Инструкция по установке ПО

1 Развертывание системы на проприетарных серверах

В этом разделе приведена максимально подробная пошаговая инструкция по развертыванию стека сервисов, описанного в файле `docker-compose.yml`, на собственных серверах под управлением Linux.

1.1 1. Предварительные требования

1. Сервер под управлением Ubuntu 20.04 или совместимой системы с доступом по SSH.
2. Права суперпользователя (root) или возможность использования `sudo`.
3. Установленные Docker Engine (версии $\geq$20.10) и Docker Compose (версии $\geq$1.29).
4. Минимум 4 ГБ оперативной памяти и свободное место на диске для хранения данных.

1.2 2. Установка Docker и Docker Compose

1. Обновите индекс пакетов:

   ```
   sudo apt update
   ```

2. Установите необходимые утилиты:

   ```
   sudo apt install -y ca-certificates curl gnupg lsb-release
   ```

3. Добавьте официальный GPG-ключ Docker и репозиторий:

   ```
   sudo mkdir -p /etc/apt/keyrings
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
     | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
   ```

```
echo \
  "deb [arch=$(dpkg --print-architecture) \
signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Установите Docker Engine и плагин Compose:

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

5. Проверьте установку:

```
docker --version          # Должно вывести что-то вроде Docker version 24.0.2
docker compose version    # Должно вывести версию плагина Compose
```

1.3 3. Подготовка директорий и прав

1. Создайте корневую папку для проекта и перейдите в неё:

```
sudo mkdir -p /root/prod
cd /root/prod
```

2. Создайте необходимые поддиректории:

```
sudo mkdir -p \
  postgres/init \
  postgres/data \
  minio/data \
  certbot/www \
  certbot/conf
```

3. Установите права и владельца для PostgreSQL и MinIO (UID/GID=1000 соответствует типичному первому пользователю в контейнерах):

```
sudo chown -R 1000:1000 postgres/
sudo chown -R 1000:1000 minio/
```

1.4 4. Размещение файлов конфигурации

1.4.1 4.1. Docker Compose файл

Создайте файл `/root/prod/docker-compose.yml` со следующим содержимым:

```
version: "3.8"

# Global environment variables
x-env: &env
  postgres: &postgres-env
    POSTGRES_DB: $DB_NAME
    POSTGRES_USER: $DB_USER
    POSTGRES_PASSWORD: $DB_PASSWORD

  backend: &backend-env
    DB_NAME: $DB_NAME
    DB_USER: $DB_USER
    DB_PASSWORD: $DB_PASSWORD
    SECRET_KEY: $SECRET_KEY
    ALGORITHM: $ALGORITHM
    EMAIL_SENDER: $EMAIL_SENDER
    SMTP_SERVER: $SMTP_SERVER
    SMTP_PORT: $SMTP_PORT
    EMAIL_PASSWORD: $EMAIL_PASSWORD
    MINIO_ACCESS_KEY: $MINIO_USER
    MINIO_SECRET_KEY: $MINIO_PASSWORD
    WEB_MINIO_ENDPOINT: $WEB_MINIO_ENDPOINT

  minio: &minio-env
    MINIO_ROOT_USER: $MINIO_USER
    MINIO_ROOT_PASSWORD: $MINIO_PASSWORD

# Common service settings
x-service-settings: &service-settings
  restart: unless-stopped
  networks:
    - food_automate_prod

services:
  postgres_db_prod:
    <<: *service-settings
    image: postgres:14.8-alpine3.18
    container_name: postgres_prod_container
    ports:
      - 5432:5432
```

```
  volumes:
    - /root/prod/postgres/init:/docker-entrypoint-initdb.d
    - /root/prod/postgres/data:/var/lib/postgresql/data
  environment:
    <<: *postgres-env

minio_prod:
  <<: *service-settings
  image: minio/minio:latest
  container_name: minio_prod_container
  ports:
    - 9000:9000 # API
    - 9001:9001 # Console
  volumes:
    - /root/prod/minio/data:/data
  environment:
    <<: *minio-env
  command: server /data --console-address ":9001"

backend_prod:
  <<: *service-settings
  image: ghcr.io/neural-solutions/food-automate-backend:v0.5.38
  container_name: backend_prod_container
  ports:
    - 8000:8000
  environment:
    <<: *postgres-env
    <<: *backend-env
    DB_HOST: postgres_prod_container
    DB_PORT: 5432
    LOGGING_FILE: "false"
    LOGGING_FILE_NAME: "nds_logger.log"
    MINIO_ENDPOINT: 172.18.0.2:9000
    REQUEST_LOGGING_FILE_NAME: "nds_request_logger.log"
    PATH_PDF_TECHCARD: "/dejavu"
  volumes:
    - /root/prod/nds_logger.log:/nds_logger.log
    - /usr/share/fonts/truetype/dejavu:/dejavu
  depends_on:
    - postgres_db_prod
    - minio_prod

frontend_prod:
  <<: *service-settings
  image: ghcr.io/neural-solutions/food-automate-frontend:v0.5.36
  container_name: frontend_prod_container
```

```
    ports:
      - 3000:80
    depends_on:
      - backend_prod

  nginx_prod:
    <<: *service-settings
    image: nginx:latest
    container_name: nginx_prod_container
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./nginx_prod.conf:/etc/nginx/conf.d/default.conf
      - ./certbot/www/:/var/www/certbot/
      - ./certbot/conf/:/etc/nginx/ssl/:ro
    depends_on:
      - frontend_prod
      - backend_prod

  certbot:
    image: certbot/certbot:latest
    container_name: certbot_prod_container
    volumes:
      - ./certbot/www/:/var/www/certbot/:rw
      - ./certbot/conf/:/etc/letsencrypt/:rw

networks:
  food_automate_prod:
    driver: bridge
```

1.4.2 4.2. Файл переменных окружения

Создайте файл `/root/prod/.env` и пропишите в него конкретные значения для переменных:

```
# PostgreSQL
DB_NAME=food_automate_db
DB_USER=food_user
DB_PASSWORD=supersecurepassword

# Backend
SECRET_KEY=your_super_secret_key_here
ALGORITHM=HS256
EMAIL_SENDER=no-reply@yourdomain.com
SMTP_SERVER=smtp.yourdomain.com
```

```
SMTP_PORT=587
EMAIL_PASSWORD=email_sender_password

# MinIO
MINIO_USER=minioadmin
MINIO_PASSWORD=minioadmin123
WEB_MINIO_ENDPOINT=minio.yourdomain.com:9000
```

Убедитесь, что файл `.env` находится рядом с `docker-compose.yml` и имеет права чтения:

```
ls -l /root/prod/{docker-compose.yml,.env}
```

1.5 5. Запуск сервисов

1. Перейдите в папку проекта:

   ```
   cd /root/prod
   ```

2. Запустите все контейнеры в фоновом режиме:

   ```
   sudo docker compose up -d
   ```

3. Проверьте, что все сервисы успешно запущены:

   ```
   sudo docker ps --filter "network=food_automate_prod"
   ```

4. Для просмотра логов конкретного контейнера (например, backend):

   ```
   sudo docker compose logs -f backend_prod
   ```

5. Остановка и удаление всех контейнеров, сети и томов:

   ```
   sudo docker compose down
   ```

1.6 6. Обновление сервисов

1. Получите последние версии образов из реестра:

   ```
   sudo docker compose pull
   ```

2. Перезапустите сервисы с обновлёнными образами:

   ```
   sudo docker compose up -d
   ```

2 Основные данные

- IP-адрес сервера - 176.114.86.101
- Папка с необходимыми файлами - home
- Подключение к серверу - ssh root@176.114.86.101

3 Структура папок на сервере

На сервере в директории `home` имеется следующая структура:

- `food-automate-infra` — папка, содержащая файл `docker-compose.yml`, который используется для запуска контейнеров приложения.
- `minio` — папка-volume, используемая для хранения данных в MinIO. Эта папка передается в контейнер MinIO для сохранения данных в случае падения контейнера.
- `nds_logger.log` — файл с логами работы приложения.
- `postgres` — папка-volume для хранения данных PostgreSQL, используется для долговременного сохранения данных при падении контейнера.

4 Запущенные контейнеры

На сервере запущены следующие контейнеры:

- **nginx_container**
  - Контейнер: `nginx:latest`
  - Команда: `/docker-entrypoint.sh`
  - Порты: `80 -> 80` (HTTP)
  - Описание: Этот контейнер работает как обратный прокси-сервер, перенаправляющий запросы на соответствующие контейнеры с приложением.
- **frontend_container**
  - Контейнер: `ghcr.io/neural-solutions/food-automate-frontend:v0.5.18`
  - Команда: `nginx -g 'daemon off;'`
  - Порты: `3000 -> 80` (Frontend интерфейс)
  - Описание: Этот контейнер отвечает за работу фронтенда вашего приложения, доступного через порт 3000.
- **backend_container**

  - Контейнер: `ghcr.io/neural-solutions/food-automate-backend:v0.5.13`
  - Команда: `python3 app.py`
  - Порты: `8000 -> 8000` (Backend API)
  - Описание: Контейнер, в котором работает серверная часть приложения. Этот контейнер отвечает за обработку запросов и взаимодействие с базой данных и другими сервисами.

- **minio_container**
  - Контейнер: `minio/minio:latest`
  - Команда: `/usr/bin/docker-entrypoint.sh`
  - Порты: `9000-9001 -> 9000-9001` (MinIO сервис)
  - Описание: Контейнер MinIO, используемый для хранения объектов, таких как изображения или документы. Использует папку `minio` для долговременного хранения данных.

- **postgres_container**
  - Контейнер: `postgres:14.8-alpine3.18`
  - Команда: `/docker-entrypoint.sh`
  - Порты: `5432 -> 5432` (PostgreSQL база данных)
  - Описание: Контейнер с PostgreSQL, в котором хранится база данных для приложения. Использует папку `postgres` для долговременного хранения данных.

5 Как подключаться к контейнерам

5.1 Подключение к контейнеру `nginx_container`

Контейнер `nginx_container` отвечает за обработку HTTP-запросов, как обратный прокси-сервер.

- Для доступа к контейнеру можно использовать любой браузер и перейти по адресу `http://<IP-сервер>:80`.
- Если требуется подключение через командную строку, используйте команду:

  ```
  docker exec -it nginx_container /bin/bash
  ```

  Эта команда откроет терминал внутри контейнера.

5.2 Подключение к контейнеру `frontend_container`

Контейнер `frontend_container` обрабатывает запросы на фронтенд.

- Для доступа к веб-интерфейсу можно использовать любой браузер и перейти по адресу `http://<IP-сервер>:3000`.
- Для подключения к контейнеру через терминал выполните:

```
docker exec -it frontend_container /bin/bash
```

5.3 Подключение к контейнеру `backend_container`

Контейнер `backend_container` обрабатывает запросы от фронтенда и взаимодействует с базой данных.

- Для работы с API можно отправлять HTTP-запросы на `http://<IP-сервер>:8000`.
- Для подключения к контейнеру используйте команду:

```
docker exec -it backend_container /bin/bash
```

5.4 Подключение к контейнеру `minio_container`

Контейнер `minio_container` предоставляет доступ к объектному хранилищу.

- Для доступа к MinIO используйте веб-интерфейс по адресу `http://<IP-сервер>:9000`.
- Для подключения через терминал выполните:

```
docker exec -it minio_container /bin/bash
```

5.5 Подключение к контейнеру `postgres_container`

Контейнер `postgres_container` содержит базу данных PostgreSQL.

- Для подключения к базе данных можно использовать любой клиент PostgreSQL (например, `psql`):

```
psql -h <IP-сервер> -U <пользователь> -d <база данных> -p 5432
```

- Для подключения к контейнеру и работы с базой данных используйте команду:

```
docker exec -it postgres_container /bin/bash
```

6 Управление контейнерами

Для управления контейнерами используйте файл `docker-compose.yml`, расположенный в папке `food-automate-infra`.

- **Запуск всех контейнеров**:

  ```
  docker-compose up -d
  ```

  Эта команда запускает все контейнеры в фоновом режиме.

- **Остановка всех контейнеров**:

  ```
  docker-compose down
  ```

  Эта команда остановит все контейнеры.

- **Проверка статуса контейнеров**:

  ```
  docker ps
  ```

  Эта команда покажет, какие контейнеры запущены и их статус.

- **Перезапуск контейнеров**:

  ```
  docker-compose restart
  ```

  Эта команда перезапустит все контейнеры.

7 Логи

Все логи работы приложения записываются в файл `nds_logger.log`. Для просмотра логов можно использовать команду:

```
tail -f nds_logger.log
```

Это позволит вам отслеживать логи в реальном времени.

8 Частые проблемы и их решения

- **Ошибка подключения к базе данных:** Убедитесь, что контейнер с PostgreSQL запущен, и порты на сервере настроены правильно. Проверьте логи контейнера с PostgreSQL.
- **MinIO не доступен:** Убедитесь, что контейнер MinIO запущен, и порты 9000 и 9001 не заняты другими процессами.

9 Завершение работы

Если нужно остановить сервер и все контейнеры, используйте команду:

```
docker-compose down
```

Чтобы восстановить работу, выполните команду:

```
docker-compose up -d
```