АО "НТЦ ИТ РОСА"

**ПРОГРАММНЫЙ КОМПЛЕКС**

**"ПОРТАЛ РАЗРАБОТЧИКА РОСА"**

**Версия 2.0**

**Руководство администратора**

РСЮК.10901-02 32 01

Листов 46

2025

АННОТАЦИЯ

Данное руководство предназначено для системных администраторов, осуществляющих установку, настройку и поддержку работы программного комплекса "Портал разработчика РОСА" (шифр – РСЮК.10901) (далее – Портал разработчика, Комплекс).

В руководстве содержится общая информация о Комплексе и его архитектуре, а также сведения, необходимые для его установки, настройки, запуска и мониторинга.

Перед эксплуатацией Комплекса рекомендуется внимательно ознакомиться с настоящим руководством.

Для разработки документа использованы ссылки на следующие стандарты:

* ГОСТ Р 2.105-2019 "Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам";
* ГОСТ 2.601 "Единая система программной документации. Виды программных документов";
* ГОСТ 19.101-77 "Единая система программной документации.  
  Виды программ и программных документов";
* ГОСТ 19.105-78 "Единая система программной документации. Общие требования к программным документам";
* ГОСТ 19.503-79 "Единая система программной документации. Руководство системного программиста".

Настоящий документ подготовлен в соответствии с технологической инструкцией "РОСА. Регламент формирования документации к программным продуктам" (шифр РСЮК.11001-02 90 01).

СОДЕРЖАНИЕ

[1 Общие сведения 5](#__RefHeading___1)

[1.1 Назначение 5](#__RefHeading___2)

[1.2 Функции 5](#__RefHeading___3)

[2 Архитектура комплекса 6](#__RefHeading___4)

[2.1 Основные компоненты 6](#__RefHeading___5)

[2.2 Состав плагинов 6](#__RefHeading___6)

[2.3 Система управления файлами 7](#__RefHeading___7)

[2.4 База данных 7](#__RefHeading___8)

[2.5 Интеграция с сервисом РОСА ID 7](#__RefHeading___9)

[2.6 Winter CMS 8](#__RefHeading___10)

[2.7 Сетевое взаимодействие и шифрование трафика 8](#__RefHeading___11)

[2.8 Обеспечение отказоустойчивости 9](#__RefHeading___12)

[2.9 Масштабируемость 9](#__RefHeading___13)

[3 Технические и программные требования 10](#__RefHeading___14)

[3.1 Требования к программному обеспечению 10](#__RefHeading___15)

[3.2 Требования к аппаратному обеспечению 11](#__RefHeading___16)

[3.3 Требования к клиентской части 11](#__RefHeading___17)

[4 Установка и запуск комплекса 14](#__RefHeading___18)

[4.1 Развёртывание программного окружения 14](#__RefHeading___19)

[4.1.1 Установка среды контейнеризации Docker 14](#__RefHeading___20)

[4.1.2 Развёртывание маршрутизатора входящих запросов Traefik 15](#__RefHeading___21)

[4.1.3 Установка и подготовка PostgreSQL 17](#__RefHeading___22)

[4.1.4 Подготовка постоянного хранилища 18](#__RefHeading___23)

[4.2 Установка Комплекса 19](#__RefHeading___24)

[4.3 Запуск Комплекса 19](#__RefHeading___25)

[4.3.1 Загрузка через командную строку 19](#__RefHeading___26)

[4.3.2 Вход в комплекс 20](#__RefHeading___27)

[5 Тонкая настройка операционной системы 21](#__RefHeading___28)

[5.1 Рекомендованные настройки производительности 21](#__RefHeading___29)

[5.1.1 Инструменты настройки ядра 21](#__RefHeading___30)

[5.1.2 Сетевые параметры 23](#__RefHeading___31)

[5.1.3 Мониторинг и корректировка параметров 24](#__RefHeading___32)

[5.1.4 Управление памятью 24](#__RefHeading___33)

[5.1.5 Оптимизация файловой системы и дискового ввода-вывода 26](#__RefHeading___34)

[5.1.6 Мониторинг и корректировка параметров ввода-вывода 27](#__RefHeading___35)

[5.1.7 Планирование процессов 28](#__RefHeading___36)

[5.1.8 Привязка к процессору и управление CPU 29](#__RefHeading___37)

[5.1.9 Мониторинг использования CPU и корректировка 30](#__RefHeading___38)

[5.2 Рекомендованные настройки безопасности 30](#__RefHeading___39)

[5.2.1 Изоляция процессов и данных 30](#__RefHeading___40)

[5.2.2 Шифрование 31](#__RefHeading___41)

[5.2.3 Брандмауэр 31](#__RefHeading___42)

[5.3 Логирование 31](#__RefHeading___43)

[6 Мониторинг и диагностика 33](#__RefHeading___44)

[6.1 Расположение программного обеспечения и журналов событий 33](#__RefHeading___45)

[6.2 Хранение постоянных данных 36](#__RefHeading___46)

[6.3 Рекомендации по мониторингу состояния Комплекса 37](#__RefHeading___47)

[6.3.1 Архитектура системы мониторинга 38](#__RefHeading___48)

[6.3.2 Мониторинг серверной инфраструктуры 38](#__RefHeading___49)

[6.3.3 Мониторинг доступности веб-интерфейса 39](#__RefHeading___50)

[7 Управление контентом через Winter CMS 41](#__RefHeading___51)

[7.1 Назначение и возможности Winter CMS 41](#__RefHeading___52)

[7.2 Языковая поддержка 41](#__RefHeading___53)

[7.3 Плагины и расширения 41](#__RefHeading___54)

[7.4 Резервное копирование контента 42](#__RefHeading___55)

[7.5 Рабочий интерфейс 42](#__RefHeading___56)

[7.5.1 Дашборд администратора 42](#__RefHeading___57)

[7.5.2 Управление структурой сайта (раздел CMS) 42](#__RefHeading___58)

[7.5.3 Медиа 43](#__RefHeading___59)

[7.5.4 Страницы 43](#__RefHeading___60)

[7.5.5 Блог 43](#__RefHeading___61)

[7.5.6 Пользователи 44](#__RefHeading___62)

[7.5.7 Авторы 44](#__RefHeading___63)

[7.5.8 Настройки 44](#__RefHeading___64)

[7.5.9 Аккаунт 45](#__RefHeading___65)

[Перечень сокращений 46](#__RefHeading___66)

1. Общие сведения
   1. Назначение

Программный комплекс "Портал разработчика РОСА" предназначен для предоставления разработчикам доступа к инструментам, полному комплекту технической документации, примерам кода и другой информации, необходимой для создания, тестирования и публикации программного обеспечения, разрабатываемого под управление мобильной операционной системы "РОСА Мобайл".

Комплекс обеспечивает централизованный доступ ко всем ресурсам, необходимым для эффективной разработки программных продуктов, соответствующих требованиям ОС "РОСА Мобайл".

* 1. Функции

Комплекс реализует следующие основные функции:

* предоставление разработчикам доступа к актуальной и структурированной технической документации;
* размещение готовых примеров фрагментов исходного кода, которые могут быть использованы как стартовые точки для новых проектов;
* обеспечение условий для ускоренной разработки, создания прототипов и верификации программного обеспечения под операционную систему "РОСА Мобайл".

1. Архитектура комплекса

Программный комплекс "Портал разработчика" построен по модульной архитектуре. Архитектура реализована на базе Winter CMS и включает серверную часть (бэкенд) для администрирования и управления контентом, а также клиентскую часть, доступную пользователям через веб-интерфейс браузера. Клиентская часть предоставляет разработчикам доступ к функциональности портала, включая документацию, шаблоны, примеры кода и другие ресурсы, необходимые для разработки ПО для "РОСА Мобайл".

Комплекс разворачивается в контейнеризированной среде (Docker) и взаимодействует с внешними сервисами аутентификации и хранения данных.

* 1. Основные компоненты

Состав архитектуры Комплекса включает в себя следующие части:

* CMS-сервер (Winter CMS) — основной элемент серверной части. Обеспечивает управление структурой, содержимым и логикой портала;
* плагины Winter CMS — модули, расширяющие функциональность CMS;
* обратный прокси (Traefik) — обеспечивает маршрутизацию и балансировку трафика между компонентами;
* СУБД PostgreSQL — хранение основной структурированной информации;
* кеширующая система Redis — хранение сессий и промежуточных данных;
* система поиска Typesense — полнотекстовый поиск по материалам портала;
* интеграция с внешней системой аутентификации РОСА ID.
  1. Состав плагинов

В состав Комплекса входят следующие служебные плагины локальной разработки:

* Local – модуль расширения ядра CMS и настройки среды;
* Local.Author – управление авторами контента;
* Local.OAuth – реализация авторизации через сервис ROSA ID;
* Local.PagesViewBag – передача дополнительных данных на страницы;
* Local.Search – интеграция поиска с использованием Typesense;
* Local.Snippets – внедрение переиспользуемых блоков контента;
* Local.VideoBlog – реализация функциональности видеоблога.

Сторонние и системные плагины в составе Комплекса:

* Winter.Blog – настройка блога;
* Winter.Pages – настройки страницы;
* Winter.Sitemap – настройки карты сайта;
* Winter.Translate – настройки мультиязычности;
* Winter.User — стандартные модули CMS Winter;
* StudioBosco.SeoExtension — настройка параметров SEO;
* WebVPF.Robots — настройка правил индексации (robots.txt).
  1. Система управления файлами

В состав Комплекса входит файловое хранилище, доступное из интерфейса CMS и через директории Docker-контейнеров. Файлы включают:

* загружаемые пользователями ресурсы (медиафайлы);
* журналы событий;
* скомпилированные шаблоны страниц;
* кешированные данные.

В Комплексе реализовано двухуровневое кеширование данных:

* на диске — редко изменяемые статические ресурсы;
* в системе Redis — сессии, задачи и динамически изменяемые данные.

Работа с Комплексом осуществляется под выделенной учетной записью, при этом доступ к файлам ограничен на уровне файловой системы, что обеспечивает дополнительный уровень безопасности.

Каждое окружение (стенд) использует уникальное хранилище, исключающее пересечения и взаимное влияние данных.

* 1. База данных

В качестве системы управления базами данных в составе Комплекса используется PostgreSQL. Контейнеры базы данных запускаются с использованием отдельной системной учетной записи, что обеспечивает изоляцию и повышенную безопасность.

Доступ к данным СУБД ограничен на уровне ОС и конфигурации контейнеров.

* 1. Интеграция с сервисом РОСА ID

Комплекс интегрирован с системой аутентификации РОСА ID, через которую осуществляется авторизация пользователей и администраторов Комплекса. РОСА ID выполняет роль idP (Identity Provider), который реализован по стандарту OAuth с поддержкой OpenID. В сервисе используется механизм JSON Web Token (JWT) для защиты сессий и запросов.

Функциональные зоны РОСА ID:

* неавторизованная зона — регистрация, вход в Портал разработчика;
* личный кабинет пользователя — доступ к персональным данным, настройкам безопасности (включая двухфакторную аутентификацию), управлению устройствами и подключению внешних сервисов.
  1. Winter CMS

В качестве системы управления содержимым в составе Комплекса используется Winter CMS — программная платформа, основанная на фреймворке Laravel. Winter CMS обеспечивает управление содержимым, настройку пользовательского интерфейса, а также взаимодействие с модулями и плагинами, реализующими функциональные компоненты Портала разработчика.

Winter CMS предназначена для:

* создания и редактирования веб-страниц с использованием визуального редактора;
* управления статическим и динамическим содержимым;
* организации структуры Портала разработчика;
* настройки ролей пользователей и прав доступа;
* централизованного управления медиафайлами;
* интеграции с расширениями и плагинами, реализующими дополнительный функционал.

Winter CMS реализует гибкую систему разграничения прав доступа. Управление осуществляется на основе ролей, которым присваиваются соответствующие разрешения.

Подробнее о рабочем интерфейсе Winter CMS см. в разделе 7. Управление контентом через Winter CMS.

* 1. Сетевое взаимодействие и шифрование трафика

Сетевое взаимодействие в Комплексе организовано по модели клиент-сервер, при которой взаимодействие между пользовательскими устройствами и серверной частью Комплекса осуществляется через защищённые каналы связи.

Для обмена данными используется протокол HTTPS, обеспечивающий защищённую передачу информации между клиентской и серверной частями.

Шифрование трафика реализуется с использованием протокола TLS (Transport Layer Security), что предотвращает возможность перехвата, модификации или подмены передаваемых данных.

Управление шифрованием и маршрутизацией запросов осуществляется с использованием обратного прокси-сервера Traefik, который автоматически обрабатывает сертификаты, управляет SSL-терминацией и обеспечивает маршрутизацию запросов между компонентами.

* 1. Обеспечение отказоустойчивости

Комплекс реализует базовую отказоустойчивость за счёт использования встроенных механизмов мониторинга состояния контейнеров в среде Docker.

Каждый контейнер, задействованный в работе Комплекса, конфигурируется с использованием директивы HEALTHCHECK, которая позволяет Комплексу проверять текущее состояние сервиса.

При выявлении нарушения работоспособности (отсутствие отклика, ошибки приложения и т. д.) контейнер считается неработоспособным, и Комплекс автоматически инициирует его перезапуск.

Данный механизм позволяет минимизировать простой сервисов и обеспечить базовую автоматическую реакцию на внутренние сбои без участия администратора.

* 1. Масштабируемость

Архитектура Портала разработчика предусматривает вертикальное масштабирование, что позволяет адаптировать Комплекс к увеличению нагрузки без изменения количества узлов.

Масштабирование реализуется за счёт увеличения вычислительных ресурсов выделенных серверов:

* оперативной памяти (RAM);
* вычислительной мощности процессора (CPU).

Вертикальный подход к масштабированию позволяет повысить производительность сервисов без необходимости внедрения кластеризации или распределённых вычислений, упрощая сопровождение Комплекса и снижая требования к инфраструктуре.

1. Технические и программные требования

Настоящий раздел содержит сведения о минимальных требованиях к программному и аппаратному обеспечению, необходимых для установки и корректного функционирования серверной и клиентской части Комплекса.

* 1. Требования к программному обеспечению

Для обеспечения безопасности и стабильной работы Комплекса необходимо соблюдение совместимости версий операционной системы, серверного программного обеспечения и сопутствующих компонентов.

Перечень программных компонентов, а также их минимально допустимые версии представлены в таблице 1.

Таблица 1 – Требования к программному обеспечению для работы Комплекса

|  |  |
| --- | --- |
| Наименование | Требования |
| Операционная система | На базе ОС Linux |
| Ядро Linux | Версия 3.10 и выше |
| Дополнительное ПО | Docker версии 27 или выше  iptables версии 1.4 или выше |
| Веб-сервер | Traefik версии 3.0 или выше Openresty/Nginx версии 1.27 |
| Язык программирования | PHP версии 8.3. |
| Фреймворки | Laravel версии 9.52.16 |
| СУБД | PostgreSQL версии 13  Redis версии 7.4  Typesense версии 27 |
| CMS | WinterCMS (совместимость с версией Laravel) |
| Языки и форматы обмена | SQL (в соответствии со стандартом ISO/IEC 9075:2011)  JSON (в соответствии с RFC 8259) |

* 1. Требования к аппаратному обеспечению

Серверное оборудование должно соответствовать следующим минимальным параметрам, указанным в таблице 2.

Таблица 2– Требования к серверному оборудованию

|  |  |
| --- | --- |
| Наименование | Требования |
| Тип сервера | Физический сервер или виртуальная машина |
| Устройство хранения | На базе SSD- или NVMe-дисков |
| Операционная система | На базе ОС Linux |
| Ядро Linux | Версия 3.10 и выше |
| Дополнительное ПО | Docker версии 27 или новее  iptables версии 1.4 или выше |

Рекомендуемые вычислительные ресурсы зависят от планируемого количества одновременно обслуживаемых пользователей на Портале разработчика. Соотношение количества пользователей и рекомендуемых вычислительных параметров приведено в таблице 3.

Таблица 3 – Зависимость пользователей и вычислительных мощностей

|  |  |  |  |
| --- | --- | --- | --- |
| Количество пользователей | CPU (vCPU) | RAM (ГБ) | DISK (ГБ) |
| До 100 | 4 | 4 | 25 |
| До 500 | 8 | 16 | 25 |
| До 1000 | 16 | 32 | 25 |

Примечание– В столбце "DISK (ГБ)" указаны минимальные значения параметра. Реальные значения зависят от объема созданного на платформе содержимого.

* 1. Требования к клиентской части

Для доступа к Комплексу с пользовательских устройств требуется современный веб-браузер с поддержкой стандартов HTML5, CSS3 и JavaScript, а также стабильное подключение к сети Интернет.

Минимальные версии браузеров, операционных систем и процессоров, в которых обеспечивается работа клиентской части Портала разработчика, представлены в таблице 4.

Таблица 4 – Поддерживаемые операционные системы и веб-браузеры для клиентской части Комплекса

| Операционная система | Версия ОС | Процессор |
| --- | --- | --- |
| Google Chrome | | |
| Windows | Windows 10  Windows Server 2016 | Intel Pentium 4 и поздние модели с поддержкой SSE3 |
| MAC | macOS Big Sur 11 | Intel или Apple Silicon (M1 и выше) |
| Linux | 64-битные версии Ubuntu 18.04  Debian 10  openSUSE 15.5  Fedora Linux 39 | Intel Pentium 4 и поздние модели с поддержкой SSE3 |
| IOS | iOS 14 | Процессор, поддерживающий iOS 14 и версии выше |
| Android | Android 8.0 | Процессор, поддерживающий Android 8.0 и версии выше |
| Yandex Browser | | |
| Windows | Windows 7 | Intel Pentium 4 |
| MAC | macOS 10.15 (Catalina) | Intel или Apple M1 (на архитектуре ARM) |
| Linux | 64-bit Ubuntu 18.04  Debian 10  openSUSE 15.2  Fedora Linux 32 | Intel Pentium 4 |
| IOS | iOS 12 | Процессор, поддерживающий iOS 12 и версии выше |
| Android | Android 5.0 или более поздняя версия | Процессор, поддерживающий Android 5.0 и версии выше |
| Mozilla Firefox | | |
| Windows | Windows 8 | Intel Pentium 4 и поздние модели с поддержкой SSE3 |
| MAC | macOS 10.12 (Sierra) | Intel или Apple Silicon (M1 и выше) |
| Linux | Ubuntu 18.04  Debian 10  Fedora 32  openSUSE 15.2 | Intel Pentium 4 и поздние модели с поддержкой SSE3 |
| IOS | iOS 12.2 | Процессор, поддерживающий iOS 12.2 и версии выше |
| Android | Android 5.0 | Процессор, поддерживающий Android 5.0 и версии выше |
| Safari | | |
| MAC | macOS 10.15 (Catalina) | Intel или Apple Silicon (M1 и выше) |
| IOS | iOS 12.0 | Процессор, поддерживающий iOS 12.0 и версии выше |

1. Установка и запуск комплекса

Установка Комплекса осуществляется на сервер, соответствующий требованиям по аппаратному обеспечению и программной совместимости, указанным в разделах 3.1 и 3.2 соответственно.

Развёртывание компонентов комплекса производится с использованием технологии контейнеризации (Docker). Все основные серверные компоненты функционируют внутри контейнеров, сборка и запуск которых выполняются с применением системы Docker Compose.

Для обеспечения корректной работы Комплекса необходимо предварительно установить и настроить программное окружение, а также выполнить развёртывание вспомогательных сервисов, обеспечивающих маршрутизацию и хранение данных.

В стандартной конфигурации требуемые программные компоненты разворачиваются автоматически внутри контейнеров, сборка которых производится из подготовленных сценариев из GitLab CI/CD.

Рекомендуется производить установку в пределах арендованной облачной инфраструктуры, организованной по модели IaaS. При этом обеспечивается требуемая степень изоляции, управляемости и масштабируемости среды исполнения Комплекса.

* 1. Развёртывание программного окружения
     1. Установка среды контейнеризации Docker

Перед запуском контейнеров необходимо установить и настроить систему контейнеризации Docker. Для этого нужно выполнить следующие шаги:

1. Установить пакеты docker и docker-compose из репозиториев ОС:

sudo dnf install docker docker-compose

Убедиться, что установка завершена без ошибок.

1. Включить автоматический запуск службы Docker при загрузке ОС:

sudo systemctl enable --now docker.service

1. Проверить работоспособность среды запуска контейнеров командой:

sudo docker run hello-world

Ожидаемый результат – информационное сообщение об успешной работе Docker.

* + 1. Развёртывание маршрутизатора входящих запросов Traefik

Для обработки входящих HTTP(S)-запросов, терминации TLS-соединений и маршрутизации трафика между контейнерами используется сервис Traefik.

* + - 1. Подготовка конфигурационных файлов

Для установки сервиса Traefik необходимо подготовить файл docker-compose.yaml со следующим содержанием:

version: '3'

services:

reverse-proxy:

image: traefik:v3.3.2

restart: always

command:

- --api.insecure=true

- --providers.docker

- --entrypoints.websecure.address=:443

- --entrypoints.web.address=:80

- --entryPoints.web.http.redirections.entryPoint.to=websecure

- --entryPoints.web.http.redirections.entryPoint.scheme=https

- --certificatesresolvers.le.acme.email=<LE info email>

- --certificatesresolvers.le.acme.storage=/acme.json

- --certificatesresolvers.le.acme.tlschallenge=true

- --api

- --providers.file.filename=/opt/traefik/traefik.yml

- --providers.docker.exposedByDefault=false

- --providers.docker.network=traefik\_default

ports:

- "80:80"

- "443:443"

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- /var/acme.json:/acme.json

- /opt/devstg.rosa.ru/traefik:/opt/traefik

labels:

- "traefik.enable=true"

- "traefik.http.routers.traefik.rule=Host(`<TRAEFIK\ DASHBOARD URL>`)"

- "traefik.http.routers.traefik.service=api@internal"

- "traefik.http.routers.traefik.middlewares=admin"

- "traefik.http.routers.traefik.tls.certresolver=le"

- "traefik.http.routers.traefik.entrypoints=websecure"

- "traefik.http.middlewares.admin.basicauth.users=<ADMIN USER>:<HASHED PASS>"

- "traefik.http.middlewares.compress.compress=true"

где необходимо заменить следующие переменные:

* <LE info email> — адрес электронной почты для получения уведомлений от файла хранения данных Let's Encrypt;
* <TRAEFIK DASHBOARD URL> — адрес веб-интерфейса Traefik;
* <ADMIN USER> — логин администратора;
* <HASHED PASS> — хеш пароля (формирование описано в официальной документации: <https://doc.traefik.io/traefik/middlewares/http/basicauth/>).
  + - 1. Создание вспомогательных файлов

Далее необходимо подготовить файлы на сервере для работы сервиса Traefik:

1. Создать служебный файл для хранения данных Let's Encrypt:

sudo touch /var/acme.json

sudo chmod 600 /var/acme.json

1. Создать структуру директорий /opt/devstg.rosa.ru/traefik/ssl/ и разместить в ней TLS-сертификат и закрытый ключ. Структура новой директории представлена ниже (рисунок 1 ).

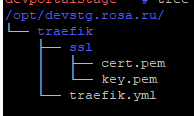


Рисунок 1 – Пример структуры директории /opt/devstg.rosa.ru/traefik/ssl/

1. Создать конфигурационный файл /opt/devstg.rosa.ru/traefik/traefik.yml:

tls:

certificates:

- certFile: /opt/traefik/ssl/cert.pem

keyFile: /opt/traefik/ssl/key.pem

stores:

- default

Подробности о работе с TLS-сертификатами приведены в официальной документации разработчиков сервиса: <https://doc.traefik.io/traefik/https/tls/>

* + - 1. Запуск сервиса

Затем необходимо запустить сервис Traefik из директории с файлом docker-compose.yaml:

sudo docker compose up -d --force-recreate

**Следует убедиться**, что запуск завершён без ошибок. Проверить статус контейнера можно командой:

sudo docker ps

После запуска сервиса Traefik нужно зайти в его дашборд по адресу <TRAEFIK DASHBOARD URL> и убедиться в доступности веб-интерфейса маршрутизатора.

* + 1. Установка и подготовка PostgreSQL

Для установки СУБД PostgreSQL и ее первоначальной настройки для работы с Комплексом требуется выполнить следующие шаги:

1. Установить PostgreSQL с помощью команды:

sudo apt install postgresql

1. Получить последнюю версию СУБД с помощью следующих команд:

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb\_release -cs`-pgdg\ main" >> /etc/apt/sources.list.d/pgdg.list'

wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc\ -O - | sudo apt-key add -

sudo apt-get install postgresql

1. Создать базу данных для Комплекса:

createdb wbii\_stat

1. Подключиться к СУБД:

sudo su

psql

1. Создать отдельного пользователя и его пароль:

CREATE USER wbii WITH PASSWORD 'указать\_сложный\_пароль';

GRANT ALL PRIVILEGES ON DATABASE wbii\_stat TO wbii;

где в первой строке необходимо задать пароль для пользователя.

Пароль должен состоять минимум из 12 символов, включать строчные и прописные латинские буквы, цифры, знаки препинания и специальные символы. Пробелы использовать запрещено.

* + 1. Подготовка постоянного хранилища

Для обеспечения долговременного хранения данных необходимо создать тома Docker (Docker volume), в которых будут размещаться:

* журналы работы (logs);
* файлы приложения (app);
* темы оформления интерфейса (themes);
* исходный код и конфигурации (code).

Том Docker volume представляет собой область на диске хост-системы, которая может быть подключена к одному или нескольким контейнерам. Данные, размещённые в Docker volume, не теряются при пересоздании контейнера, что делает его основным инструментом для хранения постоянной информации.

Создание томов осуществляется следующими командами:

sudo docker volume create storage\_logs

sudo docker volume create storage\_app

sudo docker volume create themes

sudo docker volume create code

Далее следует убедиться в успешном создании томов Docker с помощью команды:

docker volume ls

Данная команда выведет список всех созданных Docker-томов. В результате в списке должны появиться тома с именами:

* storage\_logs;
* storage\_app;
* themes;
* code.
  1. Установка Комплекса

Запуск Комплекса осуществляется с использованием системы CI/CD, настроенной в инфраструктуре GitLab, размещённой по адресу: https://git.rosa.ru/

Установка Портала разработчика выполняется следующими командами:

git checkout <название\_ветки>

git pull

php artisan winter:up

где необходимо заменить параметр <название\_ветки>:

* ветки с именами rc\_ХХ автоматически ассоциируются с тестовым контуром;
* ветка release предназначена для развертывания в промышленной среде.
  1. Запуск Комплекса
     1. Загрузка через командную строку

В штатном режиме работа Комплекса начинается автоматически при загрузке операционной системы. В случае необходимости запуска вручную требуется выполнить команды:

systemctl start postgresql

systemctl start apache2

Для запуска отдельных компонентов Комплекса вручную следует использовать следующие команды:

report/moralprep/index.php

report/statusrelative/index.php

report/whitespots/index.php

report/neuroschedule/index.php

* + 1. Вход в комплекс

Для начала работы с Комплексом администратору или пользователю необходимо:

* открыть веб-браузер и перейти по адресу: <https://developer.rosa.ru/>;
* перейти в раздел "Установить" для выбора подходящей ОС;
* запустить процедуру установки программного окружения и компонентов Комплекса;
* пройти процедуру аутентификации.

После прохождения процедуры аутентификации будет получен полный доступ к интерфейсу и функциям Комплекса.

1. Тонкая настройка операционной системы

Настоящий раздел содержит рекомендации по тонкой настройке ОС для обеспечения стабильной и безопасной работы Комплекса в производственной среде. Описанные меры включают оптимизацию параметров ядра, конфигурацию сетевой подсистемы, настройку изоляции и безопасности контейнеров, шифрование передаваемых данных, управление доступом через брандмауэр, а также организацию логирования.

Соблюдение приведённых ниже рекомендаций позволяет повысить отказоустойчивость Комплекса, минимизировать сетевые задержки, обеспечить надёжную защиту данных и упростить диагностику при эксплуатации.

* 1. Рекомендованные настройки производительности
     1. Инструменты настройки ядра

Для управления параметрами ядра Linux в реальном времени и при запуске ОС применяются различные инструменты. Их использование позволяет упростить администрирование и повысить предсказуемость поведения компонентов Комплекса.

В пунктах 5.1.1.1-5.1.1.3 рассмотрены доступные утилиты и инструменты для настройки параметров ядра с примерами их эффективного использования. Использование этих инструментов способствует стабильной и эффективной работе компонентов Комплекса в продуктивной среде.

* + - 1. sysctl

sysctl – основная утилита для чтения и изменения параметров ядра. Она взаимодействует с каталогом /proc/sys/, где доступны все системные параметры.

В качестве примера настройки параметров утилитой, чтобы изменить параметр net.core.somaxconn, который отвечает за максимальное число соединений в очереди для принятия сокетом, можно использовать следующую команду:

sysctl -w net.core.somaxconn=1024

Для сохранения настроек нужно добавить строку в файл /etc/sysctl.conf (или создать отдельный файл в /etc/sysctl.d/) со следующим содержанием:

net.core.somaxconn = 1024

* + - 1. Tuned

Tuned – утилита для применения готовых и пользовательских профилей оптимизации и динамической адаптивной настройки ОС. Она включает профили, ориентированные на разные типы нагрузки (например, throughput-performance, latency-performance).

Установка утилиты производится командами:

yum install tuned # Для ОС на основе RHEL

apt-get install tuned # Для ОС на основе Debian

Для активации профиля нужно ввести:

tuned-adm profile throughput-performance

Процесс создания пользовательского профиля для Kubernetes состоит из выполнения следующих команд:

1. создать каталог для профиля:

mkdir /etc/tuned/kubernetes-optimize

1. создать файл tuned.conf со следующими настройками:

[main]

include=throughput-performance

[sysctl]

net.ipv4.ip\_local\_port\_range = 10240 65535

net.core.somaxconn = 1024

net.ipv4.tcp\_max\_syn\_backlog = 2048

1. активировать профиль:

tuned-adm profile kubernetes-optimize

ktune

Утилита ktune из пакета tuned применяет пользовательские параметры настройки ОС при запуске на основе выбранного профиля. Хотя ktune реже используется в современных дистрибутивах, понимание его роли помогает оценить эволюцию автоматических решений для настройки ОС.

* + - 1. Дополнительные утилиты

Утилиты numactl и numad используются для ОС с неравномерным доступом к памяти (NUMA). Эти инструменты управляют политиками распределения памяти и привязкой к процессорам, что особенно важно для приложений, интенсивно использующих память.

Утилита cpufrequtils позволяет управлять политиками масштабирования частоты процессора, что помогает оптимизировать энергопотребление и производительность в зависимости от характеристик нагрузки.

Демон irqbalance распределяет аппаратные прерывания между процессорами в многопроцессорной операционной системе. Это повышает производительность для приложений с высокой интенсивностью ввода-вывода, избегая перегрузки одного процессора.

* + 1. Сетевые параметры

Для повышения производительности сетевой подсистемы ОС рекомендуется изменить ряд параметров ядра Linux. Настройка данных параметров позволяет снизить задержки при установлении соединений, расширить допустимое количество одновременных подключений и предотвратить исчерпание доступных портов при загруженном сетевом взаимодействии.

Для настройки производительности сетевых параметров в кластерах Kubernetes с высокой нагрузкой рекомендуется внести изменения в следующие параметры из таблицы 5.

Таблица 5 – Настройка сетевых параметров в кластерах Kubernetes

| Параметр | Команды |
| --- | --- |
| net.core.somaxconn  Задаёт максимальное количество соединений, которые могут находиться в очереди на прием серверным приложением. При высоких нагрузках значение по умолчанию может ограничивать число обрабатываемых подключений | Проверка текущего значения:  sysctl net.core.somaxconn  Рекомендуемое значение:  sysctl -w net.core.somaxconn=1024  Для сохранения значения после перезагрузки необходимо добавить в файл /etc/sysctl.conf строку:  net.core.somaxconn = 1024 |
| net.ipv4.tcp\_max\_syn\_backlog  Определяет максимальное количество входящих TCP-соединений, ожидающих подтверждения. Увеличение значения улучшает поведение сервера при большом количестве одновременных попыток подключения | Проверка текущего значения:  sysctl net.ipv4.tcp\_max\_syn\_backlog  Рекомендуемое значение:  sysctl -w\ net.ipv4.tcp\_max\_syn\_backlog=2048  Для сохранения после перезагрузки:  net.ipv4.tcp\_max\_syn\_backlog = 2048 |
| net.ipv4.ip\_local\_port\_range  Определяет диапазон локальных портов, используемых для установления исходящих подключений. Расширение диапазона позволяет избежать исчерпания доступных портов при интенсивной сетевой активности | Проверка текущего диапазона:  sysctl net.ipv4.ip\_local\_port\_range  Рекомендуемый диапазон:  sysctl -w\ net.ipv4.ip\_local\_port\_range="10240\ 65535"  Для постоянного применения:  net.ipv4.ip\_local\_port\_range = 10240 65535 |

* + 1. Мониторинг и корректировка параметров

После применения настроек рекомендуется контролировать их влияние на производительность ОС. Для анализа состояния сетевых соединений и очередей можно использовать следующие утилиты:

* netstat — отображение активных подключений и статистики по протоколам;
* ss — более современный аналог netstat с поддержкой фильтрации;
* ipvsadm — просмотр состояния балансировщиков при использовании IP Virtual Server.

Рекомендуется производить изменения параметров поэтапно, контролируя метрики нагрузки и стабильности работы. Оптимальные значения могут варьироваться в зависимости от характеристик оборудования и типа размещения Комплекса.

* + 1. Управление памятью

Эффективное управление оперативной памятью критически важно для обеспечения стабильной работы узлов, особенно в условиях высокой нагрузки и большого количества запущенных контейнеров. ОС Linux предоставляет ряд параметров, позволяющих гибко управлять распределением памяти и поведением ОС при её нехватке.

Для настройки параметров управления памятью рекомендуется использовать значения, приведённые в таблице 6**Ошибка! Источник ссылки не найден.**.

Таблица 6 – Рекомендуемые параметры управления памятью

| Параметр | Команды |
| --- | --- |
| vm.swappiness  Управляет склонностью ядра использовать swap. Высокое значение указывает на склонность использовать swap даже при наличии свободной оперативной памяти, тогда как низкое значение заставляет ядро избегать свопирования. Для узлов Kubernetes часто рекомендуется более низкое значение, чтобы приложения оставались в оперативной памяти для более быстрого доступа | Проверка текущего значения:  sysctl vm.swappiness  Чтобы уменьшить использование swap, можно установить более низкое значение, например 10, что поможет сохранить больше данных в физической памяти:  sysctl -w vm.swappiness=10  Чтобы изменения сохранялись после перезагрузки, следует добавить следующую строку в файл /etc/sysctl.conf:  vm.swappiness = 10 |
| vm.overcommit\_memory  Определяет политику распределения памяти при превышении её доступного объёма.  Существуют три настройки параметра:   * 0 — ядро использует эвристическую оценку; * 1 — разрешает переполнение памяти без ограничений; * 2 — запрещает запросы, если памяти и swap недостаточно.   Для узлов Kubernetes значение 1 может быть полезным, если рабочие нагрузки требуют много памяти, но фактически используют только ее часть | Проверка текущего значения:  sysctl vm.overcommit\_memory  Для разрешения переполнения памяти можно установить значение 1:  sysctl -w vm.overcommit\_memory=1  Чтобы изменения сохранялись после перезагрузки, следует добавить следующую строку в файл /etc/sysctl.conf:  vm.overcommit\_memory = 1 |
| vm.dirty\_ratio  Определяет максимальный процент оперативной памяти, который может быть заполнен "грязными" страницами (страницами, которые изменены и еще не записаны на диск) перед тем, как процессы будут вынуждены самостоятельно записывать эти данные | Проверка текущего значения:  sysctl vm.dirty\_ratio  Для балансировки между использованием памяти и дискового ввода-вывода можно установить значение 15:  sysctl -w vm.dirty\_ratio=15  Чтобы изменения сохранялись после перезагрузки, следует добавить следующую строку в файл /etc/sysctl.conf:  vm.dirty\_ratio = 15 |
| vm.dirty\_background\_ratio  Задает процент памяти, при котором ядро начинает асинхронно записывать "грязные" страницы на диск, чтобы избежать чрезмерного накопления таких страниц | Проверка текущего значения:  sysctl vm.dirty\_background\_ratio  Для снижения задержек записи данных на диск можно установить значение 5:  sysctl -w vm.dirty\_background\_ratio=5  Чтобы изменения сохранялись после перезагрузки, следует добавить следующую строку в файл /etc/sysctl.conf:  vm.dirty\_background\_ratio = 5 |

Все настройки необходимо применять с учётом объема доступной памяти и характера нагрузок на узлы, проводя предварительное тестирование в пилотной среде.

* + 1. Оптимизация файловой системы и дискового ввода-вывода

Корректная настройка параметров работы с файловой системой и вводом-выводом позволяет обеспечить стабильную работу компонентов Комплекса, интенсивно взаимодействующих с хранилищем данных. Это особенно актуально для компонентов, работающих с журналируемыми данными, логами и временными файлами.

Рекомендуется выполнить настройку параметров, указанных в таблице 7.

Таблица 7 – Параметры файловой системы и ввода-вывода

| Параметр | Команды |
| --- | --- |
| fs.file-max  Устанавливает максимальное количество файловых дескрипторов, которые может выделить ядро. Это особенно важно для узлов Kubernetes, на которых запускается множество контейнеров или приложений, открывающих много файлов одновременно | Проверка текущего значения:  sysctl fs.file-max  Увеличить лимит, например, до 500000:  sysctl -w fs.file-max=500000  Чтобы изменение сохранилось после перезагрузки, следует добавить строку в файл /etc/sysctl.conf:  fs.file-max = 500000 |
| fs.inotify.max\_user\_watches  Определяет максимальное количество файлов, которые могут отслеживаться на изменения с помощью inotify. Это особенно важно для приложений, реагирующих на изменения в режиме реального времени, таких как инструменты live-reload или сервисы синхронизации файлов | Проверка текущего значения:  sysctl fs.inotify.max\_user\_watches  Увеличить лимит, например, до 524288:  sysctl -w\ fs.inotify.max\_user\_watches=524288  Чтобы изменение сохранилось после перезагрузки, следует добавить строку в файл /etc/sysctl.conf:  fs.inotify.max\_user\_watches = 524288 |
| Настройка планировщика ввода-вывода (I/O Scheduler)  Выбор подходящего планировщика I/O влияет на производительность операций ввода-вывода, включая пропускную способность, задержки и число операций в секунду (IOPS). Для большинства случаев рекомендуется использовать deadline или mq-deadline | Проверка текущего планировщика для устройства (например, /dev/sda):  cat /sys/block/sda/queue/scheduler  Изменить планировщик на deadline для снижения задержек:  echo deadline >\ /sys/block/sda/queue/scheduler  Для систем с SSD или высокопроизводительными накопителями планировщики noop или mq-deadline могут предложить лучшую производительность за счет простоты и меньшей нагрузки |

Рекомендуется использовать соответствующие udev-правила или скрипты инициализации, чтобы выбранный планировщик применялся автоматически при запуске ОС.

* + 1. Мониторинг и корректировка параметров ввода-вывода

После применения настроек файловой системы и параметров дискового ввода-вывода (см. раздел 5.1.5) необходимо контролировать их влияние на производительность. Для оценки используются стандартные инструменты мониторинга, такие как:

* iostat — анализирует статистику по дисковому вводу-выводу;
* iotop — отслеживает текущую активность ввода-вывода процессов;
* sar —собирает и отображает статистику производительности различных подсистем ОС.

Кроме того, Kubernetes предоставляет встроенные метрики и журналы, позволяющие анализировать I/O-паттерны приложений. Это обеспечивает возможность адаптировать параметры под реальные рабочие нагрузки и добиться устойчивой производительности компонентов Комплекса.

* + 1. Планирование процессов

Параметры планировщика процессов ядра Linux оказывают прямое влияние на производительность и отклик приложений в среде Kubernetes, особенно в условиях высокой конкуренции за ресурсы CPU. Рекомендуется внести следующие настройки в параметры ядра, указанные в таблице 8.

Таблица 8 – Параметры настройки планирования процессов

| Параметр | Команды |
| --- | --- |
| kernel.sched\_migration\_cost\_ns  Определяет минимальное время (в наносекундах) выполнения процесса на текущем CPU до его возможной миграции на другой CPU. Это важно для среды Kubernetes, где поды могут часто перемещаться между CPU. Понижение значения может сделать планировщик более агрессивным в перемещении процессов, улучшая балансировку нагрузки, но с риском увеличения числа промахов кеша | Проверка текущего значения:  sysctl kernel.sched\_migration\_cost\_ns  Для повышения отзывчивости можно установить значение, например, 500000 (0,5 мс)  sysctl -w\ kernel.sched\_migration\_cost\_ns=500000  Чтобы изменение сохранилось после перезагрузки, следует добавить строку в файл /etc/sysctl.conf:  kernel.sched\_migration\_cost\_ns=500000 |
| kernel.sched\_autogroup\_enabled  Включает функцию автоматической группировки задач с похожими шаблонами выполнения для улучшения отзывчивости ОС. Эта настройка полезна для настольных систем, но в серверной среде, особенно при работе с Kubernetes, такое поведение может привести к неравномерному распределению ресурсов CPU между подами | Для проверки, включена ли автогруппировка:  sysctl kernel.sched\_autogroup\_enabled  Для равномерного распределения ресурсов CPU можно отключить функцию:  sysctl -w\ kernel.sched\_autogroup\_enabled=0  Чтобы изменение сохранилось после перезагрузки, следует добавить строку в файл /etc/sysctl.conf:  kernel.sched\_autogroup\_enabled=0 |

Применение вышеуказанных параметров требует предварительного тестирования в условиях, приближенных к промышленной среде, и регулярного анализа нагрузки на CPU.

* + 1. Привязка к процессору и управление CPU

Для повышения производительности и предсказуемости работы критически важных компонентов в составе Портала разработчика может применяться привязка контейнеров к определённым процессорным ядрам (CPU pinning), а также управление распределением ресурсов процессора с использованием средств Kubernetes.

Конфигурации развертывания Kubernetes описываются в pod – наименьшей единице развертывания в Kubernetes, задаются в формате YAML (реже — JSON). Параметры pod определяют, какие контейнеры запускаются и какие ресурсы они используют, какие политики применяются и как организовано взаимодействие с другими элементами ОС.

Kubernetes предоставляет следующие механизмы управления ресурсами CPU:

* привязка контейнеров к конкретным процессорам (CPU pinning);
* использование наборов процессоров (CPU sets);
* указание запросов и ограничений на использование CPU (параметры requests и limits в спецификации pod).

Такие настройки особенно актуальны для компонентов, чувствительных к задержкам или требующих высокой вычислительной мощности, поскольку позволяют:

* исключить миграцию процессов между ядрами;
* повысить эффективность кеширования;
* сократить накладные расходы на переключение контекста;
* обеспечить размещение модулей ближе к необходимым узлам NUMA.

Пример спецификации pod с указанием ресурсов и узловой привязки:

apiVersion: v1

kind: Pod

metadata:

name: cpu-affinity-example

spec:

containers:

- name: container1

image: nginx

resources:

requests:

cpu: "2"

limits:

cpu: "4"

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: kubernetes.io/e2e-az-name

operator: In

values:

- e2e-az1

- e2e-az2

Данный фрагмент демонстрирует задание минимального (requests) и максимального (limits) количества CPU для контейнера, а также настройку привязки к определённым зонам доступности. Это позволяет Kubernetes запланировать выполнение пода на узле, соответствующем заданным условиям, и тем самым обеспечить требуемый уровень производительности.

* + 1. Мониторинг использования CPU и корректировка

Для анализа нагрузки на CPU применяются как стандартные утилиты, так и специализированные средства мониторинга в рамках Kubernetes:

* htop, mpstat — отслеживают использование процессоров и процессов;
* Prometheus и Grafana — используются для сбора и визуализации метрик кластера.

Регулярный мониторинг позволяет выявить узкие места и провести корректировку конфигурации, учитывая характеристики архитектуры и тип рабочих нагрузок.

* 1. Рекомендованные настройки безопасности
     1. Изоляция процессов и данных

Для обеспечения безопасности и отказоустойчивости все компоненты Комплекса запускаются в отдельных контейнерах. Это обеспечивает изоляцию процессов и снижает риск того, что сбой или компрометация одного из компонентов повлияет на работу остальных.

Для повышения уровня безопасности рекомендуется выполнять следующие настройки по изоляции контейнеров:

* создавать изолированную внутреннюю сеть Docker (bridge), в которой взаимодействуют только контейнеры Комплекса;
* ограничивать сетевой доступ к каждому контейнеру, разрешая только необходимые порты:
* PostgreSQL: доступ разрешён только из контейнеров, выполняющих PHP-код;
* Redis: доступен только из контейнера с приложением;
* устанавливать лимиты на использование ресурсов (CPU и память) для каждого контейнера, чтобы предотвратить избыточное потребление и влияние одного контейнера на другие.
  + 1. Шифрование

Передача данных между пользователем и сервером осуществляется по защищённым TLS-соединениям. Рекомендуется использовать следующие настройки:

* все входящие соединения обрабатываются по протоколу TLS версии 1.2 или 1.3;
* TLS-сертификаты настраиваются и обновляются автоматически через Traefik;
* в незашифрованный порт (80) осуществляется только перенаправление на HTTPS (порт 443).
  + 1. Брандмауэр

Для защиты сетевой инфраструктуры следует настроить брандмауэр следующим образом:

* разрешить только следующие порты:
* 80/tcp — перенаправление HTTP-запросов на HTTPS;
* 443/tcp — основной порт для защищённого взаимодействия с системой;
* все остальные порты должны быть закрыты. Попытки соединения через другие порты должны блокироваться.
  1. Логирование

Система логирования настраивается для всех компонентов Комплекса с целью обеспечения наблюдаемости и возможности оперативного реагирования на возникающие ошибки.

На уровне приложения логирование конфигурируется через переменную окружения LOG\_CHANNEL. Поддерживаются следующие режимы:

* stack / single — непрерывная запись логов в один файл;
* daily — создание отдельного файла для каждого дня; файлы хранятся до 14 суток;
* syslog — отправка логов в системный журнал;
* null — отключение логирования.

Логи приложения сохраняются в каталоге storage/logs. Кроме того, все записи логов доступны для просмотра в административной панели CMS в разделе "Система → Журнал событий" (/backend/system/eventlogs).

1. Мониторинг и диагностика

Для обеспечения бесперебойной работы Портала разработчика требуется соблюдать ряд мер по мониторингу и диагностике. Администратору Комплекса следует своевременно выявлять отклонения в работе служб, предотвращать потерю данных и оперативно реагировать на сбои.

В данной главе приведены рекомендации по обеспечению устойчивого функционирования Комплекса, включая правила хранения постоянных данных, создание резервных копий и настройку системы мониторинга.

* 1. Расположение программного обеспечения и журналов событий

Поскольку работа Комплекса развёрнута в изолированных контейнерах Docker, для мониторинга состояния компонентов и выявления сбоев реализовано централизованное логирование через стандартный поток вывода (stdout) каждого контейнера. Это позволяет фиксировать события выполнения и обеспечивать оперативный доступ к журналам.

Для получения информации о состоянии запущенных контейнеров и анализа логов используются стандартные команды Docker. В таблице 9 приведён перечень типовых команд и описание их выводов.

Таблица 9 – Типовые команды управления контейнерами Docker

| Команда Docker | Назначение | Описание вывода |
| --- | --- | --- |
| docker ps | Выводит список всех запущенных контейнеров Docker | Выводы (по умолчанию):   * CONTAINER ID: Уникальный идентификатор контейнера (хеш); * IMAGE: Имя образа, из которого был создан контейнер; * COMMAND: Команда, которая выполняется внутри контейнера; * CREATED: Время, прошедшее с момента создания контейнера. * STATUS: Текущий статус контейнера (например, “Up 5 seconds”, “Exited (0) 2 hours ago”). * PORTS: Опубликованные порты (если есть)   NAMES: Имя контейнера (заданное пользователем или сгенерированное Docker) |
| docker ps -a или docker ps --all | Выводит список всех контейнеров, включая остановленные |
| docker ps -l или docker ps --latest | Показывает только последний созданный контейнер (запущен или нет) |
| docker ps -q | Выводит только идентификаторы контейнеров |
| docker ps -n [число] | Показывает информацию о последних N созданных контейнерах |
| docker ps -s | Выводит размеры контейнеров |
| docker inspect | Отображает подробную информацию о контейнере | Вывод будет в формате JSON и содержать подробные сведения о контейнере:   * ID: ID контейнера; * Name: имя контейнера; * Image: ID образа; * Config: конфигурация контейнера (команда, переменные окружения, порты и т.д.); * State: текущее состояние контейнера (статус, запущен или нет, код выхода и т.д.). Ключевое поле Status здесь показывает текущий статус; * NetworkSettings: настройки сети (IP-адреса, порты и т.д.); * Mounts: информация о примонтированных томах (volumes) и дополнительная информация |
| docker logs | Просматривает журналы контейнера | Формат логов внутри контейнера определяется самим приложением. Docker logs просто извлекает этот вывод |
| docker events | Просматривает события Docker, включая изменения состояния контейнеров. | Формат вывода по умолчанию:   * time: время события (Unix timestamp). * timeNano: время события в наносекундах. * status: тип события (например: start, stop, create, destroy). * id: ID объекта, к которому относится событие (например, ID контейнера или ID образа). * from: имя образа (если событие связано с образом). * Type: тип ресурса, к которому относится событие (например, container, image, network). * Action: действие, которое произошло (например: start, stop, create, destroy); * Actor: объект, который является источником события; * scope: область действия события (local или swarm) |
| docker ps --format | Позволяет настроить формат вывода. Выводит только выбранную информацию, например, ID и имена контейнеров | Формат вывода определяется шаблоном, указанным в кавычках после --format.  Основные доступные поля:   * .ID: ID контейнера. * .Image: имя образа. * .Command: команда, выполняющаяся внутри контейнера. * .CreatedAt: время создания контейнера (в формате RFC3339). * .RunningFor: время, в течение которого контейнер работает (например, 2 minutes ago). * .Ports: опубликованные порты. * .Names: имена контейнеров. * .Labels: метки (labels) контейнера. * .Mounts: информация о примонтированных томах. * .State: текущее состояние контейнера (например, running, exited). * .Status: статус контейнера (например, Up 5 seconds, Exited (0) 2 hours ago). * .Size: размер контейнера (с дисками и без дисков) |
| docker ps --filter | Фильтрует вывод по определенным критериям. | Формат вывода идентичен стандартному формату docker ps (или тому, который задан с помощью --format), но отображаются только контейнеры, удовлетворяющие условиям фильтра.  Колонки:   * CONTAINER ID; * IMAGE; * COMMAND; * CREATED; * STATUS; * PORTS, NAMES |

* 1. Хранение постоянных данных

Для обеспечения сохранности данных при перезапуске, обновлении или удалении контейнеров используется механизм томов Docker (Docker volumes). Эти тома создаются при первоначальной установке Комплекса (см. раздел 4.1.4). В рамках мониторинга и диагностики важно отслеживать их текущее состояние, объем занятого пространства, а также корректность подключения к контейнерам. Повреждение данных в этих томах может привести к полной или частичной недоступности функций Комплекса.

Для диагностики постоянных хранилищ необходимо понимать правильное назначение томов в Комплексе. Их перечень с описанием представлен в таблице 10.

Таблица 10 – Назначение томов Docker

| Название тома | Назначение |
| --- | --- |
| code | Хранение исходного кода программного обеспечения Комплекса |
| content | Сохранение пользовательского контента (загруженные файлы, изображения, документы и др.) |
| rosa\_postgres | Хранилище данных СУБД PostgreSQL. Обеспечивает сохранность при перезапуске контейнера |
| rosa\_redis | Хранение данных Redis при включенной персистентности |
| rosa\_typesense | Сохранение индексов и данных поискового движка Typesense |
| themes | Хранение тем оформления интерфейса и веб-страниц, созданных в CMS |

Примечание – В отличие от раздела 4.1.4. Подготовка постоянного хранилища в данной таблице представлены все фактически используемые тома, включая content, rosa\_postgres, rosa\_redis и rosa\_typesense.

Рекомендации по мониторингу томов Docker:

1. Проверка наличия и статуса томов. Для получения списка всех созданных томов использовать команду:

docker volume ls

1. Мониторинг объема хранилища. Для оценки занятого пространства можно использовать команды:

docker system df -v

или:

docker inspect <имя\_тома>

В выводах команды следует обратить внимание на поле Size.

1. Настройка регулярного резервного копирования критичных томов, особенно:

* rosa\_postgres — база данных;
* content — пользовательские файлы;
* code; themes — конфигурации и визуальное оформление.
  1. Рекомендации по мониторингу состояния Комплекса

Для обеспечения стабильной работы Комплекса рекомендуется настроить постоянный мониторинг его компонентов с использованием специализированного программного обеспечения. В качестве базового решения может быть использована система мониторинга Zabbix (<https://www.zabbix.com.ru>) — свободно распространяемое ПО с открытым исходным кодом, лицензированное под GNU AGPLv3.

* + 1. Архитектура системы мониторинга

Для корректной работы Zabbix необходимо развертывание двух основных компонентов:

* сервер мониторинга — устанавливается на отдельный сервер под управлением \*nix-системы. Отвечает за сбор, обработку и хранение данных мониторинга.
* Zabbix-агент — устанавливается на сервер, на котором размещён Комплекс. Отправляет серверу мониторинга данные о состоянии ОС и служб.

Рекомендуется настроить оповещения по электронной почте или в мессенджере (например, Telegram) при возникновении сбоев или достижении критических значений.

* + 1. Мониторинг серверной инфраструктуры

Мониторинг должен охватывать как общесистемные показатели, так и состояния ключевых сервисов, входящих в состав Комплекса. В таблице 11 приведён перечень служб и их состояний для отслеживания.

Таблица 11 – Список служб и их метрик

| Отслеживаемые службы | Состояния метрик |
| --- | --- |
| Операционная система | * Отсутствие соединения с агентом Zabbix; * Неактивность обязательных служб; * Нехватка свободного места на диске; * Недостаток свободной оперативной памяти; * Высокая нагрузка на дисковую подсистему; * Высокая нагрузка на оперативную память; * Перегрузка сетевого интерфейса; * Иные параметры, критически влияющие на работоспособность Комплекса |
| Служба Docker | * Отсутствие сведений о статусе Docker; * Служба Docker не запущена; * Изменение версии Docker; * Ошибки в статусе контейнеров; * Аварийное завершение контейнеров (exit code ≠ 0); * Перегрузка процессора службой Docker; * Повышенное потребление памяти службой Docker |
| Traefik | * Недоступность службы; * Неактивность процесса; * Изменение версии; * Повышенная загрузка CPU или RAM |
| Nginx | * Недоступность службы; * Неактивность процесса; * Изменение версии; * Перегрузка по CPU или RAM; * Рост количества ошибок (4XX, 5XX); * Превышение лимитов по соединениям и отклику |
| PHP-FPM | * Недоступность или остановка службы; * Изменение версии; * Перегрузка процессора или памяти; * Превышение лимитов: * активных и idle процессов; * соединений в секунду; * медленных запросов; * очередей подключений |
| Redis | * Недоступность службы; * Неактивность процесса; * Изменение версии; * Повышенная нагрузка на CPU или RAM; * Превышение лимитов по соединениям, процессам, очередям, медленным операциям |
| PostgreSQL | * Отсутствие доступа к службе; * Служба неактивна; * Изменение версии; * Повышенное использование CPU или RAM; * Превышение лимитов подключений, активных процессов, частоты запросов |
| Typesense | * Отсутствие информации о состоянии службы; * Служба не запущена.; * Изменение версии; * Перегрузка по CPU или памяти |

* + 1. Мониторинг доступности веб-интерфейса

Необходимо регулярно проверять доступность и корректность отображения ключевых страниц Портала разработчика:

* главная страница;
* панель управления администратора;
* ошибки 4XX и 5XX на любых страницах;

Рекомендуется использовать внешние проверки (например, через HTTP-мониторинг Zabbix или специализированные скрипты), чтобы контролировать отклик и корректность работы Комплекса как с точки зрения пользователя, так и администратора.

1. Управление контентом через Winter CMS
   1. Назначение и возможности Winter CMS

Winter CMS представляет собой гибкую систему управления контентом, предназначенную для администрирования и сопровождения веб-интерфейса Портала разработчика. CMS реализует современные подходы к работе с веб-контентом, обеспечивает поддержку шаблонов, расширений (плагинов), мультиязычности и интеграций с внешними сервисами.

Winter CMS имеет удобный интерфейс для настройки, обновления и публикации информационных материалов, размещаемых на веб-платформе.

Основные функциональные возможности Winter CMS:

* управление страницами сайта (создание, редактирование, удаление);
* настройка меню и маршрутов;
* установка и активация плагинов;
* редактирование шаблонов оформления;
* ведение логов и мониторинг состояния;
* управление мультиязычным контентом.
  1. Языковая поддержка

Winter CMS поддерживает два языка: русский и английский. По умолчанию при создании новых элементов контента и установке плагинов используется русский язык.

Переводы системных сообщений и элементов интерфейса осуществляются вручную с использованием специализированного плагина перевода. Администратор должен обеспечить добавление и актуализацию переводов при необходимости поддержки многоязычного интерфейса.

* 1. Плагины и расширения

Для расширения функциональности Winter CMS применяется система плагинов. Установка и настройка плагинов осуществляется через административный интерфейс.

Примеры доступных плагинов:

* управление мультиязычностью;
* формы обратной связи;
* интеграции с внешними API;
* расширенные редакторы контента.

Перед установкой плагина рекомендуется ознакомиться с его документацией и протестировать совместимость с текущей версией CMS и Портала разработчика.

* 1. Резервное копирование контента

Все данные, размещённые через Winter CMS, должны входить в состав регулярных резервных копий ОС. Рекомендуется настраивать автоматическое копирование:

* базы данных, содержащей структуру и содержимое страниц;
* директории с пользовательскими файлами и шаблонами;
* конфигурационных файлов CMS и установленных плагинов.

Настройка резервного копирования выполняется в рамках общей процедуры резервного копирования Комплекса (см. раздел 6.2).

* 1. Рабочий интерфейс
     1. Дашборд администратора

Дашборд является основной точкой входа в административную панель системы управления контентом Winter CMS. На данной странице отображается сводная информация о состоянии сайта. Администратор может просматривать текущее состояние системы CMS, получать доступ к логам событий и запросов, а также отслеживать ключевые метрики и сообщения, касающиеся функционирования сайта.

Дашборд обеспечивает навигацию по основным разделам CMS и служит интерфейсом начального взаимодействия при администрировании контента.

* + 1. Управление структурой сайта

Раздел CMS предоставляет средства для работы с основными элементами структуры и оформления веб-интерфейса. Шаблоны создаются с применением языка шаблонов Twig, поддерживается также использование PHP для реализации более сложной логики, например, подключения жизненного цикла страниц.

Компоненты служат механизмом подключения плагинов и модулей, используемых для расширения функциональности. Основная навигация по разделу осуществляется через левое навигационное меню, включающее следующие вкладки:

* Страницы — управление контентом и параметрами отдельных страниц, включая страницы ошибок (например, 404);
* Фрагменты — элементы HTML-шаблонов, используемые для отображения составных частей интерфейса. Подразделяются на категории:
* blog — элементы навигации;
* blogPost — шаблоны для блога;
* site — оформление (заголовки, подвал);
* snippets — универсальные контентные блоки;

– Шаблоны — базовые структуры страниц, редактируемые в HTML. Поддерживается настройка имени, описания и содержимого;

– Content — управление контентом на статических страницах;

– Assets — доступ к ресурсам: CSS, JavaScript, изображения, шрифты;

– Компоненты — добавление и конфигурация функциональных блоков на страницах.

* + 1. Медиа

Раздел "Медиа" представляет собой встроенную медиабиблиотеку, предназначенную для хранения и администрирования изображений, видеоматериалов, документов и других файлов. Поддерживаются функции поиска, сортировки и группировки файлов по папкам, что облегчает их использование и организацию.

* + 1. Страницы

Раздел "Страницы" используется для управления как статическим, так и динамическим контентом сайта. Предоставляется возможность создания, редактирования и удаления страниц, настройки SEO-параметров, метатегов, выбора языка, указания авторов и использования сниппетов — повторно используемых фрагментов кода.

Интерфейс включает визуальный редактор, позволяющий редактировать заголовки, адреса страниц и их свойства. Поддерживается предпросмотр, копирование страниц и работа с шаблонами. Вкладка "Дополнительно" содержит настройки групп свойств для дочерних страниц.

* + 1. Блог

Раздел "Блог" предназначен для создания и управления записями блога, в том числе с использованием категорий. Поддерживается вставка мультимедийного контента, включая видео с платформ Rutube и ВКонтакте. Предусмотрена настройка SEO-параметров для каждой записи.

* + 1. Пользователи

Winter CMS поддерживает управление фронтенд-пользователями, включая зарегистрированных и незарегистрированных. Для зарегистрированных пользователей предусмотрена последующая интеграция с системой РОСА ID.

Дополнительно реализована возможность создания групп пользователей с целью разграничения прав доступа и настройки ролевых политик.

* + 1. Авторы

Раздел "Авторы" позволяет создавать и редактировать страницы авторов, а также привязывать их к контенту блога или иным страницам. Для каждого автора указываются:

– имя;

– URL;

– тематическое изображение.

* + 1. Настройки

Раздел "Настройки" предоставляет административные параметры для конфигурации CMS, внешнего вида, SEO, интеграции с внешними сервисами и управления CMS. Структура раздела включает следующие подсекции:

1. Мои настройки: поисковая оптимизация — генерация и настройка метатегов.
2. CMS:

* фронтенд темы — выбор темы оформления, изменение цветов, иконок, добавление JavaScript-виджетов;
* robots.txt — настройка файла управления индексированием;
* режим обслуживания — отображение заглушки в период технических работ;
* карта сайта (sitemap.xml) — создание и настройка структуры для поисковой индексации;
* фрагменты — глобальные параметры сниппетов, используемых на сайте.

1. Система:

* обновления и плагины — установка, переустановка, обновление плагинов, сброс настроек. Включает плагины для robots.txt, sitemap.xml и SEO;
* администраторы — добавление администраторов и настройка их прав;
* персонализация панели управления — настройка логотипа, фавиконки, цветовой схемы и вида меню;
* настройки редактора кода — конфигурация параметров редактора: стили, классы, теги, панели инструментов.

1. Блог: включение и отключение функциональности блога.
2. Пользователи: настройка параметров зарегистрированных пользователей, включая будущую интеграцию с РОСА ID.
3. Translate:

* управление языками — добавление новых языков, установка плагина перевода. Русский язык — по умолчанию;
* перевод сообщений — редактирование системных сообщений.

1. Почта:

* шаблоны писем — редактирование и управление шаблонами уведомлений;
* настройки почты — конфигурация SMTP, почтовых ящиков и портов;
* брендирование писем — настройка внешнего вида шаблонов.

1. Логи:

* лог событий — журнал регистрации ошибок и действий;
* журнал доступа — история входов;
* настройки логирования — базовая конфигурация.
  + 1. Аккаунт

Раздел "Мой аккаунт" содержит настройки персонального профиля администратора. Здесь можно изменить отображаемое имя, аватар, язык интерфейса и другие параметры, не влияющие на глобальные настройки портала.

Перечень сокращений

|  |  |
| --- | --- |
| Сокращение | Определение |
| ОС | Операционная система |
| ПО | Программное обеспечение |
| СУБД | Система управления базами данных |
| CMS | Content Management System — система управления содержимым веб-сайта |
| CI/CD | Continuous Integration / Continuous Delivery — непрерывная интеграция и доставка изменений в программное обеспечение |
| CPU | Central Processing Unit — центральный процессор, основное устройство обработки данных в компьютере |
| HTTPS | Hypertext Transfer Protocol Secure — защищённый протокол передачи гипертекста через TLS/SSL |
| ID | Identifier — уникальный идентификатор объекта или пользователя |
| idP | Identity Provider — поставщик удостоверяющей информации, обеспечивающий аутентификацию пользователей |
| IOPS | Input/Output Operations Per Second — число операций ввода-вывода в секунду, метрика производительности дисковой системы |
| IP | Internet Protocol — протокол, обеспечивающий адресацию и маршрутизацию данных в сети |
| JWT | JSON Web Token — компактный и безопасный способ передачи данных между участниками |
| RAM | Random Access Memory — оперативная память, предназначенная для временного хранения данных во время работы |
| SEO | Search Engine Optimization — оптимизация веб-страниц для повышения их позиций в поисковой выдаче |
| SSL | Secure Sockets Layer — устаревший криптографический протокол для защиты интернет-соединений |
| TLS | Transport Layer Security — современный криптографический протокол защиты передаваемых данных |
| URI | Uniform Resource Identifier — унифицированный идентификатор ресурса в сети, определяющий его местоположение или имя |