**Система формирования коммерческих предложений**

Инструкция по развертыванию

Москва

2025

Оглавление

[1. Установка Docker и Docker Compose на Ubuntu 4](#_Toc205067861)

[1.1 Обновить систему 4](#_Toc205067862)

[1.2 Установить зависимости 4](#_Toc205067863)

[1.3 Добавить GPG-ключ Docker 4](#_Toc205067864)

[1.4 Добавить репозиторий Docker 4](#_Toc205067865)

[1.5 Установить Docker Engine 4](#_Toc205067866)

[1.6 Проверить, что Docker работает 4](#_Toc205067867)

[1.7 (Опционально): Разрешить запуск Docker без sudo 4](#_Toc205067868)

[1.8 Проверить Docker Compose 4](#_Toc205067869)

[1.9 Если не работает — установить вручную 4](#_Toc205067870)

[2 Описание и установка необходимых Docker-контейнеров 5](#_Toc205067871)

[2.1 db — postgres 5](#_Toc205067872)

[2.2 pgadmin — dpage/pgadmin4 5](#_Toc205067873)

[2.3 app — kp-docker/kp-generation-service:latest 5](#_Toc205067874)

[2.4 ui — nginx:alpine 5](#_Toc205067875)

[3 Запуск системы 6](#_Toc205067876)

[3.1 Создание структуры каталогов 6](#_Toc205067877)

[3.2 Каталог docker 6](#_Toc205067878)

[3.3 Каталог image 6](#_Toc205067879)

[3.4 Каталог nginx 6](#_Toc205067880)

[3.5 Каталог ui 6](#_Toc205067881)

[3.6 SSL сертификаты 6](#_Toc205067882)

[3.7 Запуск системы 6](#_Toc205067883)

[4 Сборка серверной части проекта 9](#_Toc205067884)

[4.1 Предварительные требования 9](#_Toc205067885)

[4.2 Открытие проекта 9](#_Toc205067886)

[4.3 Проверка JDK и Gradle 9](#_Toc205067887)

[4.3.1 JDK 9](#_Toc205067888)

[4.3.2 Gradle 9](#_Toc205067889)

[4.4 Сборка проекта 10](#_Toc205067890)

[4.4.1 Через IntelliJ 10](#_Toc205067891)

[4.4.2 Через терминал внутри IntelliJ 10](#_Toc205067892)

[5 Сборка клиентской части 12](#_Toc205067893)

[5.1 Предварительные требования 12](#_Toc205067894)

[5.2 Сборка проекта 12](#_Toc205067895)

[6 Выгрузка и загрузка локального докер-образа 13](#_Toc205067896)

[7 Регистрация нового пользователя 14](#_Toc205067897)

# Установка Docker и Docker Compose на Ubuntu

## Обновить систему

*sudo apt update  
sudo apt upgrade -y*

## Установить зависимости

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

## Добавить GPG-ключ Docker

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

## Добавить репозиторий Docker

*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*

## Установить Docker Engine

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

## Проверить, что Docker работает

*sudo docker version*

## (Опционально): Разрешить запуск Docker без sudo

*sudo usermod -aG docker $USER  
newgrp docker*

## Проверить Docker Compose

*docker compose version*

## Если не работает — установить вручную

*sudo curl -SL https://github.com/docker/compose/releases/download/v2.24.0/docker-compose-linux-x86\_64 \  
 -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
docker-compose –version*

# Описание и установка необходимых Docker-контейнеров

## db — postgres

Контейнер PostgreSQL, используется для хранения данных.

Команды для загрузки:

*docker pull postgres*

## pgadmin — dpage/pgadmin4

Веб-интерфейс для управления PostgreSQL.

Команды для загрузки:

*docker pull dpage/pgadmin4*

## app — kp-docker/kp-generation-service:latest

Бэкенд-приложение для генерации коммерческих предложений.

Команды для загрузки:

*# Собрать локальный образ на машине разработчика:*

*docker build -t kp-docker/kp-generation-service:latest .*

Либо сохранить его в tar-архив если образ уже загружен в локальный репозиторий Докера:

*docker save -o kp-generation-service.tar kp-docker/kp-generation-service:latest*

## ui — nginx:alpine

Фронтенд на React, обслуживается через Nginx.

Команды для загрузки:

*docker pull nginx:alpine*

# Запуск системы

## Создание структуры каталогов

На сервере создать каталог /home/system, в нем 4 подкаталога:

* docker
* image
* nginx
* ui

## Каталог docker

В каталоге docker разместить файлы docker-compose.yml и docker-compose.prod.yml из поставки программного кода.

## Каталог image

В каталоге image разместить tar-архив собранного докер-образа серверной части приложения kp-generation-service.tar. После размещения архива выполнить загрузку образа в локальный репозиторий Докера с помощью команды:

*docker load -i kp-generation-service.tar*

## Каталог nginx

В каталоге nginx разместить разместить конфигурационный файл веб-сервера NGINX (nginx.conf) из поставки программного кода.

## Каталог ui

В каталоге ui разместить каталог build (собранный REACT проект клиентской части) из поставки программного кода.

## SSL сертификаты

В системе необходимо разместить (в случае необходимости) SSL сертификаты для обеспечения шифрованного соединения между клиентской и серверной частью. В конфигурационном файле веб-сервера nginx.conf необходимо указать путь до этих сертификатов:

ssl\_certificate /etc/letsencrypt/live/uyta-co.ru/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/uyta-co.ru/privkey.pem;

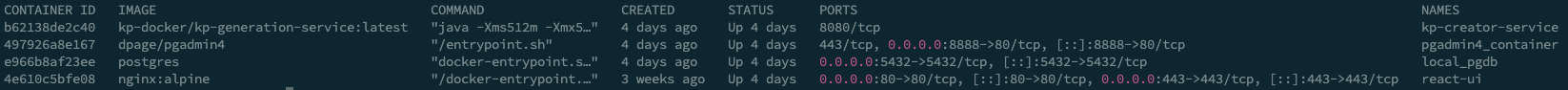
В данном случае использовались бесплатные сертификаты от кампании LetsEncypt инструкцию по генерации сертификатов можно посмотреть на их официальном сайте. Если сертификаты не требуются и запуск будет производиться в локальной сети, то необходимо модифицировать конфигурационный файл таким образом, как в Приложении 1. Но так делать настоятельно не рекомендуется и лучше всего все-таки сделать сертификаты и использовать защищённое соединение.

## Запуск системы

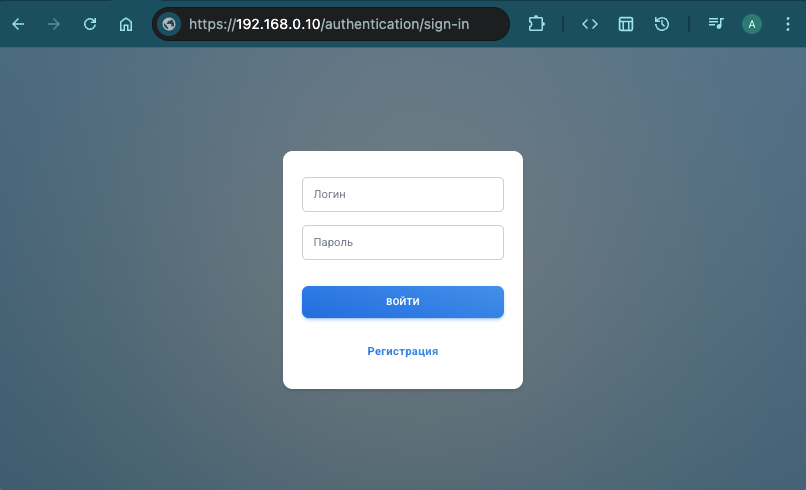
Для запуска системы перейдите в каталог *docker* и выполните команду:

*docker compose -f docker-compose.yml -f docker-compose.prod.yml up -d*

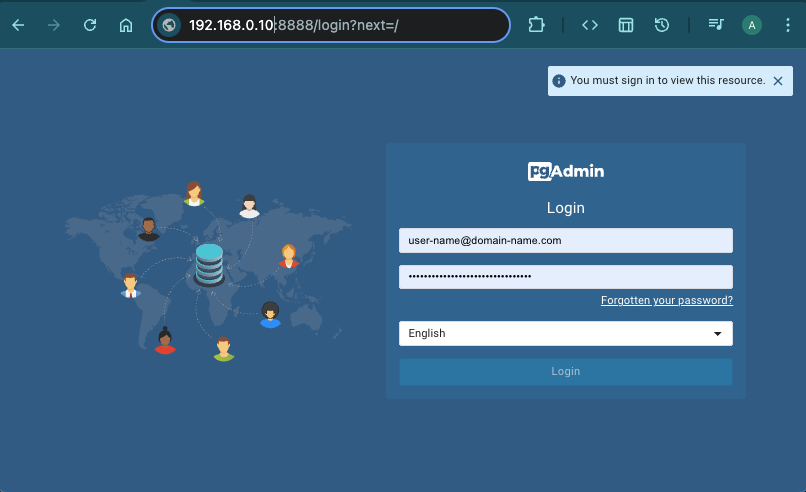
Перейдите в консоль управления системы Docker и проверьте работающие контейнеры используя команду *docker ps*, вы должны увидеть что-то подобное:

****

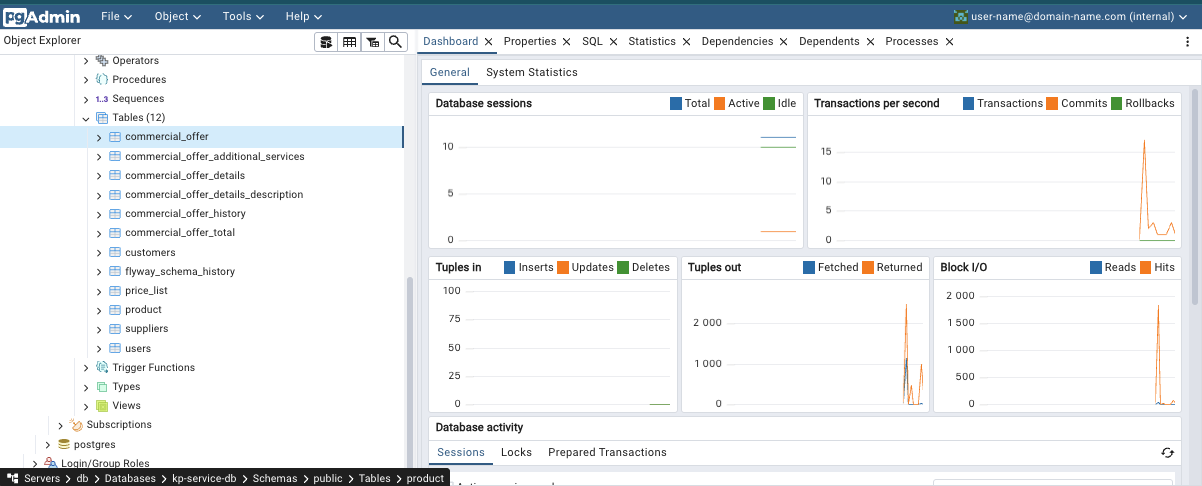
Если контейнеры поднялись и работают, то их статус будет UP. В случае ошибок старта проверяйте логи контенеров стандартными средствами. Серверная часть приложения работает на порту 8080, база данных на порту 5432, вручную управлять базой данных можно через приложение pgadmin4, которое будет доступно на порту 8888. После перехода браузером на адрес вашего сервера по 80 порту, вы увидите приветственное окно системы:

****

При переходе по порту 8888 вы увидите приветственное окно системы управления базой данных:

****

Для подключения к системе управления необходимо использовать реквизиты описанные в файле docker-compose.yml из поставки программного кода PGADMIN\_DEFAULT\_EMAIL и PGADMIN\_DEFAULT\_PASSWORD, для подключения к самой базе данных использовать реквизиты параметров SPRING\_DATASOURCE\_USERNAME и SPRING\_DATASOURCE\_PASSWORD. При успешном подключении вы увидите полный список таблиц базы данных и сможете вручную осуществлять управление данными:

****

# Сборка серверной части проекта

## Предварительные требования

Необходимо установить:

* IntelliJ IDEA 2025.1.1.1 (Community Edition) или выше;
* Java 17 или выше;
* Gradle 8.8 (можно использовать встроенный Gradle из IntelliJ);
* Docker-Desktop 4.16.2 (95914) или выше.

## Открытие проекта

Запустите IntelliJ IDEA.

Код клиентской части находится в поставке kp/server.

Выберите "Open", укажите путь к вашему проекту (где находится build.gradle или settings.gradle).

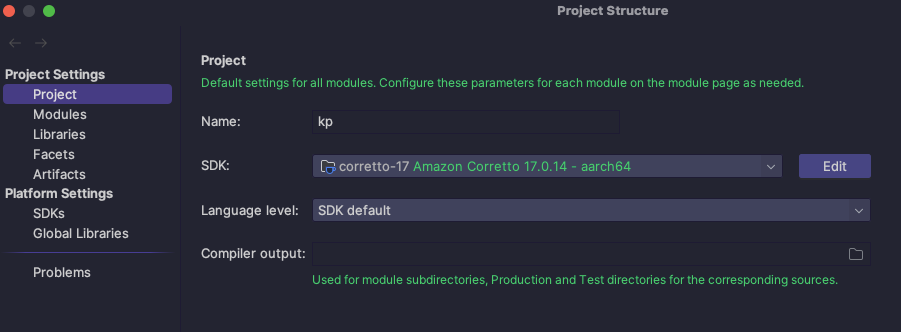
После открытия IDE автоматически определит, что проект — Gradle-проект.

## Проверка JDK и Gradle

### JDK

Перейдите в File → Project Structure → Project.

Убедитесь, что в поле Project SDK выбрана Java 17.



### Gradle

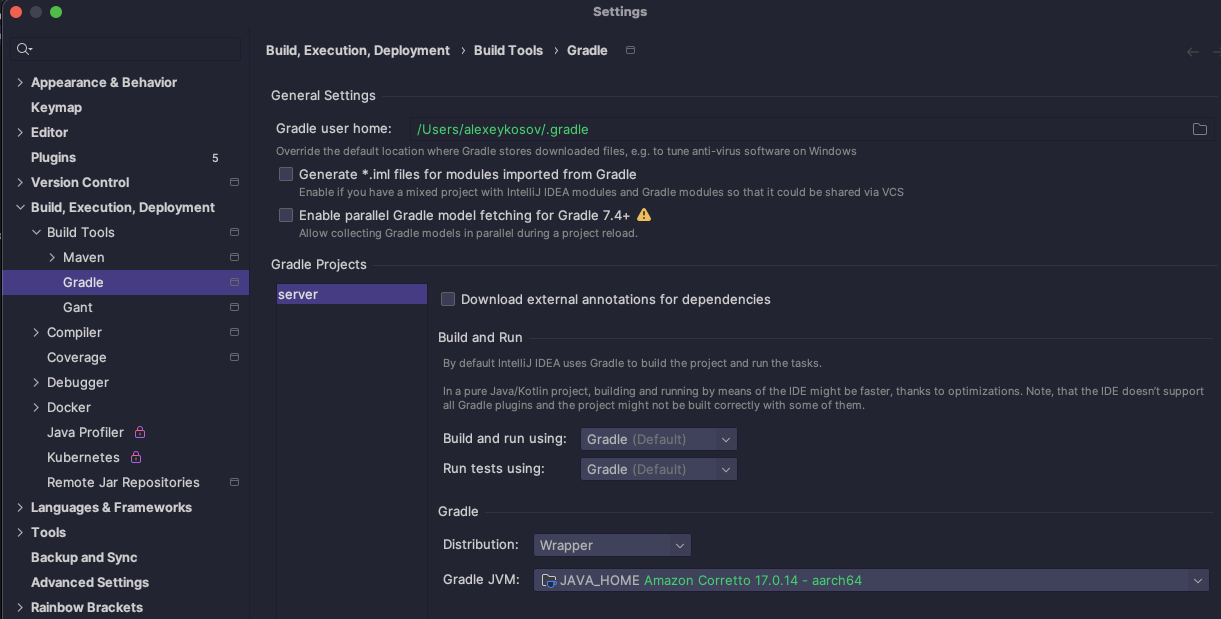
Откройте File → Settings → Build, Execution, Deployment → Build Tools → Gradle.

Убедитесь, что:

- Use Gradle from: Wrapper task in gradle-wrapper.properties или Specified location (указать путь к Gradle 8.8).

- Build and run using: Gradle.

- Run tests using: Gradle.



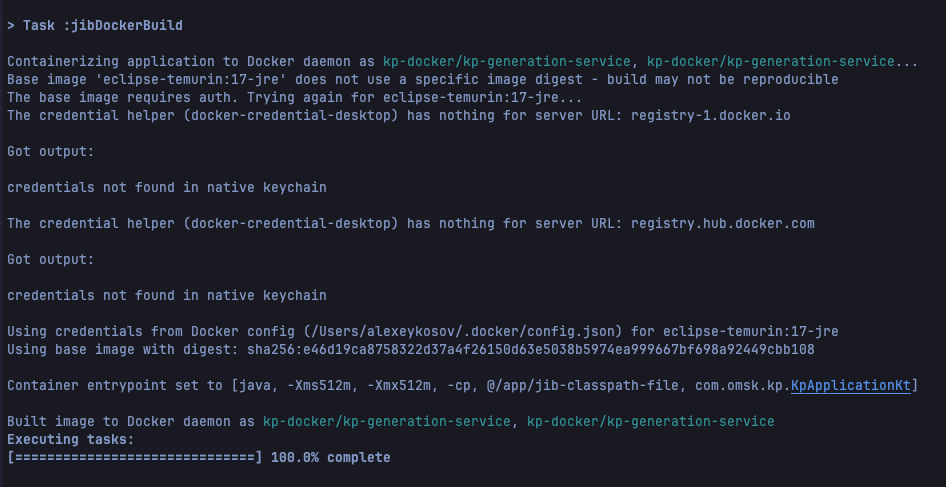
## Сборка проекта

### Через IntelliJ

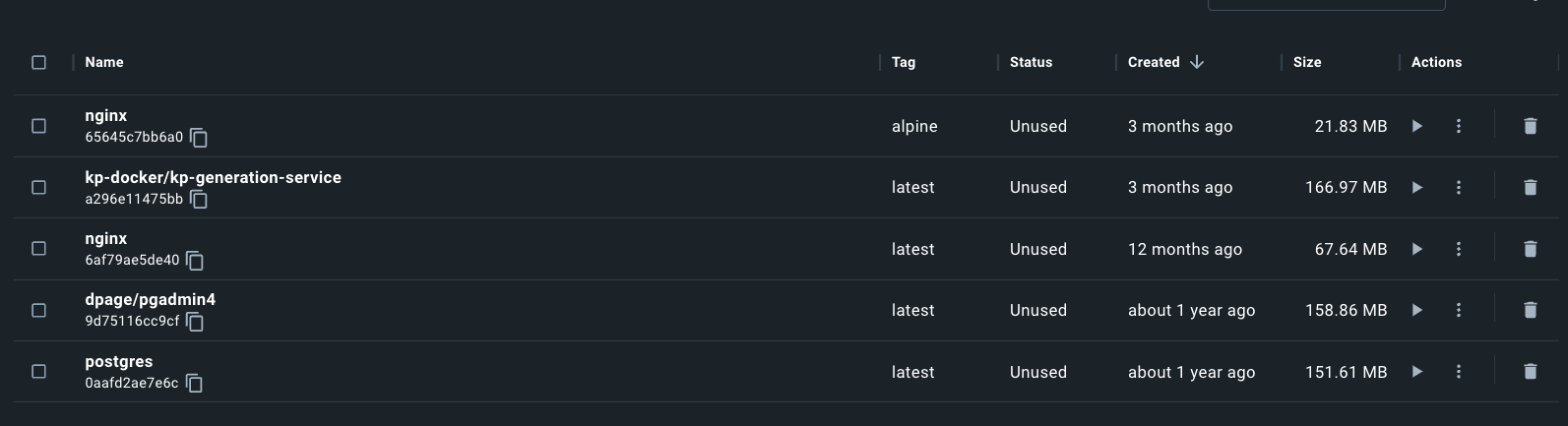
* Перейдите в левую панель Gradle (справа значок слона 🐘).
* Откройте структуру проекта → Tasks → build.
* Дважды щелкните по задаче build.
* После завершения сборки артефакт будет находиться в build/libs/.

### Через терминал внутри IntelliJ

* Команда: gradle build.
* Перейдите в терминале в каталог server по соответствующему пути, где вы разместили исходный код например, ../kp/server.
* Выполните команду gradle build результат ее выполнения будет таким:



Результатом выполнения данной команды будет сборка проекта и автоматическое формирование докер-образа серверной части и автоматическая регистрация нового образа kp-docker/kp-generation-service в локальной версии докера:

****

# Сборка клиентской части

## Предварительные требования

* Убедитесь, что установлены Node.js и npm. Для проверки выполните команды node -v и npm -v. Если они не установлены — скачайте с официального сайта https://nodejs.org и установите LTS-версию.
* Выполните в корне проекта npm install. Эта команда установит все зависимости из package.json.

## Сборка проекта

* Код клиентской части находится в поставке kp/ui.
* Выполните в корне проекта npm install. Эта команда установит все зависимости из package.json.
* Для создания production-сборки выполните команду npm run build. Результат будет в папке /build ее можно размещать на сервере.

# Выгрузка и загрузка локального докер-образа

После локальной сборки серверной части приложения необходимо выполнить команду, которая сохранит на жесткий диск образ серверной части в виде архива для последующей загрузки на сервер:

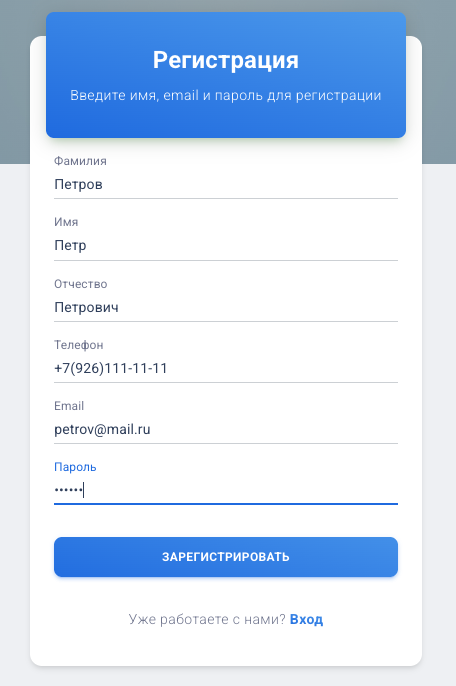
*docker save -o kp-generation-service.tar kp-docker/kp-generation-service:latest*

После этого локально будет сохранен файл *kp-generation-service.tar* который в последствии можно сохранить на удалённый сервер. После копирования образа на удалённый сервер необходимо заново развернуть его в системе Docker с помощью команды:

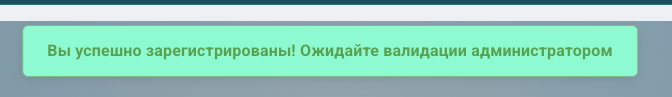
*docker load -i kp-generation-service.tar*

# Регистрация нового пользователя

При регистрации нового пользователя заполняются его данные на форме регистрации:



После нажатия на кнопку «Зарегистрировать» выдается сообщение о валидации данных со стороны администратора:



Для проведения валидации администратор системы должен присвоить новому пользователю роль в статусной модели. Это необходимо сделать через базу данных. Необходимо найти нового пользователя в таблице *USERS* и обновить поле *role* вместо значения *EMPTY* присвоить значения: USER или ADMIN. После этого новый пользователь сможет полноценно использовать систему в соответствии со своей ролью.