|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Инструкция по установке**

“Программный комплекс удаленного мониторинга и контроля за работой медицинского оборудования с предиктивным механизмом выявления наступления неисправностей”

**Москва**

**2024**

# 

[**1. Введение 3**](#_fccwrsmsvvye)

[**2. Начало установки 4**](#_1oa6o4eswtm)

[2.1. Обеспечение установки 4](#_j7cpsxfn0y60)

[2.2. Запуск установки и ожидаемый результат 4](#_odfzv4bh152v)

[2.2.1. Результат установки 5](#_l18261v9cgst)

[**3. Процесс выполнения установки 5**](#_hn590n2nrr38)

[3.1. Установка зависимостей 6](#_z5wsklwi5lnf)

[3.2. Установка Docker 6](#_evmpve6iavx)

[3.3. Проверка версий 7](#_827flbhtvfs3)

[3.4. Разворачивание проекта 7](#_v06z4m1ze0f)

[3.5. Запуск контейнеров 7](#_1w7eno6fuznu)

[3.6. Ожидание поднятия контейнера PostgreSQL 8](#_z6tg9y8pgk90)

[3.7. Выполнение команд внутри контейнера 8](#_c7uv7g5g94lp)

[3.8. Перезапуск контейнеров 9](#_ay25wytqbnss)

[**4. Используемые технологии 9**](#_k33axbbequcg)

[**14. Перечень сокращений 11**](#_xgxiree3hajj)

# 

# Введение

Документ описывает последовательность шагов установки облачного решения на сервер под управлением ОС Debian 12. Перед началом установки необходимо произвести развертывание сервера или виртуальной машины под управлением ОС Debian, отвечающего следующим техническим требованиям:

1. Количество ядер процессора: 4
2. Оперативная память: 16Гб
3. Жесткий диск NVME: 160Гб

Для успешной установки Комплекса необходимо следовать приведенным ниже шагам по установке.

# Начало установки

## Обеспечение установки

1. Необходимо иметь скрипт установки install.sh и архив с кодом archive.tar.gz;
2. Архив с кодом обязательно должен называться archive.tar.gz для упрощения процесса установки и избегания ошибок;
3. Необходимо разместить оба файла рядом в директорию, в которой предположительно должен размещаться проект;

На данный момент предполагается, что на сервер исходный код и команда попадают любым удобным для пользователя способом:

* + Через FTP менеджер
  + Через SFTP менеджер
  + Через ssh соединение (например командой scp archive.tar.gz user@server\_id:/path/to/project/)

## Запуск установки и ожидаемый результат

Команда bash install.sh запустит и проведет весь процесс установки зависимостей и самого приложения, от инженера, выполняющего установку, требуется только запустить команду.

### 

### Результат установки

1. При успешной установке будет сообщение “=== Процесс завершен ===”;
2. При выполнении команды docker compose ps можно увидеть все запущенные контейнеры;
3. Приложение доступно по IP адресу сервера;

# Процесс выполнения установки

Весь процесс установки происходит последовательно, ниже приведены этапы и команды для успешного выполнения каждого этапа. Для успешного завершения процесса установки все этапы должны выполняться строго последовательно. Процесс установки состоит из следующих этапов:

1. Установка зависимостей
2. Установка Docker
3. Проверка версий
4. Разворачивание проекта
5. Запуск контейнеров
6. Ожидание запуска контейнера PostgreSQL
7. Выполнение команд внутри контейнера
8. Перезапуск контейнеров

Ниже подробно описан каждый из этапов установки

## 

## Установка зависимостей

Выполняется следующий набор команд:

1. sudo apt-get update
2. sudo apt-get install -y \
3. apt-transport-https \
4. ca-certificates \
5. curl \
6. gnupg \
7. lsb-release

Этот шаг обновляет список доступных пакетов и устанавливает необходимые зависимости для работы с Docker. В частности, пакет apt-transport-https позволяет использовать HTTPS для обращения к репозиториям APT.

## Установка Docker

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

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

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io

Здесь добавляются ключи и источники для загрузки Docker из официального репозитория, а затем Docker устанавливается на систему.

## Проверка версий

docker --version

Эта команда выводит версию установленного Docker, чтобы убедиться, что установка прошла успешно.

## Разворачивание проекта

tar -xvf archive.tar.gz

mv ./html/\* .

mv ./html/.\* .

rm -rf html

chmod -R 777 storage

chmod -R 777 public

cp .env.example .env

Здесь происходит разархивирование архива проекта, перемещение файлов из директории html в текущую папку, установка прав доступа к папкам storage и public и копирование файла .env.example в .env для настройки окружения.

## Запуск контейнеров

docker-compose up -d --build

Эта команда использует Docker Compose для запуска контейнеров, описанных в файле docker-compose.yml. Ключ -d означает "в фоновом режиме", а --build означает, что образы Docker будут созданы заново перед запуском.

## Ожидание поднятия контейнера PostgreSQL

sleep 15

Данная команда добавляет паузу в 15 секунд перед выполнением следующих команд. Это необходимо для того, чтобы дать достаточно времени для запуска контейнера PostgreSQL перед выполнением команд внутри контейнера.

## Выполнение команд внутри контейнера

docker-compose exec -T backend bash << 'EOF'

PGPASSWORD=password psql -U developer -h base-pgsql -d core -w < core.dump

composer install

php artisan optimize:clear

php artisan key:generate

php artisan storage:link

npm install

npm run build

EOF

Этот блок кода выполняет различные команды внутри контейнера backend, такие как загрузка дампа базы данных PostgreSQL, установка зависимостей PHP и JavaScript, генерация ключа Laravel, создание символической ссылки на хранилище и т. д.

## Перезапуск контейнеров

docker-compose down

docker-compose up -d --build

Контейнеры повторно запускаются с новыми настройками. Это важно сделать для корректной работы supervisor’a.

# Используемые технологии

1. Debian 12: Базовый образ операционной системы, на котором будет развернуто приложение. Debian 12 выбрана как широко используемый и удобный для настройки дистрибутив Linux. При необходимости его поддержка и модификация куда удобнее, чем у аналогов;
2. PHP 8.2: Язык программирования, используемый для разработки бэкенда приложения. Включает в себя ряд расширений для работы с базами данных, изображениями и другими компонентами.
3. Node.js и npm: Платформа для выполнения JavaScript на сервере и его пакетный менеджер, используемые для управления зависимостями фронтенда и выполнения сборки.
4. Composer: Пакетный менеджер для PHP, используемый для управления зависимостями бэкенда.
5. Docker: Платформа для разработки, доставки и запуска приложений в контейнерах. Обеспечивает изоляцию, масштабируемость и переносимость приложения.
6. Nginx: Веб-сервер и прокси-сервер, используемый для обслуживания веб-страниц и маршрутизации запросов. Обычно используется в качестве прокси-сервера для передачи запросов к бэкенду.
7. Supervisor: Программа для управления процессами операционной системы, таких как веб-серверы и фоновые задачи. Используется для запуска и контроля нескольких процессов одновременно.
8. PostgreSQL и Redis: Системы управления базами данных, используемые для хранения данных приложения. PostgreSQL используется для хранения структурированных данных, а Redis - для кэширования и хранения неструктурированных данных.
9. TLS (SSL) сертификаты: Используются для обеспечения безопасного соединения между клиентом и сервером по HTTPS протоколу. Обеспечивают шифрование данных и подтверждение подлинности сервера.

# Перечень сокращений

Раздел определяет примененные в инструкции термины и сокращения с приведением их краткого описания (Расшифровка). Применяемые в Техническом задании термины и сокращения приведены в таблице 6.1

Таблица 6.1 Термины и сокращения

| Сокращение | Расшифровка |
| --- | --- |
| Комплекс | Программный комплекс удаленного мониторинга и контроля за работой медицинского оборудования с предиктивным механизмом выявления наступления неисправностей |
| Протокол | Набор команд и структура данных, передаваемых между участниками Комплекса |
| REST | (от англ. Representational State Transfer) - «передача репрезентативного состояния» или «передача „самоописываемого“ состояния» |
| API | (англ. Application Programming Interface[1]) - описание способов взаимодействия одной компьютерной программы с другими |
| JSON | (англ. JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript |
| Балансировщик | Программно-аппаратное средство для маршрутизации и организации потока запросов к обслуживающим эти запросы системам |
| Сервер | Отказоустойчивая ЭФМ увеличенной мощности, расположенная в защищенных помещениях |
| Виртуальный сервер | Логически выделенная мощностная (ресурсная) часть Сервера, предназначенная для выполнения определенных задач |
| НСД | Несанкционированный доступ к функциям или данным |
| HTTP | HyperText Transfer Protocol – протокол передачи данных. |
| HTTPS | Hypertext Transfer Protocol Secure– расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS, тем самым обеспечивается защита этих данных. |
| БД | База данных |
| ГОСТ | Государственный стандарт. |
| ЕСКД | Единая система конструкторской документации |
| ЕСПД | Единая система программной документации |
| ОС | Операционная система |
| Пользователь | Сотрудник КО, КНВШ или ЦО, эксплуатирующий портал для достижения целей по управлению олимпиадным движением и научно-исследовательской деятельности |
| РД | Руководящий документ |
| РФ | Российская Федерация |
| СУБД | Система управления базой данных |
| ЭВМ | Электронно-вычислительная машина |

# 