

???????

- [Контейнеризация приложения из git](#)
- [Запуск из консоли и простые dockerfile](#)
- [3 образа](#)
- [Postgresql](#)
- [Nginx](#)
- [Файловый сервер](#)
- [Mariadb](#)

??

?? git

Создаем папку app и переходим в нее.

Клонируем git

```
git clone https://github.com/sudaka/irksecrets.git
```

Создаем dockerfile

```
FROM ubuntu:latest
LABEL maintainer="..."
RUN apt-get update
RUN apt install -y python3 python3-pip uvicorn
RUN mkdir /var/www
WORKDIR /var/www
COPY ./irksecrets /var/www
RUN python3 -m pip install -r requirements.txt
EXPOSE 8000
ENTRYPOINT ["uvicorn", "irksecrets:app", "--host", "0.0.0.0", "--port", "8000"]
```

Создаем образ

```
docker build -t irkscweb:2.0 .
```

Создаем контейнер

```
docker run -d --name fa2 -p 8000:8000 irkscweb:2.0
```

Сейчас контейнер сервиса должен заработать, по адресу 127.0.0.1/docs должна быть страница сервиса. Останавливаем сервис.

Информация по официальному образу postgres:

```
POSTGRES_PASSWORD=mysecretpassword
POSTGRES_USER
POSTGRES_DB #при отсутствии будет создана БД
POSTGRES_HOST_AUTH_METHOD
```

все файлы .sql .sh в папке docker-entrypoint-initdb.d исполняются при инициализации БД

Поднимаемся на один уровень и создаем папку db.

Создаем скрипт настройки

```
create role irksecrets with login superuser;
alter role irksecrets with encrypted password 'Password';
create table secrets (chash char(64) primary key, enctext bytea);
alter database irksecrets owner to irksecrets;
```

```
FROM postgres
LABEL maintainer="sa@yandex.ru"
ENV POSTGRES_PASSWORD 123458
ENV POSTGRES_HOST_AUTH_METHOD md5
ENV POSTGRES_DB: irksecrets
COPY *.sql /docker-entrypoint-initdb.d/ #инит скрипты, при наличии БД не запускаются
EXPOSE 5432
```

Создаем compose файл. Директории app db рядом с .yaml файлом

```
services
  webapp:
    build: app/.
    ports:
      - target: 8000
        published: 8000
    networks:
      - isnet
  dbhost:
    build: db/.
    environment:
      - POSTGRES_PASSWORD=Password
      - POSTGRES_HOST_AUTH_METHOD=md5
      - POSTGRES_DB=irksecrets
    ports:
      - target: 5432
        published: 5432
    networks:
      - isnet
networks:
  networks:
```

isnet:

?????? ?? ??????? ? ????????

dockerfile

Запуск bash в контейнере:

```
docker run -i -t debian /bin/bash
```

Удаление всех остановленных контейнеров

```
docker rm -v $(docker ps -aq -f status=exited)
```

Создание контейнера, установка доп. приложения и запуск

```
docker run -it --name cowsay --hostname cowsay debian bash
root@cowsay:/# apt-get update
root@cowsay:/# apt-get install -y cowsay fortune
root@cowsay:/# exit
docker commit cowsay test/cowsayimage
docker run test/cowsayimage /usr/games/cowsay "Moo"
```

Создание образа из docker файла

Dockerfile:

```
FROM debian:wheezy
RUN apt-get update && apt-get install -y cowsay fortune
ENTRYPOINT ["/usr/games/cowsay"]
```

Создание образа из dockerfile

```
docker build -t test/cowsay-dockerfile .
```

Пример скрипта вызова разных приложений при установленной точке входа

```
nano entrypoint.sh
```

```
#!/bin/bash
if [ $# -eq 0 ]; then
```

```
/usr/games/fortune | /usr/games/cowsay  
else  
/usr/games/cowsay "$@"
```

```
chmod +x entrypoint.sh
```

DockerFile:

```
FROM debian  
RUN apt-get update && apt-get install -y cowsay fortune  
COPY entrypoint.sh /  
ENTRYPOINT ["/entrtypoint.sh"]
```

3 ???????

```
services:
  identitydock:
    build: . #build ссылается на docker file. Либо build, либо image.
    ports:
      - "5000:5000"
    environment:
      ENV: DEV
    volumes:
      - ./app:/app # старое описание
      - type: volume
        source: counter-vol
  target: /app
  links:
    - dnmonster
    - redis

dnmonster:
  image: amouat/dnmonster

redis:
  image: redis
volumes:
  counter-vol:
```

Postgresql

[Основа инструкции на хабре](#)

Минимальный compose файл с возможностью подключиться извне, логином и паролем:

```
services:
  postgres:
    image: postgres:16.3
    environment:
      POSTGRES_DB: "testdb"
      POSTGRES_USER: "testuser"
      POSTGRES_PASSWORD: "testpass"
    ports:
      - "5432:5432"
```

Точка входа для инициализации базы данных: **docker-entrypoint-initdb.d** Все *.sql или *.sh файлы в этом каталоге - скрипты для инициализации БД. Детали использования:

1. если БД уже была проинициализирована ранее, то никакие изменения к ней применяться не будут;
2. если в каталоге присутствует несколько файлов, то они будут отсортированы по имени с использованием текущей локали (по умолчанию en_US.utf8).

```
services:
  postgres:
    image: postgres:16.3
    environment:
      POSTGRES_DB: "testdb"
      POSTGRES_USER: "testuser"
      POSTGRES_PASSWORD: "testpass"
    volumes:
      - ./docker-entrypoint-initdb.d
    ports:
      - "5432:5432"
```

Для постоянного размещения БД нужно подмонтировать соответствующий каталог (куда будут сохраняться данные) в контейнер и при необходимости переопределить переменную окружения **PGDATA**

```
services:
  postgres:
    image: postgres:16.3
    environment:
      POSTGRES_DB: "testdb"
      POSTGRES_USER: "testuser"
      POSTGRES_PASSWORD: "testpass"
      PGDATA: "/var/lib/postgresql/data/pgdata"
    volumes:
      - ./docker-entrypoint-initdb.d
      - mydata:/var/lib/postgresql/data
    ports:
      - "5432:5432"
  volumes:
    mydata:
```

Подключение к базе через psql

```
psql -d testdb -U testuser -W -h 127.0.0.1 -p 5430
```

Nginx

Создать директорию nginx_config.conf и внутри файл python_microservices

```
server {  
    listen 8080;  
  
    location /api/firstendpoint {  
        proxy_pass http://firstendpoint:8000/api/firstendpoint;  
    }  
  
    location /api/secondendpoint {  
        proxy_pass http://secondendpoint:8000/api/secondendpoint;  
    }  
}
```

Compose:

```
version: '3.7'  
  
services:  
  nginx:  
    image: nginx:latest  
    ports:  
      - "8080:8080"  
    volumes:  
      - ./nginx_config.conf:/etc/nginx/conf.d/default.conf
```

[Источник](#)

NGINX reverse proxy с https терминацией

Обслуживает несколько доменов, для одного из доменов путь /auth ведет на отдельный сервер

Структура проекта:

```
>certs  
.env
```

```
docker-compose.yaml
nginx.conf.template
```

Размещение статических файлов

```
location /static {
    alias /app/static;
    # Опционально: настройки для статики
    expires 30d;
    add_header Cache-Control "public, immutable";
}
```

Директория certs: сертификаты в формате <domain name>.cert и <domain name>.key

.env

```
# Backend сервисы
WOOD_BACKEND_IP=192.168.1.194
WOOD_BACKEND_PORT=8000
HUB_BACKEND_IP=192.168.1.194
HUB_BACKEND_PORT=8021
HUB_BACKEND_WEB_PORT=8020

WOOD_AUTH_IP=192.168.1.194
WOOD_AUTH_PORT=8001

# Домены
HUB_DOMAIN=hub.bobrobotirk.ru
HUB_WEB_DOMAIN=hubui.bobrobotirk.ru
WOOD_DOMAIN=wood.bobrobotirk.ru
```

nginx.conf.template

```
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    include mime.types;
```

```
default_type application/octet-stream;
sendfile on;
keepalive_timeout 65;

# SSL-настройки
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384';
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;

# Upstreams
upstream hub_backend {
    server ${HUB_BACKEND_IP}:${HUB_BACKEND_PORT};
}

upstream hub_backend_web {
    server ${HUB_BACKEND_IP}:${HUB_BACKEND_WEB_PORT};
}

upstream wood_backend {
    server ${WOOD_BACKEND_IP}:${WOOD_BACKEND_PORT};
}

upstream wood_auth_backend {
    server ${WOOD_AUTH_IP}:${WOOD_AUTH_PORT};
}

# HTTP → HTTPS редирект
server {
    listen 80;
    server_name ${HUB_DOMAIN} ${WOOD_DOMAIN};
    return 301 https://$host$request_uri;
}

# Конфиг для hub.bobrobotirk.ru
server {
    listen 443 ssl;
    server_name ${HUB_DOMAIN};
```

```
ssl_certificate /etc/nginx/certs/${HUB_DOMAIN}.cert;
ssl_certificate_key /etc/nginx/certs/${HUB_DOMAIN}.key;

location / {
    proxy_pass http://hub_backend;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}

server {
    listen 443 ssl;
    server_name ${HUB_WEB_DOMAIN};

    ssl_certificate /etc/nginx/certs/${HUB_WEB_DOMAIN}.cert;
    ssl_certificate_key /etc/nginx/certs/${HUB_WEB_DOMAIN}.key;

    location / {
        proxy_pass http://hub_backend_web;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

# Конфиг для wood.bobrobotirk.ru
server {
    listen 443 ssl;
    server_name ${WOOD_DOMAIN};

    ssl_certificate /etc/nginx/certs/${WOOD_DOMAIN}.cert;
    ssl_certificate_key /etc/nginx/certs/${WOOD_DOMAIN}.key;

    location /auth {
        proxy_pass http://wood_auth_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

```
location / {
    proxy_pass http://wood_backend;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
}
```

Алгоритм добавления еще одного endpoint:

Файл .env Константы, связанные с endpoint

```
ELSE_BACKEND_IP=192.168.1.198
ELSE_BACKEND_PORT=8000
```

Файл docker-compose .yaml

Добавляем эти константы в envsubst для машины nginx

```
envsubst '${ELSE_BACKEND_IP} ${ELSE_BACKEND_PORT} ...'
```

И добавляем еще один сервис

```
else-backend:
  image: else-backend:0.1
  container_name: else-backend
  #restart: unless-stopped # Автоперезапуск при ошибках
  ports:
    - "${ELSE_BACKEND_PORT}:8000" # host:container
  environment:
    - APP_ENV=production
    - LOG_LEVEL=info
    - WORKERS_PER_CORE=1 # Оптимально для CPU-bound приложений
    - MAX_WORKERS=4 # Лимит на случай автоскейлинга
    - TIMEOUT=120 # Макс. время выполнения запроса (сек)
    - KEEP_ALIVE=5 # Поддержка keep-alive соединений
  command: sh -c "uvicorn main:app --host=0.0.0.0 --port=8000 --reload"
  volumes:
    - ./backend/app:/app
```

В файле nginx.conf.template добавляем два раздела

```

...
# Upstreams
upstream else-backend {
    server ${ELSE_BACKEND_IP}:${ELSE_BACKEND_PORT};
}
...
location /else {
    proxy_pass http://else-backend;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

```

docker-compose.yml

```

services:
  nginx-proxy:
    image: nginx:latest
    container_name: nginx-proxy
    hostname: nginx-proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf.template:/etc/nginx/templates/nginx.conf.template
      - ./certs:/etc/nginx/certs
    env_file:
      - .env # Подключаем переменные из файла
    command: >
      /bin/sh -c "
        envsubst '${HUB_BACKEND_IP} ${HUB_BACKEND_PORT} ${HUB_BACKEND_WEB_PORT}
        ${WOOD_BACKEND_IP}
        ${WOOD_BACKEND_PORT} ${WOOD_AUTH_IP} ${WOOD_AUTH_PORT}
        ${HUB_DOMAIN} ${HUB_WEB_DOMAIN} ${WOOD_DOMAIN}'
        < /etc/nginx/templates/nginx.conf.template
        > /etc/nginx/nginx.conf
        && nginx -g 'daemon off;'
      "
    restart: no

```


????????? ????????

Один из самых простых S3-совместимых серверов - minio ([официальный сайт](#)) Еще есть FreeNAS.

Запуск сервера

Compose файл для тестов:

```
services:
  minio:
    image: minio/minio:latest
    container_name: minio
    restart: unless-stopped
    volumes:
      - minio-storage:/data
      - minio-config:/root/.minio
    environment:
      - MINIO_ROOT_USER=miniuser
      - MINIO_ROOT_PASSWORD=admin123
    command: server /data --console-address ":9001"
    ports:
      - "9000:9000"
      - "9001:9001"
volumes:
  minio-storage:
  minio-config:
```

Запускает сервер API на 9000 порту, web интерфейс управления на 9001

[+ сертификаты](#)

[Тоже интересно](#)

Настройка bucket

Создается группа с правом readwrite, создается пользователь и привязывается к группе. Создается bucket.

Из-под пользователя создается ключ доступа.

Mariadb

Установка клиента mariadb

```
sudo apt install mariadb-client
mysql -u user -p -h 127.0.0.1 -P 3306
```

Dockerfile

```
FROM mariadb:latest

# Устанавливаем переменные окружения
ENV MYSQL_ROOT_PASSWORD=...
ENV MYSQL_DATABASE=...
ENV MYSQL_USER=...
ENV MYSQL_PASSWORD=...
#ENV MYSQL_ROOT_HOST= '%' # Разрешить root-подключения с любого хоста
(опционально)

# Копируем SQL-скрипт для дополнительных прав
COPY ./init.sql /docker-entrypoint-initdb.d/
```

init.sql

```
GRANT ALL PRIVILEGES ON wood_db.* TO 'wooduser'@'%';
FLUSH PRIVILEGES;
```

docker-compose.yml

```
services:
  mariadb:
    image: hub.bobrobotirk.ru/appone-db:0.1.0
    container_name: mariadb
    volumes:
      - ./dbdata:/var/lib/mysql
    ports:
      - "${MYSQL_PORT}:3306"
    restart: unless-stopped
```

Еще вариант:

```
services:
  mariadb:
    image: mariadb:10.11
    container_name: mariadb
    restart: unless-stopped
    environment:
      MARIADB_ROOT_PASSWORD: rootpassword
      MARIADB_DATABASE: mydb
      MARIADB_USER: myuser
      MARIADB_PASSWORD: mypassword
    ports:
      - "3306:3306"
    volumes:
      - ./mariadb_data:/var/lib/mysql
      - ./init:/docker-entrypoint-initdb.d
```