1. Получение исходного кода

клонировать код из предоставленного репозитория, переходим в папку с проектом и далее команды отдаем находясь в ней

2. Создание переменных окружения

создаем файл с переменными окружения

```
cp template.box.env .env
```

установить требуемые значения переменных. Файл .env содержит пояснения для каждой переменной. Часть переменных можно оставить без изменений.

3. Подготовка файловой структуры

Перед запуском сервиса БД нужно создать папки для файлов базы данных, имена папок должны совпадать со значениями указанными в переменных

```
# переменная dbs - файлы БД
mkdir -p dbs/mongodb/database

# переменная FILE_STORAGE_PATH - пользовательские файлы, те файлы хранимые сервисом
mkdir -p dbs/files

# переменная SECURE_FILES - файлы сертификатов необходимые для коммуникации с другими сервисами
mkdir -p dbs/.secure_files

# переменная API_LOGS_STORAGE_PATH - логи апи
mkdir -p dbs/logs/api-logs

# переменная SFC_LOGS_STORAGE_PATH - логи файлового хранилища
mkdir -p dbs/logs/sfc-logs

# тоже одной командой
mkdir -p dbs/mongodb/database dbs/logs/sfc-logs dbs/logs/api-logs dbs/.secure_files dbs/files
```

4. Запуск СУБД

производим запуск СУБД

```
docker compose up mongodb

# просмотр логов контейнера
docker logs -f mongodb
```

для сервиса mongodb статус должен быть - healfy .

В результате будут созданы общие файлы mongodb , административный доступ к созданному экземпляру можно получить, выполнив, для контейнера БД, команду:

```
mongosh -u "$MONGO_INITDB_ROOT_USERNAME" -p
"$MONGO_INITDB_ROOT_PASSWORD" --port="$mongo_port"
```

только если экземпляр СУБД уже существовал, перед стартом апи нужно будет создать БД и пользователя отдельной командой:

```
# создать БД:
use db

# создать пользователя для апи:
db.createUser(
 {
 user: "username",
 pwd: "password",
 roles: [ { role: "readWrite", db: "db" } ], passwordDigestor: "server"  }
)
# username - значение переменной mongo_login
# password - значение переменной mongo_password
```

если СУБД запускается "*с нуля*", то необходимые команды по созданию БД и пользователя выполнятся автоматически.

5. Запуск сервиса апи

```
docker compose up api
```

сразу после запуска апи, автоматически, должны выполниться миграции. Проверить статус апи можно выполнив GET запрос к адресу указанному для него. Ответ должен быть Cluster mode .

6. Запуск сервиса админ-панель

```
docker compose up admin
# первый запуск сервиса требует значительного времени, так как происходит компиляция и
сборка статических ресурсов. В дальнейшем запуск будет занимать незначительное время.
```

Для входа в личный кабинет администратора нужно использовать учетные данные которые были созданы на этапе запуска auth сервиса. После входа, при условии, что учетная запись имеет статус суперпользователя, на открывшейся странице должны отобразиться все пользователи зарегистрированные в системе auth.

7. Добавление компании

Для корректной работы системы требуется связать пользователя и компанию. Сделать это можно перейдя по относительной ссылке `/companies` и добавив нужную компанию с помощью соответствующей формы.

Вернувшись на страницу `/users` нужно выбрать соответствующего пользователя и на открывшейся форме добавить созданную компанию данному пользователю, и возможно, включить переключатель `Застройщик` .

8. Запуск сервиса управления пользовательскими файлами

```
docker compose up files
```

9. Запуск ЕРП сервиса

```
docker compose up erp
```

Для входа в личный кабинет администратора нужно использовать учетные данные которые были созданы на этапе запуска `auth` сервиса.
После входа, при условии, что учетная запись имеет, как минимум, одну связанную организацию будут доступны основные функции системы.

10. Полный запуск/перезапуск системы

```
# полная остановка системы
docker compose stop

# запуск всех сервисов системы
docker compose up -d
```

В результате вызова команды все сервисы должны находится в активном состоянии, а статус БД должен быть - `healfy`.