Настройка SSL сертификатов

OpenSSH

(1) Enable SSH connection via db.Dockerfile

> составляем докерфайл, чтобы при загрузке образа запускалась возможность создавать ssh соединение напрямую с контейнером

db.Dockerfile

```
FROM postgres

ARG ssh_user
ARG ssh_password

RUN useradd -ms /bin/bash $ssh_user
RUN echo "$ssh_user:$ssh_password" | chpasswd

RUN apt update && apt -y install openssh-server
RUN mkdir -p /var/run/sshd
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
CMD [ "/usr/sbin/sshd", "-D"]
```

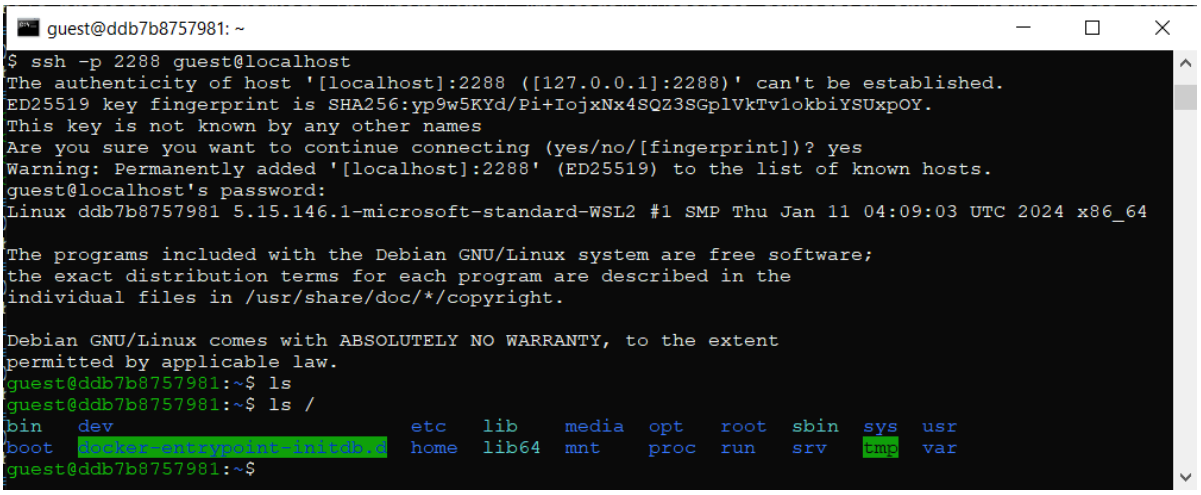
(2) Open port in docker compose

> открываем соответствующий порт в контейнере, остальные порты автоматически недоступны по политикам доступа Docker
> создаем специального пользователя для защищенной работы

```
  db:
    build:
      context: .
      dockerfile: db.Dockerfile
      args:
        ssh_user: guest
        ssh_password: ${ROOT_PASS}
    ports:
      - "5433:5432"
      - "2288:22"
    environment:
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_DB: ${POSTGRES_DB}
      PGDATA: ${POSTGRES_PGDATA}
    volumes:
      - ./db/data:/var/lib/postgresql/data
      - ./db/initdb.d:/docker-entrypoint-initdb.d
    networks:
      localnet:
        ipv4_address: ${DB_ADDRESS}
```

(3) Check ssh connection

> с клиентского окна подключаемся к сети контейнера через ssh



Настройка SSL

(4) Создаем pg_hba.conf

```
102  #
103  # If you want to allow non-local connections, you need to add more
104  # "host" records.  In that case you will also need to make PostgreSQL
105  # listen on a non-local interface via the listen_addresses
106  # configuration parameter, or via the -i or -h command line switches.
107
108  # CAUTION: Configuring the system for local "trust" authentication
109  # allows any local user to connect as any PostgreSQL user, including
110  # the database superuser.  If you do not trust all your local users,
111  # use another authentication method.
112
113
114  # TYPE  DATABASE        USER            ADDRESS                 METHOD
115
116  local   all             all                                     trust
117  hostssl all             all             all                     cert
118  host    all             all             all                     reject
119
120  # Allow replication connections from localhost, by a user with the
121  # replication privilege.
122  local   replication     all                                     trust
123  host    replication     all             127.0.0.1/32            trust
124  host    replication     all             ::1/128                 trust
125
126  # host all all all scram-sha-256
```

(5) Создаем sh скрипт для генерации ключей сервера и клента

> генерация ключей сервера

openssl > postgres > $ generate_server_keys.sh

```
#!/bin/bash

KEY_SECRET=2123wdqwid2e98qdh3iud3
openssl req -new -text -passout pass:$KEY_SECRET -subj /CN=localhost -keyout privkey.pem -out server.req
openssl rsa -in privkey.pem -passin pass:$KEY_SECRET -out server.key
openssl req -x509 -in server.req -text -key server.key -out server.crt
```

> генерация ключей клиента

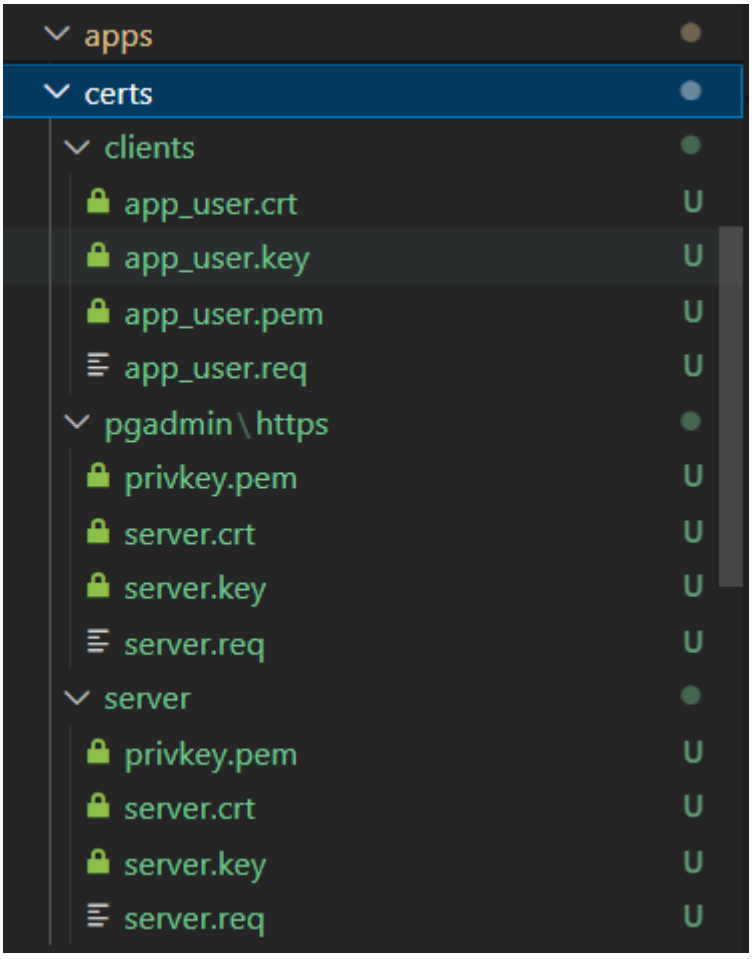
openssl > postgres > $ generate_appuser_keys.sh

```
#!/bin/bash

KEY_SECRET=12345678
USERNAME=app_user
openssl req -new -text -passout pass:$KEY_SECRET -subj /CN=app_user -keyout $USERNAME.pem -out $USERNAME.req
openssl rsa -in $USERNAME.pem -passin pass:$KEY_SECRET -out $USERNAME.key
openssl req -x509 -in $USERNAME.req -text -key $USERNAME.key -out $USERNAME.crt
```

> [Важно]
> для генерации https соединения можно воспользоваться файлом для сервера

(6) Генерируем ключи



(7) Заполняем конфиги docker-compose

(7.1) для сервера

```yaml
db:
  build:
    context: .
    dockerfile: db.Dockerfile
    args:
      ssh_user: guest
      ssh_password: ${ROOT_PASS}
  ports:
    - "5433:5432"
    - "2288:22"
  environment:
    POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    POSTGRES_USER: ${POSTGRES_USER}
    POSTGRES_DB: ${POSTGRES_DB}
    PGDATA: ${POSTGRES_PGDATA}
  volumes:
    - ./db/data:/var/lib/postgresql/data
    - ./db/initdb.d:/docker-entrypoint-initdb.d
    - ./config/pg/pg_hba.conf:/var/lib/postgresql/data/pg_hba.conf:ro
  networks:
    localnet:
      ipv4_address: ${DB_ADDRESS}
```

(7.2) для админки

```yaml
  db-ui:
    image: dpage/pgadmin4
    ports:
      - "5050:80"
      - "5053:443"
    environment:
      PGADMIN_DEFAULT_EMAIL: ${PGADMIN_EMAIL}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_PWD}
      PGADMIN_LISTEN_ADDRESS: '0.0.0.0'
      PGADMIN_ENABLE_TLS: true
      PGADMIN_CONFIG_SERVER_MODE: "False"
      PGADMIN_CONFIG_MASTER_PASSWORD_REQUIRED: "False"
    volumes:
      - ./db/pgadmin:/var/lib/pgadmin
      - ./config/pgadmin/servers.json:/pgadmin4/servers.json
      # HTTPS certificates mapping. NB! server.crt -> server.cert, server.key -> server.key
      - ./certs/pgadmin/https/server.crt:/certs/server.cert:ro
      - ./certs/pgadmin/https/server.key:/certs/server.key:ro
      # TLS superuser client certificates mapping
      - ./certs/clients/app_user.crt:/certs/app_user.crt:ro
      - ./certs/clients/app_user.key:/certs/app_user.key:ro
    restart: unless-stopped
    deploy:
      resources:
        limits:
          cpus: '0.5'
          memory: 1G
    networks:
      localnet:
        ipv4_address: ${ADMIN_ADDRESS}

```

(8) Проверка ssl на сервере

```
$ docker exec -it 2c8 psql -U app_user "sslmode=require dbname=wirehouse_db"
psql (16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

wirehouse_db=# _
```

(9) Админ панель с самоподписным сертификатом