Инструкция по установке экземпляра программного обеспечения «М-Сервис» для экспертизы

Содержание

1. Назначение документа

Настоящая инструкция описывает порядок установки и первичной настройки экземпляра программного обеспечения «М-Сервис» для проведения экспертизы. Документ предназначен для технических специалистов, осуществляющих развертывание системы в тестовой инфраструктуре.

2. Состав поставки

Для развертывания экземпляра ПО «М-Сервис» правообладатель предоставляет:

1. Docker-образы:
    - образ backend-сервиса: `m-service-backend-core:<tag>` (например: `m-service-backend-core:latest` или конкретный тег сборки);
    - образ frontend-приложения: `m-service-frontend:<tag>` (например: `m-service-frontend:latest`);
    - образ PostgreSQL (опционально, при необходимости отдельного образа с преднастроенной версией): `postgres:<version>`;
    - образ MinIO (при использовании собственного образа): `minio:<version>`.
2. SQL-скрипты и/или миграции:
    - каталог `migrations/` внутри backend-проекта (Doctrine Migrations);
    - при необходимости — SQL-скрипты для инициализации схемы БД и минимального набора тестовых данных.
3. Файл примера настроек окружения backend:
    - `backend/.env.example` (или аналогичный файл с перечнем переменных окружения).
4. Файл примера настроек окружения frontend:**
    - frontend/.env.example` / `.env.local.example` (перечень `NEXT_PUBLIC_*` переменных).
5. Описание Kubernetes- или Docker-манифестов (по согласованию):
    - манифесты Kubernetes (Deployment/Service/Ingress/CronJob) могут поставляться отдельным пакетом (например, каталог `manifests/`), либо правообладатель предоставляет упрощенную схему развертывания на одном сервере.
6. Тестовые учетные данные:
    - логин/пароль тестового администратора;
    - логин/пароль тестового клиентского пользователя;
    - учётные данные тестового юнита (плеера).

3. Требования к среде

3.1. Аппаратные требования

Минимальные требования к серверу для тестового развертывания:

- CPU: не менее 4 vCPU;
- RAM: не менее 8 GB;
- Диск: не менее 80 GB SSD;
- Постоянное подключение к сети Интернет.

3.2. Программные требования

- ОС: Ubuntu 22.04 LTS или эквивалентная;
- Docker (актуальная стабильная версия);

- Односерверный кластер Kubernetes (k3s/kubeadm/minikube в режиме single-node) или эквивалентная контейнерная среда;
- `kubectl` для управления кластером;
- при использовании Kubernetes-манифестов — установленный Ingress-контроллер (например, nginx-ingress).

4. Подготовка Docker-образов

Если образы переданы как `*.tar`:

```bash
docker load -i m-service-backend-core.tar
docker load -i m-service-frontend.tar
# при необходимости:
docker load -i minio.tar
docker load -i postgres.tar
```

После загрузки: ```bash docker images | grep m-service```
Убедиться, что образы `m-service-backend-core` и `m-service-frontend` доступны с требуемыми тегами.

5. Развертывание базы данных PostgreSQL

1. Запустить контейнер/под PostgreSQL (через Kubernetes или docker-compose) с указанием:
    a. имени базы данных (например, `mservice`);
    b. пользователя (например, `mservice`);
    c. пароля.
2. Конфигурация подключения задаётся в backend-переменной:
    a. `DATABASE_URL=postgresql://mservice:password@postgres:5432/mservice`
    b. `DATABASE_SERVER_VERSION=<версия PostgreSQL>` (например, `15`).
3. После запуска БД необходимо применить миграции Doctrine. Варианты:
    a. непосредственно в контейнере backend: ```bash php bin/console doctrine:migrations:migrate latest --no-interaction```
    b. либо через Makefile (при наличии): ```bash make migrate```

Миграции находятся в каталоге `migrations/` и автоматически создают всю необходимую структуру базы данных. Фикстуры не используются, структура и служебные данные формируются миграциями.

6. Настройка S3-хранилища (MinIO)

1. Запустить сервис MinIO (контейнер или pod) с заданными:
    - access key / secret key;
    - endpoint (например, `http://minio:9000`);
    - именем bucket (например, `mservice-audio`).
2. В backend прописать переменные окружения:
    - `S3_BUCKET` — имя bucket (например, `mservice-audio`);
    - `S3_PATH` — базовый путь внутри bucket (может быть пустым);
    - `S3_VERSION` — версия API (`latest` или используемая версия);
    - `S3_REGION` — регион (для MinIO может быть произвольным);
    - `S3_ACCESS_KEY` — access key;

    - `S3_SECRET_KEY` — secret key;
    - `S3_ENDPOINT` — endpoint (например, `http://minio:9000`);
    - `S3_USE_PATH_STYLE_ENDPOINT=true` — для MinIO, как правило, включается path-style.

7. Базовые переменные окружения backend

При запуске экземпляра backend необходимо задать (как минимум):

- Общесистемные:**
    - `APP_ENV=prod`
    - `APP_DEBUG=0`
    - `APP_SECRET=<случайная_строка>`
    - `APP_LOCALE=ru` (по умолчанию)
    - `APP_DEFAULT_TIMEZONE=Europe/Moscow` (или используемый часовой пояс)
- База данных:**
    - `DATABASE_URL=postgresql://user:password@host:port/dbname`
    - `DATABASE_SERVER_VERSION=<версия PostgreSQL>`
- JWT:**
    - `JWT_SECRET_KEY` — ключ подписи JWT;
    - `JWT_PASSPHRASE` — при использовании зашифрованных ключей;
    - `JWT_LIFETIME` — время жизни access-токена.
- S3/MinIO: (см. раздел 6 выше)
- WebSocket (при использовании отдельного сервера):**
    - `WEB_SOCKET_DSN` — DSN;
    - `WEB_SOCKET_SERVER_URI` — URI WebSocket-сервера.
- Почта (при необходимости отправки писем):**
    - `MAILER_DSN`
    - `MAILER_DEFAULT_FROM_EMAIL`
    - `MAILER_DEFAULT_FROM_NAME`
- HTTP/Frontend:**
    - `HTTP_PROTOCOL=http` или `https`;
    - `REQUEST_HOST=<host backend-части>`;
    - `FRONT_URN=<относительный путь фронтенда, при необходимости>`.

Остальные переменные (`SLACK_*`, `SENTRY_DSN`, `WEB_ANALYTICS_ENABLED` и т.п.) могут оставаться не заданными в тестовом окружении, если соответствующий функционал не используется при экспертизе.

8. Переменные окружения frontend

Frontend (Next.js) использует набор переменных с префиксом `NEXT_PUBLIC_`, которые встраиваются в сборку:
Обязательные/ключевые:

- `NEXT_PUBLIC_API_BASE_URL`
    - Базовый URL для API (REST, GraphQL, WebSocket).
    - Пример: `https://mservice.example.com`
    - На основе этого URL формируются пути `/api`, `/graphql`, `/ws`.

Рекомендуемые:

- `NEXT_PUBLIC_LOGS_ENABLED` — включение логирования на фронте (`true/false`);
- `NEXT_PUBLIC_ENABLE_METRICS` — включение сбора метрик (`true/false`);

- `NEXT_PUBLIC_PLAYER_DEV_TOOLS` — dev-инструменты для плеера (обычно `false` в prod);
- `NEXT_PUBLIC_ADMIN_DEV_TOOLS` — dev-инструменты для админ-панели;
- `NEXT_PUBLIC_ASSETS_VERSION` — строка версии ассетов (для сброса кэша при обновлении);
- `NEXT_PUBLIC_SENTRY_DSN` — DSN Sentry (при использовании).

Эти переменные задаются в `.env.local` (или аналогичном файле), который попадает в образ при сборке.

9. Развертывание backend и frontend

9.1. Запуск backend-контейнера

```bash
docker run -d --name mservice-backend \
  --env-file ./backend.env \
  --network=mservice-net \
  m-service-backend-core:latest
```

Где `backend.env` содержит все переменные окружения из раздела 7.

9.2. Запуск frontend-контейнера

```bash
docker run -d --name mservice-frontend \
  --env-file ./frontend.env \
  --network=mservice-net \
  -p 80:3000 \
  m-service-frontend:latest
```

Frontend будет обслуживать web-интерфейс и страницу плеера.
При использовании Kubernetes конкретные Deployment/Service/Ingress описываются в поставляемых манифестах и применяются командой `kubectl apply -f ...`.

10. Настройка маршрутизации и доступа

В типовой конфигурации:

- frontend доступен по HTTP(S) на порту 80/443;
- backend-API доступен по путям `/api` и `/graphql`;
- WebSocket-подключение для плеера и админ-панели — по `/ws`.

При использовании Ingress:

- `https://mservice.example.com/` → frontend;
- `https://mservice.example.com/api` → backend REST;
- `https://mservice.example.com/graphql` → backend GraphQL;
- `wss://mservice.example.com/ws` → WebSocket.

11. Применение миграций базы данных

Программное обеспечение использует механизм миграций Doctrine Migrations для управления схемой базы данных.

При развертывании экземпляра программного обеспечения миграции запускаются автоматически.
При необходимости ручного запуска доступны стандартные команды:

- php bin/console doctrine:migrations:migrate latest — применение всех актуальных миграций;
- php bin/console doctrine:migrations:list — просмотр списка миграций;
- php bin/console doctrine:migrations:diff — генерация новой миграции на основе изменений в Entity.

12. Тестовые учетные данные

Для целей экспертизы правообладатель предоставляет экспертам:

- учетную запись администратора/суперадминистратора с полным доступом к функционалу системы;
- учетную запись клиентского пользователя, привязанного к тестовой компании;
- учетную запись тестового юнита (плеера), привязанного к соответствующей точке вещания.

Конкретные значения логинов и паролей для указанных учетных записей передаются в сопроводительном письме вместе с дистрибутивом программного обеспечения и не включаются в настоящий документ.

13. Проверка работоспособности

После развертывания необходимо убедиться в корректной работе экземпляра:

1. Вход в админ-панель:
    a. открыть в браузере URL фронтенда;
    b. авторизоваться с использованием учетных данных администратора;
    c. убедиться, что доступны разделы компаний, пользователей, юнитов, контента и расписаний.
2. Проверка личного кабинета клиента:
    a. войти под тестовым клиентским пользователем;
    b. проверить отображение юнитов и настроек эфира.
3. Проверка плеера:
    a. открыть страницу плеера (специальный URL или раздел интерфейса);
    b. авторизовать юнит при необходимости;
    c. убедиться, что расписание загружается и контент воспроизводится.
4. Проверка обмена данными:
    a. убедиться, что в административном интерфейсе появляются:
    b. события плеера (старт/стоп, ошибки, смена громкости и т.п.);
    c. логи фронтенда;
    d. метрики (скорость интернета, размер кэша, ошибки загрузки треков и т.д.).
5. Проверка WebSocket:
    a. в админ-панели отобразить статус юнита (онлайн/офлайн);
    b. убедиться, что статус корректно меняется при подключении/отключении плеера.

14. Готовность к экспертизе

После выполнения всех шагов и успешной проверки:

- backend, frontend, БД и MinIO работают в целевом окружении;
- доступ к веб-интерфейсу и API обеспечен;

- плеер подключается, получает расписания, воспроизводит контент;
- события, логи и метрики корректно сохраняются.

Экземпляр программного обеспечения «М-Сервис» считается готовым для проведения экспертизы.