

Инструкция системного администратора — Wialon Message Hub

Раздел эксплуатации сервера (не веб-панели). Составлен по фактическому Dockerfile, docker-compose.yml, nginx/nginx.conf, app/core/config.py, действующей эксплуатации на VPS 93.188.206.185. Реальные значения секретов текущего продакшн-сервера не публикуются.

1. Операционная система

Ubuntu (VPS-хостинг). Требования к конкретной версии ОС в проекте не зафиксировано — единственная жёсткая зависимость хоста — установленный Docker Engine с поддержкой Docker Compose v2 (команда `docker compose`, не устаревшая `docker-compose`).

2. Системные требования

Фактически эксплуатируется на VPS с **4 vCPU** и **8 ГБ RAM**. Это ограничение существенно: параллельная тяжёлая пересборка Docker-образов (например, одновременная загрузка браузерного бинарника Playwright) поднимала load average до 25+ на этом сервере и на 20+ минут делала сервис недоступным. Рекомендация для эксплуатации: не запускать тяжёлые операции (полная пересборка всех образов `--no-cache`, установка браузерных зависимостей) параллельно с работающим продакшеном; собирать короткими шагами по одному сервису.

3. Директория установки

Приложение и docker-compose-манифест разворачиваются в отдельной директории на сервере (в текущей эксплуатации — `/opt/wmg`, подтверждено фактическим расположением на боевом сервере в ходе эксплуатации; в самом репозитории этот путь нигде не захардкожен — при развёртывании на другом сервере может быть любым).

4. Конфигурационные файлы

Файл	Назначение
<code>.env</code> (в корне, создаётся на сервере вручную по образцу <code>.env.example</code> , не коммитится в git)	Все секреты и параметры окружения
<code>docker-compose.yml</code>	Определение всех сервисов, портов, зависимостей
<code>nginx/nginx.conf</code>	Конфигурация обратного прокси
<code>alembic.ini</code>	Конфигурация Alembic (адрес БД читается через <code>.env</code>)

5. Переменные окружения

Полный список (`.env.example`, `app/core/config.py`):

```

POSTGRES_HOST=db
POSTGRES_PORT=5432
POSTGRES_DB=wmg
POSTGRES_USER=wmg
POSTGRES_PASSWORD=<пароль БД>

# Опционально, переопределяет POSTGRES_* целиком:
# DATABASE_URL=postgresql+asyncpg://wmg:<пароль>@db:5432/wmg

APP_MASTER_KEY=<ключ Fernet, генерируется Fernet.generate_key()>
SESSION_SECRET_KEY=<произвольная случайная строка>

PUBLIC_BASE_URL=http://<адрес сервера>
WEB_PORT=8080
LOG_LEVEL=INFO

```

APP_MASTER_KEY **критичен** — это ключ шифрования всех секретов компаний (пароли и токены Wialon, токены Telegram и MAX) в базе данных. При его утере или замене расшифровать уже сохранённые секреты станет невозможно — их придётся вводить заново для каждой компании. Ключ должен резервироваться отдельно от резервной копии самой базы данных.

6. База данных

PostgreSQL 16 (образ postgres:16-alpine), поднимается собственным контейнером db в составе docker-compose.yml, данные — в именованном Docker-томе db_data (персистентны между пересозданием контейнера, но физически хранятся на диске того же хоста).

7. Необходимые порты

Порт	Кем используется	Публикуется наружу
80	nginx	Да — единственный публично открытый порт
8080	web (uvicorn)	Нет, только expose внутри Docker-сети
5432	db (PostgreSQL)	Нет

8. Доменное имя

В текущей эксплуатации сервис доступен по голому IP-адресу (<http://93.188.206.185>), без выделенного домена.

9. SSL / TLS

Не настроен. nginx.conf слушает только порт 80, директив listen 443/ssl_certificate нет. Сервис работает по обычному HTTP. Подробности и риски — см. документ «Выявленные проблемы и рекомендации по развитию».

10. Запуск

Первый запуск на новом сервере:

```
git clone <репозиторий> /opt/wmg
cd /opt/wmg
cp .env.example .env
# отредактировать .env: задать пароли БД, сгенерировать APP_MASTER_KEY и SESSION_SECRET_KEY
docker compose up -d --build
```

Порядок старта контейнеров определяется зависимостями в `docker-compose.yml`: `db` (дожидается `healthcheck`) → `migrate` (однократно накатывает миграции, `alembic upgrade head`) → `web` / `collector` / `dispatcher` (стартуют только после успешного завершения `migrate`) → `nginx`.

11. Остановка

```
docker compose down
```

Останавливает все контейнеры, том `db_data` (данные PostgreSQL) сохраняется. Для полной остановки без удаления состояния можно использовать `docker compose stop`.

12. Перезапуск

Полный перезапуск:

```
docker compose restart
```

Перезапуск отдельного сервиса (например, после точечного обновления кода одного компонента):

```
docker compose up -d --build <имя_сервиса>
# например: docker compose up -d --build dispatcher
```

13. Автозапуск

Все долгоживущие сервисы (`db`, `web`, `collector`, `dispatcher`, `nginx`) объявлены с `restart: unless-stopped` — Docker автоматически поднимает их при перезапуске хоста или падении процесса, если только они не были остановлены явной командой. Сервис `migrate` — одноразовая задача, политики перезапуска не имеет.

14. Резервное копирование

В текущей версии не реализовано — ни встроенного механизма в приложении, ни `cron`-задачи, ни отдельного сервиса резервного копирования в `docker-compose.yml` нет. Это открытый пункт, вынесенный отдельно в документ «Выявленные проблемы и рекомендации по развитию».

Рекомендуемая процедура на переходный период (до появления штатного механизма):

Создание резервной копии

```
docker compose exec db pg_dump -U wmg wmg > backup_$(date +%F).sql
cp /opt/wmg/.env /secure/backup/location/.env.$(date +%F)
```

Резервировать нужно **и** дампы базы, **и** файл `.env` (в первую очередь — значение `APP_MASTER_KEY`, без него расшифровать секреты из бэкапа базы будет невозможно).

Восстановление из резервной копии

```
docker compose exec -T db psql -U wmg wmg < backup_2026-08-22.sql
```

При восстановлении на новый сервер убедитесь, что восстановленный `.env` содержит тот же `APP_MASTER_KEY`, что был активен на момент создания дампа.

15. Расположение логов

Логи всех процессов идут в `stdout` контейнеров (простой текстовый формат `%(asctime)s - %(name)s - %(levelname)s - %(message)s`), просматриваются штатной командой Docker:

```
docker compose logs -f web
docker compose logs -f collector
docker compose logs -f dispatcher
```

Отдельно от системных логов Docker, в самой базе данных ведётся таблица `system_log` — она же отображается в разделе «Журнал» веб-панели и переживает пересоздание контейнеров.

16. Обновление приложения

```
cd /opt/wmg
git pull # либо иная доставка изменённых файлов
docker compose build web collector dispatcher
docker compose up -d --no-build web collector dispatcher
```

Новые миграции применяются автоматически при следующем `docker compose up`, так как сервис `migrate` перезапускается вместе с остальными и выполняет `alembic upgrade head` до старта `web/collector/dispatcher`.

На проде с ограниченными ресурсами (см. раздел 2) собирайте образы по одному сервису, не запускайте `docker compose build` без аргументов на все сервисы одновременно при высокой боевой нагрузке.

17. Известное эксплуатационное ограничение — Playwright/Chromium

Пакет `playwright` установлен в образе (`requirements.txt`), но браузерный бинарник Chromium **намеренно не устанавливается** при сборке (закомментировано в `Dockerfile` с пояснением о перегрузке CPU при установке). Это означает, что функция автоматического получения токена Wialon по логину/паролю (веб-кнопка «Получить токен по логину и паролю» и авто-переоформление токена внутри Collector) в текущем боевом образе **упадёт в рантайме** при попытке реального запуска браузера, пока Chromium не будет установлен отдельным контролируемым шагом:

```
docker compose exec web python -m playwright install chromium
docker compose exec collector python -m playwright install chromium
```

Выполнять эту команду рекомендуется в период низкой нагрузки на сервер, не параллельно с активной сборкой других образов.