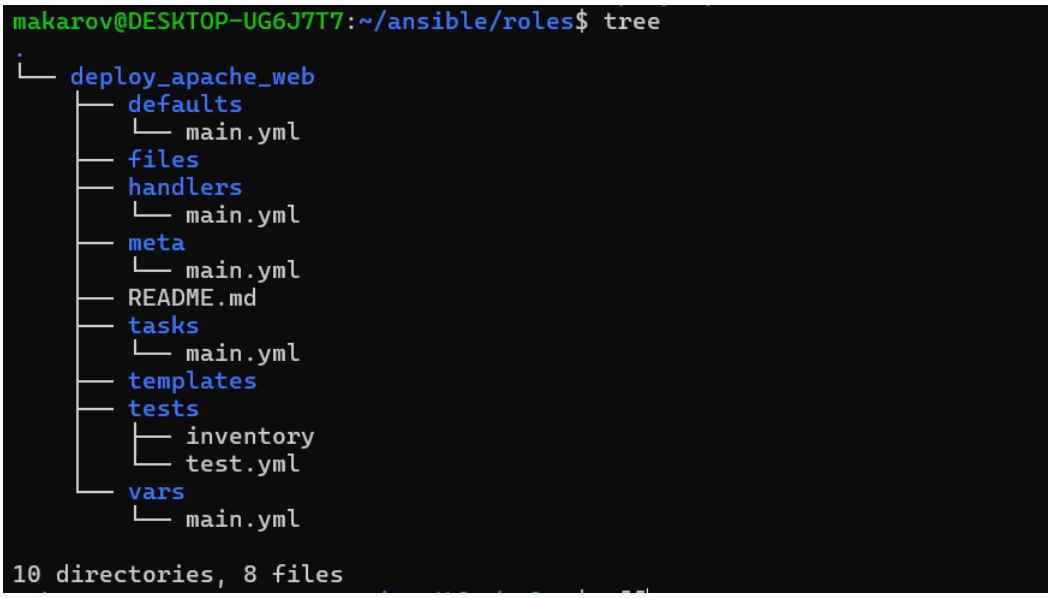
Homework_Lesson20_Ansible_3

Цель: получить практический опыт работы с Ansible-ролями
Задание 1: просмотрите и повторите все манипуляции, указанные в видеоуроке по ссылке https://www.youtube.com/watch?v=9pHMZnb3JDQ
В данном ролике рассматриваются настройки Ansible-ролей.
Задание 2: (опционально, на выбор):
● Роль для настройки базы данных. Напишите роль, которая устанавливает и настраивает базу данных (например, MySQL или PostgreSQL). Роль должна принимать переменные для конфигурирования базы данных, такие как пароль для root-пользователя и порт, на котором база данных будет слушать.

Выполнение 1-ого задания:

Создание ролей в Ansible:

$ mkdir roles	\\ создаем директорию roles
$ ansible-galaxy init deploy_apache_web	\\ создание роли с названием deploy_apache_web
$ tree	\\ показывает структуру директории roles



$ nano deploy_apache_web/defaults/main.yml

Далее, мы наш playbook для настройки/установки чего либо превращаем в роль, т.е. содержимое playbook переносится в роль, типо tasks playbook переносится в tasks -> main.yml нашей созданной роли.

$ ansible-playbook playbook6.yml

Выполнение 2-ого задания:

Создаем роль для настройки базы данных Postgresql.

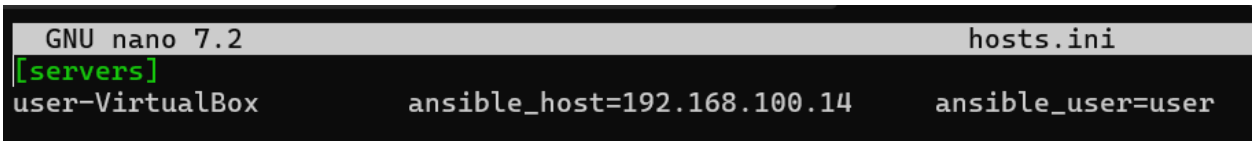
На WSL выполняем следующие действия:

$ ssh-keygen -t rsa -b 4096	\\ создание ключа
$ ssh-copy-id user@192.168.100.14	\\ копируем наш ключ на ВМ

Создаем отдельную директорию ansible_postgres.

$ mkdir ansible_postgres
$ cd ./ansible_postgres

Создаем файлы hosts.ini и конфигурационный файл ansible.cfg.
$ touch hosts.ini
$ touch ansible.cfg
$ nano hosts.ini

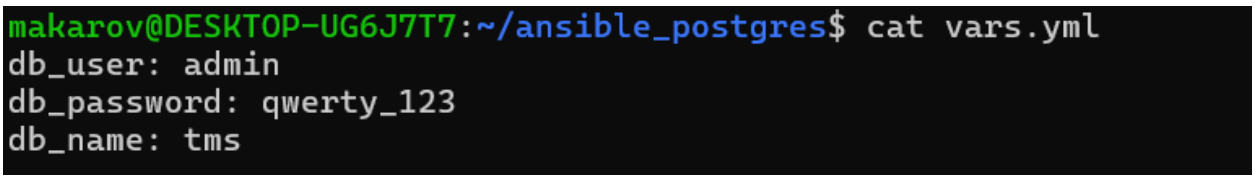


В файле hosts.ini прописываем наши сервера на котором будем устанавливать и настраивать Postgresql

$ nano ansible.cfg

Указываем файл hosts.ini

$ touch vars.yml \\ создаем файл переменных
$ nano vars.yml \\ добавляем пользователя БД, имя БД и пароль



Данные переменные, нам понадобятся для создания пользователя и БД в PostgreSQL

Создаем playbook для установки/настройки Postgres. Сначала впишем несколько строк для установки PostgreSQL и попробуем проверить:

$ nano install_pg.yml

Добавляем следующие строки:

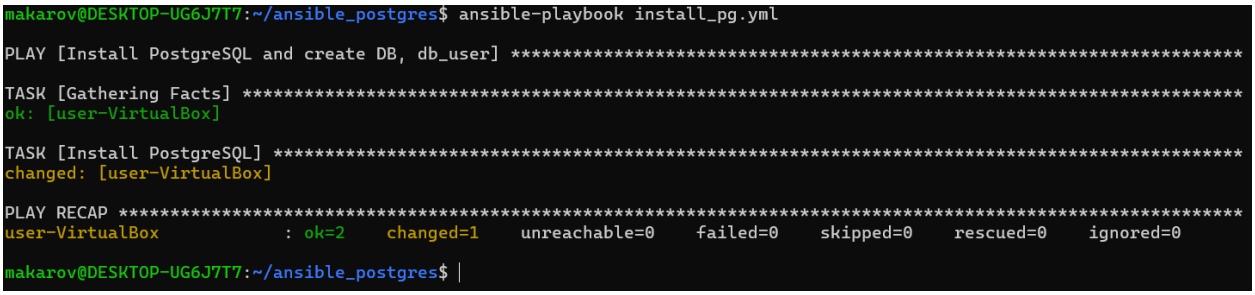
```
---
- name: Install PostgreSQL and create DB, db_user
  hosts: servers
  become: yes
  vars_files:
  - vars.yml

  pre_tasks:
    - name: Install PostgreSQL
      ansible.builtin.apt:
```

name: postgresql
state: latest
update_cache: true

Пробуем запустить playbook:

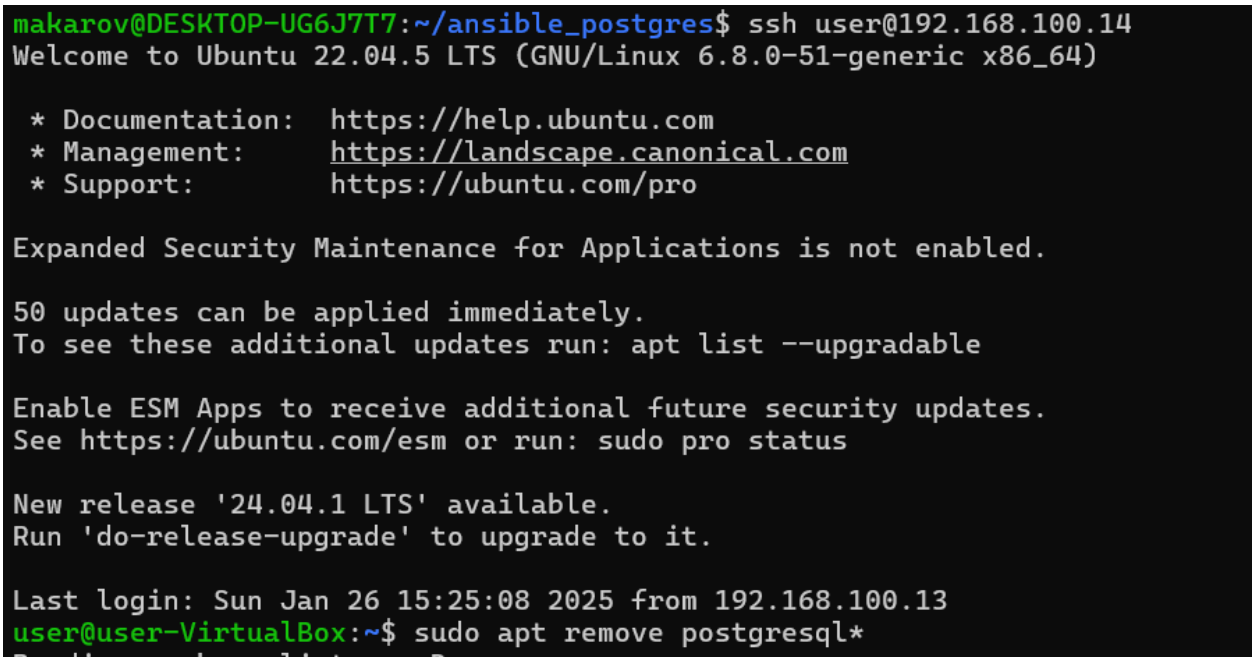
$ ansible-playbook install_pg.yml



Установка PostgreSQL успешна

Заходим удаленно по SSH на нашу ВМ:

$ ssh user@192.168.100.14
$ sudo apt remove postgresql* \\ удаляем PostgreSQL



$ exit \\ возвращаемся к нашей WSL (Ansible-Master)

Добавляем в playbook поднятие сервиса postgresql и добавляем в автозагрузку:

```
tasks:
  - name: Start and enable services
    service:
      name: postgresql
      state: started          \\ Запускаем сервис
      enabled: yes            \\ добавляем в автозагрузку (enabled)
```

Переходим к наполнению нашего playbook (install_pg.yml). Нам нужно, чтоб после установки СУБД PostgreSQL, у нас залогинилось пользователем postgres (который по умолчанию появляется, после установки PostgreSQL) и указать: создание базы данных tms и юзера admin. Также можно будет добавить строки на загрузку каких-нибудь данных в созданную БД tms.

Добавляем создание БД и пользователя, которых мы указали в vals.yml, в файл install_pg.yml:

```
 - name: Create database
   postgresql_db:
     state: present
     name: "{{  db_name  }}"
   become: yes
   become_user: postgres

 - name: Create db user
   postgresql_user:
     state: present
     name: "{{ db_user }}"
     password: "{{ db_password }}"
   become: yes
   become_user: postgres
```

Пробуем для тестирования запустить сборку playbook (install_pg.yml):

$ ansible-playbook install_pg.yml

Выдает ошибку, что нехватает библиотек.

```
makarov@DESKTOP-UG6J7T7:~/ansible_postgres$ ansible-playbook install_pg.yml

PLAY [Install PostgreSQL and create DB, db_user] ********************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [user-VirtualBox]

TASK [Install PostgreSQL] *******************************************************************************************
changed: [user-VirtualBox]

TASK [Start and enable services] ************************************************************************************
ok: [user-VirtualBox]

TASK [Create database] **********************************************************************************************
fatal: [user-VirtualBox]: FAILED! => {"changed": false, "msg": "Failed to import the required Python library (psycopg2)
on user-VirtualBox's Python /usr/bin/python3. Please read the module documentation and install it in the appropriate loc
ation. If the required library is installed, but Ansible is using the wrong Python interpreter, please consult the docum
entation on ansible_python_interpreter"}

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

Ошибки из-за нехватки библиотеки Python psycopg2 на стороне ВМ

Покопавшись в инете, необходимо в наш playbook добавить задачу на установку пакетов python3-psycopg2. Данная библиотека позволяет использовать модули PostgreSQL, т.е. postgresql_db, postgresql_user, postgresql_pg_hba и postgresql_privs. Добавим его в pre_tasks где указана установка пакетов PostgreSQL:

```
    - name: Install Python packages
      apt:
            name: python3-psycopg2
            state: present
```

```
makarov@DESKTOP-UG6J7T7:~/ansible_postgres$ cat install_pg.yml
---
- name: Install PostgreSQL and create DB, db_user
  hosts: servers
  become: yes
  vars_files:
  - vars.yml

  pre_tasks:
    - name: Install PostgreSQL
      ansible.builtin.apt:
        name: postgresql
        state: latest
        update_cache: true

    - name: Install Python packages
      apt:
        name: python3-psycopg2
        state: present
```

Поправленный вариант install_pg.yml

Пробуем повторный запуск:

$ ansible-playbook install_pg.yml

```
makarov@DESKTOP-UG6J7T7:~/ansible_postgres$ ansible-playbook install_pg.yml

PLAY [Install PostgreSQL and create DB, db_user] ********************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [user-VirtualBox]

TASK [Install PostgreSQL] *******************************************************************************************
ok: [user-VirtualBox]

TASK [Install Python packages] **************************************************************************************
changed: [user-VirtualBox]

TASK [Start and enable services] ************************************************************************************
ok: [user-VirtualBox]

TASK [Create database] **********************************************************************************************
changed: [user-VirtualBox]

TASK [Create db user] ***********************************************************************************************
changed: [user-VirtualBox]

PLAY RECAP **********************************************************************************************************
user-VirtualBox            : ok=6    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Успешный запуск, после добавления установки пакетов

На ВМ user-VirtualBox(192.168.100.14) появилась БД tms и пользователь admin:

```
                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 tms       | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(4 rows)
```

```
postgres=# \du
                             List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 admin     |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

Добавляем также привилегии пользователю admin для БД tms и к примеру выполнение SQL-скрипта на добавление таблицы с сообщением:

Добавляем привилегии пользователю (admin):

```
- name: Grant user db_user to db
    postgresql_privs:
      type: database
      database: "{{ db_name }}"
      roles: "{{ db_user }}"
      grant_option: no
      privs: all
    become: yes
    become_user: postgres
```

Перед добавление задачи в Ansible, создаем файл CREATE_TABLE_ANSIBLE.sql и переносим его на ВМ либо даем доступ к этому файлу:

```
CREATE TABLE IF NOT EXISTS test (
  message varchar(255) NOT NULL
  );
INSERT INTO test(message) VALUES('Ansible is fun by tms');
ALTER TABLE test OWNER TO "admin";
```

Добавляем саму задачу в наш playbook:

```
   - name: Add data to our db
     become: true
     become_user: postgres
     shell: "psql -d {{ db_name }} -f /ALL_FILES/CREATE_TABLE_ANSIBLE.sql"
```

Можно добавить смену порта, по умолчанию в PostgreSQL порт 5432, попробуем поменять его на 5433. Будем использовать модуль postgresql_set. Эту задачу нужно добавить сразу после задачи установки самого PostgresSQL. Т.е. мы её добавляем в pre_tasks:.

```
        - name: Change listen port 5433 to PG
                postgresql_set:
                name: port
                value: 5433
                state: present
                db: “{{ db_name }}”
                notify: Restart PostgreSQL
```

Добавим notify который используется для связи задач с обработчиками handlers, для запуска определенных действий, в нашей ситуации, это перезапуск службы PostgreSQL, после глобальных настроек, типа смены портов:

```
        handlers:
         - name: Restart PostgreSQL
                systemd:
                name: postgresql
                state: restarted
```