.8.4
  config file = /home/user/ansible-troubleshooting/ansible.cfg
  configured module search path = [u'/home/user/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 2.7.5 (default, Aug  7 2019, 00:51:29) [GCC 4.8.5 20150623 (Red
Hat 4.8.5-39)]
Using /home/user/ansible-troubleshooting/ansible.cfg as config file

PLAYBOOK: ansible-troubleshooting.yml
**********************************************************************************************
**************************************************
1 plays in ./ansible-troubleshooting.yml

PLAY [Korzystanie z modulu debug i opcji register]
**********************************************************************************************
****************************************
Perform task: TASK: Gathering Facts (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Gathering Facts (N)o/(y)es/(c)ontinue:
**********************************************************************************************
********************************

TASK [Gathering Facts]
**********************************************************************************************
*********************************************************************
task path: /home/user/ansible-troubleshooting/ansible-troubleshooting.yml:2
ok: [gdynia]
META: ran handlers
Perform task: TASK: Data (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Data (N)o/(y)es/(c)ontinue:
**********************************************************************************************
****************************************

TASK [Data]
```

```
*********************************************************************************************************
*************************************************************************
task path: /home/user/ansible-troubleshooting/ansible-troubleshooting.yml:5
changed: [gdynia] => {"changed": true, "cmd": "/usr/bin/date", "delta":
"0:00:00.002582", "end": "2019-10-12 15:40:46.186678", "rc": 0, "start": "2019-10-12
15:40:46.184096", "stderr": "", "stderr_lines": [], "stdout": "Sat Oct 12 15:40:46 CEST
2019", "stdout_lines": ["Sat Oct 12 15:40:46 CEST 2019"]}
Perform task: TASK: Debug dla data (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Debug dla data (N)o/(y)es/(c)ontinue:
*********************************************************************************************************
*********************************

TASK [Debug dla data]
*********************************************************************************************************
*********************************************************************
task path: /home/user/ansible-troubleshooting/ansible-troubleshooting.yml:9
ok: [gdynia] => {
    "msg": {
        "changed": true,
        "cmd": "/usr/bin/date",
        "delta": "0:00:00.002582",
        "end": "2019-10-12 15:40:46.186678",
        "failed": false,
        "rc": 0,
        "start": "2019-10-12 15:40:46.184096",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "Sat Oct 12 15:40:46 CEST 2019",
        "stdout_lines": [
            "Sat Oct 12 15:40:46 CEST 2019"
        ]
    }
}
Perform task: TASK: instalacja pakietu ktorego nie ma (N)o/(y)es/(c)ontinue: y

Perform task: TASK: instalacja pakietu ktorego nie ma (N)o/(y)es/(c)ontinue:
*********************************************************************************************************
**************

TASK [instalacja pakietu ktorego nie ma]
*********************************************************************************************************
************************************************
```

```
task path: /home/user/ansible-troubleshooting/ansible-troubleshooting.yml:14
fatal: [gdynia]: FAILED! => {"changed": false, "msg": "No package matching 'scren' found available, installed or updated", "rc": 126, "results": ["No package matching 'scren' found available, installed or updated"]}
...ignoring

Perform task: TASK: Debug dla yum (N)o/(y)es/(c)ontinue: y

Perform task: TASK: Debug dla yum (N)o/(y)es/(c)ontinue:
*********************************************************************************************************
**********************************

TASK [Debug dla yum]
*********************************************************************************************************
*************************************************************************
task path: /home/user/ansible-troubleshooting/ansible-troubleshooting.yml:20
ok: [gdynia] => {
    "msg": {
        "changed": false,
        "failed": true,
        "msg": "No package matching 'scren' found available, installed or updated",
        "rc": 126,
        "results": [
            "No package matching 'scren' found available, installed or updated"
        ]
    }
}
META: ran handlers
META: ran handlers

PLAY RECAP
*********************************************************************************************************
*************************************************************************
gdynia                     : ok=5   changed=1   unreachable=0    failed=0    skipped=0
rescued=0    ignored=1
```

Rozdział 9 - Ansible AWX

Ćwiczenie 1 - QUIZ - Ansible AWX

PYTANIE 1:

Z której bazy danych korzysta projekt Ansible AWX?

Odpowiedź A:

postgresql

Odpowiedź B:

mysql

Odpowiedź C:

mongodb

Odpowiedź D:

oracle

PYTANIE 2:

Czy Ansible AWX potrzebuje zainstalowanego Ansible?

Odpowiedź A:

tak

Odpowiedź B:

nie

PYTANIE 3:

Czego nie ma Ansible AWX?

Odpowiedź A:

Zaimplementowanego podejścia RBAC

Odpowiedź B:

Możliwości centralnego logowania

Odpowiedź C:

REST API

Odpowiedź D:

Kreatora do tworzenia playbooków

Odpowiedzi: 1A, 2B, 3D

Rozdział 10 - LAB

LAB: Ćwiczenie końcowe

Kroki wstępne:	Zresetuj maszyny: **gdynia, krakow**
Cel:	Ćwiczenie zawierające różne elementy Ansible takie jak: - Tworzenie inventory - Tworzenie ról - Używanie ról - Używanie różnego rodzaju modułów - Templates - Handlers
Opis:	Stwórz playbooka, który użyje ról do implementacji serwera WWW z PHP z przykładową stroną i zainstaluje serwer bazy danych

1. Informacje ogólne:
    a. Host gdynia powinien być w grupie www
    b. Host krakow powinien być w grupie db
    c. Role powinny być w katalogu “./roles”
    d. Stwórz trzy role: web, db i common
    e. Hosty z grupy db powinny aplikować rolę db
    f. Hosty z grupy www powinny aplikować rolę web
    g. Rola common jest dla wszystkich hostów
2. Rola web:
    a. Ma dwie zmienne wykorzystywane jako tytuł strony WWW i treść
    b. Istnieje handler do restartowania usługi WWW
    c. Domyślna strona WWW ma wyświetlać tytuł, treść i zrzut informacyjny z funkcji phpinfo()
    d. Rola powinna konfigurować autostart i firewalla
3. Rola db:
    a. Instalacja bazy danych
    b. Autostart
    c. Firewall
    d. Definicja portu i nazwa usługi powinny być definiowane w zmiennych grupowych dla db

4. Rola common aktualizuje wszystkie systemy do najnowszych wersji pakietów

LAB: Ćwiczenie końcowe - rozwiązanie

```
[user@warszawa ansible-lab]$ tree ./
./
├── ansible.cfg
├── group_vars
│   └── db
├── inventory
├── roles
│   ├── common
│   │   ├── defaults
│   │   │   └── main.yml
│   │   ├── files
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── meta
│   │   │   └── main.yml
│   │   ├── README.md
│   │   ├── tasks
│   │   │   └── main.yml
│   │   ├── templates
│   │   ├── tests
│   │   │   ├── inventory
│   │   │   └── test.yml
│   │   └── vars
│   │       └── main.yml
│   ├── db
│   │   ├── defaults
│   │   │   └── main.yml
│   │   ├── files
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── meta
│   │   │   └── main.yml
│   │   ├── README.md
│   │   ├── tasks
│   │   │   ├── install.yml
│   │   │   ├── main.yml
│   │   │   └── service.yml
│   │   ├── templates
│   │   ├── tests
```

```
│   │   │   ├── inventory
│   │   │   └── test.yml
│   │   └── vars
│   │       └── main.yml
│   └── web
│       ├── defaults
│       │   └── main.yml
│       ├── files
│       ├── handlers
│       │   └── main.yml
│       ├── meta
│       │   └── main.yml
│       ├── README.md
│       ├── tasks
│       │   ├── configure.yml
│       │   ├── install.yml
│       │   ├── main.yml
│       │   └── service.yml
│       ├── templates
│       │   └── index.html.j2
│       ├── tests
│       │   ├── inventory
│       │   └── test.yml
│       └── vars
│           └── main.yml
└── site.yml

29 directories, 35 files

[user@warszawa ansible-lab]$ cat inventory
[www]
gdynia

[db]
krakow
[user@warszawa ansible-lab]$ cat ansible.cfg
[defaults]
inventory=./inventory
remote_user=devops
roles_path=./roles
[privilege_escalation]
become=True
become_method=sudo
```

```
become_user=root

[user@warszawa ansible-lab]$ cat site.yml
---
- name: Podstawowa konfiguracja na wszystkich hostach
  hosts: all
  roles:
    - common

- name: Implementacja DB
  hosts:
    - db
  roles:
    - db

- name: Implementacja serwera WWW
  hosts:
    - www
  roles:
    - web

[user@warszawa ansible-lab]$ cat group_vars/db
---
# Zmienne dla bazy danych
db_service: mariadb
db_port: 3306/tcp

[user@warszawa ansible-lab]$ cat  roles/common/tasks/main.yml
---
# tasks file for common
- name: Aktualizacja pakietow
  yum:
          state: latest
          name: '*'

[user@warszawa ansible-lab]$ cat roles/db/defaults/main.yml
---
# defaults file for db
db_service: mysqld
db_port: 3306/tcp

[user@warszawa ansible-lab]$ cat roles/db/tasks/install.yml
---
```

```
- name: mariadb packages installation
  yum:
        name:
        - mariadb
        - mariadb-server
        state: latest

[user@warszawa ansible-lab]$ cat roles/db/tasks/service.yml
---
- name: mysql autostart and start
  service:
        name: "{{ db_service }}"
        state: started
        enabled: true

- name: firewalld for httpd
  firewalld:
        port: "{{ db_port }}"
        state: enabled
        permanent: yes
        immediate: yes

[user@warszawa ansible-lab]$ cat roles/db/tasks/main.yml
---
# tasks file for db
- import_tasks: install.yml
- import_tasks: service.yml

[user@warszawa ansible-lab]$ cat roles/web/defaults/main.yml
---
# defaults file for web
tytul: Domyslny tytul
napis: Domyslny napis

[user@warszawa ansible-lab]$ cat roles/web/handlers/main.yml
---
# handlers file for server_www

- name: httpd restart
  service:
    name: httpd
    state: restarted
```

```
[user@warszawa ansible-lab]$ cat roles/web/templates/index.html.j2
<!doctype html>
<html>
 <head>
  <title>{{ tytul }}</title>
 </head>
 <body>
    <p>{{ napis }}</p>
    <?php phpinfo() ?>
 </body>
</html>

[user@warszawa ansible-lab]$ cat roles/web/tasks/main.yml
---
# tasks file for server_www
- include_tasks: install.yml
- include_tasks: configure.yml
- include_tasks: service.yml

[user@warszawa ansible-lab]$ cat roles/web/tasks/install.yml
---
- name: install httpd package
  yum:
        name: httpd
        state: latest

- name: install php package
  yum:
        name: mod_php
        state: latest
  notify: httpd restart
[user@warszawa ansible-lab]$ cat roles/web/tasks/configure.yml
---
- name: index.php
  template:
        src: index.html.j2
        dest: /var/www/html/index.html

- name: httpd config to use php inside html files
  copy:
        content: "AddHandler application/x-httpd-php .html"
        dest: /etc/httpd/conf.d/custom.conf
  notify: httpd restart
```

```
[user@warszawa ansible-lab]$ cat roles/web/tasks/service.yml
---
- name: httpd autostart and start
  service:
        name: httpd
        state: started
        enabled: true

- name: firewalld for httpd
  firewalld:
        service: http
        state: enabled
        permanent: yes
```

LAB: Ćwiczenie końcowe 2

Kroki wstępne:	Zresetuj maszyny: **gdynia, krakow**
Cel:	Ćwiczenie zawierające różne elementy Ansible
Opis:	Stwórz playbooka, który rekonfiguruje/zabezpiecza usługę SSH na maszynie krakow.

1. Informacje ogólne:
    a. SSHD ma nasłuchiwać na porcie 12345/tcp (pamiętaj o firewall, selinux)
    b. Komunikat “Serwer prywatny” ma pojawiać się przed procesem uwierzytelniania do usługi
    c. Logowanie na roota jest niemożliwe
2. Uwaga: Zauważ co się stanie jeśli play zadziała - czy będziesz mógł dalej zarządzać hostem **krakow** za pomocą ansible? Po zaimplementowaniu play’a popraw tak plik inventory, aby wszystko dalej działało

LAB: Ćwiczenie końcowe 2 - rozwiązanie

```
[user@warszawa ansible-lab2]$ cat sshd.yml
---
- name: sshd tuning
  hosts: krakow
  vars:
    port: 12345
    message_file: "/etc/ssh/ssh_message.txt"
    message: "To jest prywatny serwer\n"
    sshd_config: "/etc/ssh/sshd_config"
  tasks:

  - name: port in config file
    lineinfile:
      dest: "{{ sshd_config }}"
      regexp: "^Port"
      line: "Port {{ port }}"

  - name: port selinux
    seport:
      ports: "{{ port }}"
      proto: "tcp"
      setype: "ssh_port_t"
      state: "present"

  - name: port firewalld
    firewalld:
      port: "{{ port}}/tcp"
      permanent: yes
      state: enabled
      immediate: true

  - name: create ssh_message.txt
    copy:
      dest: "{{ message_file }}"
      content: "{{ message }}"

  - name: banner in sshd
    lineinfile:
      dest: "{{ sshd_config }}"
```

```
    regexp: "^Banner"
    line: "Banner {{ message_file }}"

- name: blokada roota
  lineinfile:
    dest: "{{ sshd_config }}"
    regexp: "^PermitRootLogin"
    line: "PermitRootLogin no"

- name: sshd restart
  service:
    name: sshd
    state: restarted
```

Inventory po zmianach:

```
[user@warszawa ansible-lab2]$ cat inventory
gdynia
krakow ansible_port=12345
```