Общая схема запуска

Файл docker-compose.yml описывает многоконтейнерное окружение приложения для отслеживания сна: веб-сервис Django, базу данных, брокер задач, систему мониторинга, логирование и сопутствующие сервисы. Команда docker-compose up из каталога проекта создаёт и запускает все описанные в файле сервисы, автоматически поднимая их зависимости и агрегируя их логи в одном выводе терминала. При первом запуске при необходимости выполняется сборка образов и загрузка недостающих образов из реестра, а при последующих запусках переиспользуются уже созданные контейнеры и тома данных.

Архитектура контейнеров

- В docker-compose.yml логично выделены несколько групп сервисов:
- Основное веб-приложение Django (Gunicorn + Nginx), использующее общий том для статических файлов и конфигурации.
- Инфраструктурные компоненты: PostgreSQL как основная БД, Redis как брокер для Celery и кеш, Celery worker и Celery beat для фоновых задач.
- Подсистема мониторинга и логирования: Prometheus для сбора метрик, Grafana для их визуализации, Loki и Promtail для централизованного сбора логов.
- Хранилища и вспомогательные сервисы: ClickHouse (аналитические данные), MinIO (объектное хранилище), Qdrant (векторная БД), Ollama (LLM-модели) и т.п., для которых, описаны отдельные тома для долговременного хранения данных.

Между контейнерами определены внутренние Docker-сети: как минимум одна «backend»-сеть для взаимодействия между приложением и БД/брокером, а также, при необходимости, публичная сеть для Nginx, публикующего HTTP-порт наружу. Для долговременного хранения данных в docker-compose.yml привязаны именованные тома к PostgreSQL, ClickHouse, MinIO, Loki, Prometheus и Qdrant, что позволяет пересоздавать контейнеры без потери пользовательских и аналитических данных.

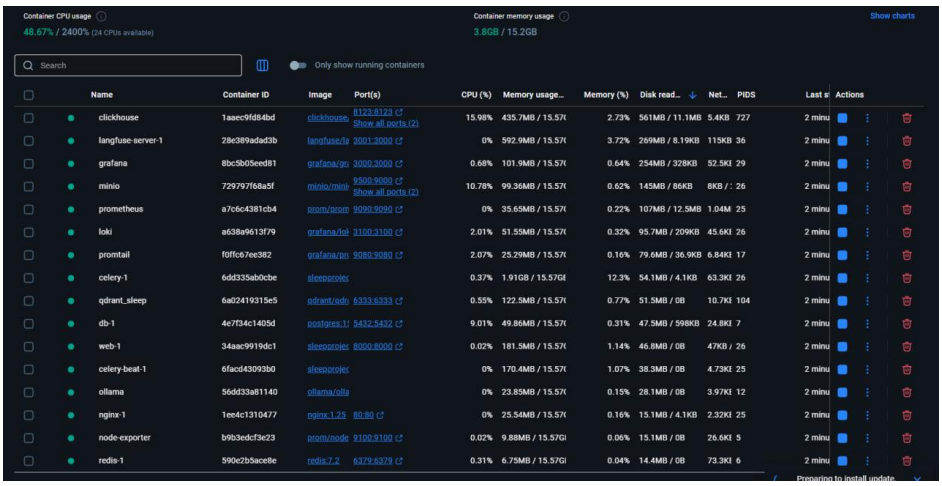

Рисунок 2 – развернутый докер

Управление окружением

Управление конфигурацией основано на файле .env в корне проекта, который создаётся пользователем на основе шаблона .env.example. Docker Compose по умолчанию подхватывает .env из текущего каталога и использует его переменные для подстановки значений в docker-compose.yml (механизм variable interpolation), что позволяет не хранить секреты и специфичные настройки окружения прямо в YAML-файле. Дополнительно отдельные сервисы могут использовать директиву env_file: и/или секцию environment: для передачи своих переменных окружения, а при необходимости можно запускать окружение с альтернативным набором переменных через флаг --env-file, указав другой путь к файлу окружения.

Типичные группы переменных окружения включают: параметры доступа к БД (хост, порт, имя БД, логин, пароль), секретный ключ Django и настройки DEBUG, параметры подключения к Redis и брокеру Celery, креденшелы для MinIO и внешних сервисов, а также базовые URL и учётные записи доступа к Prometheus/Grafana/Loki. Такое разделение позволяет легко переключаться между локальной разработкой, тестовой и продуктивной средой, изменяя только содержимое .env, не модифицируя сам docker-compose.yml.

Развёртывание с нуля и время

Развёртывание окружения «с нуля» включает установку Docker и Docker Compose, подготовку .env и первый запуск. После клонирования репозитория и создания файла .env по шаблону демонстрационный запуск выполняется

через docker-compose up -d --build, что в одном шаге собирает образы (если указаны секции build:), загружает недостающие образы из реестра и создаёт/запускает все сервисы в фоне.

Общее время первого развёртывания определяется скоростью сети и производительностью диска, так как на этом этапе скачиваются образы баз данных, брокеров, систем мониторинга и самого веб-приложения, а также выполняется их начальная инициализация. После завершения первого цикла docker-compose up повторные запуски выполняются заметно быстрее, поскольку образы уже присутствуют локально, а данные хранятся в именованных томах, и Compose лишь пересоздаёт/запускает контейнеры на основе существующего состояния.

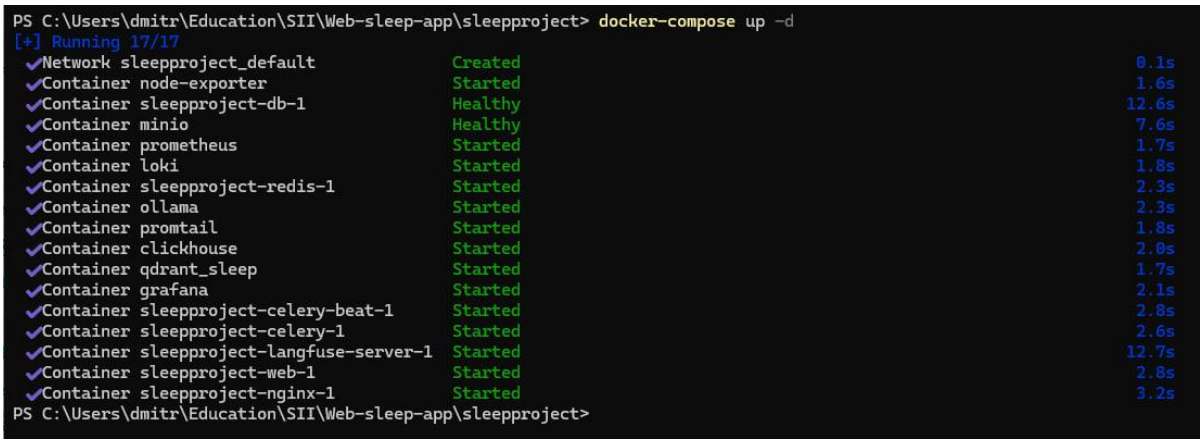

Рисунок 2 – запуск контейнеров