Руководство администратора «Л1 Первая Линия Техподдержки»

Руководство администратора «Л1 Первая Линия Техподдержки»

Содержание

Документ	Назначение
Системные требования	Состав системы, ресурсы, платформа, диски и сетевые доступы.
Развёртывание	Подготовка сервера, установка в открытом или закрытом контуре и первоначальная проверка.
Конфигурация	Правила работы с `.env` и справочник переменных окружения.
Эксплуатация	Запуск и остановка, журналы, резервное копирование, обновление, откат и диагностика.
Альт	Совместимость Л1 Первая Линия техподдержки с дистрибутивами Альт

Быстрый маршрут

1. Проверьте системные и сетевые требования.
2. Выберите способ установки:
    - быстрая установка через `install.sh` — только для Container Registry;
    - ручная установка — для Container Registry или закрытого контура.
3. Сохраните `.env`, `APP_SECRETS_KEY` и сведения о версиях в защищённом хранилище.
4. Настройте резервное копирование PostgreSQL и MinIO по руководству по эксплуатации.

> **Предупреждение.** `install.sh` пока работает только с Container Registry. Возможность закрытого контура сохраняется, , но его установка выполняется вручную.

Критерии успешного развёртывания

Развёртывание завершено, если:

- сервисы `postgres`, `redis`, `minio`, `backend` и `frontend` имеют состояние `Up` и `healthy`;
- одноразовые сервисы `migrate` и `minio-mc` завершились с кодом `0`;
- веб-интерфейс открывается по `FRONTEND_PUBLIC_URL`;
- пользовательский компьютер может загрузить и скачать тестовый файл через `S3_PUBLIC_ENDPOINT`;

Инструкция рассчитана на администратора, знакомого с Linux, Docker, Docker Compose, DNS и reverse proxy.

Системные требования

Требования ниже относятся к односерверному развёртыванию.

Состав развёртывания

Сервис	Назначение	Постоянные данные
`frontend`	Веб-интерфейс и проксирование запросов к backend.	Нет.
`backend`	API, бизнес-логика и фоновые обработчики.	Да, volumes `l1-support-backend-config` и `l1-support-backend-logs`.
`postgres`	Основные данные системы.	Да, Docker volume `postgres_data`.
`redis`	Кэш и временная координация фоновых задач.	Нет, данные считаются временными.
`minio`	Файлы и вложения.	Да, Docker volume `minio_data`.
`migrate`	Применение миграций PostgreSQL перед запуском backend.	Нет, одноразовый сервис.
`minio-mc`	Создание бакета и технического пользователя MinIO.	Нет, одноразовый сервис.

Все сервисы по умолчанию запускаются на одном Docker-сервере в общей bridge-сети.

Аппаратные ресурсы

Контур	CPU	RAM	Системный диск	Диск данных
Демонстрация и функциональное тестирование	2 vCPU	4 ГБ	20–30 ГБ SSD	от 20 ГБ
Начальная промышленная конфигурация	4 vCPU	8 ГБ	40–60 ГБ SSD	от 100 ГБ
Повышенная нагрузка	8 vCPU	16 ГБ	80–100 ГБ SSD/NVMe	от 250 ГБ
Высокая нагрузка или повышенные требования к доступности	По результатам нагрузочного тестирования	от 32 ГБ	NVMe	отдельные PostgreSQL и S3-хранилище

Это предварительные ориентиры, а не гарантированные показатели производительности. Итоговая конфигурация зависит от:

- числа одновременно работающих пользователей;
- количества создаваемых обращений и сообщений;
- интенсивности IMAP/SMTP-обмена;
- количества и размера вложений;
- срока хранения файлов и резервных копий;
- требований к времени отклика и восстановлению после сбоя.

Программная платформа

- 64-разрядный Linux;
- архитектура хоста, совпадающая с архитектурой образов поставки; базовый целевой вариант — `amd64`;
- Docker Engine 24 или новее;
- Docker Compose Plugin v2.20 или новее; Compose v1 не поддерживается;
- включённый автозапуск Docker после перезагрузки сервера;
- `curl` и `openssl` для проверки установки и генерации секретов;
- Nginx и Certbot, если TLS завершается непосредственно на Docker-сервере;
- корректная синхронизация времени через NTP.

Рекомендуемые базовые дистрибутивы: минимальная установка Ubuntu Server 22.04/24.04 LTS или Debian 12 без графической оболочки.

> Применение Astra Linux, ALT Linux, RED OS и других дистрибутивов требует предварительной проверки совместимости конкретной версии Docker Engine и образов поставки.

Проверьте окружение перед установкой

```
uname -m
docker version
docker compose version
systemctl is-enabled docker
timedatectl status
```

Для рабочих мест рекомендуется актуальная версия Chromium-based браузера или Firefox ESR с включёнными JavaScript, cookies и доступом к обоим публичным адресам системы.

Входящие соединения

В промышленной среде снаружи публикуются только порты reverse proxy.

Откуда	Куда	Порт	Назначение
Let's Encrypt и рабочие места	reverse proxy	`80/TCP`	HTTP-01 и перенаправление HTTP на HTTPS.

Откуда	Куда	Порт	Назначение
Рабочие места	reverse proxy	443/TCP	Веб-интерфейс, REST API и WebSocket.
Рабочие места, схема с одним DNS	reverse proxy	9443/TCP	S3 API, загрузка и скачивание файлов.
Только reverse proxy на Docker-сервере	frontend	9090/TCP локально	Веб-интерфейс.
Только reverse proxy на Docker-сервере	backend	9100/TCP локально	REST API и WebSocket.
Только Docker-сервер	PostgreSQL	9101/TCP локально	Администрирование и диагностика БД.
Только Docker-сервер	Redis	9102/TCP локально	Диагностика Redis.
Только Docker-сервер	MinIO Console	9103/TCP локально	Администрирование MinIO.
Только reverse proxy на Docker-сервере	MinIO S3 API	9104/TCP локально	Файлы и вложения.

При двух DNS-именах веб-интерфейс и S3 API публикуются через разные виртуальные хосты на 443/TCP, поэтому 9443/TCP не требуется. При временном запуске без reverse proxy рабочим местам нужны прямые доступы к 9090/TCP и 9104/TCP. В промышленной конфигурации порты 9090 и 9100–9104 не должны быть доступны из внешней сети.

Исходящие соединения

Откуда	Куда	Порт	Когда требуется
Docker-сервер	Container Registry	443/TCP или порт registry, например 5050/TCP	Загрузка образов в контуре с registry.
Docker-сервер и backend	DNS-серверы	53/UDP, при необходимости 53/TCP	Разрешение имён.
Docker-сервер	NTP-сервер	123/UDP	Синхронизация времени.
Backend	LDAP/LDAPS	389/636 TCP	Аутентификация через LDAP/AD.
Backend	IMAP/IMAPS	143/993 TCP	Получение входящей почты.
Backend	SMTP	25/465/587 TCP	Отправка почты.
Backend	DaData	443/TCP	Подсказки организаций при включённой интеграции.
Рабочие места	Публичный S3 endpoint	443/TCP или 9443/TCP	Загрузка и скачивание файлов.

Если используется LDAP/AD, контейнеру backend должны быть доступны корпоративный DNS, контроллеры домена и требуемые LDAP/LDAPS-порты.

DNS и TLS

Для промышленной установки предпочтительно подготовьте два DNS-имени, например:

- `l1.example.org` — веб-интерфейс;
- `s3.l1.example.org` — S3 API.

> Оба адреса должны быть доступны с рабочих мест и защищены действующими TLS-сертификатами. Внутреннее имя контейнера `minio` нельзя использовать как публичный S3 endpoint.

Если выделено только одно DNS-имя, допустима публикация S3 API на отдельном TLS-порту:

- `https://l1.example.org` — веб-интерфейс, REST API и WebSocket;
- `https://l1.example.org:9443` — S3 API.

Один сертификат для `l1.example.org` можно использовать на обоих портах.

Reverse proxy должен разрешать загрузку файлов настроенного максимального размера, передавать заголовки `Host`, `X-Forwarded-For`, `X-Forwarded-Proto`, WebSocket-заголовки `Upgrade`, `Connection`, `Sec-WebSocket-Protocol` и поддерживать долгоживущие соединения приложения.

Развёртывание

Перед установкой ознакомьтесь с системными требованиями и подготовьте DNS, сетевые доступы и диски.

> **Предупреждение.** Быстрый мастер `install.sh` пока не поддерживает закрытый контур. Для закрытого контура используйте ручную установку из архивов образов, описанную ниже. Параметр `--offline` у быстрого мастера недоступен.

1. Получение комплекта поставки

Архив содержит каталог верхнего уровня `l1-support`:

```
l1-support
├── install.sh
├── docker-compose.yaml
└── .env.example
```

Также должны быть предоставлены точные версии образов backend и frontend. Для установки через Container Registry нужны учётные данные, если registry требует авторизацию. Для закрытого контура отдельно запросите у поставщика архивы всех образов из `docker-compose.yaml`, включая PostgreSQL, Redis, MinIO и MinIO Client.

Архив комплекта поставки: скачать `l1-support-release.zip`.

Ссылка открывает публичную папку в Облаке Mail. Выберите файл `l1-support-release.zip` и нажмите **«Скачать»**. Архив содержит установочные файлы для сценария с Container Registry; образы для закрытого контура в него не входят.

2. Подготовка каталога

2.1. Распакуйте комплект поставки

```
unzip ~/Downloads/l1-support-release.zip -d ~/l1-support-release
```

Команда создаст каталог `~/l1-support-release`. Внутри него должен находиться каталог `l1-support` с файлами `install.sh`, `docker-compose.yaml` и `.env.example`.

2.2. Создайте рабочий каталог приложения

```
sudo install -d -o "$USER" -g "$(id -gn)" /opt/l1-support
```

Эта команда создаёт постоянный рабочий каталог `/opt/l1-support` и назначает текущего пользователя его владельцем. Благодаря этому следующие команды и сам установщик можно запускать без sudo.

> Если приложение нужно разместить в другом месте, замените `/opt/l1-support` на выбранный абсолютный путь во всех последующих командах.

2.3. Скопируйте файлы приложения

```
cp -a ~/l1-support-release/l1-support/. /opt/l1-support/
cd /opt/l1-support
```

`/.` в конце исходного пути означает «скопировать всё содержимое каталога», включая скрытый файл `.env.example`. После копирования команда `cd` переводит терминал в рабочий каталог приложения.

2.4. Проверьте комплект поставки

Проверьте, что необходимые файлы находятся на месте:

```
ls -la
```

В выводе должны присутствовать `install.sh`, `docker-compose.yaml` и `.env.example`.

После подготовки каталога выберите один из двух способов установки ниже.

Быстрая установка

Быстрый интерактивный мастер доступен для серверов с доступом к Container Registry:

```
cd /opt/l1-support
bash install.sh
```

> **Важно.** Быстрый установщик не настраивает DNS, TLS-сертификаты и reverse proxy. При выборе HTTPS подготовьте их заранее в соответствии с разделом «Публикация через HTTPS».

В HTTPS-режиме мастер предложит выбрать два DNS-имени на стандартном порту 443 либо одно DNS-имя с отдельным S3-портом, по умолчанию 9443. Локальные порты frontend и MinIO при этом привязываются к `127.0.0.1`. В режиме прямого HTTP-доступа они привязываются к `0.0.0.0` и должны быть ограничены сетевыми правилами сервера.

Сервер с доступом к Container Registry

Если на сервере ещё нет действующих учётных данных, выполните авторизацию в Container Registry. Используйте выделенную учётную запись:

- логин: `robot$l1-support+trial-license`;
- пароль: `CYcAL6USeE2cH4JNOOAoTWJyKklLN7zX`.

Чтобы пароль не сохранился в истории команд, запустите авторизацию без указания пароля в командной строке:

```
docker login registry.legionsecurity.ru
```

После запроса `Username` введите логин, а после запроса `Password` — пароль, указанные выше.

После успешного завершения откройте показанный мастером Web-адрес и перейдите к первоначальной настройке.

Ручная установка

Ручной сценарий подходит как для Container Registry, так и для закрытого контура.

1. Настройка окружения

При первой ручной установке создайте `.env` из шаблона и ограничьте доступ к нему:

```
cp .env.example .env
chmod 600 .env
```

> При обновлении существующий `.env` не заменяется шаблоном из новой поставки. Новые и изменённые параметры нужно переносить вручную после сравнения файлов.

Полный перечень параметров приведён в справочнике конфигурации. До первого запуска обязательно настройте следующие группы в `.env`.

Версии образов

Укажите пути и точные неизменяемые версии:

```
BACKEND_IMAGE=registry.legionsecurity.ru/l1-support/l1-server
BACKEND_VERSION=v1.8.0
FRONTEND_IMAGE=registry.legionsecurity.ru/l1-support/l1-client
FRONTEND_VERSION=v1.8.0
```

Тег `latest` в промышленной установке не используется.

Публичные адреса

Укажите адреса, по которым frontend и файловое хранилище будут доступны с рабочих мест пользователей.

Для временного запуска без reverse proxy замените `192.168.1.10` на IP-адрес сервера:

```
FRONTEND_BIND_ADDRESS=0.0.0.0
FRONTEND_PUBLIC_URL=http://192.168.1.10:9090
MINIO_API_BIND_ADDRESS=0.0.0.0
S3_PUBLIC_ENDPOINT=http://192.168.1.10:9104
```

Порты 9090 и 9104 должны быть доступны с рабочих мест пользователей.

Для промышленной установки с двумя DNS-именами укажите:

```
FRONTEND_BIND_ADDRESS=127.0.0.1
FRONTEND_PUBLIC_URL=https://l1.example.org
MINIO_API_BIND_ADDRESS=127.0.0.1
S3_PUBLIC_ENDPOINT=https://s3.l1.example.org
```

Если заказчик выделил только одно DNS-имя, используйте разные HTTPS-порты:

```
FRONTEND_BIND_ADDRESS=127.0.0.1
FRONTEND_PUBLIC_URL=https://l1.example.org
MINIO_API_BIND_ADDRESS=127.0.0.1
S3_PUBLIC_ENDPOINT=https://l1.example.org:9443
```

В этом варианте reverse proxy принимает веб-интерфейс на стандартном порту 443, а S3 API — на отдельном порту 9443. Один TLS-сертификат для `l1.example.org` действует на обоих портах: номер порта не входит в сертификат.

DNS-имена должны разрешаться в IP-адрес reverse proxy, а указанные порты — быть доступны с рабочих мест пользователей. Не настраивайте HTTPS frontend вместе с HTTP-адресом S3: браузер может заблокировать загрузку и скачивание файлов как mixed content.

`S3_PUBLIC_ENDPOINT` должен быть доступен из браузеров пользователей. Укажите только базовый адрес — без имени бакета и дополнительного пути. Backend использует этот адрес при формировании подписанных URL, по которым браузеры напрямую загружают и скачивают файлы.

> Не используйте `localhost` или `127.0.0.1`: для пользователя они указывают на его собственный компьютер, а не на сервер.

Секреты

Для генерации секретов на сервере должен быть установлен `openssl`.

Сгенерируйте независимые значения для каждого секрета:

```
printf 'JWT_SECRET=%s\n' "$(openssl rand -hex 32)"
printf 'APP_SECRETS_KEY=%s\n' "$(openssl rand -base64 32)"
printf 'DB_PASSWORD=%s\n' "$(openssl rand -hex 24)"
printf 'REDIS_PASSWORD=%s\n' "$(openssl rand -hex 24)"
printf 'MINIO_ROOT_PASSWORD=%s\n' "$(openssl rand -hex 24)"
printf 'S3_SECRET_KEY=%s\n' "$(openssl rand -hex 24)"
```

Перенесите полученные значения в `.env`.

> Сохраните `.env` и `APP_SECRETS_KEY` в защищённом хранилище. При потере `APP_SECRETS_KEY` ранее зашифрованные настройки интеграций станут недоступны.

Корпоративный DNS

Этот раздел требуется, если для авторизации используется LDAP/Active Directory и контейнеру backend необходимо разрешать корпоративные доменные имена.

Замените значения на параметры корпоративной сети:

```
DOCKER_DNS_SERVER=192.168.20.100
DOCKER_DNS_SEARCH=corp.example.org
```

- `DOCKER_DNS_SERVER` — IP-адрес корпоративного DNS-сервера.
- `DOCKER_DNS_SEARCH` — домен, который будет автоматически добавляться при обращении по короткому имени, например `dc01` → `dc01.corp.example.org`.

DNS-сервер и контроллеры домена должны быть доступны с Docker-сервера и из контейнера backend. Если LDAP/Active Directory не используется или корпоративные имена уже разрешаются без дополнительной настройки, эти параметры можно не указывать.

Проверка конфигурации

```
grep -n 'CHANGE_ME' .env
ENVFILE=.env docker compose --env-file .env config --quiet
```

Строки в выводе `grep` означают, что соответствующие параметры необходимо заполнить. Команда `docker compose config` не должна возвращать ошибок.

2. Загрузка образов

Перед выполнением команд перейдите в каталог развёртывания `/opt/l1-support` или в каталог, выбранный при установке. В нём должны находиться `docker-compose.yaml` и настроенный файл `.env`.

Сервер с доступом к Container Registry

Если на сервере ещё нет действующих учётных данных, выполните авторизацию в Container Registry. Используйте выделенную учётную запись:

- логин: `robot$l1-support+trial-license`;
- пароль: `CYcAL6USeE2cH4JNOOAoTWJyKklLN7zX`.

Чтобы пароль не сохранился в истории команд, запустите авторизацию без указания пароля в командной строке:

```
docker login registry.legionsecurity.ru
```

После запроса `Username` введите логин, а после запроса `Password` — пароль, указанные выше.

Загрузите образы приложения:

```
docker compose pull
```

Повторная авторизация перед каждым запуском не требуется. Выполните `docker login` повторно, только если текущие учётные данные перестали действовать.

Закрытый контур: ручная загрузка образов

Быстрый мастер `install.sh` для этого сценария не используется. Запросите архивы образов у поставщика, передайте их на сервер разрешённым способом и загрузите каждый архив вручную:

```
for archive in /path/to/images/*.tar; do
  docker load --input "$archive"
done
```

Убедитесь, что загружены все образы и их теги совпадают со значениями в `.env` и `docker-compose.yaml`:

```
docker image ls
ENVFILE=.env docker compose --env-file .env config --images
```

В закрытом контуре не выполняйте `docker compose pull`. Если хотя бы один образ отсутствует или его тег не совпадает, запуск завершится ошибкой получения образа.

3. Запуск

```
docker compose up -d
docker compose ps -a
```

Ожидаемое состояние:

Сервис	Статус
`postgres`, `redis`, `minio`, `backend`, `frontend`	`Up` и `healthy`.
`migrate`, `minio-mc`	`Exited (0)`.

`migrate` и `minio-mc` являются одноразовыми сервисами. Их завершение с кодом `0` является штатным.

Если запуск не завершился успешно:

```
docker compose ps -a
docker compose logs --tail=200 migrate minio-mc backend
```

После исправления причины повторите `docker compose up -d`.

4. Проверка на Docker-сервере

```
curl -f http://127.0.0.1:9100/api/v1/system/bootstrap/status
curl -fI http://127.0.0.1:9090
curl -f http://127.0.0.1:9104/minio/health/live
```

Если локальные проверки проходят, а система недоступна с рабочего места, проверьте DNS, межсетевой экран, маршрутизацию и reverse proxy.

Публикация через HTTPS

Быстрый мастер записывает публичные адреса в `.env`, но не устанавливает и не настраивает DNS, TLS-сертификаты, межсетевой экран или reverse proxy. Эти компоненты настраиваются отдельно.

Варианты публикации

Вариант	Веб-интерфейс	S3 API	Входящие порты
Два DNS-имени	`https://l1.example.org`	`https://s3.l1.example.org`	`80`, `443`
Одно DNS-имя, разные порты	`https://l1.example.org`	`https://l1.example.org:9443`	`80`, `443`, `9443`

Порт `80` нужен для перенаправления на HTTPS и проверки HTTP-01 при выпуске и автоматическом продлении сертификата Let's Encrypt. Если используется другой способ проверки сертификата, например DNS-01, необходимость внешнего порта `80` определяется инфраструктурой заказчика.

Для варианта с одним DNS настройте NAT или внешний firewall:

```
PUBLIC_IP:80   -> DOCKER_SERVER_IP:80
PUBLIC_IP:443  -> DOCKER_SERVER_IP:443
PUBLIC_IP:9443 -> DOCKER_SERVER_IP:9443
```

Порты приложения `9090`, `9100`, `9101`, `9102`, `9103` и `9104` не публикуйте во внешнюю сеть. Reverse proxy на Docker-сервере обращается к ним локально.

Установка Nginx и Certbot

Этот пример предназначен для Ubuntu/Debian, когда Nginx установлен на том же сервере, что и Docker:

```
sudo apt-get update
sudo apt-get install -y nginx certbot python3-certbot-nginx
```

До запроса сертификата создайте HTTP-конфигурацию `/etc/nginx/sites-available/l1-support`:

```
server {
    listen 80;
    listen [::]:80;
    server_name l1.example.org;

    location / {
        proxy_pass http://127.0.0.1:9090;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Активируйте конфигурацию и убедитесь, что домен доступен извне по HTTP:

```
sudo ln -s /etc/nginx/sites-available/l1-support /etc/nginx/sites-enabled/l1-support
sudo nginx -t
sudo systemctl enable --now nginx
sudo systemctl reload nginx
```

Получите сертификат. Для одного DNS-имени:

```
sudo certbot certonly --nginx -d l1.example.org
```

Для двух DNS-имён выпустите один сертификат с обоими именами:

```
sudo certbot certonly --nginx -d l1.example.org -d s3.l1.example.org
```

Перед этой командой добавьте `s3.l1.example.org` в `server_name` временной HTTP-конфигурации. После выпуска проверьте фактические пути сертификата командой `sudo certbot certificates`. Если Certbot добавил суффикс, например `-0001`, используйте показанные им пути в итоговой конфигурации Nginx.

Если HTTP-01 завершается с `timeout` или `connection refused`, проверьте DNS, NAT и доступность входящего `80/TCP`.

Готовая конфигурация Nginx для одного DNS

После выпуска сертификата замените конфигурацию `l1-support` следующим содержимым. В примере API и WebSocket направляются непосредственно в backend, веб-интерфейс — во frontend, S3 API — в MinIO:

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {
```

```
    listen 80;
    listen [::]:80;
    server_name l1.example.org;

    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name l1.example.org;

    ssl_certificate /etc/letsencrypt/live/l1.example.org/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/l1.example.org/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /api/ {
        proxy_pass http://127.0.0.1:9100;
        proxy_http_version 1.1;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Sec-WebSocket-Protocol $http_sec_websocket_protocol;

        proxy_buffering off;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
    }

    location / {
        proxy_pass http://127.0.0.1:9090;
        proxy_http_version 1.1;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }
}

server {
    listen 9443 ssl;
    listen [::]:9443 ssl;
    server_name l1.example.org;

    ssl_certificate /etc/letsencrypt/live/l1.example.org/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/l1.example.org/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # Значение должно быть не меньше FILES_MAX_SIZE_BYTES.
    client_max_body_size 1g;

    location / {
        proxy_pass http://127.0.0.1:9104;
```

```
        proxy_http_version 1.1;

        # Для подписанных S3 URL важно сохранить исходный Host вместе с :9443.
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;

        proxy_request_buffering off;
        proxy_buffering off;
        proxy_read_timeout 3600s;
        proxy_send_timeout 3600s;
    }
}
```

Для схемы с двумя DNS измените последний блок: используйте `listen 443 ssl`, укажите `server_name s3.l1.example.org` и пути к сертификату, содержащему оба DNS-имени.

Проверьте и примените конфигурацию:

```
sudo nginx -t
sudo systemctl reload nginx
sudo ss -lntp | grep -E ':(443|9443)[[:space:]]'
```

Настройка приложения и проверка

Для одного DNS укажите в `.env`:

```
FRONTEND_BIND_ADDRESS=127.0.0.1
FRONTEND_PUBLIC_URL=https://l1.example.org
MINIO_API_BIND_ADDRESS=127.0.0.1
S3_PUBLIC_ENDPOINT=https://l1.example.org:9443
```

Примените изменения:

```
docker compose up -d --force-recreate backend frontend
docker compose ps -a
```

Проверьте HTTPS и автоматическое продление сертификата:

```
curl -fI https://l1.example.org
curl -f https://l1.example.org:9443/minio/health/live
sudo certbot renew --dry-run
```

В браузере дополнительно проверьте:

- WebSocket `/api/v1/notifications/live` возвращает `101 Switching Protocols`;
- загрузка файла выполняется методом PUT по адресу из `S3_PUBLIC_ENDPOINT` и возвращает `200 OK`;
- скачивание ранее загруженного файла работает;
- URL загрузки и скачивания начинается с `https://`, а не с внутреннего IP-адреса.

Публичные адреса reverse proxy должны в точности совпадать со значениями `FRONTEND_PUBLIC_URL` и `S3_PUBLIC_ENDPOINT`.

Первоначальная настройка

1. Откройте `FRONTEND_PUBLIC_URL` в браузере.
2. Создайте первого администратора на начальной странице.
3. Войдите в систему.
4. Загрузите и скачайте тестовый файл.
5. Настройте почту и используемые провайдеры аутентификации.
6. Проверьте внешнее резервное копирование PostgreSQL и MinIO.

Дальнейшие процедуры приведены в руководстве по эксплуатации.

Конфигурация

Основная конфигурация развёртывания находится в файле `.env`. Создавайте его только из `.env.example` той же версии поставки.

Правила работы с `.env`

- Используйте разные случайные значения для всех паролей и ключей.
- Не меняйте APP_SECRETS_KEY после первоначальной настройки без отдельной процедуры ротации.
- Не используйте тег образа `latest` в промышленной среде.
- Перед запуском проверяйте конфигурацию командой `ENVFILE=.env docker compose --env-file .env config --quiet`.
- При обновлении сравнивайте новый `.env.example` с действующим `.env`; не заменяйте рабочий файл целиком.

Приложение

Переменная	Назначение	Пример
BACKEND_IMAGE	Образ серверной части.	registry.legionsecurity.ru/l1-support/l1-server
BACKEND_VERSION	Точная версия серверной части.	v1.8.0
FRONTEND_IMAGE	Образ клиентской части.	registry.legionsecurity.ru/l1-support/l1-client
FRONTEND_VERSION	Точная версия клиентской части.	v1.8.0
SERVICE_NAME	Имя сервиса в журналах.	l1-support
APP_ENV	Режим работы.	prod
SERVER_ADDRESS	Адрес прослушивания внутри контейнера.	0.0.0.0
SERVER_PORT	Внутренний порт backend.	3000
SERVER_HOST_PORT	Диагностический порт backend на localhost.	9100
FRONTEND_BIND_ADDRESS	Адрес публикации frontend на хосте.	127.0.0.1
FRONTEND_HOST_PORT	Порт веб-интерфейса на хосте.	9090
FRONTEND_PUBLIC_URL	Публичный адрес веб-интерфейса.	https://l1.example.org
DOCKER_DNS_SERVER	Корпоративный DNS-сервер для контейнера backend. Используется для LDAP/AD	192.168.20.100

Переменная	Назначение	Пример
DOCKER_DNS_SEARCH	Поисковый домен DNS.	corp.example.org

PostgreSQL

Переменная	Назначение	Пример
DB_TYPE	Тип базы данных.	postgres
DB_HOST	Имя сервиса PostgreSQL.	postgres
DB_PORT	Внутренний порт PostgreSQL.	5432
DB_USER	Пользователь базы данных.	l1_support
DB_PASSWORD	Пароль пользователя.	<случайный_пароль>
DB_NAME	Имя базы данных.	l1_support
DB_HOST_PORT	Диагностический порт PostgreSQL на localhost.	9101

Redis

Переменная	Назначение	Пример
REDIS_HOST	Имя сервиса Redis.	redis
REDIS_PORT	Внутренний порт Redis.	6379
REDIS_HOST_PORT	Диагностический порт Redis на localhost.	9102
REDIS_PASSWORD	Пароль Redis.	<случайный_пароль>
REDIS_DB	Номер логической базы.	0
REDIS_MAX_RETRIES	Число повторов подключения.	3
REDIS_DIAL_TIMEOUT	Тайм-аут подключения.	5s
REDIS_READ_TIMEOUT	Тайм-аут чтения.	3s
REDIS_WRITE_TIMEOUT	Тайм-аут записи.	3s

MinIO и S3

Переменная	Назначение	Пример
MINIO_ROOT_USER	Администратор MinIO.	l1-minio-admin
MINIO_ROOT_PASSWORD	Пароль администратора MinIO.	<случайный_пароль>
MINIO_API_BIND_ADDRESS	Адрес публикации S3 API на хосте.	127.0.0.1

Переменная	Назначение	Пример
MINIO_API_HOST_PORT	Порт S3 API на хосте.	9104
MINIO_CONSOLE_HOST_PORT	Порт консоли MinIO на localhost.	9103
S3_BUCKET	Имя приватного бакета.	l1-files
S3_REGION	Регион S3.	us-east-1
S3_ACCESS_KEY	Технический пользователь backend.	l1-app
S3_SECRET_KEY	Пароль технического пользователя.	<случайный_пароль>
S3_ENDPOINT	Внутренний адрес MinIO для backend.	http://minio:9000
S3_PUBLIC_ENDPOINT	Базовый адрес S3 API в подписанных URL.	https://s3.l1.example.org
S3_USE_PATH_STYLE	Использование path-style адресов S3.	true

В штатной поставке используется MinIO из docker-compose.yaml:

- backend обращается к хранилищу по внутреннему адресу S3_ENDPOINT; для встроенного MinIO оставьте http://minio:9000;
- backend использует S3_PUBLIC_ENDPOINT при формировании временных подписанных URL, по которым браузеры напрямую загружают и скачивают файлы;
- S3_PUBLIC_ENDPOINT указывается без имени бакета, пути к объекту и служебного пути /minio/health/live, например https://s3.l1.example.org, https://l1.example.org:9443 или http://192.168.1.10:9104;
- S3_USE_PATH_STYLE=true требуется для встроенного MinIO.

Если FRONTEND_PUBLIC_URL использует HTTPS, S3_PUBLIC_ENDPOINT также должен использовать HTTPS, иначе браузер может заблокировать файловые запросы как mixed content. При одном DNS-имени задайте разные порты, например https://l1.example.org для frontend и https://l1.example.org:9443 для S3. Нельзя указывать одинаковый адрес для frontend и S3: оба сервиса используют корневые URL и не разделяются дополнительным путём.

При локальном reverse proxy оставьте FRONTEND_BIND_ADDRESS=127.0.0.1 и MINIO_API_BIND_ADDRESS=127.0.0.1. Значение 0.0.0.0 используйте только для временного прямого HTTP-доступа или когда доступ к портам отдельно ограничен внешним межсетевым экраном.

Сервис minio-mc при запуске автоматически создаёт приватный бакет S3_BUCKET, технического пользователя S3_ACCESS_KEY и назначает ему политику readwrite. Backend работает с хранилищем под этим техническим пользователем. Пара MINIO_ROOT_USER/MINIO_ROOT_PASSWORD используется для администрирования MinIO и первоначальной настройки; не используйте её вместо S3_ACCESS_KEY/S3_SECRET_KEY.

S3_ENDPOINT и S3_PUBLIC_ENDPOINT не взаимозаменяемы. Публичный адрес должен быть доступен с каждого рабочего места пользователя. Подключение внешнего S3-

совместимого хранилища требует отдельной конфигурации Compose и не входит в штатную схему развёртывания.

Аутентификация и шифрование

Переменная	Назначение	Пример
JWT_SECRET	Секрет подписи JWT.	<случайный_секрет>
ACCESS_TOKEN_TTL	Время жизни access token.	5m
REFRESH_TOKEN_TTL	Время жизни refresh token.	720h
PASSWORD_RESET_TOKEN_TTL	Время жизни токена восстановления пароля.	30m
APP_SECRETS_KEY	Base64-ключ шифрования настроек.	результат openssl rand -base64 32
APP_TOTP_ISSUER	Название в приложении-аутентификаторе.	Л1 Первая Линия Техподдержки

> Значение APP_SECRETS_KEY должно декодироваться в 32 байта. Его потеря или несогласованная замена сделает сохранённые зашифрованные пароли интеграций недоступными.

Файловый модуль

Переменная	Назначение	Пример
FILES_CONFIG_PATH	Путь к конфигурации файлов внутри контейнера.	config/files.json
FILES_CONFIG_RELOAD_INTERVAL	Интервал перечитывания конфигурации.	30s
FILES_MAX_SIZE_BYTES	Максимальный размер одного файла.	500mb
FILES_MULTIPART_THRESHOLD_BYTES	Порог multipart-загрузки.	50mb
FILES_MULTIPART_PART_SIZE_BYTES	Размер части multipart-загрузки.	16mb
FILES_UPLOAD_URL_TTL_SECONDS	Время жизни ссылки на загрузку.	3600
FILES_DOWNLOAD_URL_TTL_SECONDS	Время жизни ссылки на скачивание.	300
FILES_CLEANUP_INTERVAL	Интервал очистки незавершённых загрузок.	20m
FILES_CLEANUP_BATCH_SIZE	Размер пачки при очистке.	100

Ограничение размера запроса на reverse proxy должно быть не меньше FILES_MAX_SIZE_BYTES.

Интеграции и фоновые обработчики

Переменная	Назначение	Пример
APPEARANCE_CONFIG_PATH	Конфигурация внешнего вида.	config/appearance.json
NOTIFICATION_TEMPLATES_CONFIG_PATH	Шаблоны уведомлений.	config/notification_templates.json
NOTIFICATION_TEMPLATES_RELOAD_INTERVAL	Интервал перечитывания шаблонов.	30s
DADATA_DEFAULT_TOKEN	Токен DaData.	пусто
DADATA_BASE_URL	Адрес API DaData.	https://suggestions.dadata.ru
DADATA_TIMEOUT	Тайм-аут обращения к DaData.	5s
DADATA_CONFIG_PATH	Конфигурация DaData.	config/dadata.json
DADATA_CONFIG_RELOAD_INTERVAL	Интервал перечитывания конфигурации.	30s
EMAIL_WORKERS_ENABLED	Обработка электронной почты.	true
EMAIL_IMAP_POLL_INTERVAL	Интервал проверки входящей почты.	30s
EMAIL_OUTBOUND_POLL_INTERVAL	Интервал отправки исходящей почты.	30s
EMAIL_WORKER_BATCH_SIZE	Размер пачки почтовых сообщений.	100
EMAIL_LOCK_TIMEOUT	Время блокировки почтовой задачи.	15m
SLA_WORKER_ENABLED	Обработка SLA.	true
SLA_POLL_INTERVAL	Интервал обработки SLA.	30s
SLA_WORKER_BATCH_SIZE	Размер пачки SLA-записей.	100
REPORTS_EXPORT_RETENTION	Срок хранения выгруженных отчётов.	720h
REPORTS_EXPORT_MAX_PER_USER	Максимум выгрузок на пользователя.	50
REPORTS_EXPORT_CLEANUP_INTERVAL	Интервал очистки истории выгрузок.	1h
REPORTS_EXPORT_CLEANUP_BATCH_SIZE	Размер пачки при очистке выгрузок.	100

Параметры конкретных LDAP, IMAP и SMTP-подключений задаются в интерфейсе системы. До их включения обеспечьте сетевые доступы, перечисленные в системных требованиях.

Эксплуатация

Все команды выполняются из каталога установки. В текущем описании указан `/opt/l1-support`.

Проверка состояния

```
cd /opt/l1-support
docker compose ps -a
docker compose logs --tail=200
```

Постоянные сервисы `postgres`, `redis`, `minio`, `backend` и `frontend` должны быть `Up` и `healthy`. Состояние `Exited (0)` для одноразовых сервисов `migrate` и `minio-mc` является штатным.

Проверка локальных endpoint:

```
curl -f http://127.0.0.1:9100/api/v1/system/bootstrap/status
curl -fI http://127.0.0.1:9090
curl -f http://127.0.0.1:9104/minio/health/live
```

Если система опубликована через HTTPS, дополнительно проверьте публичные endpoint и продление сертификата:

```
curl -fI https://l1.example.org
curl -f https://l1.example.org:9443/minio/health/live
sudo certbot renew --dry-run
```

Замените адреса в примере на фактические значения `FRONTEND_PUBLIC_URL` и `S3_PUBLIC_ENDPOINT` из `.env`.

Журналы

```
docker compose logs --tail=200
docker compose logs -f backend
docker compose logs -f postgres redis minio
docker compose logs --since=30m backend
```

Не передавайте `.env`, токены, cookies, пароли и полные диагностические архивы третьим лицам без проверки содержимого.

Остановка и запуск

Остановить и запустить существующие контейнеры:

```
docker compose stop
docker compose start
```

Применить изменённую конфигурацию или пересоздать необходимые контейнеры:

```
ENVFILE=.env docker compose --env-file .env config --quiet
docker compose up -d
```

Не используйте `docker compose down -v`: параметр `-v` удаляет volumes PostgreSQL, MinIO, конфигурации и журналов backend вместе с данными.

Резервное копирование

В резервную копию должны входить:

- дамп PostgreSQL;
- содержимое бакета MinIO;
- `.env` и неизменяемый `APP_SECRETS_KEY`;
- содержимое volume `l1-support-backend-config`;
- номера версий backend и frontend.

Redis содержит временные данные и отдельно не резервируется.

PostgreSQL

```
docker compose exec -T postgres sh -c \
  'pg_dump -U "$POSTGRES_USER" -d "$POSTGRES_DB" --format=custom' \
  > l1-support-postgres-$(date +%F-%H%M%S).dump
```

Команда должна завершиться с кодом `0`, а полученный файл — иметь ненулевой размер.

MinIO

Данные находятся в volume `minio_data`. Настройте регулярное копирование бакета во внешнее S3-совместимое или файловое хранилище, например средствами `mc mirror` или корпоративной системы резервного копирования.

> Не ограничивайтесь копированием файлов активного Docker volume без остановки MinIO: такая копия может быть несогласованной.

Требования к резервным копиям

- PostgreSQL и MinIO должны относиться к одному согласованному периоду.
- Копии должны храниться за пределами Docker-сервера.
- Доступ к `.env` и `APP_SECRETS_KEY` должен быть ограничен.
- Срок хранения и допустимая потеря данных должны соответствовать принятым RPO/RTO.
- Восстановление необходимо регулярно проверять на отдельном стенде.

Для строгой согласованности остановите запись данных на время создания обеих копий:

```
docker compose stop backend frontend
# Создайте дамп PostgreSQL и копию бакета MinIO.
docker compose start backend frontend
```

Восстановление

> Перед восстановлением сохраните текущее состояние и убедитесь, что выбраны согласованные копии PostgreSQL и MinIO.

Остановите приложение:

```
docker compose stop backend frontend
```

Восстановите PostgreSQL:

```
docker compose exec -T postgres sh -c \
  'pg_restore -U "$POSTGRES_USER" -d "$POSTGRES_DB" --clean --if-exists' \
  < l1-support-postgres-YYYY-MM-DD-HHMMSS.dump
```

Восстановите бакет MinIO из соответствующей копии принятой в организации процедурой, затем запустите и проверьте систему:

```
docker compose start backend frontend
docker compose ps -a
```

Проверьте API, веб-интерфейс, открытие существующего вложения и загрузку нового файла.

Обновление

Перед обновлением:

1. Зафиксируйте текущие `BACKEND_VERSION` и `FRONTEND_VERSION`.
2. Создайте согласованные резервные копии PostgreSQL и MinIO.
3. Сравните новый `.env.example` с действующим `.env`.
4. Проверьте примечания к релизу и совместимость отката схемы БД.

После переноса новых файлов поставки:

```
cd /opt/l1-support
ENVFILE=.env docker compose --env-file .env config --quiet
```

В контуре с registry загрузите новые образы:

```
docker compose pull backend frontend migrate
```

В закрытом контуре вместо `pull` загрузите отдельно запрошенные у поставщика архивы образов через `docker load` и проверьте теги командой `docker compose config --images`.

Примените обновление:

```
docker compose up -d
docker compose ps -a
```

Compose применит миграции до запуска новой версии backend. После обновления проверьте API, веб-интерфейс, загрузку и скачивание файлов, почтовые обработчики и используемые провайдеры аутентификации.

Откат

> Откат только образов допустим, если предыдущая версия приложения совместима с уже применённой схемой базы данных.

Если схема совместима, верните предыдущие версии в `.env`, обеспечьте наличие образов на сервере и выполните:

```
docker compose up -d --no-deps backend frontend
docker compose ps -a
```

В контуре с registry отсутствующие предыдущие образы можно предварительно получить командой `docker compose pull backend frontend`.

> Если предыдущая версия несовместима с новой схемой, остановите приложение и восстановите согласованные резервные копии PostgreSQL и MinIO. Не запускайте down-миграции без явно предусмотренной для релиза процедуры.

Диагностика

Проблема	Что проверить	Команда или действие
`pull access denied` или `manifest unknown`	Путь к образу, точную версию и авторизацию в registry.	`docker compose config --images`, `docker login <registry>`.
В закрытом контуре выполняется обращение к registry	На сервер загружены не все образы или не совпадают теги.	`docker image ls`, `docker compose config --images`.
`migrate` завершился с `Exited (1)`	Параметры PostgreSQL, готовность БД и журнал миграций.	`docker compose logs --tail=200 migrate postgres`.
Изменения `.env` не применились	Конфигурацию Compose и состояние контейнеров.	Проверить и применить изменённую конфигурацию.

Проблема	Что проверить	Команда или действие
Backend имеет состояние `unhealthy`	PostgreSQL, Redis, MinIO, завершение миграций и значения `.env`.	`docker compose ps -a`, `docker compose logs --tail=200 backend`.
Интерфейс работает, файлы не загружаются	`S3_PUBLIC_ENDPOINT`, DNS, TLS, лимит reverse proxy и доступ рабочих мест к S3.	Открыть `${S3_PUBLIC_ENDPOINT}/minio/health/live` с рабочего места.
WebSocket `notifications/live` возвращает `401`	Передачу `Upgrade`, `Connection` и `Sec-WebSocket-Protocol` через все proxy.	Проверить запрос в DevTools и access log reverse proxy; ожидается `101`.
WebSocket возвращает `400`, `426` или `502`	HTTP/1.1, WebSocket upgrade и доступ reverse proxy к backend `127.0.0.1:9100`.	Сверить Nginx с разделом публикации через HTTPS.
Система работает локально, но недоступна пользователям	DNS, firewall, reverse proxy и совпадение публичных URL.	Сравнить `.env` с конфигурацией reverse proxy.
LDAP не подключается	Корпоративный DNS, маршрутизацию, сертификат и порты `389/636`.	Проверить разрешение имени и TCP-доступ из сети Docker-сервера.
Письма не приходят или не отправляются	IMAP/SMTP, DNS, TLS и журналы почтовых обработчиков.	`docker compose logs --since=30m backend`.
После перезагрузки сервисы не запустились	Автозапуск Docker и состояние контейнеров.	`systemctl status docker`, `docker compose ps -a`.
На диске заканчивается место	Docker Root Dir, volumes, журналы и незавершённые загрузки.	`df -h`, `docker system df`, `docker volume inspect ...`.

Если проблема сохраняется, соберите версии образов, вывод `docker compose ps -a`, относящийся к ошибке фрагмент журналов и время возникновения. Перед передачей материалов удалите секреты и персональные данные.

Обратиться в техническую поддержку можно по адресу 1ltp.dev@legionsecurity.ru.