

**Программное обеспечение CyberStudio – платформа для
предиктивной аналитики оборудования и оптимизации режимов
технологических процессов
v 3.20.0
Руководство администратора**



Таблица 1.

История изменений

Версия	Дата	Комментарий	Автор
1	07.04.2026	Первая версия документа	Гавриленков С.И.

СОДЕРЖАНИЕ

1	Термины и сокращения.....	10
2	Назначение документа	10
2.1	Целевая аудитория	11
2.2	Связанные документы	11
3	Термины и сокращения.....	11
4	Назначение документа	12
4.1	Целевая аудитория	12
4.2	Связанные документы	13
5	Архитектура системы.....	13
5.1	Схема взаимодействия компонентов	13
6	Компоненты системы	14
6.1	Прикладные сервисы (backend API).....	14
6.2	Фронтенд-приложения	14
6.3	Инфраструктурные сервисы	15
7	Инфраструктурные сервисы	15
7.1	PostgreSQL	15
7.2	ClickHouse.....	15
7.3	Kafka.....	16
7.4	Redis.....	16
7.5	MinIO.....	16
7.6	Keycloak	16
7.7	Traefik.....	16
7.8	Вспомогательные сервисы	17
8	Сетевая архитектура.....	17
8.1	Docker-сеть	17
8.2	Маршрутизация запросов (Traefik)	17
8.3	Цепочка аутентификации	18
8.4	Проброс портов на хост.....	18
9	Стек мониторинга.....	18
9.1	Архитектура мониторинга	18
9.2	Grafana Alloy.....	19
9.3	VictoriaMetrics	19
9.4	Loki	19

9.5	Grafana.....	20
9.6	Алертинг	20
9.7	Prometheus-метрики сервисов	20
10	Системные требования.....	21
10.1	Требования к виртуальной машине.....	21
10.2	Требования к программному обеспечению	21
10.3	Сетевые требования	21
10.4	Подготовка каталогов данных	21
11	Структура compose-проекта	22
11.1	Структура каталогов	22
11.2	Корневой compose.yml.....	24
11.3	Системные сервисы	24
12	Конфигурация переменных окружения.....	24
12.1	Глобальные переменные (.env)	24
12.2	Переменные сервисов	25
12.3	Редактирование конфигурации.....	26
13	Конфигурация Traefik	26
13.1	Файлы конфигурации	26
13.2	Статическая конфигурация (traefik.yml).....	26
13.3	Динамическая конфигурация middleware	27
13.4	Маршрутизация сервисов (Docker labels)	27
13.5	Настройка TLS.....	28
14	Конфигурация Keycloak.....	28
14.1	Файлы конфигурации	28
14.2	Переменные окружения (keycloak.env).....	28
14.3	Realm cyberstudio.....	29
14.4	Интеграция с OAuth2-Proxy	29
14.5	Генерация секретов	30
15	Конфигурация PostgreSQL.....	30
15.1	Файлы конфигурации	30
15.2	Параметры конфигурации (postgresql.conf).....	30
15.3	Базы данных.....	31
15.4	Healthcheck.....	31
15.5	Хранение данных	31
16	Конфигурация ClickHouse	31

16.1	Файлы конфигурации	31
16.2	Интерфейсы	32
16.3	Healthcheck.....	32
16.4	Миграции схемы	32
16.5	Хранение данных	32
17	Конфигурация Kafka	32
17.1	Файлы конфигурации	33
17.2	Параметры брокера (kafka.env).....	33
17.3	Healthcheck.....	33
17.4	Schema Registry.....	33
17.5	Kafka Connect.....	33
17.6	Kafka UI.....	34
17.7	Kafka Exporter	34
17.8	Управление топиками.....	34
18	Конфигурация Redis	34
18.1	Файлы конфигурации	34
18.2	Параметры конфигурации (redis.conf)	35
18.3	Healthcheck.....	35
18.4	Распределение баз данных	35
18.5	Хранение данных	35
18.6	Мониторинг	35
19	Конфигурация MinIO	35
19.1	Файлы конфигурации	36
19.2	Переменные окружения (minio.env).....	36
19.3	Интерфейсы	36
19.4	Бакет и структура	36
19.5	Параметры S3 в сервисах	36
19.6	Хранение данных	37
20	Конфигурация мониторинга.....	37
20.1	Grafana Alloy.....	37
20.2	VictoriaMetrics	38
20.3	Loki	38
20.4	Grafana.....	39
21	Smoke-тесты после развёртывания	39
21.1	Проверка состояния контейнеров.....	39

21.2	Проверка инфраструктурных сервисов.....	40
21.3	Проверка маршрутизации Traefik.....	41
21.4	Проверка прикладных сервисов	41
21.5	Проверка мониторинга	41
21.6	Проверка цепочки аутентификации	41
22	Запуск и остановка	41
22.1	Запуск всей системы	41
22.2	Остановка всей системы.....	42
22.3	Запуск отдельного сервиса.....	42
22.4	Остановка отдельного сервиса	42
22.5	Перезапуск сервиса.....	42
22.6	Пересоздание всех контейнеров	43
22.7	Список имён сервисов	43
23	Проверка состояния.....	43
23.1	Общий статус контейнеров	43
23.2	Healthcheck сервисов	43
23.3	Использование ресурсов.....	43
23.4	Проверка дискового пространства	44
23.5	Проверка Docker-сети.....	44
23.6	Проверка инфраструктурных сервисов.....	44
23.7	Автоматический мониторинг (Autoheal)	44
24	Просмотр логов.....	44
24.1	Логи через Docker Compose	44
24.2	Фильтрация логов	45
24.3	Логи через Grafana / Loki.....	45
24.4	Уровни логирования	45
24.5	Ротация логов Docker.....	45
25	Резервное копирование и восстановление	46
25.1	Общие принципы	46
25.2	PostgreSQL	46
25.3	ClickHouse.....	47
25.4	MinIO.....	47
25.5	Keycloak (экспорт realm)	47
25.6	Конфигурация развёртывания	48
25.7	Проверка целостности резервных копий	48

26	Обновление версии.....	48
26.1	Порядок обновления	48
26.2	Обновление версии образов	48
26.3	Загрузка новых образов	48
26.4	Применение обновления	49
26.5	Проверка после обновления.....	49
26.6	Откат обновления.....	49
26.7	Обновление инфраструктурных компонентов	49
26.8	Очистка после обновления.....	50
27	Масштабирование.....	50
27.1	Горизонтальное масштабирование сервисов	50
27.2	Вертикальное масштабирование	50
27.3	Ограничения single-node деплоя.....	51
27.4	Рекомендации	51
28	TLS-сертификаты	51
28.1	Общие сведения	51
28.2	Автоматическое получение (ACME / Let's Encrypt)	51
28.3	Ручная установка сертификата	52
28.4	Обновление переменных окружения	53
28.5	Продление сертификата.....	53
29	Управление пользователями	53
29.1	Общие сведения	53
29.2	Управление через Keycloak.....	53
29.3	Управление через CyberStudio	54
29.4	Интеграция с корпоративным SSO.....	54
29.5	Безопасность учётных данных.....	55
30	Общий подход к диагностике.....	55
30.1	Алгоритм диагностики	55
30.2	Основные команды диагностики.....	56
30.3	Типовые сценарии.....	56
30.4	Сбор диагностической информации	57
31	Диагностика PostgreSQL.....	57
31.1	Симптомы и решения	57
31.2	Проверка подключения	57
31.3	Проверка активных подключений.....	58

31.4 Медленные запросы.....	58
31.5 Размер баз данных.....	58
31.6 Слоты репликации (CDC).....	58
31.7 Дисковое пространство	58
32 Диагностика ClickHouse	59
32.1 Симптомы и решения	59
32.2 Проверка доступности.....	59
32.3 Текущие запросы.....	59
32.4 Размер таблиц	59
32.5 Проблема «Too many parts».....	60
32.6 Мерж (фоновые операции).....	60
32.7 Дисковое пространство	60
33 Диагностика Kafka.....	60
33.1 Симптомы и решения	60
33.2 Проверка состояния кластера	61
33.3 Управление топиками.....	61
33.4 Consumer lag	61
33.5 Kafka Connect.....	61
33.6 Schema Registry.....	62
33.7 Дисковое пространство	62
33.8 Kafka UI.....	62
34 Диагностика Traefik.....	62
34.1 Симптомы и решения	62
34.2 Дашборд Traefik	63
34.3 Проверка роутеров	63
34.4 Проверка middleware.....	63
34.5 Проверка сервисов	63
34.6 Ошибка 502 Bad Gateway	63
34.7 Ошибка 504 Gateway Timeout	63
34.8 Логи Traefik	64
35 Диагностика Keycloak.....	64
35.1 Симптомы и решения	64
35.2 Проверка доступности.....	64
35.3 Логи Keycloak.....	64
35.4 Проблемы OAuth2-Proxy	64

35.5	Проверка конфигурации клиента	65
35.6	Проверка токенов	65
35.7	Экспорт realm для отладки	65
36	Диагностика мониторинга	65
36.1	Симптомы и решения	65
36.2	Grafana Alloy	66
36.3	VictoriaMetrics	66
36.4	Loki	67
36.5	Grafana	67
37	Диагностика сервисов CyberStudio	67
37.1	Общие принципы	67
37.2	Симптомы и решения	68
37.3	Диагностика конкретного сервиса	68
37.4	OOM Killed (Exit code 137)	68
37.5	Restart loop	69
37.6	Dependencies chain	69
37.7	Проверка datalake-worker	69
37.8	Логирование в формате JSON	69

1 ТЕРМИНЫ И СОКРАЩЕНИЯ

Таблица 2.

Термины и сокращения

Термин / сокращение	Определение
API	Application Programming Interface — программный интерфейс взаимодействия
CDC	Change Data Capture — технология захвата изменений данных в реальном времени
CLI	Command Line Interface — интерфейс командной строки
ClickHouse	Колоночная СУБД для аналитических запросов (БДВР)
Docker	Платформа контейнеризации приложений
Docker Compose	Инструмент для описания и запуска многоконтейнерных Docker-приложений
FQDN	Fully Qualified Domain Name — полное доменное имя
Grafana	Платформа визуализации метрик и логов
Grafana Alloy	Единый агент сбора метрик и логов
ИС	Индекс состояния — числовая оценка отклонения параметра от модели исправного состояния
JWT	JSON Web Token — формат токена аутентификации
Kafka	Распределённая платформа потоковой передачи событий
Keycloak	Сервер управления идентификацией и доступом (Identity Provider)
KRaft	Kafka Raft — режим работы Kafka без ZooKeeper
Loki	Система агрегации и хранения логов (Grafana Loki)
MinIO	S3-совместимое объектное хранилище
OAuth2	Протокол авторизации
OAuth2-Proxy	Обратный прокси для аутентификации через OAuth2/OIDC
OIDC	OpenID Connect — протокол аутентификации поверх OAuth2
ОС	Операционная система
PostgreSQL	Реляционная СУБД
Redis	Хранилище данных в оперативной памяти (кэш, сессии)
S3	Simple Storage Service — протокол объектного хранения
Schema Registry	Реестр схем сообщений Kafka (формат Avro)
SMTP	Simple Mail Transfer Protocol — протокол отправки электронной почты
SSO	Single Sign-On — технология единого входа
СУБД	Система управления базами данных
TLS	Transport Layer Security — протокол защиты транспортного уровня
Traefik	Обратный прокси-сервер и балансировщик нагрузки
VictoriaMetrics	Хранилище временных рядов метрик
ВМ	Виртуальная машина
БДВР	База данных временных рядов (ClickHouse)
ПО	Программное обеспечение
СА	Сервис администрирования
СБЛ	Сервис бизнес-логики
СДМ	Сервис дизайна мнемосхем
СДП	Сервис дизайна проекта
СИ	Сервис исторический
СМ	Сервис мониторинга
СМД	Сервис моделирования
СП	Сервис прогноза
ССД	Сервис сбора данных

2 НАЗНАЧЕНИЕ ДОКУМЕНТА

Настоящий документ представляет собой руководство администратора ПО CyberStudio — платформы предиктивной аналитики и диагностики оборудования.

Документ предназначен для системных администраторов, ответственных за эксплуатацию и сопровождение развёрнутого экземпляра ПО CyberStudio. Руководство охватывает следующие аспекты:

- обзор архитектуры и компонентов системы;
- конфигурирование инфраструктурных и прикладных сервисов;
- повседневная эксплуатация: запуск, остановка, проверка состояния, просмотр логов;
- резервное копирование и восстановление данных;
- обновление версии ПО;
- диагностика и устранение типовых неисправностей.

Целевая конфигурация развёртывания — единственная виртуальная машина с Docker Compose. Все команды и примеры в документе ориентированы на данный сценарий.

2.1 ЦЕЛЕВАЯ АУДИТОРИЯ

Документ рассчитан на специалистов со следующими компетенциями:

- администрирование ОС семейства Linux (Ubuntu/Debian/RHEL);
- базовые навыки работы с Docker и Docker Compose;
- понимание сетевых протоколов (HTTP/HTTPS, TCP, DNS);
- навыки работы в командной строке (bash).

2.2 СВЯЗАННЫЕ ДОКУМЕНТЫ

- Руководство по развёртыванию — первоначальная установка и настройка ПО CyberStudio;
- Руководство пользователя — работа с функциональными модулями системы;
- Программа и методика испытаний — верификация корректности развёртывания.

3 ТЕРМИНЫ И СОКРАЩЕНИЯ

Таблица 3.

Термины и сокращения

Термин / сокращение	Определение
API	Application Programming Interface — программный интерфейс взаимодействия
CDC	Change Data Capture — технология захвата изменений данных в реальном времени
CLI	Command Line Interface — интерфейс командной строки
ClickHouse	Колоночная СУБД для аналитических запросов (БДВР)
Docker	Платформа контейнеризации приложений
Docker Compose	Инструмент для описания и запуска многоконтейнерных Docker-приложений
FQDN	Fully Qualified Domain Name — полное доменное имя
Grafana	Платформа визуализации метрик и логов
Grafana Alloy	Единый агент сбора метрик и логов
ИС	Индекс состояния — числовая оценка отклонения параметра от модели исправного состояния
JWT	JSON Web Token — формат токена аутентификации
Kafka	Распределённая платформа потоковой передачи событий
Keycloak	Сервер управления идентификацией и доступом (Identity Provider)
KRaft	Kafka Raft — режим работы Kafka без ZooKeeper
Loki	Система агрегации и хранения логов (Grafana Loki)
MinIO	S3-совместимое объектное хранилище

Термин / сокращение	Определение
OAuth2	Протокол авторизации
OAuth2-Proxy	Обратный прокси для аутентификации через OAuth2/OIDC
OIDC	OpenID Connect — протокол аутентификации поверх OAuth2
ОС	Операционная система
PostgreSQL	Реляционная СУБД
Redis	Хранилище данных в оперативной памяти (кэш, сессии)
S3	Simple Storage Service — протокол объектного хранения
Schema Registry	Реестр схем сообщений Kafka (формат Avro)
SMTP	Simple Mail Transfer Protocol — протокол отправки электронной почты
SSO	Single Sign-On — технология единого входа
СУБД	Система управления базами данных
TLS	Transport Layer Security — протокол защиты транспортного уровня
Traefik	Обратный прокси-сервер и балансировщик нагрузки
VictoriaMetrics	Хранилище временных рядов метрик
ВМ	Виртуальная машина
БДВР	База данных временных рядов (ClickHouse)
ПО	Программное обеспечение
СА	Сервис администрирования
СБЛ	Сервис бизнес-логики
СДМ	Сервис дизайна мнемосхем
СДП	Сервис дизайна проекта
СИ	Сервис исторический
СМ	Сервис мониторинга
СМД	Сервис моделирования
СП	Сервис прогноза
ССД	Сервис сбора данных

4 НАЗНАЧЕНИЕ ДОКУМЕНТА

Настоящий документ представляет собой руководство администратора ПО CyberStudio — платформы предиктивной аналитики и диагностики оборудования.

Документ предназначен для системных администраторов, ответственных за эксплуатацию и сопровождение развёрнутого экземпляра ПО CyberStudio. Руководство охватывает следующие аспекты:

- обзор архитектуры и компонентов системы;
- конфигурирование инфраструктурных и прикладных сервисов;
- повседневная эксплуатация: запуск, остановка, проверка состояния, просмотр логов;
- резервное копирование и восстановление данных;
- обновление версии ПО;
- диагностика и устранение типовых неисправностей.

Целевая конфигурация развёртывания — единственная виртуальная машина с Docker Compose. Все команды и примеры в документе ориентированы на данный сценарий.

4.1 ЦЕЛЕВАЯ АУДИТОРИЯ

Документ рассчитан на специалистов со следующими компетенциями:

- администрирование ОС семейства Linux (Ubuntu/Debian/RHEL);
- базовые навыки работы с Docker и Docker Compose;
- понимание сетевых протоколов (HTTP/HTTPS, TCP, DNS);
- навыки работы в командной строке (bash).

4.2 СВЯЗАННЫЕ ДОКУМЕНТЫ

- Руководство по развёртыванию — первоначальная установка и настройка ПО CyberStudio;
- Руководство пользователя — работа с функциональными модулями системы;
- Программа и методика испытаний — верификация корректности развёртывания.

5 АРХИТЕКТУРА СИСТЕМЫ

ПО CyberStudio построено на микросервисной архитектуре. Система состоит из набора независимых сервисов, каждый из которых реализует отдельную функциональную область: сбор данных, моделирование, мониторинг, прогнозирование и др.

Основные архитектурные принципы:

— **Микросервисы** — каждый функциональный модуль представляет собой отдельный контейнер (API-сервис + фронтенд-приложение). Сервисы взаимодействуют между собой через REST API и шину сообщений Kafka;

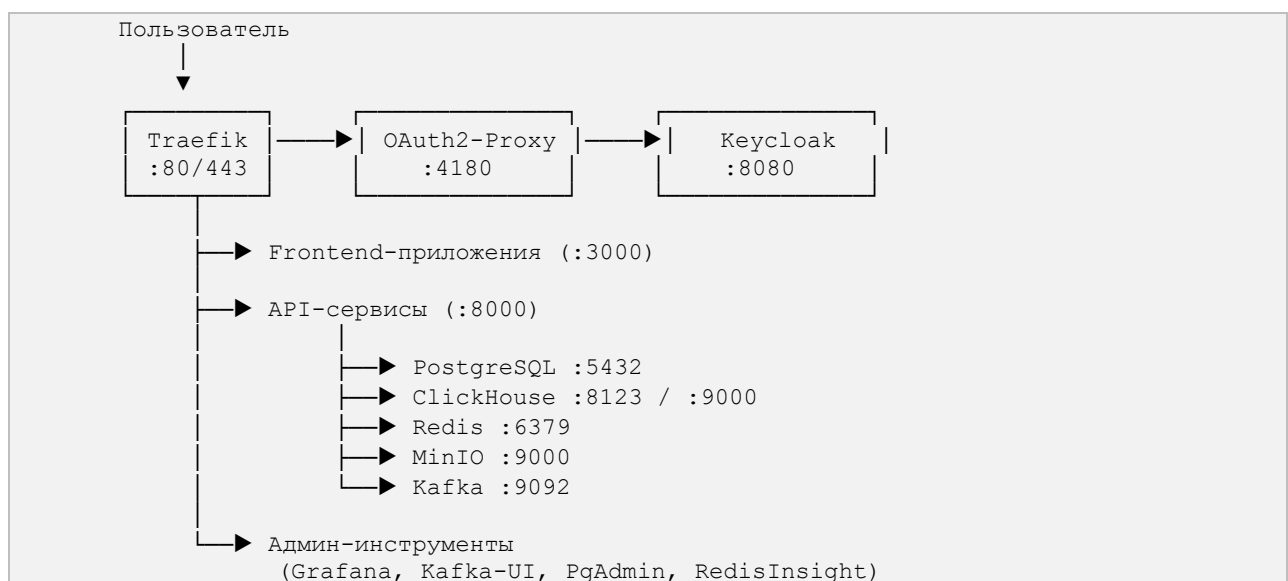
— **Шина сообщений (Kafka)** — асинхронный обмен данными между сервисами. Kafka обеспечивает приём потоков данных от внешних источников, передачу результатов расчётов и координацию задач;

— **Обратный прокси (Traefik)** — единая точка входа для всех HTTP-запросов. Traefik выполняет маршрутизацию запросов к сервисам по URL-путям, терминацию TLS и интеграцию с OAuth2-Proxy для аутентификации;

— **Аутентификация (Keycloak + OAuth2-Proxy)** — Keycloak выступает в роли Identity Provider (realm cyberstudio). OAuth2-Proxy обеспечивает проверку аутентификации на уровне обратного прокси: каждый запрос к защищённому сервису проходит через middleware oauth2-verify, который валидирует JWT-токен;

— **Контейнеризация (Docker Compose)** — все компоненты системы развёрнуты в виде Docker-контейнеров. Управление осуществляется через Docker Compose с модульной структурой: корневой `compose.yml` включает файлы инфраструктурных и прикладных сервисов.

5.1 СХЕМА ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ



Все компоненты размещаются на одной виртуальной машине и взаимодействуют через внутреннюю Docker-сеть cyberstudio (bridge, подсеть 172.16.0.0/24).

6 КОМПОНЕНТЫ СИСТЕМЫ

ПО CyberStudio состоит из трёх групп компонентов: прикладные сервисы (backend API), фронтенд-приложения и инфраструктурные сервисы.

6.1 ПРИКЛАДНЫЕ СЕРВИСЫ (BACKEND API)

Каждый API-сервис реализует определённую функциональную область платформы. Все API-сервисы работают на порту 8000, предоставляют эндпоинт /healthcheck для проверки состояния и /metrics для сбора метрик Prometheus.

Таблица 4.

Прикладные сервисы CyberStudio

Сервис	Код	Назначение
Сервис администрирования	admin-api	Управление пользователями, ролями, пространствами и проектами
Сервис сбора данных	datalake-api	Теги платформы, виртуальные теги, источники данных, группы тегов
Обработчик данных	datalake-worker	Приём и обработка потоков данных из Kafka, запись в ClickHouse
Сервис дизайна проекта	design-api	Иерархия активов, расчётные объекты, картирование, рабочие режимы
Сервис моделирования	model-api	Обучение моделей, AutoML, версионирование, метрики качества
Сервис мониторинга	monitoring-api	Виджеты, дашборды, события
Сервис бизнес-логики	business-logic-api	Процессы, задачи, автоматизации, шаблоны
Сервис исторический	historical-api	Визуализация временных рядов, группы графиков
Сервис дизайна мнемосхем	canvas-api	Конструктор дашбордов
Сервис прогноза	prognosis-api	Блоки прогноза, остаточный ресурс
Сервис неисправностей	malfunction-api	Классификатор дефектов, правила, прогонка
Сервис оптимизации	optimization-api	Оптимизатор технологических параметров
Сервис экспорта	export-api	Экспорт настроек проекта
Сервис импорта	import-api	Импорт настроек проекта и данных
Исполнитель расчётов	executor-api	Координация вычислительных задач
Менеджер Kafka	kafka-manager-api	Управление топиками Kafka, коннекторами Kafka Connect
Миграции ClickHouse	clickhouse-migrations	Создание и обновление схемы ClickHouse

6.2 ФРОНТЕНД-ПРИЛОЖЕНИЯ

Фронтенд-приложения представляют собой веб-интерфейсы для работы с соответствующими API-сервисами. Все 15 фронтенд-приложений упакованы в единый контейнер frontend. Каждое приложение доступно по отдельному URL-пути через Traefik.

Таблица 5.

Фронтенд-приложения

Приложение	URL-путь	Назначение
Администрирование	/cyberstudio/admin	Управление пользователями и настройками
Сбор данных	/cyberstudio/datalake	Работа с тегами и источниками данных

Приложение	URL-путь	Назначение
Дизайн проекта	/cyberstudio/design	Проектирование структуры моделей
Моделирование	/cyberstudio/model	Настройка и обучение моделей
Мониторинг	/cyberstudio/monitoring	Дашборды и события
Бизнес-логика	/cyberstudio/business-logic	Процессы и автоматизации
Исторические данные	/cyberstudio/historical	Графики временных рядов
Мнемосхемы	/cyberstudio/canvas	Конструктор дашбордов
Прогноз	/cyberstudio/prognosis	Прогнозирование и остаточный ресурс
Неисправности	/cyberstudio/malfunction	Классификатор дефектов
Оптимизация	/cyberstudio/optimization	Оптимизатор параметров
Экспорт	/cyberstudio/export	Экспорт настроек
Импорт	/cyberstudio/import	Импорт настроек и данных
Выбор проекта	/cyberstudio/select-project	Начальная страница выбора проекта

6.3 ИНФРАСТРУКТУРНЫЕ СЕРВИСЫ

Перечень инфраструктурных сервисов приведён в разделе «Инфраструктурные сервисы».

7 ИНФРАСТРУКТУРНЫЕ СЕРВИСЫ

Инфраструктурные сервисы обеспечивают работу прикладных компонентов ПО CyberStudio: хранение данных, обмен сообщениями, аутентификацию, маршрутизацию и мониторинг.

7.1 POSTGRESQL

Реляционная СУБД, основное хранилище конфигурации и метаданных системы. Каждый прикладной сервис использует отдельную базу данных в рамках общего экземпляра PostgreSQL. Версия: 17.x.

Ключевые параметры:

- до 500 одновременных подключений (`max_connections`);
- логическая репликация включена (`wal_level=logical`) для поддержки CDC через Kafka Connect;
- расширения: `pgvector`, `pg_stat_statements`.

7.2 CLICKHOUSE

Колоночная СУБД для хранения временных рядов (БДВР). Используется сервисами сбора данных, мониторинга и исторических данных для записи и выборки значений тегов.

Ключевые характеристики:

- высокая скорость аналитических запросов по большим объёмам данных;
- HTTP-интерфейс на порту 8123, нативный протокол на порту 9000;
- эндпоинт `/ping` для проверки доступности.

7.3 KAFKA

Распределённая платформа потоковой передачи событий. Работает в режиме KRaft (без ZooKeeper). Обеспечивает асинхронный обмен данными между сервисами и приём потоков данных от внешних источников.

Состав:

- **Kafka** — брокер сообщений (порт 9092);
- **Schema Registry** — реестр Avro-схем сообщений (порт 8081);
- **Kafka Connect** — платформа коннекторов для CDC (Change Data Capture) через Debezium (порт 8083);
- **Kafka UI** — веб-интерфейс для просмотра топиков, consumer groups и коннекторов.

7.4 REDIS

Хранилище данных в оперативной памяти. Используется для кэширования, хранения сессий и промежуточных результатов вычислений. Каждый сервис использует отдельную базу данных Redis (DB 0–15).

Ключевые параметры:

- лимит памяти: 900 МВ;
- политика вытеснения: `allkeys-lru` (вытесняются наименее используемые ключи);
- периодическое сохранение на диск (RDB-снапшоты).

7.5 MINIO

S3-совместимое объектное хранилище. Используется для хранения файлов: загруженных данных, моделей, результатов экспорта. Все сервисы работают с единым бакетом `cyberstudio`.

Ключевые параметры:

- API на порту 9000;
- веб-консоль на порту 9001;
- поддержка S3 Select для серверной фильтрации данных.

7.6 KEYCLOAK

Сервер управления идентификацией и доступом (Identity Provider). Обеспечивает аутентификацию пользователей по протоколу OpenID Connect. Realm `cyberstudio` содержит конфигурацию клиентов, ролей и пользователей.

Связанные компоненты:

- **OAuth2-Proxy** — обратный прокси, встроенный в цепочку Traefik. Перехватывает запросы к защищённым сервисам и перенаправляет неаутентифицированных пользователей на страницу входа Keycloak.

7.7 TRAEFIK

Обратный прокси-сервер и балансировщик нагрузки. Единая точка входа для всех HTTP/HTTPS-запросов к системе. Версия: 3.x.

Функции:

- маршрутизация запросов к сервисам по URL-путям (path-based routing);
- интеграция с OAuth2-Proxy через forward auth middleware;
- автоматическое обнаружение сервисов через Docker labels;
- терминация TLS (поддержка ACME/Let's Encrypt и ручных сертификатов);
- HTTP Basic Auth для защиты административных инструментов (Grafana, Kafka-UI, PgAdmin, RedisInsight).

7.8 ВСПОМОГАТЕЛЬНЫЕ СЕРВИСЫ

Таблица 6.

Вспомогательные сервисы

Сервис	Назначение
Autoheal	Автоматический перезапуск контейнеров, не прошедших healthcheck
volume-init	Инициализация каталогов данных с корректными правами доступа
PgAdmin	Веб-интерфейс администрирования PostgreSQL
RedisInsight	Веб-интерфейс администрирования Redis

8 СЕТЕВАЯ АРХИТЕКТУРА

8.1 DOCKER-СЕТЬ

Все контейнеры CyberStudio объединены в единую Docker-сеть типа bridge:

- имя сети: cyberstudio;
- подсеть: 172.16.0.0/24;
- шлюз: 172.16.0.1;
- драйвер: bridge.

Контейнеры обращаются друг к другу по имени сервиса (DNS-имя контейнера в Docker-сети). Например, API-сервис подключается к PostgreSQL по адресу `postgres:5432`, к Kafka — по адресу `kafka:9092`.

8.2 МАРШРУТИЗАЦИЯ ЗАПРОСОВ (TRAEFIK)

Traefik является единственной точкой входа для внешних HTTP/HTTPS-запросов. Порты 80 (HTTP) и 443 (HTTPS) проброшены на хостовую машину.

Маршрутизация выполняется по URL-пути. Все пути начинаются с префикса `/cyberstudio/`:

`https://<APP_HOST>/cyberstudio/<сервис>` → контейнер сервиса

Таблица 7.

Маршруты Traefik

URL-путь	Целевой сервис	Порт	Защита
<code>/cyberstudio/api/<service></code>	API-сервисы	8000	OAuth2-Proxy
<code>/cyberstudio/admin</code>	Frontend (admin)	3000	OAuth2-Proxy
<code>/cyberstudio/datalake</code>	Frontend (datalake)	3000	OAuth2-Proxy
<code>/cyberstudio/design</code>	Frontend (design)	3000	OAuth2-Proxy
<code>/cyberstudio/model</code>	Frontend (model)	3000	OAuth2-Proxy
<code>/cyberstudio/monitoring</code>	Frontend (monitoring)	3000	OAuth2-Proxy
<code>/cyberstudio/keycloak</code>	Keycloak	8080	—
<code>/cyberstudio/oauth2</code>	OAuth2-Proxy	4180	—
<code>/cyberstudio/grafana</code>	Grafana	3000	HTTP Basic Auth

URL-путь	Целевой сервис	Порт	Защита
/cyberstudio/kafka-ui	Kafka-UI	8080	HTTP Basic Auth
/cyberstudio/pgadmin	PgAdmin	80	HTTP Basic Auth
/cyberstudio/redisinsight	RedisInsight	5540	HTTP Basic Auth
/cyberstudio/minio	MinIO Console	9001	HTTP Basic Auth
/cyberstudio/alloy	Grafana Alloy UI	12345	HTTP Basic Auth
/cyberstudio/victoria-metrics	VictoriaMetrics	8428	HTTP Basic Auth
/cyberstudio/traefik	Traefik Dashboard	8080	HTTP Basic Auth

8.3 ЦЕПОЧКА АУТЕНТИФИКАЦИИ

Запросы к прикладным сервисам проходят следующую цепочку:

1. Пользователь обращается по HTTPS к Traefik.
2. Traefik применяет middleware `oauth2-verify` — forward auth запрос к OAuth2-Proxy.
3. OAuth2-Proxy проверяет наличие и валидность cookie сессии.
4. Если сессия отсутствует или истекла, OAuth2-Proxy возвращает HTTP 401, а middleware `oauth2-signin` перенаправляет пользователя на `/cyberstudio/oauth2/sign_in`.
5. OAuth2-Proxy инициирует OIDC-поток с Keycloak (realm `cyberstudio`).
6. После успешной аутентификации OAuth2-Proxy устанавливает cookie и передаёт в заголовках: `X-Auth-Request-Email`, `X-Auth-Request-Groups`, `X-Auth-Request-Access-Token`.
7. API-сервис получает JWT-токен из заголовка и выполняет авторизацию.

Административные инструменты (Grafana, Kafka-UI, PgAdmin, RedisInsight) защищены отдельным middleware `auth@file`, который использует HTTP Basic Auth. Учётные данные задаются в файле `infra/traefik/dynamic/middlewares.yml`.

8.4 ПРОБРОС ПОРТОВ НА ХОСТ

По умолчанию на хостовую машину проброшены только порты Traefik:

- 80 — HTTP;
- 443 — HTTPS.

PostgreSQL (порт 5432) может быть дополнительно проброшен на указанный в переменной `EXPOSED_IP` IP-адрес для прямого доступа из внешних инструментов.

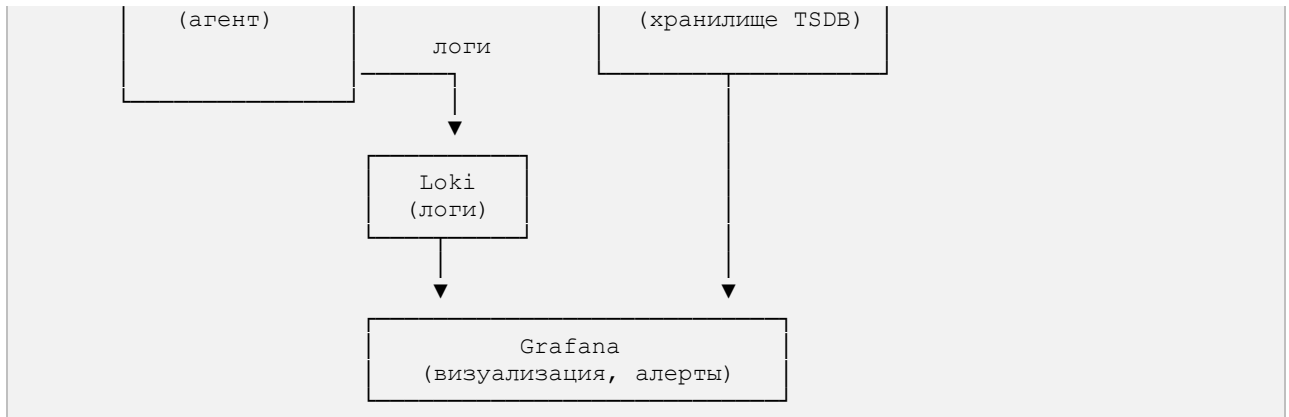
Примечание: все остальные сервисы доступны только через Docker-сеть. Прямой доступ к портам контейнеров с хостовой машины не рекомендуется в продуктивной среде.

9 СТЕК МОНИТОРИНГА

Система мониторинга ПО CyberStudio построена на стеке Grafana и обеспечивает сбор метрик, агрегацию логов, визуализацию и алертинг.

9.1 АРХИТЕКТУРА МОНИТОРИНГА





9.2 GRAFANA ALLOY

Единый агент сбора телеметрии (замена Prometheus + Promtail). Alloy выполняет:

- **Сбор метрик контейнеров** — автообнаружение Docker-контейнеров с Prometheus-лейблами. Метрики собираются с интервалом 60 секунд;
 - **Метрики cAdvisor** — потребление CPU, памяти, дискового I/O и сети каждого контейнера;
 - **Метрики node exporter** — системные метрики хостовой машины (CPU, RAM, диск, сеть);
 - **Экспортёры баз данных** — встроенные экспортёры для PostgreSQL (автообнаружение баз данных) и Redis;
 - **Сбор логов Docker** — перехват stdout/stderr всех контейнеров;
 - **Системные логи** — сбор логов из /var/log/ и syslog (порт 5140);
 - **Remote write** — отправка метрик в VictoriaMetrics и логов в Loki.
- Веб-интерфейс Alloy доступен по адресу /cyberstudio/alloy (HTTP Basic Auth).
Ограничения ресурсов: CPU — 2 ядра, RAM — 2 ГБ.

9.3 VICTORIAMETRICS

Хранилище временных рядов метрик, совместимое с Prometheus. Принимает данные от Alloy по протоколу Remote Write.

Ключевые параметры:

- период хранения: 14 дней (-retentionPeriod=14d);
- путь хранения: /data;
- формат логов: JSON;
- API доступен по адресу /cyberstudio/victoria-metrics.

9.4 LOKI

Система агрегации и хранения логов. Принимает логи от Alloy и обеспечивает их индексацию и поиск через Grafana.

Ключевые параметры:

- период хранения: 14 дней;
- хранение DEBUG-логов: 1 день (отдельная политика);
- максимальная длительность запроса: 14 дней;
- таймаут запроса: 5 минут;

- параллелизм запросов: 12;
- WAL включён для надёжности приёма;
- компактификация каждые 10 минут с включённым удалением.

9.5 GRAFANA

Платформа визуализации метрик и логов. Предоставляет дашборды для мониторинга состояния всех компонентов системы.

Конфигурация:

- анонимная аутентификация включена с ролью Admin (доступ защищён HTTP Basic Auth на уровне Traefik);
- provisioning: источники данных, дашборды и правила алертинга загружаются автоматически из конфигурационных файлов;
- адрес: `/cyberstudio/grafana`.

Предустановленные источники данных:

- VictoriaMetrics (метрики);
- Loki (логи).

Предустановленные дашборды включают мониторинг:

- контейнеров (CPU, RAM, сеть, перезапуски);
- PostgreSQL (подключения, транзакции, размер БД);
- ClickHouse (запросы, мерж, размер таблиц);
- Kafka (lag, партиции, throughput);
- Redis (память, ключи, соединения);
- хостовой машины (node exporter).

9.6 АЛЕРТИНГ

Grafana настроена с предустановленными правилами алертинга и точками контакта (contact points). Конфигурация алертов загружается из файлов provisioning:

- `alerting/alerts.yaml` — правила алертинга;
- `alerting/contactpoints.yaml` — точки контакта (email, webhook).

SMTP-сервер для отправки email-уведомлений настраивается в `grafana.ini` (параметры `smtp.host`, `smtp.from_address`).

9.7 PROMETHEUS-МЕТРИКИ СЕРВИСОВ

Все прикладные сервисы CyberStudio экспортируют метрики в формате Prometheus:

- эндпоинт: `/metrics`;
- порт: 8000;
- интервал сбора: 60 секунд.

Дополнительные экспортёры:

Таблица 8.

Экспортёры метрик

Компонент	Экспортёр	Порт	Путь
Kafka	kafka-exporter	9308	/metrics
Traefik	встроенный	8082	/metrics
ClickHouse	встроенный	8124	/metrics

Компонент	Экспортёр	Порт	Путь
MinIO	встроенный	9000	/minio/v2/metrics/cluster
Loki	встроенный	3100	/metrics
PostgreSQL	Alloy (встроенный)	—	—
Redis	Alloy (встроенный)	—	—

10 СИСТЕМНЫЕ ТРЕБОВАНИЯ

10.1 ТРЕБОВАНИЯ К ВИРТУАЛЬНОЙ МАШИНЕ

Таблица 9.

Минимальные требования к ВМ

Параметр	Минимальное значение	Рекомендуемое значение
ОС	Ubuntu 22.04 LTS / Debian 12 / RHEL 8+	Ubuntu 24.04 LTS
CPU	16 vCPU	32 vCPU
RAM	32 ГБ	64 ГБ
Диск (система + данные)	200 ГБ SSD	500 ГБ SSD
Диск (мониторинг)	50 ГБ	100 ГБ
Сетевой интерфейс	1 Гбит/с	1 Гбит/с

Примечание: объём дискового пространства для данных зависит от количества тегов, частоты опроса и глубины хранения. Приведённые значения рассчитаны на типовой сценарий: ~5000 тегов, опрос раз в секунду, хранение 1 год.

10.2 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Таблица 10.

Необходимое ПО

Компонент	Минимальная версия	Назначение
Docker Engine	24.0+	Среда выполнения контейнеров
Docker Compose	2.20+ (плагин V2)	Управление многоконтейнерными приложениями
curl	любая	Проверка доступности сервисов
jq	1.6+	Парсинг JSON-ответов при диагностике

Внимание: используется Docker Compose V2 (плагин `docker compose`, не отдельный бинарный файл `docker-compose`). Все команды в документе приведены в формате `docker compose`.

10.3 СЕТЕВЫЕ ТРЕБОВАНИЯ

Для корректной работы системы необходимо обеспечить:

- доступ к ВМ по портам 80 (HTTP) и 443 (HTTPS) для пользователей;
- доступ к ВМ по порту 22 (SSH) для администратора;
- DNS-запись (A-запись), указывающая FQDN системы на IP-адрес ВМ;
- доступ ВМ к container registry для загрузки Docker-образов (по умолчанию: `cr.yandex`);
- (опционально) доступ к внешнему SMTP-серверу для отправки email-уведомлений из Grafana.

10.4 ПОДГОТОВКА КАТАЛОГОВ ДАННЫХ

Данные сервисов хранятся в двух корневых каталогах, определяемых переменными окружения:

— DATA_BASE_DIR — основные данные (PostgreSQL, ClickHouse, Kafka, Redis, MinIO и др.);

— MONITORING_DATA_BASE_DIR — данные мониторинга (Grafana, Loki, PgAdmin, RedisInsight).

Каталоги должны располагаться на SSD-разделе с достаточным объёмом свободного пространства.

```
# Пример: создание каталогов данных
sudo mkdir -p /data
sudo mkdir -p /monitoring-data
sudo chown -R $USER:$USER /data /monitoring-data
```

11 СТРУКТУРА COMPOSE-ПРОЕКТА

Конфигурация развёртывания ПО CyberStudio организована в виде модульного Docker Compose проекта. Корневой файл `compose.yml` включает `compose`-файлы всех сервисов через директиву `include`.

11.1 СТРУКТУРА КАТАЛОГОВ

```
deployment/
├── .env                                # Глобальные переменные окружения
├── compose.yml                        # Корневой compose-файл (include)
├── infra/                             # Инфраструктурные сервисы
│   ├── alloy/                         # Grafana Alloy (метрики + логи)
│   │   ├── compose.yml
│   │   └── config/config.hcl
│   ├── clickhouse/                   # ClickHouse (БДВР)
│   │   ├── compose.yml
│   │   ├── clickhouse.env
│   │   └── config/config.xml
│   ├── grafana/                      # Grafana (визуализация)
│   │   ├── compose.yml
│   │   ├── grafana.ini
│   │   └── provisioning/
│   ├── kafka/                        # Дашборды, datasources, алерты
│   │   ├── compose.yml               # Kafka (шина сообщений)
│   │   └── kafka.env
│   ├── kafka-connect/               # Kafka Connect (CDC)
│   │   ├── compose.yml
│   │   └── kafka-connect.env
│   ├── kafka-ui/                    # Kafka UI
│   │   ├── compose.yml
│   │   └── dynamic_config.yaml
│   ├── keycloak/                    # Keycloak (IDP)
│   │   ├── compose.yml
│   │   ├── keycloak.env
│   │   └── realm.json
│   ├── loki/                         # Loki (логи)
│   │   ├── compose.yml
│   │   └── config.yml
│   ├── minio/                       # MinIO (S3)
│   │   ├── compose.yml
│   │   └── minio.env
│   ├── oauth2-proxy/                # OAuth2-Proxy
│   │   ├── compose.yml
│   │   └── oauth2-proxy.cfg
│   ├── pgadmin/                     # PgAdmin
│   │   └── compose.yml
│   ├── postgres/                    # PostgreSQL
│   │   ├── compose.yml
│   │   ├── postgres.env
│   │   ├── postgresql.conf
│   │   └── initdb/db-create.sh
│   └── redis/                       # Redis
```

```

├── compose.yml
├── redis.conf
├── redisinsight/                                # RedisInsight
│   ├── compose.yml
├── schema-registry/                            # Schema Registry
│   ├── compose.yml
│   └── schema-registry.env
├── traefik/                                    # Traefik (reverse proxy)
│   ├── compose.yml
│   ├── traefik.yml
│   └── dynamic/middlewares.yml
├── victoria-metrics/                          # VictoriaMetrics
│   └── compose.yml
├── cyberstudio/                               # Прикладные сервисы
│   ├── admin/
│   │   ├── compose.yml
│   │   └── admin-api.env
│   ├── business-logic/
│   │   ├── compose.yml
│   │   └── business-logic-api.env
│   ├── canvas/
│   │   ├── compose.yml
│   │   └── canvas-api.env
│   ├── clickhouse-migrations/
│   │   ├── compose.yml
│   │   └── clickhouse-migrations.env
│   ├── datalake/
│   │   ├── compose.yml
│   │   ├── datalake-common.env
│   │   └── datalake-worker.env
│   ├── design/
│   │   ├── compose.yml
│   │   └── design-api.env
│   ├── executor/
│   │   ├── compose.yml
│   │   └── executor-api.env
│   ├── export/
│   │   ├── compose.yml
│   │   └── export-api.env
│   ├── frontend/
│   │   ├── compose.yml
│   │   └── frontend.env
│   ├── historical/
│   │   ├── compose.yml
│   │   └── historical-api.env
│   ├── import/
│   │   ├── compose.yml
│   │   └── import-api.env
│   ├── kafka-manager/
│   │   ├── compose.yml
│   │   └── kafka-manager-api.env
│   ├── malfunction/
│   │   ├── compose.yml
│   │   └── malfunction-api.env
│   ├── model/
│   │   ├── compose.yml
│   │   └── model-api.env
│   ├── monitoring/
│   │   ├── compose.yml
│   │   └── monitoring-api.env
│   ├── optimization/
│   │   ├── compose.yml
│   │   └── optimization-api.env
│   ├── prognosis/
│   │   ├── compose.yml
│   │   └── prognosis-api.env
├── addons/
│   ├── emulator/                             # Эмулятор данных (опционально)
│   │   └── emulator-api.env

```

11.2 КОРНЕВОЙ COMPOSE.YML

Корневой файл `compose.yml` определяет Docker-сеть, системные сервисы и подключает все модули через `include`:

```
name: cyberstudio
networks:
  default:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.16.0.0/24
          gateway: 172.16.0.1
services:
  volume-init:
    # Инициализация каталогов с корректными правами
    ...
  autoheal:
    # Автоматический перезапуск упавших контейнеров
    ...
include:
- infra/alloy/compose.yml
- infra/clickhouse/compose.yml
- infra/grafana/compose.yml
- infra/kafka/compose.yml
- infra/kafka-connect/compose.yml
# ... остальные инфраструктурные сервисы
- cyberstudio/admin/compose.yml
- cyberstudio/business-logic/compose.yml
# ... остальные прикладные сервисы
```

Порядок строк в секции `include` не влияет на порядок запуска — зависимости определяются через `depends_on` в `compose`-файлах сервисов.

11.3 СИСТЕМНЫЕ СЕРВИСЫ

В корневом `compose.yml` определены два системных сервиса:

— **volume-init** — одноразовый контейнер, который устанавливает права доступа 777 на каталоги данных сервисов (Kafka, Loki, RedisInsight, PgAdmin, Grafana). Завершается после выполнения;

— **autoheal** — мониторинг здоровья контейнеров. Каждые 30 секунд проверяет статус `healthcheck` всех контейнеров и автоматически перезапускает нездоровые. Начинает работу через 600 секунд после старта (`grace period` для первоначального запуска сервисов).

12 КОНФИГУРАЦИЯ ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

Конфигурация ПО CyberStudio разделена на два уровня: глобальные переменные и переменные отдельных сервисов.

12.1 ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ (.ENV)

Файл `.env` в корне проекта содержит переменные, используемые в `compose.yml` и наследуемые сервисами.

Таблица 11.

Глобальные переменные окружения

Переменная	Описание	Пример значения
<code>DATA_BASE_DIR</code>	Корневой каталог данных	<code>/data</code>

Переменная	Описание	Пример значения
	сервисов	
MONITORING_DATA_BASE_DIR	Корневой каталог данных мониторинга	/monitoring-data
APP_HOST	FQDN хоста системы	cyberstudio.example.com
APP_VER	Версия Docker-образов CyberStudio	cyberstudio-3.18.0
CR_URL	URL container registry	cr.yandex/crpuat8boltmm3jaaddg
DOCKERHUB_URL	URL Docker Hub (или зеркало)	docker.io
ENV_NAME	Имя окружения (имя compose-проекта)	cyberstudio
WEB_SCHEME	Схема URL (http или https)	http
EXPOSED_IP	IP для проброса портов на хост	127.0.0.1

Внимание: перед запуском системы обязательно замените значения APP_HOST, CR_URL и APP_VER на актуальные для вашего окружения.

12.2 ПЕРЕМЕННЫЕ СЕРВИСОВ

Каждый сервис хранит свои переменные в отдельном .env файле в каталоге сервиса. Формат именования: <имя-сервиса>.env (например, admin-api.env, datalake-common.env).

12.2.1 ОБЩИЕ ПЕРЕМЕННЫЕ ПРИКЛАДНЫХ СЕРВИСОВ

Все API-сервисы CyberStudio используют набор общих переменных:

Таблица 12.

Общие переменные API-сервисов

Переменная	Описание	Значение по умолчанию
APP_DEBUG	Режим отладки (0/1)	0
AUTH_FLOW	Тип аутентификации	keycloak
APP_ROOT_PATH	Базовый путь API	/cyberstudio/api/<service>
KAFKA_BOOTSTRAP_SERVERS	Адрес Kafka	kafka:9092
KAFKA_TOPIC_PREFIX	Префикс топиков Kafka	cyberstudio
KAFKA_ENCODING_TYPE	Кодировка сообщений	utf-8
PG_HOST	Хост PostgreSQL	postgres
PG_PORT	Порт PostgreSQL	5432
PG_USER	Пользователь PostgreSQL	(замените на свой)
PG_PASSWORD	Пароль PostgreSQL	(замените на свой)
PG_DATABASE	Имя базы данных	<имя сервиса>
S3_ENDPOINT	Эндпоинт MinIO	minio:9000
S3_BUCKET	Имя бакета	cyberstudio
S3_SHARED_BUCKET_DIR	Общий каталог в бакете	shared
S3_DOWNLOAD_LINK_EXPIRY	Срок действия ссылки на скачивание	600
S3_VERIFY_SSL	Проверка SSL для S3	false

12.2.2 РАСПРЕДЕЛЕНИЕ БАЗ ДАННЫХ REDIS

Сервисы, использующие Redis, работают с отдельными базами данных:

Таблица 13.

Распределение баз данных Redis

Сервис	Номер базы Redis
business-logic-api	2
design-api	4
executor-api	5
monitoring-api	6
model-api	7
prognosis-api	9
malfunction-api	10
optimization-api	11

12.3 РЕДАКТИРОВАНИЕ КОНФИГУРАЦИИ

Для изменения параметров сервиса:

1. Откройте файл `.env` соответствующего сервиса.
2. Измените необходимые значения.
3. Перезапустите сервис:

```
docker compose restart <имя_сервиса>
```

Примечание: изменение глобальных переменных в корневом `.env` требует пересоздания контейнеров: `docker compose up -d`.

13 КОНФИГУРАЦИЯ TRAEFIK

Traefik выполняет роль обратного прокси и единой точки входа для всех HTTP/HTTPS-запросов к ПО CyberStudio. Версия: 3.5.

13.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 14.

Файлы конфигурации Traefik

Файл	Назначение
infra/traefik/compose.yml	Определение контейнера Traefik
infra/traefik/traefik.yml	Статическая конфигурация (entrypoints, providers)
infra/traefik/dynamic/middlewares.yml	Динамическая конфигурация (middleware)

13.2 СТАТИЧЕСКАЯ КОНФИГУРАЦИЯ (TRAEFIK.YML)

```
global:
  checkNewVersion: false
  sendAnonymousUsage: false
api:
  basePath: /cyberstudio/traefik
  dashboard: true
  insecure: true
metrics:
  prometheus:
    entryPoint: metrics
entryPoints:
  http:
    address: :80
    transport:
      respondingTimeouts:
        readTimeout: 3600
traefik:
  address: :8080
```

```
metrics:
  address: :8082
providers:
  docker:
    exposedByDefault: false
  file:
    directory: /etc/traefik/dynamic
```

Ключевые параметры:

- **entryPoints.http** — основная точка входа на порту 80. Таймаут чтения увеличен до 3600 секунд для поддержки длительных операций (обучение моделей, загрузка данных);
 - **api.dashboard** — включён дашборд Traefik по адресу /cyberstudio/traefik;
 - **providers.docker** — автоматическое обнаружение сервисов через Docker labels.
- Параметр `exposedByDefault: false` означает, что маршруты создаются только для контейнеров с явной меткой `traefik.enable=true`;
- **providers.file** — загрузка динамической конфигурации middleware из каталога /etc/traefik/dynamic.

13.3 ДИНАМИЧЕСКАЯ КОНФИГУРАЦИЯ MIDDLEWARE

Файл `infra/traefik/dynamic/middlewares.yml` определяет middleware для HTTP Basic Auth:

```
http:
  middlewares:
    auth:
      basicAuth:
        users:
          - "user:$apr1$..." # замените на свои учётные данные
```

Этот middleware (`auth@file`) используется для защиты административных инструментов: Grafana, Kafka-UI, PgAdmin, RedisInsight, Alloy, VictoriaMetrics, MinIO Console, Traefik Dashboard.

Для генерации хеша пароля:

```
# Установка утилиты htpasswd
sudo apt-get install apache2-utils

# Генерация записи user:password
htpasswd -nb <пользователь> <пароль>
```

13.4 МАРШРУТИЗАЦИЯ СЕРВИСОВ (DOCKER LABELS)

Маршруты Traefik задаются через Docker labels в compose-файлах сервисов. Пример типичного набора labels для API-сервиса:

```
labels:
  traefik.enable: true
  traefik.http.routers.<service>-api.rule: >-
    Host(`${APP_HOST}`) &&
    PathPrefix(`/cyberstudio/api/<service>`)
  traefik.http.routers.<service>-api.middlewares: >-
    oauth2-signin,oauth2-verify,<service>-api-strip
  traefik.http.middlewares.<service>-api-strip.stripprefix.prefixes: >-
    /cyberstudio/api/<service>
  traefik.http.services.<service>-api.loadbalancer.server.port: 8000
```

Для защищённых сервисов добавляются middleware `oauth2-signin` и `oauth2-verify` (forward auth через OAuth2-Proxy). Для административных инструментов используется middleware `auth@file` (HTTP Basic Auth).

13.5 НАСТРОЙКА TLS

По умолчанию Traefik работает по HTTP (порт 80). Для включения HTTPS необходимо:

1. Добавить entrypoint для HTTPS в traefik.yml:

```
entryPoints:
  http:
    address: :80
    http:
      redirections:
        entryPoint:
          to: https
          scheme: https
  https:
    address: :443
    http:
      tls: {}
```

1. Настроить TLS-сертификаты — через ACME (Let's Encrypt) или вручную. Подробнее см. раздел «TLS-сертификаты» в главе «Эксплуатация».
2. Изменить значение переменной WEB_SCHEME на https в корневом .env.

14 КОНФИГУРАЦИЯ KEYCLOAK

Keycloak — сервер управления идентификацией и доступом. Обеспечивает аутентификацию пользователей ПО CyberStudio по протоколу OpenID Connect. Версия: 24.x.

14.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 15.

Файлы конфигурации Keycloak

Файл	Назначение
infra/keycloak/compose.yml	Определение контейнера
infra/keycloak/keycloak.env	Переменные окружения
infra/keycloak/realm.json	Начальная конфигурация realm

14.2 ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ (KEYCLOAK.ENV)

Таблица 16.

Переменные окружения Keycloak

Переменная	Описание	Значение по умолчанию
KEYCLOAK_ADMIN	Логин администратора	(замените на свой)
KEYCLOAK_ADMIN_PASSWORD	Пароль администратора	(замените на свой)
KEYCLOAK_DATABASE_HOST	Хост PostgreSQL	postgres
KEYCLOAK_DATABASE_PORT	Порт PostgreSQL	5432
KEYCLOAK_DATABASE_USER	Пользователь БД	(замените на свой)
KEYCLOAK_DATABASE_NAME	Имя БД	keycloak
KEYCLOAK_DATABASE_PASSWORD	Пароль БД	(замените на свой)
KEYCLOAK_HTTP_RELATIVE_PATH	Базовый путь	/cyberstudio/keycloak/
KEYCLOAK_PRODUCTION	Режим production	true
KEYCLOAK_PROXY	Режим прокси	edge
KEYCLOAK_HTTP_PORT	Порт HTTP	8080
KEYCLOAK_LOG_LEVEL	Уровень логирования	INFO

Keycloak использует отдельную базу данных `keycloak` в PostgreSQL. Параметр `KEYCLOAK_PROXY=edge` указывает, что Keycloak находится за обратным прокси (Traefik), который выполняет терминацию TLS.

14.3 REALM CYBERSTUDIO

При первом запуске Keycloak импортирует конфигурацию `realm` из файла `realm.json`. Realm `cyberstudio` содержит:

- клиенты OAuth2 (включая `oauth2-proxy`);
- роли пользователей;
- настройки протоколов аутентификации.

Административная консоль Keycloak доступна по адресу:

`https://<APP_HOST>/cyberstudio/keycloak/admin/`

14.4 ИНТЕГРАЦИЯ С OAUTH2-PROXY

OAuth2-Proxy обеспечивает прозрачную аутентификацию на уровне обратного прокси. Конфигурация размещена в файле `infra/oauth2-proxy/oauth2-proxy.cfg`.

Таблица 17.

Ключевые параметры OAuth2-Proxy

Параметр	Описание	Значение
<code>provider</code>	Тип провайдера	<code>keycloak-oidc</code>
<code>client_id</code>	ID клиента в Keycloak	<code>oauth2-proxy</code>
<code>client_secret</code>	Секрет клиента	(замените на свой)
<code>cookie_secret</code>	Ключ шифрования cookie	(замените на свой)
<code>cookie_name</code>	Имя cookie сессии	<code>token</code>
<code>oidc_issuer_url</code>	URL издателя OIDC	<code>http://<APP_HOST>/cyberstudio/keycloak/realms/cyberstudio</code>
<code>redirect_url</code>	URL обратного вызова	<code>http://<APP_HOST>/cyberstudio/oauth2/callback</code>
<code>proxy_prefix</code>	Базовый путь OAuth2-Proxy	<code>/cyberstudio/oauth2</code>
<code>scope</code>	Запрашиваемые scope	<code>openid</code>
<code>code_challenge_method</code>	Метод PKCE	<code>S256</code>
<code>http_address</code>	Адрес прослушивания	<code>0.0.0.0:4180</code>

OAuth2-Proxy определяет два Traefik middleware:

- **oauth2-signin** — перенаправляет пользователя на страницу входа при получении HTTP 401;

— **oauth2-verify** — forward auth middleware, проверяющий аутентификацию для каждого запроса.

После успешной аутентификации OAuth2-Proxy передаёт в заголовках запроса:

- X-Auth-Request-Email — email пользователя;
- X-Auth-Request-Groups — группы пользователя;
- X-Auth-Request-Access-Token — JWT-токен доступа.

14.5 ГЕНЕРАЦИЯ СЕКРЕТОВ

Для генерации `cookie_secret`:

```
# Генерация 32-байтового ключа в Base64
openssl rand -base64 32
```

Для генерации `client_secret` — создайте нового клиента в консоли Keycloak или измените секрет существующего клиента `oauth2-proxy` в realm `cyberstudio`.

Внимание: при изменении `client_secret` в Keycloak необходимо обновить значение в файле `oauth2-proxy.cfg` и перезапустить контейнер OAuth2-Proxy.

15 КОНФИГУРАЦИЯ POSTGRESQL

PostgreSQL — основная реляционная СУБД ПО CyberStudio. Хранит конфигурацию, метаданные и бизнес-данные всех прикладных сервисов. Версия: 17.x.

15.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 18.

Файлы конфигурации PostgreSQL

Файл	Назначение
<code>infra/postgres/compose.yml</code>	Определение контейнера
<code>infra/postgres/postgres.env</code>	Учётные данные
<code>infra/postgres/postgresql.conf</code>	Параметры СУБД
<code>infra/postgres/initdb/db-create.sh</code>	Скрипт создания баз данных

15.2 ПАРАМЕТРЫ КОНФИГУРАЦИИ (POSTGRESQL.CONF)

Таблица 19.

Ключевые параметры PostgreSQL

Параметр	Значение	Назначение
<code>max_connections</code>	500	Максимальное число одновременных подключений
<code>shared_buffers</code>	128MB	Объём разделяемой памяти
<code>wal_level</code>	logical	Уровень WAL для поддержки CDC
<code>max_wal_senders</code>	15	Максимальное число процессов отправки WAL
<code>max_replication_slots</code>	15	Максимальное число слотов репликации

Параметры `wal_level=logical`, `max_wal_senders` и `max_replication_slots` необходимы для работы CDC-подсистемы (Debezium-коннектор Kafka Connect). Без логической репликации Kafka Connect не сможет отслеживать изменения в базе данных.

Примечание: значение `shared_buffers=128MB` является стартовым. Для продуктивных окружений рекомендуется увеличить до 25% от объёма RAM, выделенного под PostgreSQL (обычно 4–8 ГБ).

15.3 БАЗЫ ДАННЫХ

При первом запуске скрипт `initdb/db-create.sh` автоматически создаёт базы данных для всех сервисов:

Таблица 20.

Базы данных PostgreSQL

База данных	Сервис
admin	Сервис администрирования
datalake	Сервис сбора данных
historical	Сервис исторический
monitoring	Сервис мониторинга
template	Шаблоны проектов
canvas	Сервис дизайна мнемосхем
business-logic	Сервис бизнес-логики
export	Сервис экспорта
import	Сервис импорта
keycloak	Keycloak (IDP)
llm	Модуль LLM
model	Сервис моделирования
malfunction	Сервис неисправностей
mailer	Почтовый сервис
design	Сервис дизайна проекта
prognosis	Сервис прогноза
optimization	Сервис оптимизации

Все базы данных создаются с одним владельцем, указанным в переменной `POSTGRES_USER`.

15.4 HEALTHCHECK

PostgreSQL проверяется командой `pg_isready` с интервалом 10 секунд и 5 попытками. Большинство прикладных сервисов ожидают состояния `healthy` у PostgreSQL перед запуском (`depends_on: postgres: condition: service_healthy`).

15.5 ХРАНЕНИЕ ДАННЫХ

Данные PostgreSQL хранятся в томе `DATA_BASE_DIR/postgres/data`. Каталог `{DATA_BASE_DIR}/postgres/dumps` предназначен для хранения резервных копий (см. раздел «Резервное копирование»).

16 КОНФИГУРАЦИЯ CLICKHOUSE

ClickHouse — колоночная СУБД для хранения и аналитики временных рядов (БДВР). Используется сервисами сбора данных, мониторинга и исторических данных. Версия: 25.x.

16.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 21.

Файлы конфигурации ClickHouse

Файл	Назначение
<code>infra/clickhouse/compose.yml</code>	Определение контейнера
<code>infra/clickhouse/clickhouse.env</code>	Учётные данные

Файл	Назначение
infra/clickhouse/config/config.xml	Основная конфигурация сервера
infra/clickhouse/config/config.d/	Дополнительные файлы конфигурации

16.2 ИНТЕРФЕЙСЫ

Таблица 22.

Интерфейсы ClickHouse

Интерфейс	Порт	Назначение
HTTP	8123	REST API, веб-интерфейс, импорт/экспорт данных
Native	9000	Нативный протокол (высокопроизводительный доступ)
Prometheus	8124	Экспорт метрик в формате Prometheus

16.3 HEALTHCHECK

Проверка доступности выполняется HTTP-запросом к эндпоинту `/ping`:

```
wget --no-verbose --tries=1 --spider http://localhost:8123/ping
```

16.4 МИГРАЦИИ СХЕМЫ

Создание и обновление схемы ClickHouse выполняется отдельным контейнером `clickhouse-migrations`. Этот контейнер запускается при старте системы, ожидает готовности ClickHouse (`depends_on: clickhouse: condition: service_healthy`), выполняет миграции и завершается.

Параметры миграций задаются в файле `cyberstudio/clickhouse-migrations/clickhouse-migrations.env`:

Таблица 23.

Параметры миграций ClickHouse

Переменная	Описание	Значение
CLICKHOUSE_HOST	Хост ClickHouse	clickhouse
CLICKHOUSE_NATIVE_PORT	Нативный порт	9000
CLICKHOUSE_HTTP_PORT	HTTP порт	8123
CLICKHOUSE_DATABASE	Имя базы данных	default
KAFKA_TOPIC_PREFIX	Префикс топиков Kafka	cyberstudio

16.5 ХРАНЕНИЕ ДАННЫХ

Данные ClickHouse хранятся в томе `${DATA_BASE_DIR}/clickhouse/data`. Объем данных зависит от количества тегов, частоты опроса и глубины хранения. ClickHouse эффективно сжимает колоночные данные — типичный коэффициент сжатия для временных рядов составляет 10–20х.

17 КОНФИГУРАЦИЯ КАФКА

Kafka — платформа потоковой передачи событий. Обеспечивает асинхронный обмен данными между сервисами CyberStudio и приём потоков данных от внешних источников. Работает в режиме KRaft (без ZooKeeper). Версия: 4.x.

17.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 24.

Файлы конфигурации Kafka

Файл	Назначение
infra/kafka/compose.yml	Определение контейнера Kafka
infra/kafka/kafka.env	Параметры брокера
infra/schema-registry/compose.yml	Schema Registry
infra/schema-registry/schema-registry.env	Параметры Schema Registry
infra/kafka-connect/compose.yml	Kafka Connect (CDC)
infra/kafka-connect/kafka-connect.env	Параметры Kafka Connect
infra/kafka-ui/compose.yml	Kafka UI
infra/kafka-ui/dynamic_config.yaml	Конфигурация Kafka UI

17.2 ПАРАМЕТРЫ БРОКЕРА (KAFKA.ENV)

Таблица 25.

Ключевые параметры Kafka

Переменная	Описание	Значение
KAFKA_NODE_ID	Идентификатор узла	0
KAFKA_CLUSTER_ID	Идентификатор кластера	(автогенерация)
KAFKA_LISTENERS	Слушатели	PLAINTEXT://:9092,CONTROLLER://:9093
KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR	Фактор репликации	1
KAFKA_NUM_PARTITIONS	Число партиций по умолчанию	1
KAFKA_AUTO_CREATE_TOPICS_ENABLE	Автосоздание топиков	false

Примечание: фактор репликации установлен в 1, поскольку используется single-node конфигурация. В кластерной конфигурации рекомендуется увеличить до 3.

17.3 HEALTHCHECK

Проверка доступности Kafka:

```
kafka-cluster.sh cluster-id --bootstrap-server localhost:9092
```

Период ожидания запуска (start_period) увеличен до 600 секунд, поскольку при первом запуске Kafka инициализирует внутренние топики.

17.4 SCHEMA REGISTRY

Schema Registry хранит Avro-схемы сообщений Kafka. Порт: 8081. Зависит от Kafka (depends_on: kafka: condition: service_healthy).

17.5 KAFKA CONNECT

Kafka Connect обеспечивает Change Data Capture (CDC) — захват изменений из PostgreSQL через Debezium-коннектор.

Таблица 26.

Параметры Kafka Connect

Параметр	Описание	Значение
Образ	Кастомный с Debezium	cphysics/kafka-connect:3.2.1-debezium
Порт	REST API	8083
Формат ключей/значений	Сериализация	Avro (Schema Registry)
Группа	Consumer Group	source-connect
Топик конфигурации	Внутренний топик	connect-config
Топик смещений	Внутренний топик	connect-offset
Топик статусов	Внутренний топик	connect-status

Проверка доступности Kafka Connect:

```
curl -f http://localhost:8083/connectors
```

17.6 KAFKA UI

Веб-интерфейс для мониторинга Kafka: топики, consumer groups, коннекторы, Schema Registry. Доступен по адресу /cyberstudio/kafka-ui (защищён HTTP Basic Auth).

Конфигурация в файле dynamic_config.yaml:

- кластер Kafka: kafka:9092;
- Kafka Connect: http://kafka-connect:8083;
- Schema Registry: http://schema-registry:8081.

17.7 KAFKA EXPORTER

Для сбора метрик Kafka используется kafka-exporter (danielqsj/kafka-exporter:v1.9.0), который экспортирует метрики consumer lag, партиций и throughput на порту 9308.

17.8 УПРАВЛЕНИЕ ТОПИКАМИ

Создание и управление топиками Kafka осуществляется через сервис kafka-manager-api. Этот сервис автоматически создаёт необходимые топики при первом запуске и управляет коннекторами Kafka Connect.

Параметры kafka-manager-api:

- время хранения сообщений: 43 200 000 мс (12 часов);
- префикс топигов: cyberstudio.

18 КОНФИГУРАЦИЯ REDIS

Redis — хранилище данных в оперативной памяти. Используется для кэширования, хранения сессий и промежуточных результатов вычислений. Версия: 7.2.x.

18.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 27.

Файлы конфигурации Redis

Файл	Назначение
infra/redis/compose.yml	Определение контейнера
infra/redis/redis.conf	Конфигурация Redis

18.2 ПАРАМЕТРЫ КОНФИГУРАЦИИ (REDIS.CONF)

Таблица 28.

Ключевые параметры Redis

Параметр	Значение	Назначение
bind	0.0.0.0	Адрес прослушивания (все интерфейсы)
maxmemory	900mb	Максимальный объём памяти
maxmemory-policy	allkeys-lru	Политика вытеснения: удаляются наименее используемые ключи
io-threads	6	Число потоков I/O
appendonly	no	AOF-журнал отключён
rdbcompression	no	Сжатие RDB отключено

Точки сохранения RDB:

- каждые 900 секунд при наличии хотя бы 1 изменения;
- каждые 300 секунд при наличии хотя бы 10 изменений;
- каждые 60 секунд при наличии хотя бы 10 000 изменений.

Внимание: при превышении лимита maxmemory Redis начинает вытеснять ключи (eviction) в соответствии с политикой allkeys-lru. Это может привести к потере кэшированных данных и увеличению нагрузки на PostgreSQL. Контролируйте использование памяти через Grafana.

18.3 HEALTHCHECK

```
redis-cli ping
```

Интервал проверки: 10 секунд.

18.4 РАСПРЕДЕЛЕНИЕ БАЗ ДАННЫХ

Redis поддерживает до 16 логических баз данных (DB 0–15). Каждый сервис CyberStudio использует отдельную базу данных для изоляции данных. Распределение баз данных приведено в разделе «Конфигурация переменных окружения».

18.5 ХРАНЕНИЕ ДАННЫХ

Данные Redis хранятся в томе `${DATA_BASE_DIR}/redis/data`. При перезапуске контейнера данные восстанавливаются из RDB-снапшота.

18.6 МОНИТОРИНГ

Метрики Redis экспортируются через встроенный экспортёр Grafana Alloy. Ключевые метрики: использование памяти, число подключений, количество ключей, частота вытеснений.

19 КОНФИГУРАЦИЯ MINIO

MinIO — S3-совместимое объектное хранилище. Используется для хранения файлов: загруженных данных, обученных моделей, результатов экспорта. Версия: 2025.x.

19.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 29.

Файлы конфигурации MinIO

Файл	Назначение
infra/minio/compose.yml	Определение контейнера
infra/minio/minio.env	Учётные данные и параметры

19.2 ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ (MINIO.ENV)

Таблица 30.

Переменные окружения MinIO

Переменная	Описание	Значение
MINIO_ROOT_USER	Логин администратора	(замените на свой)
MINIO_ROOT_PASSWORD	Пароль администратора	(замените на свой)
MINIO_BROWSER	Включить веб-консоль	on
MINIO_BROWSER_REDIRECT_URL	URL редиректа консоли	http://localhost/cyberstudio/minio

19.3 ИНТЕРФЕЙСЫ

Таблица 31.

Интерфейсы MinIO

Интерфейс	Порт	Назначение
S3 API	9000	API для работы с объектами (S3-протокол)
Web Console	9001	Веб-интерфейс администрирования
Prometheus	9000	Метрики (/minio/v2/metrics/cluster)

Веб-консоль MinIO доступна по адресу /cyberstudio/minio/ (защищена HTTP Basic Auth через Traefik).

19.4 БАКЕТ И СТРУКТУРА

Все сервисы CyberStudio работают с единым бакетом cyberstudio. Внутри бакета используется каталог shared для общих файлов.

При первом запуске необходимо создать бакет через веб-консоль MinIO или CLI:

```
# Установка MinIO Client
curl -O https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc

# Настройка подключения
./mc alias set local http://localhost:9000 <MINIO_ROOT_USER>
<MINIO_ROOT_PASSWORD>

# Создание бакета
./mc mb local/cyberstudio
```

19.5 ПАРАМЕТРЫ S3 В СЕРВИСАХ

Все API-сервисы подключаются к MinIO со следующими параметрами:

- эндпоинт: minio:9000 (внутри Docker-сети);
- бакет: cyberstudio;

- каталог общих файлов: `shared`;
- срок действия временной ссылки на скачивание: 600 секунд;
- проверка SSL: отключена (`S3_VERIFY_SSL=false`).

Примечание: для интеграции с внешним S3-хранилищем (вместо MinIO) замените значения `S3_ENDPOINT`, `S3_ACCESS_KEY`, `S3_SECRET_KEY` и `S3_BUCKET` в `env`-файлах всех API-сервисов. Убедитесь, что хранилище поддерживает S3 Select.

19.6 ХРАНЕНИЕ ДАННЫХ

Данные MinIO хранятся в томе `${DATA_BASE_DIR}/minio/data`.

20 КОНФИГУРАЦИЯ МОНИТОРИНГА

Стек мониторинга ПО CyberStudio включает четыре компонента: Grafana Alloy (сбор телеметрии), VictoriaMetrics (хранение метрик), Loki (хранение логов) и Grafana (визуализация).

20.1 GRAFANA ALLOY

20.1.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 32.

Файлы конфигурации Alloy

Файл	Назначение
<code>infra/alloy/compose.yml</code>	Определение контейнера
<code>infra/alloy/config/config.hcl</code>	Конфигурация агента (HCL)

20.1.2 КЛЮЧЕВЫЕ ПАРАМЕТРЫ

- ограничения ресурсов: CPU — 2 ядра, RAM — 2 ГБ;
- интервал скрейпинга метрик: 60 секунд;
- веб-интерфейс: порт 12345, доступен по адресу `/cyberstudio/alloy`.

20.1.3 ИСТОЧНИКИ ДАННЫХ

Alloy собирает телеметрию из следующих источников:

Таблица 33.

Источники телеметрии Alloy

Источник	Тип	Описание
Docker containers	Метрики	Prometheus-метрики контейнеров с лейблом <code>prometheus.io/scrape=true</code>
cAdvisor	Метрики	CPU, RAM, I/O, сеть каждого контейнера
Node exporter	Метрики	Системные метрики хостовой машины
Redis exporter	Метрики	Метрики Redis (<code>redis://redis:6379</code>)
PostgreSQL exporter	Метрики	Автообнаружение всех баз данных
Docker logs	Логи	<code>stdout/stderr</code> всех контейнеров
Syslog	Логи	Системные логи (порт 5140)
System logs	Логи	Файлы из <code>/var/log/</code>

20.1.4 ОТПРАВКА ДАННЫХ

— метрики → VictoriaMetrics: <http://victoria-metrics:8428/cyberstudio/victoria-metrics/prometheus/api/v1/write>;
 — логи → Loki: <http://loki:3100/loki/api/v1/push>.

20.2 VICTORIAMETRICS

Таблица 34.

Параметры VictoriaMetrics

Параметр	Значение
Образ	victoriametrics/victoria-metrics:v1.122.0
Порт	8428
Период хранения	14d
Путь хранения	/data
Префикс пути	/cyberstudio/victoria-metrics
Формат логов	JSON

VictoriaMetrics принимает данные по протоколу Prometheus Remote Write и предоставляет Prometheus-совместимый API для запросов из Grafana.

Веб-интерфейс доступен по адресу `/cyberstudio/victoria-metrics` (HTTP Basic Auth).

20.3 LOKI

20.3.1 КЛЮЧЕВЫЕ ПАРАМЕТРЫ

Таблица 35.

Параметры Loki

Параметр	Значение
Образ	grafana/loki:3.5.3
Порт	3100
Период хранения	14d
Хранение DEBUG-логов	1d
Макс. длительность запроса	14d
Таймаут запроса	5m
Параллелизм запросов	12
Макс. число потоков	10000
Интервал компактификации	10m

20.3.2 ХРАНИЛИЩЕ

Loki использует файловую систему для хранения чанков и индексов:

- каталог чанков: `/loki/chunks`;
- каталог правил: `/loki/rules`;
- WAL: `/loki/wal`;
- каталог компактификации: `/loki/compactor`.

Данные хранятся в томе `${MONITORING_DATA_BASE_DIR}/loki/data`.

20.4 GRAFANA

20.4.1 ФАЙЛЫ КОНФИГУРАЦИИ

Таблица 36.

Файлы конфигурации Grafana

Файл / каталог	Назначение
infra/grafana/compose.yml	Определение контейнера
infra/grafana/grafana.ini	Основная конфигурация
provisioning/datasources/	Источники данных (VictoriaMetrics, Loki)
provisioning/dashboards/	Предустановленные дашборды
provisioning/alerting/	Правила алертинга и точки контакта

20.4.2 АУТЕНТИФИКАЦИЯ

Grafana настроена с анонимной аутентификацией и ролью Admin:

```
[auth]
disable_login_form = true

[auth.basic]
enabled = false

[auth.anonymous]
enabled = true
org_role = Admin
```

Доступ к Grafana защищён на уровне Traefik через HTTP Basic Auth (middleware auth@file).

20.4.3 EMAIL-УВЕДОМЛЕНИЯ

Для отправки email-уведомлений из алертов Grafana настройте SMTP в файле grafana.ini:

```
[smtp]
enabled = true
host = <smtp_host>:<smtp_port>
user = <smtp_user>
password = <smtp_password>
from_address = <email_from>
from_name = CyberStudio Alerts
```

20.4.4 PROVISIONING

Дашборды, источники данных и правила алертинга загружаются автоматически при запуске Grafana из каталога provisioning/. Для добавления пользовательских дашбордов разместите JSON-файлы в каталоге provisioning/dashboards/.

21 SMOKE-ТЕСТЫ ПОСЛЕ РАЗВЁРТЫВАНИЯ

После запуска системы необходимо выполнить проверку доступности всех компонентов. Smoke-тесты позволяют убедиться, что инфраструктура работает корректно и сервисы готовы к эксплуатации.

21.1 ПРОВЕРКА СОСТОЯНИЯ КОНТЕЙНЕРОВ

Убедитесь, что все контейнеры запущены и находятся в состоянии healthy:

```
docker compose ps
```

Ожидаемый результат: все сервисы в статусе Up или Up (healthy). Контейнер volume-init должен быть в статусе Exited (0) — это нормально, он завершается после инициализации.

21.2 ПРОВЕРКА ИНФРАСТРУКТУРНЫХ СЕРВИСОВ

21.2.1 POSTGRESQL

```
docker compose exec postgres pg_isready -U postgres
```

Ожидаемый ответ: accepting connections.

Проверка созданных баз данных:

```
docker compose exec postgres psql -U <POSTGRES_USER> -c "\l"
```

21.2.2 CLICKHOUSE

```
curl -s http://localhost:8123/ping
```

Ожидаемый ответ: Ok.

21.2.3 KAFKA

```
docker compose exec kafka kafka-cluster.sh cluster-id \
--bootstrap-server localhost:9092
```

Ожидаемый результат: идентификатор кластера.

Проверка топиков:

```
docker compose exec kafka kafka-topics.sh --list \
--bootstrap-server localhost:9092
```

21.2.4 REDIS

```
docker compose exec redis redis-cli ping
```

Ожидаемый ответ: PONG.

21.2.5 MINIO

```
curl -s http://localhost:9000/minio/health/live
```

Ожидаемый HTTP-код: 200.

21.2.6 KEYCLOAK

```
curl -s -o /dev/null -w "%{http_code}" \
http://localhost:8080/cyberstudio/keycloak/realms/master
```

Ожидаемый HTTP-код: 200.

21.2.7 SCHEMA REGISTRY

```
docker compose exec schema-registry \
bash -c "nc -z localhost 8081 && echo OK"
```

21.2.8 KAFKA CONNECT

```
curl -s http://localhost:8083/connectors
```

Ожидаемый ответ: JSON-массив (может быть пустым [] при первом запуске).

21.3 ПРОВЕРКА МАРШРУТИЗАЦИИ TRAEFIK

Убедитесь, что Traefik корректно маршрутизирует запросы:

```
# Дашборд Traefik
curl -s -o /dev/null -w "%{http_code}" \
  http://<APP_HOST>/cyberstudio/traefik/dashboard/

# Keycloak
curl -s -o /dev/null -w "%{http_code}" \
  http://<APP_HOST>/cyberstudio/keycloak/realms/cyberstudio

# OAuth2-Proxy (ожидается 403 — нет сессии)
curl -s -o /dev/null -w "%{http_code}" \
  http://<APP_HOST>/cyberstudio/oauth2/userinfo
```

21.4 ПРОВЕРКА ПРИКЛАДНЫХ СЕРВИСОВ

Проверка healthcheck всех API-сервисов:

```
# Проверка всех API одной командой
for svc in admin datalake design model monitoring \
  business-logic historical canvas prognosis \
  malfunction optimization export import executor; do
  code=$(curl -s -o /dev/null -w "%{http_code}" \
    http://<APP_HOST>/cyberstudio/api/${svc}/healthcheck)
  echo "${svc}: ${code}"
done
```

Ожидаемый HTTP-код для каждого сервиса: 200.

21.5 ПРОВЕРКА МОНИТОРИНГА

```
# VictoriaMetrics
curl -s -o /dev/null -w "%{http_code}" \
  http://localhost:8428/cyberstudio/victoria-metrics/api/v1/status/tsdb

# Loki
curl -s -o /dev/null -w "%{http_code}" \
  http://localhost:3100/ready

# Grafana
curl -s -o /dev/null -w "%{http_code}" \
  http://localhost:3000/api/health
```

21.6 ПРОВЕРКА ЦЕПОЧКИ АУТЕНТИФИКАЦИИ

1. Откройте в браузере `http://<APP_HOST>/cyberstudio/select-project`.
2. Система должна перенаправить на страницу входа Keycloak.
3. Введите учётные данные пользователя.
4. После успешной аутентификации должна отобразиться страница выбора проекта.

Примечание: если перенаправление на Keycloak не происходит или возвращается ошибка, проверьте конфигурацию OAuth2-Proxy и корректность переменных `oidc_issuer_url` и `redirect_url`.

22 ЗАПУСК И ОСТАНОВКА

22.1 ЗАПУСК ВСЕЙ СИСТЕМЫ

Для запуска всех компонентов ПО CyberStudio выполните команду из каталога развёртывания:

```
cd /path/to/deployment
docker compose up -d
```

Флаг `-d` запускает контейнеры в фоновом режиме. Docker Compose автоматически соблюдает порядок запуска, определённый через `depends_on`:

1. volume-init (инициализация каталогов) → завершается;
2. PostgreSQL, Redis, ClickHouse, Kafka (инфраструктура);
3. Keycloak (ожидает PostgreSQL);
4. Schema Registry (ожидает Kafka);
5. Kafka Connect (ожидает Kafka);
6. OAuth2-Proxy (ожидает Keycloak);
7. kafka-manager-api, clickhouse-migrations (ожидают инфраструктуру);
8. API-сервисы CyberStudio (ожидают kafka-manager-api и БД);
9. Фронтенд-приложения;
10. Traefik, Autoheal, мониторинг.

Полный запуск системы занимает 5–10 минут. Наибольшее время требуется на инициализацию Kafka (до 600 секунд при первом запуске) и Kafka Connect.

22.2 ОСТАНОВКА ВСЕЙ СИСТЕМЫ

```
docker compose down
```

Команда `down` останавливает и удаляет все контейнеры, но сохраняет тома данных и сеть. Данные PostgreSQL, ClickHouse, Kafka и других сервисов не теряются.

Внимание: команда `docker compose down -v` удаляет тома данных. Не используйте флаг `-v`, если не хотите потерять данные.

22.3 ЗАПУСК ОТДЕЛЬНОГО СЕРВИСА

```
docker compose up -d <имя_сервиса>
```

Примеры:

```
# Запуск только PostgreSQL
docker compose up -d postgres

# Запуск API сервиса администрирования
docker compose up -d admin-api
```

Docker Compose автоматически запустит все зависимости указанного сервиса.

22.4 ОСТАНОВКА ОТДЕЛЬНОГО СЕРВИСА

```
docker compose stop <имя_сервиса>
```

Примеры:

```
# Остановка сервиса моделирования
docker compose stop model-api

# Остановка фронтенда
docker compose stop frontend
```

Примечание: команда `stop` останавливает контейнер без его удаления. Для повторного запуска используйте `docker compose start <имя_сервиса>`.

22.5 ПЕРЕЗАПУСК СЕРВИСА

```
docker compose restart <имя_сервиса>
```

Перезапуск выполняет `stop + start` контейнера. Если необходимо применить изменения конфигурации (`env`-файлы, `compose.yml`), используйте `up -d`:

```
docker compose up -d <имя_сервиса>
```

Команда `up -d` пересоздаст контейнер, если обнаружит изменения в конфигурации.

22.6 ПЕРЕСОЗДАНИЕ ВСЕХ КОНТЕЙНЕРОВ

При обновлении глобальных переменных в `.env` или изменении `compose.yml`:

```
docker compose up -d --force-recreate
```

22.7 СПИСОК ИМЕН СЕРВИСОВ

Для получения списка всех сервисов:

```
docker compose config --services
```

23 ПРОВЕРКА СОСТОЯНИЯ

23.1 ОБЩИЙ СТАТУС КОНТЕЙНЕРОВ

```
docker compose ps
```

Вывод содержит: имя контейнера, образ, статус, порты и состояние `healthcheck`. Ожидаемые состояния:

Таблица 37.

Состояния контейнеров

Состояние	Значение
Up (healthy)	Контейнер работает, <code>healthcheck</code> пройден
Up	Контейнер работает, <code>healthcheck</code> не настроен или ещё не пройден
Up (unhealthy)	Контейнер работает, но <code>healthcheck</code> не проходит
Exited (0)	Контейнер завершился успешно (нормально для <code>volume-init</code> , <code>clickhouse-migrations</code>)
Restarting	Контейнер перезапускается (возможна проблема)

23.2 HEALTHCHECK СЕРВИСОВ

Все прикладные API-сервисы предоставляют эндпоинт `/healthcheck`:

```
# Проверка конкретного сервиса
curl -s http://<APP_HOST>/cyberstudio/api/<service>/healthcheck

# Массовая проверка всех API
for svc in admin datalake design model monitoring \
    business-logic historical canvas prognosis \
    malfunction optimization export import executor; do
    code=$(curl -s -o /dev/null -w "%{http_code}" \
        http://<APP_HOST>/cyberstudio/api/${svc}/healthcheck)
    printf "%-20s %s\n" "${svc}" "${code}"
done
```

23.3 ИСПОЛЬЗОВАНИЕ РЕСУРСОВ

Просмотр потребления CPU и RAM контейнерами:

```
docker stats --no-stream
```

Для постоянного мониторинга (обновление в реальном времени):

```
docker stats
```

23.4 ПРОВЕРКА ДИСКОВОГО ПРОСТРАНСТВА

```
# Общее использование диска Docker
docker system df

# Детализация по томам
docker system df -v

# Свободное место на хостовой машине
df -h /data /monitoring-data
```

23.5 ПРОВЕРКА DOCKER-СЕТИ

```
# Список контейнеров в сети cyberstudio
docker network inspect cyberstudio --format '{{range .Containers}}{{.Name}}:
{{.IPv4Address}}{{"\n"}}{{end}}'
```

23.6 ПРОВЕРКА ИНФРАСТРУКТУРНЫХ СЕРВИСОВ

```
# PostgreSQL: число активных подключений
docker compose exec postgres psql -U <POSTGRES_USER> -c \
  "SELECT count(*) FROM pg_stat_activity;"

# Redis: использование памяти
docker compose exec redis redis-cli info memory | grep used_memory_human

# Kafka: список consumer groups и lag
docker compose exec kafka kafka-consumer-groups.sh \
  --bootstrap-server localhost:9092 --list

# ClickHouse: список таблиц и их размер
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT table, formatReadableSize(total_bytes) AS size FROM
system.tables WHERE database = 'default' ORDER BY total_bytes DESC"
```

23.7 АВТОМАТИЧЕСКИЙ МОНИТОРИНГ (AUTOHEAL)

Контейнер autoheal каждые 30 секунд проверяет состояние healthcheck всех контейнеров. Контейнеры со статусом unhealthy автоматически перезапускаются.

Для просмотра событий autoheal:

```
docker compose logs autoheal --tail=20
```

24 ПРОСМОТР ЛОГОВ

24.1 ЛОГИ ЧЕРЕЗ DOCKER COMPOSE

Просмотр логов конкретного сервиса:

```
# Последние 100 строк
docker compose logs --tail=100 <имя_сервиса>

# Непрерывный вывод (follow)
docker compose logs -f <имя_сервиса>

# С временными метками
docker compose logs -t --tail=50 <имя_сервиса>
```

Просмотр логов нескольких сервисов одновременно:

```
docker compose logs -f admin-api design-api model-api
```

Просмотр логов всех сервисов:

```
docker compose logs -f --tail=10
```

24.2 ФИЛЬТРАЦИЯ ЛОГОВ

Для поиска ошибок в логах:

```
# Поиск ошибок в логах сервиса
docker compose logs <имя_сервиса> 2>&1 | grep -i "error"

# Поиск по временному диапазону
docker compose logs --since="2024-01-01T10:00:00" \
  --until="2024-01-01T11:00:00" <имя_сервиса>
```

24.3 ЛОГИ ЧЕРЕЗ GRAFANA / LOKI

Grafana предоставляет централизованный доступ ко всем логам системы через источник данных Loki.

Для доступа к логам через Grafana:

1. Откройте https://<APP_HOST>/cyberstudio/grafana (HTTP Basic Auth).
2. Перейдите в раздел **Explore**.
3. Выберите источник данных **Loki**.
4. Используйте LogQL-запросы для поиска и фильтрации.

Примеры LogQL-запросов:

```
# Все логи контейнера admin-api
{container="admin-api"}

# Ошибки во всех сервисах
{container=~".*-api"} |= "ERROR"

# Логи PostgreSQL за последний час
{container="postgres"} | line_format "{{.message}}"

# JSON-логи с фильтрацией по уровню
{container="model-api"} | json | level="error"
```

24.4 УРОВНИ ЛОГИРОВАНИЯ

API-сервисы CyberStudio поддерживают стандартные уровни логирования: DEBUG, INFO, WARNING, ERROR, CRITICAL.

Для изменения уровня логирования отредактируйте переменную LOG_LEVEL в env-файле сервиса и перезапустите его:

```
# Пример: включение DEBUG-логов для model-api
echo "LOG_LEVEL=DEBUG" >> cyberstudio/model/model-api.env
docker compose restart model-api
```

Внимание: DEBUG-логи значительно увеличивают объем данных. Включайте DEBUG только на время диагностики. Loki хранит DEBUG-логи только 1 день (в отличие от 14 дней для остальных уровней).

24.5 РОТАЦИЯ ЛОГОВ DOCKER

Docker по умолчанию хранит логи контейнеров без ограничений, что может привести к исчерпанию дискового пространства. Рекомендуется настроить ротацию логов в `/etc/docker/daemon.json`:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "50m",
    "max-file": "5"
  }
}
```

```
}
}
```

После изменения перезапустите Docker:

```
sudo systemctl restart docker
```

Примечание: ротация логов Docker не влияет на логи в Loki — они хранятся отдельно и управляются политикой хранения Loki (14 дней).

25 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

25.1 ОБЩИЕ ПРИНЦИПЫ

Регулярное резервное копирование необходимо для следующих компонентов:

Таблица 38.

Компоненты, требующие резервного копирования

Компонент	Данные	Критичность	Рекомендуемая периодичность
PostgreSQL	Конфигурация, метаданные, бизнес-данные	Высокая	Ежедневно
ClickHouse	Временные ряды (значения тегов)	Высокая	Еженедельно
MinIO	Файлы (модели, экспорт, загруженные данные)	Средняя	Еженедельно
Keycloak	Realm, пользователи, роли	Высокая	После каждого изменения
Конфигурация	Каталог deployment/ (compose, env, конфиги)	Высокая	При каждом изменении

25.2 POSTGRESQL

25.2.1 РЕЗЕРВНОЕ КОПИРОВАНИЕ

Создание дампа всех баз данных:

```
# Полный дамп всех баз
docker compose exec postgres pg_dumpall -U <POSTGRES_USER> \
  > ${DATA_BASE_DIR}/postgres/dumps/full_$(date +%Y%m%d_%H%M%S).sql

# Дамп отдельной базы данных
docker compose exec postgres pg_dump -U <POSTGRES_USER> -d <database> \
  --format=custom \
  > ${DATA_BASE_DIR}/postgres/dumps/<database>_$(date +%Y%m%d_%H%M%S).dump
```

Формат `--format=custom` создаёт сжатый дамп, который позволяет выборочное восстановление таблиц.

25.2.2 ВОССТАНОВЛЕНИЕ

```
# Восстановление полного дампа (plain SQL)
docker compose exec -T postgres psql -U <POSTGRES_USER> < full_dump.sql

# Восстановление отдельной базы (custom format)
docker compose exec -T postgres pg_restore -U <POSTGRES_USER> \
  -d <database> --clean --if-exists < database.dump
```

Внимание: восстановление базы данных требует остановки API-сервисов, использующих эту базу. Перезапустите сервисы после восстановления.

25.2.3 АВТОМАТИЗАЦИЯ

Пример cron-задачи для ежедневного резервного копирования:

```
# Добавить в crontab (crontab -e)
0 2 * * * cd /path/to/deployment && \
docker compose exec -T postgres pg_dumpall -U postgres \
> /backup/postgres/full_${date +%Y%m%d}.sql 2>&1 | \
logger -t postgres-backup
```

25.3 CLICKHOUSE

25.3.1 РЕЗЕРВНОЕ КОПИРОВАНИЕ

```
# Экспорт таблицы в формате Native
docker compose exec clickhouse clickhouse-client \
--user <user> --password <password> \
--query "SELECT * FROM default.<table> FORMAT Native" \
> /backup/clickhouse/<table>_${date +%Y%m%d}.native

# Экспорт в CSV (для небольших таблиц)
docker compose exec clickhouse clickhouse-client \
--user <user> --password <password> \
--query "SELECT * FROM default.<table> FORMAT CSVWithNames" \
> /backup/clickhouse/<table>_${date +%Y%m%d}.csv
```

Альтернативный способ — копирование каталога данных:

```
# Остановить ClickHouse
docker compose stop clickhouse

# Скопировать данные
cp -r ${DATA_BASE_DIR}/clickhouse/data /backup/clickhouse/data_${date +%Y%m%d}

# Запустить ClickHouse
docker compose start clickhouse
```

25.3.2 ВОССТАНОВЛЕНИЕ

```
# Импорт из Native-формата
docker compose exec -T clickhouse clickhouse-client \
--user <user> --password <password> \
--query "INSERT INTO default.<table> FORMAT Native" \
< /backup/clickhouse/<table>.native
```

25.4 MINIO

25.4.1 РЕЗЕРВНОЕ КОПИРОВАНИЕ

```
# Установка MinIO Client (если не установлен)
curl -O https://dl.min.io/client/mc/release/linux-amd64/mc
chmod +x mc
./mc alias set local http://localhost:9000 <MINIO_ROOT_USER>
<MINIO_ROOT_PASSWORD>

# Копирование всего бакета
./mc mirror local/cyberstudio /backup/minio/cyberstudio_${date +%Y%m%d}
```

25.4.2 ВОССТАНОВЛЕНИЕ

```
./mc mirror /backup/minio/cyberstudio_backup local/cyberstudio
```

25.5 KEYCLOAK (ЭКСПОРТ REALM)

25.5.1 ЭКСПОРТ

```
docker compose exec keycloak /opt/keycloak/bin/kc.sh export \
--dir /tmp/export --realm cyberstudio
```

```
docker compose cp keycloak:/tmp/export/cyberstudio-realm.json \
  /backup/keycloak/realm_$(date +%Y%m%d).json
```

25.5.2 ИМПОРТ

Для восстановления realm замените файл infra/keycloak/realm.json на сохранённую копию и пересоздайте контейнер Keycloak:

```
cp /backup/keycloak/realm_backup.json infra/keycloak/realm.json
docker compose up -d --force-recreate keycloak
```

Примечание: импорт realm при работающем Keycloak заменяет существующую конфигурацию. Убедитесь, что вы используете корректную версию экспорта.

25.6 КОНФИГУРАЦИЯ РАЗВЁРТЫВАНИЯ

Каталог deployment/ содержит все файлы конфигурации. Рекомендуется хранить его в системе контроля версий (Git) или создавать архив при каждом изменении:

```
tar -czf /backup/config/deployment_$(date +%Y%m%d_%H%M%S).tar.gz \
  -C /path/to deployment/
```

25.7 ПРОВЕРКА ЦЕЛОСТНОСТИ РЕЗЕРВНЫХ КОПИЙ

Периодически проверяйте возможность восстановления из резервных копий:

```
# Проверка дампа PostgreSQL
pg_restore --list database.dump > /dev/null && echo "OK" || echo "CORRUPTED"

# Проверка архива конфигурации
tar -tzf deployment_backup.tar.gz > /dev/null && echo "OK" || echo "CORRUPTED"
```

26 ОБНОВЛЕНИЕ ВЕРСИИ

26.1 ПОРЯДОК ОБНОВЛЕНИЯ

Обновление ПО CyberStudio выполняется путём замены Docker-образов на новую версию. Процедура:

1. Создать резервную копию (см. раздел «Резервное копирование»).
2. Обновить версию образов в конфигурации.
3. Загрузить новые образы.
4. Пересоздать контейнеры.
5. Дождаться завершения миграций.
6. Проверить работоспособность.

26.2 ОБНОВЛЕНИЕ ВЕРСИИ ОБРАЗОВ

Измените переменную APP_VER в корневом файле .env:

```
# Было
APP_VER=cyberstudio-3.18.0

# Стало
APP_VER=cyberstudio-3.19.0
```

26.3 ЗАГРУЗКА НОВЫХ ОБРАЗОВ

```
docker compose pull
```

Команда загрузит все образы, версия которых изменилась. Загрузка выполняется параллельно.

26.4 ПРИМЕНЕНИЕ ОБНОВЛЕНИЯ

```
docker compose up -d
```

Docker Compose автоматически определит контейнеры с обновлёнными образами и пересоздаст только их. Зависимости будут соблюдены.

При обновлении автоматически выполняются:

- миграции схемы ClickHouse (контейнер `clickhouse-migrations`);
- миграции схемы PostgreSQL (при запуске каждого API-сервиса);
- обновление топиков Kafka (через `kafka-manager-api`).

26.5 ПРОВЕРКА ПОСЛЕ ОБНОВЛЕНИЯ

```
# Убедиться, что все контейнеры запущены
docker compose ps

# Проверить healthcheck сервисов
for svc in admin datalake design model monitoring \
    business-logic historical canvas prognosis \
    malfunction optimization export import executor; do
    code=$(curl -s -o /dev/null -w "%{http_code}" \
        http://<APP_HOST>/cyberstudio/api/${svc}/healthcheck)
    printf "%-20s %s\n" "${svc}" "${code}"
done

# Проверить логи на наличие ошибок
docker compose logs --since="10m" 2>&1 | grep -i "error"
```

26.6 ОТКАТ ОБНОВЛЕНИЯ

Если после обновления обнаружены критические проблемы:

1. Верните предыдущую версию в `.env`:

```
APP_VER=cyberstudio-3.18.0
```

1. Пересоздайте контейнеры:

```
docker compose up -d
```

1. При необходимости восстановите базы данных из резервных копий.

Внимание: откат версии после применения миграций БД может потребовать восстановления из резервной копии, поскольку обратные миграции не гарантированы. Всегда создавайте резервную копию перед обновлением.

26.7 ОБНОВЛЕНИЕ ИНФРАСТРУКТУРНЫХ КОМПОНЕНТОВ

Обновление версий инфраструктурных сервисов (PostgreSQL, ClickHouse, Kafka, Redis и др.) выполняется изменением тега образа в соответствующем `compose.yml`:

```
# Пример: infra/postgres/compose.yml
services:
  postgres:
    image: postgres:17.5 # измените на новую версию
```

Внимание: обновление мажорных версий инфраструктурных сервисов (например, PostgreSQL 17 → 18) может требовать дополнительных шагов миграции. Обратитесь к документации соответствующего компонента.

26.8 ОЧИСТКА ПОСЛЕ ОБНОВЛЕНИЯ

После успешного обновления удалите неиспользуемые образы для освобождения дискового пространства:

```
docker image prune -a --filter "until=168h"
```

Команда удалит образы, не используемые контейнерами и старше 7 дней.

27 МАСШТАБИРОВАНИЕ

27.1 ГОРИЗОНТАЛЬНОЕ МАСШТАБИРОВАНИЕ СЕРВИСОВ

Docker Compose позволяет запускать несколько экземпляров одного сервиса с помощью параметра `--scale`:

```
docker compose up -d --scale model-api=3
```

Traefik автоматически обнаруживает новые экземпляры контейнеров и распределяет нагрузку между ними (round-robin).

27.1.1 СЕРВИСЫ, ДОПУСКАЮЩИЕ МАСШТАБИРОВАНИЕ

Таблица 39.

Масштабируемые сервисы

Сервис	Масштабирование	Примечания
API-сервисы (admin, design, model и др.)	Да	Stateless, масштабируются свободно
datalake-worker	Да	Обработка потоков данных параллелизуется
executor-api	Да	Параллельное выполнение вычислительных задач
frontend	Нет	Статический контент, одного экземпляра достаточно

27.1.2 СЕРВИСЫ, НЕ ДОПУСКАЮЩИЕ МАСШТАБИРОВАНИЕ

Таблица 40.

Немасштабируемые сервисы

Сервис	Причина
PostgreSQL	Single-node, требуется кластерное решение (Patroni)
ClickHouse	Одиночный экземпляр, для масштабирования нужен кластер
Kafka	Single-node в текущей конфигурации
Redis	Одиночный экземпляр
Keycloak	Stateful, требуется кластерная конфигурация
Traefik	Единая точка входа
kafka-manager-api	Управляет состоянием топиков и коннекторов

27.2 ВЕРТИКАЛЬНОЕ МАСШТАБИРОВАНИЕ

Для увеличения ресурсов отдельных контейнеров добавьте ограничения в compose-файл сервиса:

```
services:
  model-api:
```

```
deploy:
  resources:
    limits:
      cpus: '4'
      memory: 8G
    reservations:
      cpus: '2'
      memory: 4G
```

27.3 ОГРАНИЧЕНИЯ SINGLE-NODE ДЕПЛОЯ

При развёртывании на одной виртуальной машине существуют следующие ограничения:

- все контейнеры конкурируют за ресурсы одного хоста (CPU, RAM, I/O);
- отказ хоста приводит к недоступности всей системы;
- масштабирование инфраструктурных сервисов (PostgreSQL, ClickHouse, Kafka) ограничено одним экземпляром;
- общий объём оперативной памяти должен быть достаточным для всех контейнеров одновременно.

Для оценки потребностей в ресурсах используйте `docker stats` и дашборды Grafana. При систематической нехватке ресурсов рассмотрите увеличение характеристик ВМ (вертикальное масштабирование) или переход к распределённой конфигурации.

27.4 РЕКОМЕНДАЦИИ

- Начните с мониторинга фактического потребления ресурсов через Grafana в течение 1–2 недель.
- Масштабируйте в первую очередь сервисы с наибольшим временем отклика или потреблением CPU.
- При масштабировании `datalake-worker` учитывайте пропускную способность Kafka и ClickHouse.
- Для Redis контролируйте общий объём используемой памяти — при масштабировании API-сервисов нагрузка на Redis увеличивается.

28 TLS-СЕРТИФИКАТЫ

28.1 ОБЩИЕ СВЕДЕНИЯ

По умолчанию ПО CyberStudio работает по протоколу HTTP. Для продуктивного использования рекомендуется включить HTTPS с TLS-сертификатами. Traefik поддерживает два способа получения сертификатов: автоматический (ACME / Let's Encrypt) и ручной.

28.2 АВТОМАТИЧЕСКОЕ ПОЛУЧЕНИЕ (ACME / LET'S ENCRYPT)

Для автоматического получения и обновления TLS-сертификатов через Let's Encrypt добавьте конфигурацию в `infra/traefik/traefik.yml`:

```
entryPoints:
  http:
    address: :80
  http:
    redirections:
```

```

    entryPoint:
      to: https
      scheme: https
  https:
    address: :443
  certificatesResolvers:
    letsencrypt:
      acme:
        email: admin@example.com      # замените на свой email
        storage: /data/acme.json
        httpChallenge:
          entryPoint: http

```

Затем добавьте TLS-конфигурацию в Docker labels сервисов (в compose-файлах):

```

labels:
  traefik.http.routers.<service>.tls: true
  traefik.http.routers.<service>.tls.certresolver: letsencrypt

```

Требования для ACME:

- DNS-запись (A-запись) для APP_HOST должна указывать на публичный IP ВМ;
- порт 80 должен быть доступен из интернета (для HTTP-01 challenge);
- email-адрес администратора для уведомлений о продлении.

28.3 РУЧНАЯ УСТАНОВКА СЕРТИФИКАТА

Для использования существующего сертификата (корпоративный СА, купленный сертификат):

1. Разместите файлы сертификата и ключа в каталоге Traefik:

```

mkdir -p infra/traefik/certs
cp /path/to/cert.pem infra/traefik/certs/cert.pem
cp /path/to/key.pem infra/traefik/certs/key.pem

```

1. Смонтируйте каталог в контейнер Traefik (infra/traefik/compose.yml):

```

volumes:
  - ./certs:/etc/traefik/certs:ro

```

1. Создайте файл динамической конфигурации TLS (infra/traefik/dynamic/tls.yml):

```

tls:
  certificates:
    - certFile: /etc/traefik/certs/cert.pem
      keyFile: /etc/traefik/certs/key.pem
  stores:
    default:
      defaultCertificate:
        certFile: /etc/traefik/certs/cert.pem
        keyFile: /etc/traefik/certs/key.pem

```

1. Добавьте entrypoint HTTPS в traefik.yml:

```

entryPoints:
  http:
    address: :80
    http:
      redirections:
        entryPoint:
          to: https
          scheme: https
  https:
    address: :443

```

1. Перезапустите Traefik:

```

docker compose restart traefik

```

28.4 ОБНОВЛЕНИЕ ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

После включения HTTPS обновите:

1. Переменную `WEB_SCHEME` в корневом `.env`:

```
WEB_SCHEME=https
```

1. URL-адреса в `infra/oauth2-proxy/oauth2-proxy.cfg`:

```
redirect_url="https://<APP_HOST>/cyberstudio/oauth2/callback"
oidc_issuer_url="https://<APP_HOST>/cyberstudio/keycloak/realms/cyberstudio"
```

1. Root URL в `infra/grafana/grafana.ini`:

```
[server]
root_url = https://<APP_HOST>/cyberstudio/grafana/
```

1. Пересоздайте контейнеры:

```
docker compose up -d
```

28.5 ПРОДЛЕНИЕ СЕРТИФИКАТА

- **АСМЕ**: продление выполняется автоматически Traefik (за 30 дней до истечения);
- **Ручной сертификат**: замените файлы `cert.pem` и `key.pem` и перезапустите Traefik.

Для мониторинга срока действия сертификата проверяйте дашборд Traefik или используйте команду:

```
echo | openssl s_client -servername <APP_HOST> -connect <APP_HOST>:443 2>/dev/null
| \
    openssl x509 -noout -dates
```

29 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

29.1 ОБЩИЕ СВЕДЕНИЯ

Управление пользователями в ПО CyberStudio осуществляется на двух уровнях:

- **Keycloak** — создание учётных записей, назначение ролей, управление аутентификацией;
- **Сервис администрирования CyberStudio** — назначение пользователям пространств и проектов, управление бизнес-ролями внутри платформы.

29.2 УПРАВЛЕНИЕ ЧЕРЕЗ KEYCLOAK

29.2.1 ДОСТУП К КОНСОЛИ АДМИНИСТРАТОРА

Консоль администратора Keycloak доступна по адресу:

```
https://<APP_HOST>/cyberstudio/keycloak/admin/
```

Учётные данные администратора указаны в файле `infra/keycloak/keycloak.env` (переменные `KEYCLOAK_ADMIN` и `KEYCLOAK_ADMIN_PASSWORD`).

29.2.2 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЯ

1. В консоли Keycloak выберите realm **cyberstudio**.
2. Перейдите в раздел **Users** → **Add user**.
3. Заполните поля: Username, Email, First Name, Last Name.

4. Включите **Email verified** (если email подтверждён).

~~5-1~~. Нажмите **Create**.

~~6-2~~. Перейдите на вкладку **Credentials** → **Set password**. Установите пароль и снимите флаг **Temporary** (если не требуется принудительная смена при первом входе).

29.2.3 НАЗНАЧЕНИЕ РОЛЕЙ

Роли Keycloak определяют базовые права доступа к платформе. Для назначения:

1. Откройте карточку пользователя.
2. Перейдите на вкладку **Role Mappings**.
3. Назначьте необходимые роли realm или клиента.

29.2.4 БЛОКИРОВКА ПОЛЬЗОВАТЕЛЯ

Для временной блокировки доступа:

1. Откройте карточку пользователя.
2. Установите переключатель **Enabled** в положение **Off**.
3. Нажмите **Save**.

Заблокированный пользователь не сможет войти в систему. Существующие сессии остаются активными до истечения срока JWT-токена.

Для немедленного завершения сессий:

1. Перейдите на вкладку **Sessions**.
2. Нажмите **Sign out all sessions**.

29.2.5 УДАЛЕНИЕ ПОЛЬЗОВАТЕЛЯ

1. Откройте карточку пользователя.
2. Нажмите **Delete** → подтвердите действие.

Внимание: удаление пользователя из Keycloak не удаляет его данные и настройки внутри CyberStudio. Для полной очистки используйте также сервис администрирования CyberStudio.

29.3 УПРАВЛЕНИЕ ЧЕРЕЗ CYBERSTUDIO

Сервис администрирования CyberStudio (/cyberstudio/admin) предоставляет интерфейс для:

- назначения пользователей в пространства;
- назначения ролей пользователям в проектах;
- просмотра списка пользователей и их активности.

Подробнее — в руководстве пользователя, раздел «Сервис администрирования».

29.4 ИНТЕГРАЦИЯ С КОРПОРАТИВНЫМ SSO

Keycloak поддерживает интеграцию с корпоративными системами аутентификации через:

- LDAP / Active Directory (Identity Federation);

- SAML 2.0 (Identity Brokering);
- OpenID Connect (Identity Brokering).

Для настройки интеграции:

1. В консоли Keycloak перейдите в realm **cyberstudio** → **Identity Providers** (или **User Federation** для LDAP).
2. Добавьте и настройте провайдер.
3. Настройте маппинг атрибутов (email, имя, роли).

Примечание: настройка интеграции с корпоративным SSO зависит от инфраструктуры конкретного заказчика и выходит за рамки данного руководства. Обратитесь к документации Keycloak.

29.5 БЕЗОПАСНОСТЬ УЧЁТНЫХ ДАННЫХ

Рекомендации:

- смените пароль администратора Keycloak после первого развёртывания;
- используйте сложные пароли (минимум 12 символов, буквы, цифры, спецсимволы);
- включите политику паролей в Keycloak: realm → **Authentication** → **Policies** → **Password Policy**;
- настройте блокировку аккаунта после нескольких неудачных попыток входа: realm → **Authentication** → **Policies** → **Brute Force Detection**.

30 ОБЩИЙ ПОДХОД К ДИАГНОСТИКЕ

30.1 АЛГОРИТМ ДИАГНОСТИКИ

При возникновении проблемы рекомендуется следующий порядок действий:

1. **Определить масштаб** — затронут один сервис или несколько? Проблема у одного пользователя или у всех?
2. **Проверить состояние контейнеров:**

```
docker compose ps
```

1. **Проверить логи проблемного сервиса:**

```
docker compose logs --tail=100 <имя_сервиса>
```

1. **Проверить ресурсы хоста:**

```
docker stats --no-stream
df -h /data /monitoring-data
free -h
```

1. **Проверить инфраструктуру** — доступность PostgreSQL, Kafka, Redis, ClickHouse.
2. **Проверить сетевую связность** — маршрутизацию Traefik, DNS, OAuth2-Proxy.
3. **Изучить дашборды Grafana** — метрики и логи за период возникновения проблемы.

30.2 ОСНОВНЫЕ КОМАНДЫ ДИАГНОСТИКИ

Таблица 41.

Команды диагностики

Команда	Назначение
<code>docker compose ps</code>	Статус всех контейнеров
<code>docker compose logs --tail=50 <svc></code>	Последние 50 строк логов
<code>docker compose logs -f <svc></code>	Непрерывный вывод логов
<code>docker inspect <container></code>	Детальная информация о контейнере
<code>docker compose exec <svc> <cmd></code>	Выполнение команды внутри контейнера
<code>docker stats --no-stream</code>	Потребление ресурсов
<code>docker compose events</code>	Поток событий Docker
<code>docker system df</code>	Использование диска Docker

30.3 ТИПОВЫЕ СЦЕНАРИИ

30.3.1 ВСЕ СЕРВИСЫ НЕДОСТУПНЫ

Причины:

- Traefik не работает → `docker compose ps traefik`, проверить логи;
- DNS не резолвится → `nslookup <APP_HOST>`;
- порты 80/443 не проброшены → `ss -tlnp | grep -E ':(80|443)'`;
- Docker daemon не запущен → `systemctl status docker`.

30.3.2 МАССОВЫЙ ПЕРЕЗАПУСК СЕРВИСОВ

Причины:

- PostgreSQL недоступен → `docker compose ps postgres`, `docker compose logs postgres`;
- нехватка памяти → `free -h`, `dmesg | grep -i oom`;
- нехватка дискового пространства → `df -h`.

При массовом перезапуске в первую очередь проверяйте инфраструктурные сервисы: PostgreSQL, Kafka, Redis.

30.3.3 ОДИН СЕРВИС НЕ ЗАПУСКАЕТСЯ

```
# Проверить статус
docker compose ps <имя_сервиса>

# Проверить логи
docker compose logs --tail=100 <имя_сервиса>

# Проверить предыдущий запуск (если контейнер перезапускался)
docker logs <container_id> --previous 2>/dev/null

# Проверить события Docker
docker inspect <container_id> --format='{{.State.ExitCode}} {{.State.Error}}'
```

30.3.4 АУТЕНТИФИКАЦИЯ НЕ РАБОТАЕТ

Проверить цепочку: Traefik → OAuth2-Proxy → Keycloak.

```
# Keycloak доступен?
curl -s http://localhost:8080/cyberstudio/keycloak/realms/cyberstudio

# OAuth2-Proxy запущен?
```

```
docker compose ps oauth2-proxy
docker compose logs --tail=20 oauth2-proxy

# Traefik видит middleware?
curl -s http://localhost:8080/api/http/middlewares | jq '.[ ] | .name'
```

30.4 СБОР ДИАГНОСТИЧЕСКОЙ ИНФОРМАЦИИ

Для обращения в техническую поддержку соберите:

```
# Создать каталог для диагностики
mkdir -p /tmp/diag_$(date +%Y%m%d)

# Статус контейнеров
docker compose ps > /tmp/diag_$(date +%Y%m%d)/ps.txt

# Логи всех сервисов (последние 500 строк каждого)
docker compose logs --tail=500 > /tmp/diag_$(date +%Y%m%d)/logs.txt 2>&1

# Использование ресурсов
docker stats --no-stream > /tmp/diag_$(date +%Y%m%d)/stats.txt

# Дисковое пространство
df -h > /tmp/diag_$(date +%Y%m%d)/disk.txt
docker system df > /tmp/diag_$(date +%Y%m%d)/docker_disk.txt

# Системная информация
uname -a > /tmp/diag_$(date +%Y%m%d)/system.txt
free -h >> /tmp/diag_$(date +%Y%m%d)/system.txt

# Архивировать
tar -czf /tmp/diag_$(date +%Y%m%d).tar.gz /tmp/diag_$(date +%Y%m%d)/
```

31 ДИАГНОСТИКА POSTGRESQL

Дашборд Grafana: PostgreSQL.

31.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 42.

Типовые проблемы PostgreSQL

Симптом	Возможная причина	Действия
Контейнер postgres в статусе unhealthy	Ошибка запуска, повреждение данных	Проверить логи: <code>docker compose logs postgres</code>
API-сервисы не запускаются, ошибки подключения	PostgreSQL недоступен	Проверить <code>docker compose ps postgres</code> , перезапустить
Медленные запросы, высокая нагрузка CPU	Неоптимальные запросы, нехватка shared_buffers	Проверить <code>pg_stat_activity</code> , увеличить shared buffers
Ошибка too many connections	Превышен лимит max_connections	Перезапустить сервисы с утечкой подключений, увеличить лимит
Ошибка No space left on device	Заполнен диск	Очистить WAL, увеличить диск
Ошибка CDC / replication slot	Слот репликации переполнен	Проверить kafka-connect, удалить неактивные слоты

31.2 ПРОВЕРКА ПОДКЛЮЧЕНИЯ

```
# Проверить доступность
docker compose exec postgres pg_isready -U postgres

# Подключиться к PostgreSQL
docker compose exec postgres psql -U <POSTGRES_USER>
```

31.3 ПРОВЕРКА АКТИВНЫХ ПОДКЛЮЧЕНИЙ

```
docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT datname, count(*) as connections
FROM pg_stat_activity
GROUP BY datname
ORDER BY connections DESC;
"
```

Если количество подключений приближается к `max_connections` (500), определите сервисы с наибольшим числом подключений и перезапустите их.

31.4 МЕДЛЕННЫЕ ЗАПРОСЫ

```
# Текущие длительные запросы (> 30 секунд)
docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT pid, now() - pg_stat_activity.query_start AS duration,
       query, state
FROM pg_stat_activity
WHERE (now() - pg_stat_activity.query_start) > interval '30 seconds'
AND state != 'idle'
ORDER BY duration DESC;
"

# Принудительное завершение запроса
docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT pg_terminate_backend(<pid>);
"
```

31.5 РАЗМЕР БАЗ ДАННЫХ

```
docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT datname, pg_size_pretty(pg_database_size(datname)) as size
FROM pg_database
WHERE datistemplate = false
ORDER BY pg_database_size(datname) DESC;
"
```

31.6 СЛОТЫ РЕПЛИКАЦИИ (CDC)

```
# Проверить активные слоты
lag docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT slot_name, active, restart_lsn, confirmed_flush_lsn,
       pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(), restart_lsn)) AS
FROM pg_replication_slots;
"
```

Если LAG слота репликации растёт, Kafka Connect не успевает обрабатывать изменения. Проверьте состояние `kafka-connect`:

```
docker compose logs --tail=50 kafka-connect
curl -s http://localhost:8083/connectors | jq .
```

Для удаления неактивного слота (только если коннектор будет пересоздан):

```
docker compose exec postgres psql -U <POSTGRES_USER> -c "
SELECT pg_drop_replication_slot('<slot_name>');
"
```

31.7 ДИСКОВОЕ ПРОСТРАНСТВО

```
# Размер каталога данных PostgreSQL
du -sh ${DATA_BASE_DIR}/postgres/data

# Размер WAL
docker compose exec postgres psql -U <POSTGRES_USER> -c "
```

```
SELECT pg_size_pretty(sum(size)) FROM pg_ls_waldir();
```

Если WAL занимает слишком много места, проверьте слоты репликации — неактивные слоты удерживают WAL от удаления.

32 ДИАГНОСТИКА CLICKHOUSE

Дашборд Grafana: **ClickHouse**.

32.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 43.

Типовые проблемы ClickHouse

Симптом	Возможная причина	Действия
Контейнер clickhouse в статусе unhealthy	Ошибка запуска, порт занят	Проверить логи: <code>docker compose logs clickhouse</code>
Медленные запросы в сервисе исторических данных	Большой объём данных, неоптимальный запрос	Проверить <code>system.query_log</code> , добавить фильтры по времени
Ошибка Too many parts	Слишком частая вставка мелких батчей	Увеличить размер батча в <code>datalake-worker</code>
Рост дискового пространства	Накопление данных, недостаточный мерж	Проверить <code>system.parts</code> , настроить TTL
Ошибка миграций clickhouse-migrations	ClickHouse недоступен, ошибка DDL	Проверить логи миграций

32.2 ПРОВЕРКА ДОСТУПНОСТИ

```
# HTTP-интерфейс
curl -s http://localhost:8123/ping

# Подключение через CLI
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password>
```

32.3 ТЕКУЩИЕ ЗАПРОСЫ

```
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT query_id, user, elapsed, query
  FROM system.processes
  ORDER BY elapsed DESC"
```

Для принудительного завершения запроса:

```
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "KILL QUERY WHERE query_id = '<query_id>'"
```

32.4 РАЗМЕР ТАБЛИЦ

```
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT database, table,
  formatReadableSize(sum(bytes_on_disk)) AS size,
  sum(rows) AS total_rows,
  count() AS parts
  FROM system.parts
  WHERE active
  GROUP BY database, table
  ORDER BY sum(bytes_on_disk) DESC"
```

32.5 ПРОБЛЕМА «TOO MANY PARTS»

Если в логах появляется ошибка Too many parts, это означает, что данные вставляются слишком частыми мелкими батчами:

```
# Проверить количество активных частей
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT database, table, count() AS parts
          FROM system.parts
          WHERE active
          GROUP BY database, table
          HAVING parts > 300
          ORDER BY parts DESC"
```

Решение: увеличить параметр CLICKHOUSE_MAX_BATCH_SIZE в env-файле datalake-worker (по умолчанию: 30000).

32.6 МЕРЖ (ФОНОВЫЕ ОПЕРАЦИИ)

```
# Текущие операции мержа
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT database, table, elapsed, progress,
          formatReadableSize(total_size_bytes_compressed) AS size
          FROM system.merges"
```

32.7 ДИСКОВОЕ ПРОСТРАНСТВО

```
# Общий размер данных ClickHouse
du -sh ${DATA_BASE_DIR}/clickhouse/data

# Размер по базам данных
docker compose exec clickhouse clickhouse-client \
  --user <user> --password <password> \
  --query "SELECT database,
          formatReadableSize(sum(bytes_on_disk)) AS size
          FROM system.parts WHERE active
          GROUP BY database"
```

33 ДИАГНОСТИКА KAFKA

Дашборд Grafana: **Kafka**.

33.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 44.

Типовые проблемы Kafka

Симптом	Возможная причина	Действия
Контейнер kafka не запускается	Ошибка инициализации KRaft	Проверить логи, убедиться в корректности CLUSTER_ID
Высокий consumer lag	Потребители не успевают обработать сообщения	Проверить datalake-worker, увеличить число экземпляров
Kafka Connect не создаёт коннекторы	Schema Registry недоступен	Проверить docker compose ps schema-registry
Ошибка UNKNOWN_TOPIC_OR_PARTITION	Топик не создан	Проверить kafka-manager-api
Дискосвое пространство заполнено	Накопление сегментов логов	Уменьшить retention, очистить старые данные

33.2 ПРОВЕРКА СОСТОЯНИЯ КЛАСТЕРА

```
# Идентификатор кластера
docker compose exec kafka kafka-cluster.sh cluster-id \
  --bootstrap-server localhost:9092

# Список брокеров
docker compose exec kafka kafka-metadata.sh --snapshot \
  /var/lib/kafka/data/__cluster_metadata-0/00000000000000000000.log \
  --broker-list
```

33.3 УПРАВЛЕНИЕ ТОПИКАМИ

```
# Список топиков
docker compose exec kafka kafka-topics.sh --list \
  --bootstrap-server localhost:9092

# Описание топика
docker compose exec kafka kafka-topics.sh --describe \
  --topic <topic_name> \
  --bootstrap-server localhost:9092

# Создание топика (если kafka-manager-api не создал)
docker compose exec kafka kafka-topics.sh --create \
  --topic <topic_name> \
  --partitions 1 \
  --replication-factor 1 \
  --bootstrap-server localhost:9092
```

33.4 CONSUMER LAG

Consumer lag показывает, сколько сообщений не обработано потребителем.

```
# Список consumer groups
docker compose exec kafka kafka-consumer-groups.sh --list \
  --bootstrap-server localhost:9092

# Lag конкретной группы
docker compose exec kafka kafka-consumer-groups.sh --describe \
  --group <group_name> \
  --bootstrap-server localhost:9092
```

Если lag постоянно растёт:

- проверьте логи сервиса-потребителя (обычно datalake-worker);
- увеличьте число экземпляров потребителя: `docker compose up -d --scale datalake-worker=2;`
- проверьте производительность ClickHouse (куда записываются данные).

33.5 KAFKA CONNECT

```
# Список коннекторов
curl -s http://localhost:8083/connectors | jq .

# Статус коннектора
curl -s http://localhost:8083/connectors/<connector>/status | jq .

# Перезапуск коннектора
curl -X POST http://localhost:8083/connectors/<connector>/restart

# Перезапуск задачи коннектора
curl -X POST http://localhost:8083/connectors/<connector>/tasks/0/restart
```

Если коннектор в статусе FAILED:

```
# Получить описание ошибки
```

```
curl -s http://localhost:8083/connectors/<connector>/status | jq
'.tasks[0].trace'
```

33.6 SCHEMA REGISTRY

```
# Проверка доступности
docker compose exec schema-registry bash -c "nc -z localhost 8081 && echo OK"

# Список зарегистрированных схем
curl -s http://localhost:8081/subjects | jq .
```

33.7 ДИСКОВОЕ ПРОСТРАНСТВО

```
# Размер данных Kafka
du -sh ${DATA_BASE_DIR}/kafka/data

# Размер по топикам
docker compose exec kafka bash -c "du -sh /var/lib/kafka/data/*" | sort -rh |
head -20
```

Для уменьшения занимаемого места можно уменьшить retention топиков:

```
docker compose exec kafka kafka-configs.sh \
--alter --topic <topic_name> \
--add-config retention.ms=3600000 \
--bootstrap-server localhost:9092
```

33.8 KAFKA UI

Веб-интерфейс Kafka UI (/cyberstudio/kafka-ui) предоставляет визуальный интерфейс для:

- просмотра топиков и сообщений;
- мониторинга consumer groups и lag;
- управления коннекторами Kafka Connect;
- просмотра схем Schema Registry.

34 ДИАГНОСТИКА TRAEFIK

Дашборд Traefik: https://<APP_HOST>/cyberstudio/traefik/dashboard/.

34.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 45.

Типовые проблемы Traefik

Симптом	Возможная причина	Действия
Все сервисы возвращают 404	Traefik не запущен или не видит контейнеры	Проверить docker compose ps traefik, логи
Конкретный сервис возвращает 404	Контейнер сервиса не запущен или labels некорректны	Проверить docker compose ps <service>, labels
Ошибка 502 Bad Gateway	Backend-сервис не отвечает	Проверить healthcheck и логи backend-сервиса
Ошибка 504 Gateway Timeout	Backend-сервис отвечает слишком долго	Увеличить readTimeout в traefik.yml
Ошибка 401 Unauthorized	OAuth2-Прoxy не работает	Проверить oauth2-proxy и Keycloak
HTTPS не работает	Ошибка TLS-конфигурации	Проверить сертификаты, endpoints

34.2 ДАШБОРД TRAEFIK

Дашборд Traefik показывает:

- список роутеров и их статус;
- список middleware;
- список сервисов и их health;
- entrypoints.

Доступ: `https://<APP_HOST>/cyberstudio/traefik/dashboard/` (HTTP Basic Auth).

34.3 ПРОВЕРКА РОУТЕРОВ

```
# Список всех роутеров
curl -s http://localhost:8080/api/http/routers | jq '.[].name'

# Детали конкретного роутера
curl -s http://localhost:8080/api/http/routers/<router_name>@docker | jq .
```

34.4 ПРОВЕРКА MIDDLEWARE

```
# Список middleware
curl -s http://localhost:8080/api/http/middlewares | jq '.[].name'
```

Убедитесь, что middleware `oauth2-signin`, `oauth2-verify` и `auth@file` присутствуют.

34.5 ПРОВЕРКА СЕРВИСОВ

```
# Список backend-сервисов
curl -s http://localhost:8080/api/http/services | jq '.[].name'

# Статус конкретного сервиса
curl -s http://localhost:8080/api/http/services/<service_name>@docker | jq .
```

34.6 ОШИБКА 502 BAD GATEWAY

Ошибка 502 означает, что Traefik не может подключиться к backend-сервису:

1. Проверить, запущен ли контейнер:

```
docker compose ps <имя_сервиса>
```

1. Проверить healthcheck:

```
docker inspect <container_id> --format='{{json .State.Health.Status}}'
```

1. Проверить доступность порта внутри Docker-сети:

```
docker compose exec traefik wget -q --spider http://<service>:8000/healthcheck &&
echo OK
```

34.7 ОШИБКА 504 GATEWAY TIMEOUT

Ошибка 504 возникает при превышении таймаута ответа backend-сервиса. По умолчанию `readTimeout` установлен в 3600 секунд (1 час).

Если длительные операции (обучение моделей, импорт данных) превышают этот таймаут, увеличьте значение в `traefik.yml`:

```
entryPoints:
  http:
```

```
address: :80
transport:
  respondingTimeouts:
    readTimeout: 7200    # 2 часа
```

34.8 ЛОГИ TRAEFIK

```
docker compose logs --tail=50 traefik
```

Для включения детального логирования добавьте в `traefik.yml`:

```
log:
  level: DEBUG
accessLog: {}
```

Внимание: access log генерирует большой объём данных. Включайте только на время диагностики.

35 ДИАГНОСТИКА KEYCLOAK

35.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 46.

Типовые проблемы Keycloak

Симптом	Возможная причина	Действия
Keycloak не запускается	PostgreSQL недоступен	Проверить <code>docker compose ps postgres</code>
Страница входа не открывается	Контейнер не готов (start_period: 120 с)	Дождаться, проверить логи
Ошибка «Invalid redirect URL»	Некорректный <code>redirect_url</code> в OAuth2-Proxy	Проверить настройки клиента в Keycloak
Токен не валидируется	Несоответствие <code>oidc_issuer_url</code>	Проверить URL в <code>oauth2-proxy.cfg</code>
Пользователь не может войти	Аккаунт заблокирован или неверный пароль	Проверить статус в консоли Keycloak
OAuth2-Proxy возвращает 500	Keycloak недоступен или секрет клиента неверен	Проверить <code>oauth2-proxy.cfg</code> , логи <code>oauth2-proxy</code>

35.2 ПРОВЕРКА ДОСТУПНОСТИ

```
# Healthcheck Keycloak
curl -s -o /dev/null -w "%{http_code}" \
  http://localhost:8080/cyberstudio/keycloak/realms/master

# Проверка realm cyberstudio
curl -s http://localhost:8080/cyberstudio/keycloak/realms/cyberstudio | jq .realm
```

35.3 ЛОГИ KEYCLOAK

```
docker compose logs --tail=50 keycloak
```

Для увеличения детализации логов измените переменную `KEYCLOAK_LOG_LEVEL` в `keycloak.env`:

```
KEYCLOAK_LOG_LEVEL=DEBUG
```

Внимание: DEBUG-логи Keycloak очень объёмны. Включайте только на время диагностики.

35.4 ПРОБЛЕМЫ OAUTH2-PROXY

```
# Логи OAuth2-Proxy
```

```
docker compose logs --tail=50 oauth2-proxy
```

```
# Проверка конфигурации
docker compose exec oauth2-proxy cat /etc/oauth2-proxy/oauth2-proxy.cfg
```

Типичные ошибки в логах OAuth2-Proxy:

- provider failed to verify id token — несоответствие `oidc_issuer_url`. URL должен совпадать с тем, который возвращает Keycloak;
- invalid client_id or client_secret — секрет клиента `oauth2-proxy` в Keycloak не совпадает с `client_secret` в `oauth2-proxy.cfg`;
- cookie_secret must be 16, 24, or 32 bytes — невалидный `cookie_secret`.

35.5 ПРОВЕРКА КОНФИГУРАЦИИ КЛИЕНТА

Убедитесь, что клиент `oauth2-proxy` настроен корректно в Keycloak:

1. Откройте консоль администратора:
`https://<APP_HOST>/cyberstudio/keycloak/admin/`.
2. Перейдите в realm **cyberstudio** → **Clients** → **oauth2-proxy**.
3. Проверьте:

- **Valid** **Redirect** **URIs** содержит `http(s)://<APP_HOST>/cyberstudio/oauth2/callback;`
- **Web Origins** содержит `http(s)://<APP_HOST>;`
- **Client Secret** совпадает с `client_secret` в `oauth2-proxy.cfg`.

35.6 ПРОВЕРКА ТОКЕНОВ

Для проверки получения токена:

```
# Получить токен (Resource Owner Password Grant)
curl -s -X POST \

"http://localhost:8080/cyberstudio/keycloak/realms/cyberstudio/protocol/openid-connect/token" \
  -d "client_id=oauth2-proxy" \
  -d "client_secret=<client_secret>" \
  -d "grant_type=password" \
  -d "username=<user>" \
  -d "password=<password>" | jq .access_token
```

Если токен получен — Keycloak работает корректно, проблема в OAuth2-Proxy или Traefik middleware.

35.7 ЭКСПОРТ REALM ДЛЯ ОТЛАДКИ

```
docker compose exec keycloak /opt/keycloak/bin/kc.sh export \
  --dir /tmp/export --realm cyberstudio
docker compose cp keycloak:/tmp/export/ /tmp/keycloak-export/
```

36 ДИАГНОСТИКА МОНИТОРИНГА

36.1 СИМПТОМЫ И РЕШЕНИЯ

Таблица 47.

Типовые проблемы мониторинга

Симптом	Возможная причина	Действия
Нет метрик в Grafana	Alloy не собирает данные,	Проверить Alloy UI, логи alloy

Симптом	Возможная причина	Действия
	VictoriaMetrics недоступен	
Нет логов в Grafana	Alloy не отправляет логи, Loki недоступен	Проверить Loki readiness, логи alloy
Grafana не открывается	Контейнер grafana не запущен	Проверить docker compose ps grafana
Дашборды пустые	Источник данных не настроен	Проверить provisioning datasources
Алерты не срабатывают	SMTP не настроен, contact point не работает	Проверить grafana.ini, тест contact point
VictoriaMetrics: высокое потребление диска	Длинный retention, много метрик	Уменьшить retention, проверить кардинальность
Loki: ошибка ingestion rate	Слишком высокая скорость записи логов	Увеличить лимиты в config.yml

36.2 GRAFANA ALLOY

36.2.1 ПРОВЕРКА СОСТОЯНИЯ

Веб-интерфейс Alloy: https://<APP_HOST>/cyberstudio/alloy (HTTP Basic Auth).

В интерфейсе отображаются:

- статус компонентов (targets, exporters, processors);
- здоровье каждого компонента (healthy/unhealthy);
- метрики производительности агента.

```
# Логи Alloy
docker compose logs --tail=50 alloy
```

36.2.2 ПРОВЕРКА TARGETS

В веб-интерфейсе Alloy проверьте раздел targets:

- все сервисы с лейблом `prometheus.io/scrape=true` должны быть обнаружены;
- статус каждого target должен быть `up`.

Если target отсутствует, проверьте Docker labels в compose-файле сервиса.

36.2.3 ПРОБЛЕМЫ С ЭКСПОРТЁРАМИ

```
# Проверить экспортёр PostgreSQL
docker compose exec alloy wget -qO- \
http://localhost:12345/component/prometheus.exporter.postgres.postgres/metrics | head - 5

# Проверить экспортёр Redis
docker compose exec alloy wget -qO- \
http://localhost:12345/component/prometheus.exporter.redis.redis/metrics | head -5
```

36.3 VICTORIAMETRICS

```
# Проверка доступности
curl -s http://localhost:8428/cyberstudio/victoria-metrics/api/v1/status/tsdb | \
jq .

# Количество временных рядов
curl -s http://localhost:8428/cyberstudio/victoria-metrics/api/v1/status/tsdb | \
jq '.data.totalSeries'

# Использование диска
du -sh ${MONITORING_DATA_BASE_DIR}/victoria-metrics/data
```

Если VictoriaMetrics потребляет слишком много диска, уменьшите период хранения в compose-файле:

```
command:
  - "-retentionPeriod=7d"    # было 14d
```

36.4 LOKI

```
# Проверка готовности
curl -s http://localhost:3100/ready

# Статистика ingestion
curl -s http://localhost:3100/metrics | grep loki_ingester

# Логи Loki
docker compose logs --tail=50 loki
```

Типичные проблемы Loki:

- **Rate limit exceeded** — увеличьте `max_global_streams_per_user` в `config.yml`;
- **Context deadline exceeded** — увеличьте `query_timeout`;
- **WAL replay slow** — при большом WAL перезапуск Loki может занять несколько минут.

36.5 GRAFANA

```
# Проверка здоровья
curl -s http://localhost:3000/api/health | jq .

# Логи Grafana
docker compose logs --tail=50 grafana
```

36.5.1 ПРОВЕРКА ИСТОЧНИКОВ ДАННЫХ

```
# Список datasources
curl -s http://localhost:3000/api/datasources | jq '.[].name'
```

Если `datasource` отсутствует, проверьте файлы в каталоге `provisioning/datasources/`.

36.5.2 ТЕСТ EMAIL-УВЕДОМЛЕНИЙ

В интерфейсе Grafana: **Alerting** → **Contact points** → выбрать точку контакта → **Test**.
Если email не отправляется, проверьте настройки SMTP в `grafana.ini` и доступность SMTP-сервера с хоста:

```
docker compose exec grafana nc -z <smtp_host> <smtp_port> && echo OK || echo FAIL
```

37 ДИАГНОСТИКА СЕРВИСОВ CYBERSTUDIO

37.1 ОБЩИЕ ПРИНЦИПЫ

Все прикладные API-сервисы CyberStudio имеют единообразную структуру диагностики:

- эндпоинт `/healthcheck` — проверка доступности;
- эндпоинт `/metrics` — метрики Prometheus;
- логи Docker — `docker compose logs <service>`;
- зависимости — PostgreSQL, Kafka, Redis, ClickHouse (в зависимости от сервиса).

37.2 СИМПТОМЫ И РЕШЕНИЯ

Таблица 48.

Типовые проблемы сервисов CyberStudio

Симптом	Возможная причина	Действия
Сервис в статусе Restarting	Ошибка при запуске, недоступна зависимость	Проверить логи, проверить зависимости
OOM Killed (Exit code 137)	Нехватка оперативной памяти	Увеличить memory limit, проверить утечки
Ошибка подключения к БД	PostgreSQL недоступен или БД не создана	Проверить postgres, проверить env-файл
Ошибка подключения к Kafka	Kafka недоступна или топик не создан	Проверить kafka, kafka-manager-api
Ошибка подключения к Redis	Redis недоступен	Проверить redis, номер DB
Ошибка подключения к MinIO	MinIO недоступен или бакет не создан	Проверить minio, создать бакет
HTTP 500 при вызове API	Внутренняя ошибка сервиса	Проверить логи, передать в разработку
Длительное время отклика	Высокая нагрузка, медленные запросы к БД	Проверить docker stats, метрики Grafana

37.3 ДИАГНОСТИКА КОНКРЕТНОГО СЕРВИСА

```
# 1. Проверить статус
docker compose ps <имя_сервиса>

# 2. Проверить healthcheck
curl -s -o /dev/null -w "%{http_code}" \
  http://<APP_HOST>/cyberstudio/api/<service>/healthcheck

# 3. Проверить логи (последние 100 строк)
docker compose logs --tail=100 <имя_сервиса>

# 4. Проверить потребление ресурсов
docker stats --no-stream --format "table {{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}"
| \
  grep <имя_сервиса>

# 5. Проверить переменные окружения
docker compose exec <имя_сервиса> env | sort

# 6. Проверить сетевую связность с зависимостями
docker compose exec <имя_сервиса> bash -c "
  nc -z postgres 5432 && echo 'PG: OK' || echo 'PG: FAIL'
  nc -z kafka 9092 && echo 'Kafka: OK' || echo 'Kafka: FAIL'
  nc -z redis 6379 && echo 'Redis: OK' || echo 'Redis: FAIL'
"
```

37.4 OOM KILLED (EXIT CODE 137)

Если контейнер завершается с кодом 137, он был уничтожен из-за нехватки памяти:

```
# Проверить OOM в системных логах
dmesg | grep -i "oom\|killed" | tail -20

# Проверить текущее потребление памяти
docker stats --no-stream --format "table {{.Name}}\t{{.MemUsage}}\t{{.MemPerc}}"
```

Решения:

— увеличить объём RAM виртуальной машины;

— установить ограничения памяти для контейнеров через

deploy.resources.limits.memory в compose-файле;

— уменьшить пул подключений к БД (переменная `PG_POOL_SIZE` в `env`-файле сервиса).

37.5 RESTART LOOP

Если сервис постоянно перезапускается:

```
# Просмотр истории перезапусков
docker inspect <container_id> --format='{{.RestartCount}}'

# Логи предыдущего запуска
docker logs <container_id> --previous 2>/dev/null
```

Частые причины:

- недоступна зависимость (PostgreSQL, Kafka) → `autoheal` перезапускает сервис, но зависимость по-прежнему недоступна;
- ошибка миграции БД → сервис не может запуститься, пока не будет исправлена схема;
- некорректная переменная окружения → проверить `env`-файл.

37.6 DEPENDENCIES CHAIN

При проблемах с запуском проверяйте зависимости по цепочке:

```
PostgreSQL → Keycloak → OAuth2-Proxy
Kafka → Schema Registry → Kafka Connect → kafka-manager-api
kafka-manager-api → API-сервисы
ClickHouse → clickhouse-migrations → datalake-api → datalake-worker
```

Если не запускается сервис в конце цепочки, проблема может быть в любом из предшествующих компонентов.

37.7 ПРОВЕРКА DATALAKE-WORKER

`datalake-worker` — компонент с повышенными требованиями к ресурсам. При проблемах с приёмом данных:

```
# Логи worker
docker compose logs --tail=100 datalake-worker

# Consumer lag (показывает задержку обработки)
docker compose exec kafka kafka-consumer-groups.sh --describe \
--group <group_name> --bootstrap-server localhost:9092
```

Ключевые параметры `datalake-worker` (из `datalake-worker.env`):

- `MESSAGE_BATCH_COUNT=10000` — размер батча сообщений;
- `CLICKHOUSE_MAX_BATCH_SIZE=30000` — размер батча записи в ClickHouse;
- `MESSAGE_MAX_MEMORY_CHUNK=30` — максимальный размер чанка в памяти (МБ).

37.8 ЛОГИРОВАНИЕ В ФОРМАТЕ JSON

Большинство API-сервисов выводят структурированные логи в формате JSON. Для удобного просмотра:

```
docker compose logs --tail=10 model-api 2>&1 | \
python3 -m json.tool 2>/dev/null || \
docker compose logs --tail=10 model-api
```

Для поиска ошибок с использованием `jq`:

```
docker compose logs --tail=1000 model-api 2>&1 | \
grep -o '{.*}' | jq 'select(.level == "ERROR")'
```