Р·СЕРВИС

Руководство системного администратора Р-Сервис

Обзор решения Р-Сервис (RR Tech Service Management)

СОДЕРЖАНИЕ

1 ОБЗОР ПРОДУКТА

Р-Сервис (RR Tech Service Management) – это современная система управления услугами информационных технологий и корпоративными услугами (ITSM/ESM) с поддержкой концепции интеграции и управления услугами в условиях множества поставщиков (SIAM).

Система работает по модели «клиент-сервер». Для подключения к системе не требуется специальное программное обеспечение (ПО), ведь используются веб-браузеры.

В основе концепции продукта лежит понятие мультиарендности[1] – возможности изоляции данных для разных клиентов в рамках одной инсталляции. Система Р-Сервис развивает данный концепт, позволяя устанавливать доверие между разными пространствами[2], делиться сущностями (записями) и перенаправлять запросы в рамках доверия.

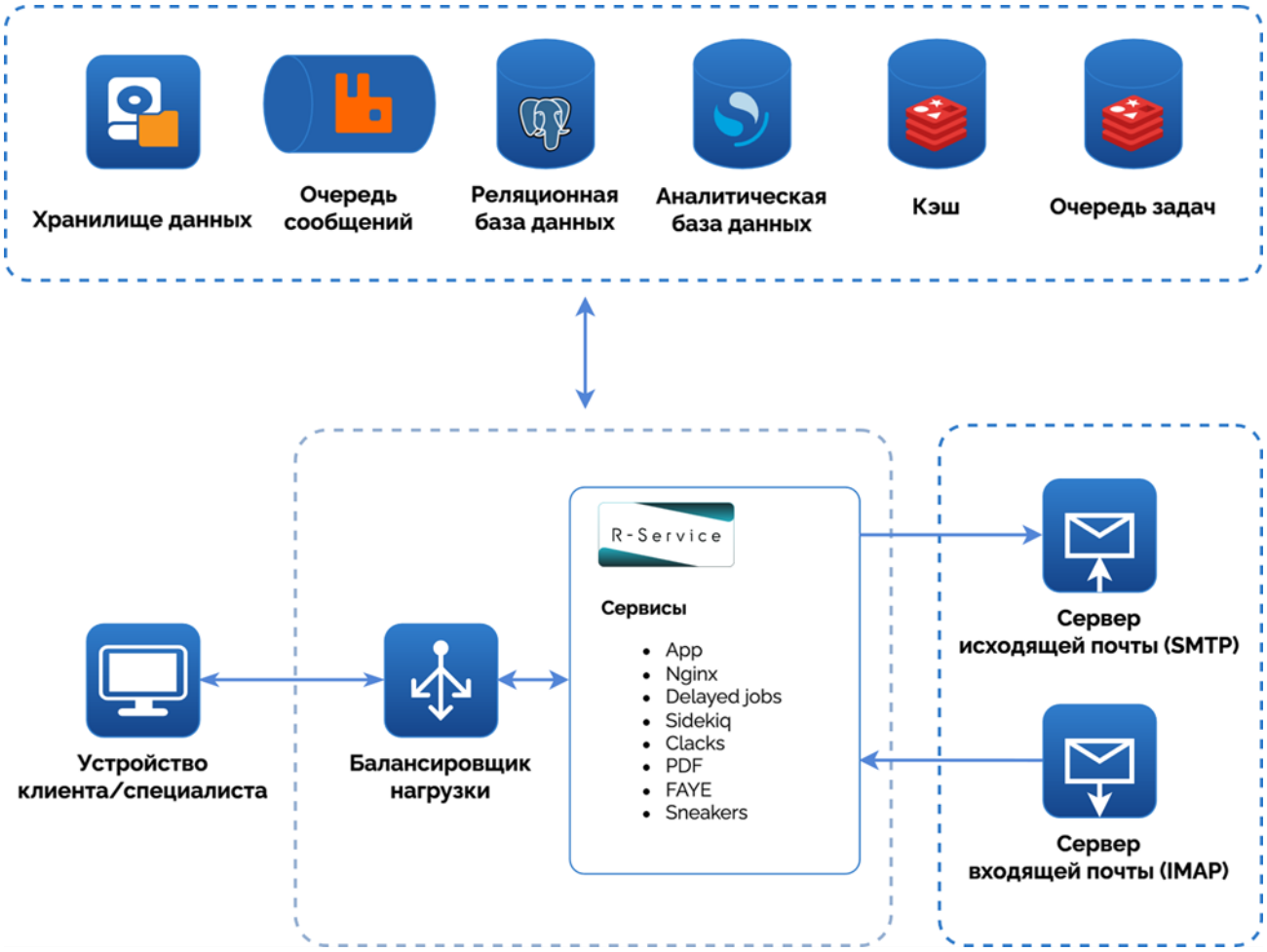
[1] Мультиарендность (то англ. «multitenancy») – это архитектура программного обеспечения, при которой один экземпляр приложения обслуживает множество независимых клиентов ("арендаторов").

[2] Пространство – ограниченная область данных, рабочая область, где работают пользователи. Можно провести аналогию с учетной записью – по своей сути пространство – это и есть учетная запись определенного клиента. См. справку о системе

2 ОБЗОР АРХИТЕКТУРЫ

Система Р-Сервис спроектирована для обеспечения отказоустойчивости, сохранности данных и возможности масштабирования даже под самые крупные инсталляции.

Система состоит из компонентов, каждый из которых работает в контейнерах.



Для начала рассмотрим общие ресурсы.

2.1 Балансировщик нагрузки

Единая входная точка клиентов в систему. Он балансирует и маршрутизирует нагрузку между копиями (репликами) серверов приложений (app) и серверов реального времени (realtime), а также выполняет завершение и расшифровку трафика на уровне защищенных слоев (SSL-терминацию).

2.2 Реляционная[3] база данных

Основным хранилищем данных системы является реляционная база данных PostgreSQL. Она выступает в роли единственного источника истины для всех записей и справочной информации, используемой системой.

[3] Реляционная база данных - способ организации информации в виде связанных таблиц (отношений), состоящих из строк и столбцов.

2.3 Аналитическая база данных

Для выполнения аналитических операций, сложной фильтрации данных и формирования отчетов система использует отдельный кластер OpenSearch. Данная аналитическая база предназначена для высокоскоростной обработки поисковых запросов и агрегаций, что существенно снижает нагрузку на основную реляционную базу и обеспечивает быстрое построение пользовательских выборок.

Актуальность данных в OpenSearch поддерживается с помощью роли «задание/работа» (job), которая реагирует на отложенные задачи, формируемые при каждом изменении данных в PostgreSQL. Роль «задание/работа» (job) выполняет выборку обновленных записей, преобразование данных в необходимый формат и последующую реиндексацию соответствующих документов в OpenSearch.

2.4 Сетевое хранилище

Система хранит бинарные данные (вложения, документы, изображения и т. д.) во внешнем сетевом хранилище. В качестве хранилища может использоваться любая технология, предоставляющая сетевую файловую систему (например, блок серверных сообщений (SMB), сетевая файловая система (NFS) или корпоративное объектное хранилище, смонтированное как файловый ресурс).

Приложение не зависит от конкретной реализации хранилища и ожидает только наличие доступной сетевой папки на серверах ролей «приложение» (app) и «задание/работа» (job).

2.5 Сервер исходящей почты

Для отправки исходящих сообщений система интегрируется с почтовым сервером посредством простого протокола передачи почты (SMTP).

2.6 Сервер входящей почты

Для приема входящих писем система использует протокол доступа к интернет-сообщениям (IMAP).

2.7 Сервер развертывания

Данный сервер используется только для развертывания. На нем выполняются необходимые скрипты[4] развертывания. Возможно использование существующего сервера, на котором есть доступ ко всем серверам системы.

Теперь рассмотрим выделенные компоненты системы. Все компоненты системы можно поделить на три основных роли: «приложение» (app), «реальное время» (realtime) и «задание/работа» (job). Рассмотрим, что делает каждая из ролей и какие компоненты на ней выполняются.

2.8 Приложение (app)

На данной роли серверов выполняется основной компонент системы. Данная роль генерирует веб-страницы на основе запросов от браузера, а также предоставляет файлы и вложения из сетевого хранилища.

[4] Небольшая программа или последовательность команд, написанная для автоматизации конкретных задач, обработки данных или создания интерактивности.

Компонент	Описание
приложение (app)	Основной компонент системы. Принимает запросы от компонента nginx, генерирует страницы и отдает их через nginx обратно клиенту.
nginx[5]	Сайдкар (*от англ. Sidecar – вспомогательный элемент, работающий параллельно с основным*) для приложения (app). Всегда работает вкупе с приложением (app). Предоставляет статические файлы и вложения. Перенаправляет запрос в приложение (app), если это не запрос на статический файл или вложение.

2.9 Задание/работа (job)

На данной роли серверов выполняются некоторые вспомогательные сервисы, а также компонент обработки отложенных задач. Данная роль выполняет обработку требовательных и длинных процессов в системе, периодических задач и генерацию PDF-файлов.

Компонент	Описание
delayed-job-periodic	Обработчики отложенных задач. Разделены на несколько очередей для возможности раздельного масштабирования и обеспечения качества обслуживания (QoS)[6] для разных типов задач.
delayed-job-slow	
delayed-job-normal	
delayed-job-fast	
delayed-job-urgent	
delayed-job-notifications	
pdf	Компонент создания PDF-файлов. Используется только обработчиками отложенных задач.
clacks	Обработчик входящей электронной почты. Читает входящие письма через протокол доступа к интернет-сообщениям (IMAP).
memcached	Сервис кэширования.

2.10 Реальное время (realtime)

На данной роли выполняются компоненты, необходимые для работы функциональности реального времени[7]. Данная роль отвечает за функцию отправки изменений в реальном времени клиентам, распознавание коллизий и отправку автоматических уведомлений.

Компонент	Описание
faye	Обработчик соединений реального времени. Данный сервис принимает запрос на установку сессии от браузера, и при

[5] Важно обратить внимание, что данный компонент НЕ является балансировщиком нагрузки. Для функционирования приложения необходим вышестоящий балансировщик.

[6] QoS - от англ. Quality of Service - качество обслуживания; технология управления сетевым трафиком, позволяющая приоритизировать важные данные над менее критичными, обеспечивая им гарантированную пропускную способность, минимальные задержки и отсутствие потерь пакетов.

[7] Под функциональностью реального времени подразумевается возможность обновлять данные на уже открытой странице без её перезагрузки у всех подключенных браузеров.

	возможности обновляет сессию до веб-сокета[8] (websocket) соединения. Хранит текущие сессии в redis.
sneakers	Обработчик очереди RabbitMQ. Принимает сообщения об изменении данных и маршрутизирует их в необходимый канал компонента faye.
rabbitmq	Сервис очереди сообщений. Получает от ролей приложения и заданий/работы (app/job) информацию об изменении данных и уведомлениях.
redis	Сервис хранилища «ключ-значение» (Key/Value). Используется ролью реального времени (realtime) для хранения сессий и ролями приложения и заданий/работы (app/job) для инкрементных[9] значений и распределенной блокировки (distributed locking).

[8] Веб-сокет (websocket) - протокол связи для обмена сообщениями между браузером и веб-сервером с постоянным соединением.

[9] Инкрементный - процесс постепенного изменения, увеличения или дополнения чего-либо, при котором обновляются только новые или измененные данные, а не вся система целиком

3 ДАННЫЕ

Следующие данные хранятся в разных компонентах системы.

Тип	Описание данных	Кем используется
Реляционная база данных	Основные данные приложения	Компоненты ролей приложения (app) и «заданий/работы» (job)
Аналитическая база данных	Копия некоторых основных данных приложения из реляционной базы данных, необходимая для выполнения аналитики	Компонент роли приложения (app) читает данные. Компонент роли «заданий/работы» (job) читает и пишет данные
Файлы	Вложения, логи импорта/экспорта и массового редактирования, аватары и прочие загружаемые и генерируемые приложением файлы.	Компоненты ролей приложения (app) и «заданий/работы» (job)
Redis	Сервис хранилища «ключ-значение» (Key/Value). Используется ролью реального времени (realtime) для хранения сессий и ролями приложения (app) и «заданий/работы» (job) для инкрементных значений и распределенной блокировки (distributed locking)	Компонент faye. Компоненты ролей приложения (app) и «заданий/работы» (job)
Memcached	Кэшированные значения.	Компоненты ролей приложения (app) и «заданий/работы» (job)
RabbitMQ	Сообщения об изменениях в данных для отправки их ролью реального времени (realtime) подключенным браузерам.	Компоненты ролей приложения (app) и «заданий/работы» (job) создают сообщения. Компонент sneakers роли реального времени (realtime) читает и удаляет сообщения.

4 ДОМЕНЫ И ПОДДОМЕНЫ[10]

Система использует поддомены для маршрутизации по пространствам. Каждому пространству выделяется поддомен, куда заходят пользователи. Из-за этого в больших инсталляциях может использоваться достаточно много поддоменов. Для оптимизации администрирования рекомендуется:

- использовать подстановочные сертификаты (wildcard SSL Certificate). Такой сертификат выпускается сразу на все поддомены определенного домена;
- использовать подстановочные записи (wildcard DNS). Многие сервера системы доменных имен поддерживают подстановочные (wildcard) записи – это записи, где поддоменом выступает символ *. Такие записи позволяют маршрутизировать все поддомены на сервера системы.

К примеру, если у вас созданы два пространства PP-Tex (rr-tech) и ПроПродукт (pro-product), при основном домене системы example.com их адресами будут являться:

- rr-tech.example.com
- pro-product.example.com

Также система имеет несколько предопределенных поддоменов, необходимых для функционирования. Список этих доменов и их назначение представлено в таблице.

Поддомен	Описание
api	Поддомен для REST API* **архитектурный стиль взаимодействия между программами*
graphql	Поддомен для GraphQL API** ***язык запросов и серверная среда с открытым исходным кодом*
oauth	Поддомен для OAuth API*** ****протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе*
assets0	Поддомены для доставки статических файлов (css/js). Возможно использование нескольких поддоменов для статических файлов для параллельной загрузки файлов при работе через http/1.1[11]
assets1	
assets2	
assets3	
realtime	Поддомен для сервиса реального времени (faye)
io	Поддомен для сервиса коротких ссылок

Если требуется, то поддомены, представленные в таблице, возможно изменить.

Учитывайте, что для разных сред (к примеру, для тестовой и продуктивной) потребуются разные основные домены. Для примера:

- Продуктивная среда – r-service.example.com
- Тестирование – r-service-qa.example.com
- Разработка – r-service-dev.example.com

[10] Поддомен - домен, являющийся частью домена более высокого уровня.

[11] http/1.1 загружает файлы друг за другом в рамках одного поддомена. http/2.0 поддерживает мультиплексирование (передача данных по нескольким логическим каналам связи в одном физическом канале) поэтому для данной версии протокола http использование нескольких поддоменов не так актуально.

5 ЭЛЕКТРОННАЯ ПОЧТА

Система отправляет и получает электронную почту с нескольких ящиков, которые должны быть доступны до начала процесса установки.

Переменная среды	Пример	Описание
ITRP_ERRORS_MAILBOX	rs-errors@example.com	Ящик, на который будут отправлены сообщения об ошибках внутри системы
ITRP_INFO_MAILBOX	rs-info@example.com	Ящик, на который будут отправлены информационные сообщения для администраторов системы, такие как общее использование лицензий или использование пространства на диске
ITRP_SUPPORT_MAILBOX	support@example.com	Используется в письмах создания пространств как «от» (from) и «ответить-кому» (reply-to) адрес
ITRP_NOREPLY_MAILBOX	noreply@example.com	Ящик, который будет использоваться как «от» (from) адрес, если ответ на письмо не предусмотрен
ITRP_INBOUND_MAILBOX	rs@example.com	Имя ящика с входящей почтой.
ITRP_ERRORS_MAILBOX	rs-errors@example.com	Ящик, на который будут отправлены сообщения об ошибках внутри системы

5.1 Входящая почта

Для маршрутизации входящей почты между пространствами в системе существует два подхода:

1. Общий ящик с маршрутизацией по домену (Catch-all)
2. Один ящик, маршрутизация через «+» (Subaddressing)

Оба варианта позволяют системе работать с множеством адресов при фактически едином физическом почтовом ящике, но различаются по способу маршрутизации и настройке на стороне почтового сервера.

5.1.1 Общий ящик с маршрутизацией по домену (Catch-all)

Принцип работы:

Общий ящик с маршрутизацией по домену (Catch-all) — это механизм, при котором все письма, отправленные на любой адрес определённого домена, перенаправляются в один общий ящик. Например, если настроен общий ящик с маршрутизацией по домену (Catch-all) для домена mail.r-service.tech, то письма, отправленные на:

- r-service@mail.r-service.tech
- support@mail.r-service.tech
- finance@mail.r-service.tech

Все будут перенаправлены в один и тот же ящик.

Использование в системе:

В этом сценарии для каждого пространства или команды создаётся уникальный адрес, например r-service@mail.r-service.tech, и все письма, отправленные на такие адреса, доставляются в общий ящик системы.

Настройка:

1. Создаётся один почтовый ящик, к которому система подключается по протоколу доступа к интернет-сообщениям (IMAP).
2. На почтовом сервере настраивается общий ящик с маршрутизацией по домену (Catch-all) для поддомена mail, который перенаправляет все письма на этот ящик.
3. Система, анализируя заголовок «Кому» (To), определяет конкретный сервис или пространство, куда относится письмо.

Преимущества:

1. Простота настройки для множества адресов.
2. Работает, даже если почтовый сервер не поддерживает один ящик с маршрутизацией через + (Subaddressing).

Недостатки:

1. Не всегда подходит для инсталляций Он-Премис (от англ.On-Premise - метод развертывания IT-систем, при котором программное обеспечение устанавливается и работает исключительно на собственных серверах, оборудовании и инфраструктуре компании).

5.1.2 Один ящик, маршрутизация через «+» (Subaddressing)

Принцип работы:

«Плюсовая адресация» (Subaddressing) — это возможность почтового сервера воспринимать символ «+» как разделитель между основным адресом и дополнительной меткой. Например: rs+r-service@r-service.tech.

Здесь:

- rs@r-service.tech — основной ящик,
- r-service — субадрес (идентификатор пространства).

Письмо доставляется в ящик rs@r-service.tech, но заголовок «Кому» (To) сохраняется в исходном виде (rs+r-service@r-service.tech), что позволяет системе определить назначение письма.

Настройка:

1. Создаётся один почтовый ящик, к которому система обращается через протокол доступа к интернет-сообщениям (IMAP). Пример: rs@r-service.tech.
2. Если почтовый сервер поддерживает «плюсовую адресацию» (subaddressing) (например, Gmail, ProtonMail, FastMail, Microsoft 365 Cloud): дополнительная настройка не требуется — сервер автоматически доставит письмо в основной ящик.
3. Если почтовый сервер не поддерживает «плюсовую адресацию» (subaddressing):

- нужно настроить алиасы (от англ. alias — псевдоним; это альтернативное, обычно короткое и запоминающееся имя, назначенное объекту, команде или адресу

вместо длинного, сложного или оригинального) для адресов с +, чтобы они перенаправлялись на основной ящик.

4. Для Он-Премис (On-Premise) Exchange рекомендуется:

- создать общий почтовый ящик (shared mailbox), принимающий письма на адреса вида rs+r- service@r-service.tech;
- настроить автоматическую пересылку на основной ящик rs@r-service.tech.

Преимущества:

1. Простота маршрутизации — субадрес прямо указывает на нужное пространство.
2. Не нужно выделять отдельный домен (подходит для многих Он-Премис (On-Premise) инсталляций).

Недостатки:

1. Требует поддержки «плюсовой адресации» (subaddressing) со стороны почтового сервера.
2. Для некоторых Он-Премис (On-Premise) решений (например, Exchange без современной проверки подлинности (modern mode)) может потребоваться ручная настройка алиасов или транспортных правил.

6 НАСТРОЙКА СИСТЕМЫ

Настройка системы производится через переменные среды. Для удобства стандартные скрипты развертывания используют четыре файла с переменными среды:

- .env – общий для всех серверов;
- .env.web – переменные для серверов роли приложения (app);
- .env.job – переменные для серверов роли «заданий/работы» (job);
- .env.realtime – переменные для серверов роли реального времени (realtime).

Список всех переменных доступен в **Приложении 1**.

7 МЕХАНИЗМЫ РАЗВЕРТЫВАНИЯ

Система Р-Сервис поддерживает два механизма развертывания, отличающиеся сложностью настройки и необходимой квалификацией системного администратора. Обычно выбор механизма развертывания зависит от размера инсталляции и имеющихся в компании ресурсов.

Оба механизма развертывания поддерживают обновление под нагрузкой за счет применения скользящих (последовательных) обновлений (rolling updates).

7.1 Инструменты Docker Swarm или Docker Compose

В данном механизме развертывания используется предопределенный набор серверов, четко поделенных на роли (как описано в Обзоре архитектуры). Для оркестрации[12] используется решение Docker Swarm.

В данной конфигурации не поддерживается авто масштабирование.

Рекомендуется для маленьких и средних инсталляций, где не требуется динамическое авто масштабирование и имеется прогнозируемая ежедневная нагрузка.

7.2 Инструмент Kubernetes

В данном механизме развертывания используется продукт Kubernetes, который позволяет динамически использовать набор серверов для распределения и масштабирования компонентов приложения.

Рекомендуется для крупных инсталляций или для компаний, где имеется существующий кластер Kubernetes и соответствующая экспертиза системных администраторов.

[12] Оркестрация - автоматизированное управление, координация и интеграция множества разрозненных программных компонентов, сервисов или инфраструктурных элементов в единый рабочий процесс.

8 ПЛАНИРОВАНИЕ РЕСУРСОВ И ТРЕБОВАНИЯ К ОБОРУДОВАНИЮ

В данном разделе рассматриваются необходимые ресурсы для инсталляции. Важно заметить, что данные ресурсы — это только лишь отправная точка, ведь в зависимости от характера нагрузки, сложности бизнес-процессов и с ростом инсталляции и объема данных потребуется масштабирование каждого компонента в зависимости от нагрузки.

Рекомендуется иметь две среды:

- продуктивная;
- тестовая.

Тестовая среда используется для проверки бизнес-сценариев и тестирования установки нового релиза.

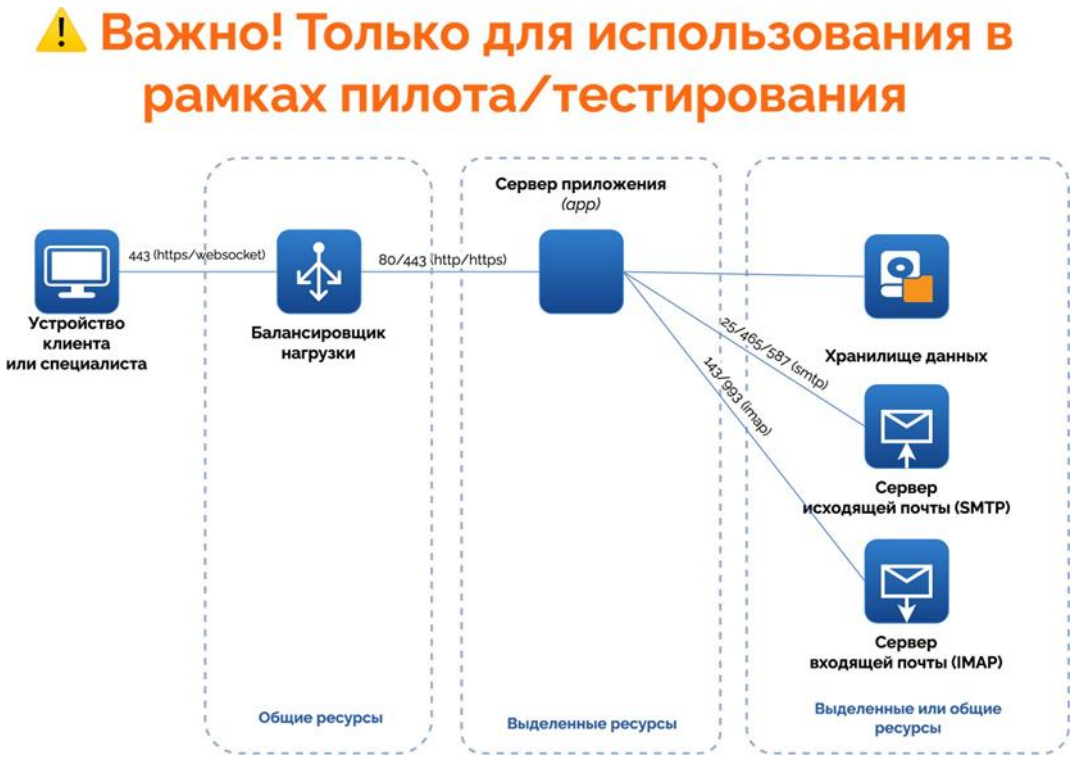
Также возможно использовать больше сред (к примеру, продуктивная, среда тестирования, среда финального тестирования, среда разработки), но обычно это не требуется.

Данная секция по большей части применима для механизма развертывания Docker Swarm, но может являться отправной точкой для планирования требуемых ресурсов в кластере Kubernetes.

8.1 Пробная/тестовая инсталляция

В данной конфигурации все компоненты разворачиваются на одном сервере. Данная конфигурация не подходит для эксплуатации в продуктивной среде.

Роль	Имя	Процессор (CPU)	Память (RAM)	Хранилище (Storage)
Прилолжение (app)	APP1	8 (100%)	32 Гб	64 Гб (ssd), 80 Гб (HDD)



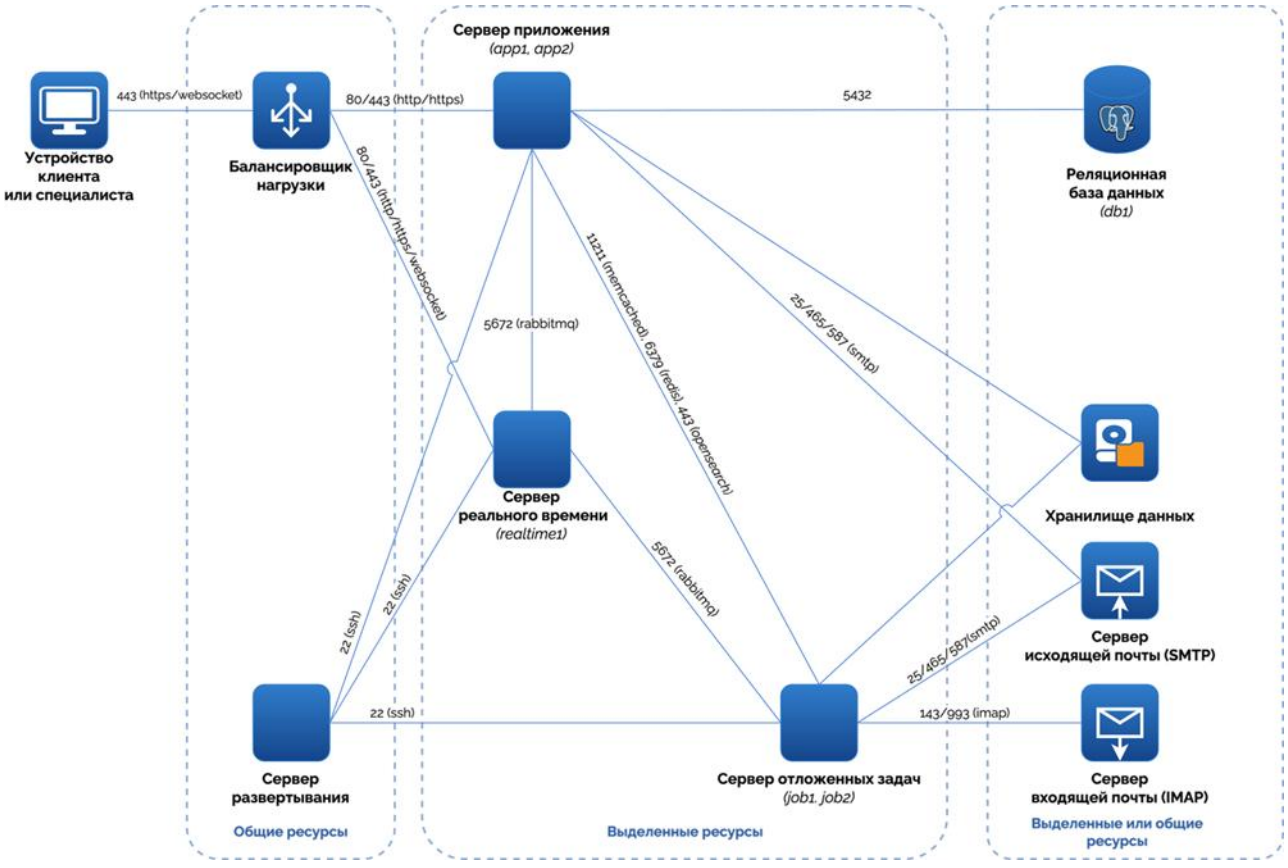
8.2 Маленькая инсталляция (примерно 500 специалистов, 8400 запросов в день)

В данной конфигурации обеспечивается отказоустойчивость ролей приложения (app) и «заданий/работы» (job). Для экономии ресурсов аналитическая база данных работает на одном из серверов роли «заданий/работы» (job).

Не обеспечивается отказоустойчивость сервера реляционной базы данных.

При необходимости возможно построение кластера.

Роль	Имя	Процессор (CPU)	Память (RAM)	Хранилище (Storage)
Прилолжение (app)	APP1	2 (100%)	4 Гб	30 Гб (HDD)
Прилолжение (app)	APP2	2 (100%)	4 Гб	30 Гб (HDD)
Задание/работа (job)	JOB1	4 (100%)	16 Гб	30 Гб (HDD), 50 Гб (SSD, OpenSearch storage)
Задание/работа (job)	JOB2	4 (100%)	16 Гб	30 Гб (HDD)
Реальное время (realtime)	REALTIME1	2 (100%)	4 Гб	30 Гб (HDD)
	DB1	4 (100%)	16 Гб	30 Гб (HDD, Boot) 50 Гб (SSD, DB Storage)
	DEPLOYMENT[13]	1 (25%)	2 Гб	30 Гб (HDD, Boot)



[13] Сервер развертывания общий для всех сред

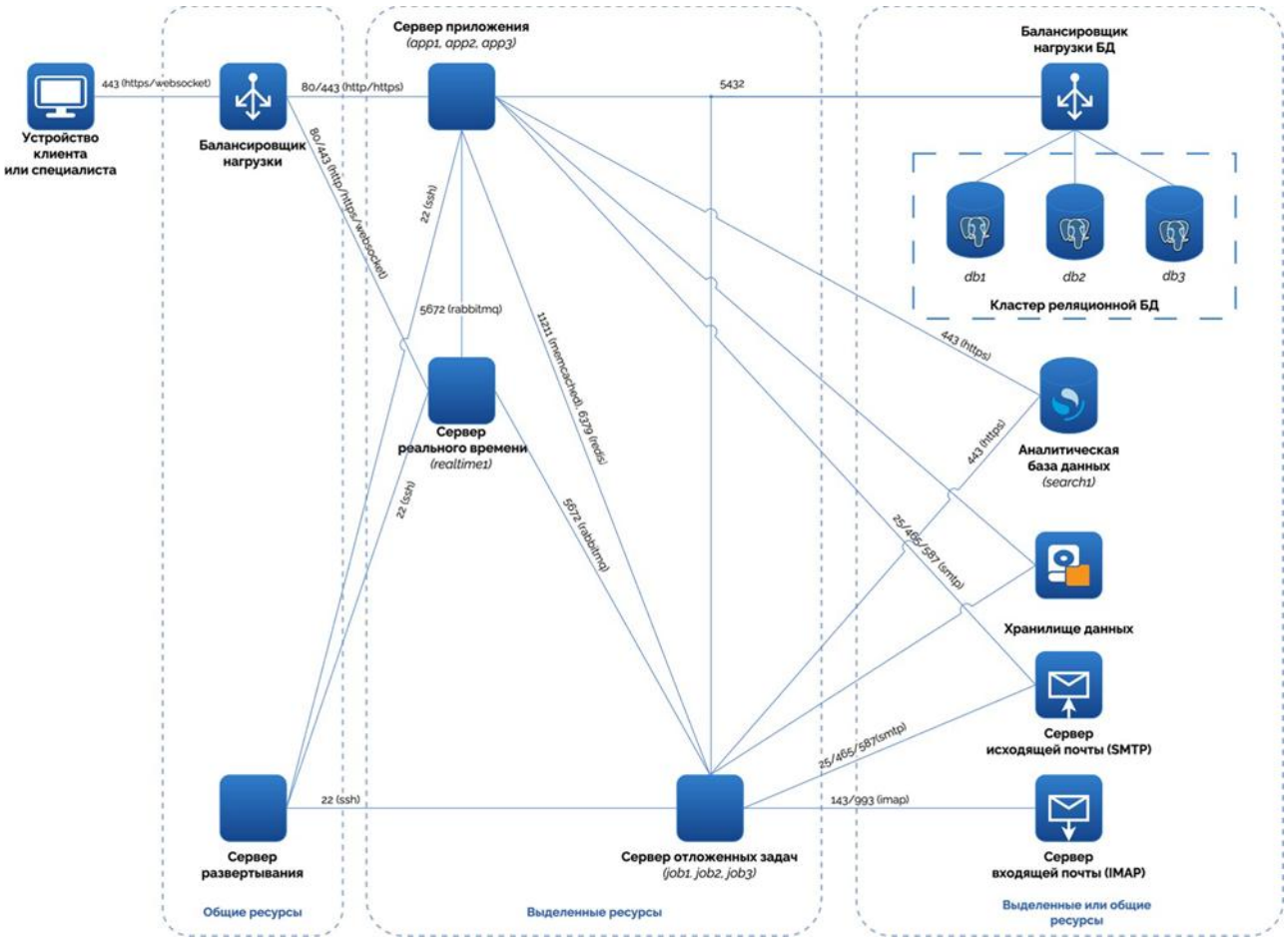
8.3 Средняя инсталляция (примерно 4000 специалистов, 20160 запросов в день)

В данной конфигурации обеспечивается отказоустойчивость ролей приложения (app) и «заданий/работы» (job), а также реляционной базы данных.

Кластер реляционной базы данных рекомендуется организовывать средствами инструмента Patroni. Также рекомендуется использовать etcd (распределенное, отказоустойчивое хранилище данных типа «ключ-значение» (key-value), написанное на языке Go), размещенный на каждом из узлов сервера базы данных. Таким образом, при отказе любого из узлов сработает механизм аварийного переключения (failover), ведь даже с двумя узлами соберется кворум в etcd.

Сервер аналитической базы данных расположен отдельно. Для обеспечения его отказоустойчивости возможно использование кластера OpenSearch.

Роль	Имя	Процессор (CPU)	Память (RAM)	Хранилище (Storage)
Прилолжение (app)	APP1	2 (100%)	4 Гб	30 Гб (HDD)
Прилолжение (app)	APP2	2 (100%)	4 Гб	30 Гб (HDD)
Прилолжение (app)	APP3	2 (100%)	4 Гб	30 Гб (HDD)
Задание/работа (job)	JOB1	4 (100%)	16 Гб	30 Гб (HDD)
Задание/работа (job)	JOB2	4 (100%)	16 Гб	30 Гб (HDD)
Задание/работа (job)	JOB3	4 (100%)	16 Гб	30 Гб (HDD)
Реальное время (realtime)	REALTIME1	2 (100%)	4 Гб	30 Гб (HDD)
	DB1	4 (100%)	16 Гб	30 Гб (HDD, Boot) 200 Гб (SSD, DB Storage)
	DB2	4 (100%)	16 Гб	30 Гб (HDD, Boot) 200 Гб (SSD, DB Storage)
	DB3	4 (100%)	16 Гб	30 Гб (HDD, Boot) 200 Гб (SSD, DB Storage)
	SEARCH1	4 (100%)	16 Гб	30 Гб (HDD, Boot) 200 Гб (SSD, DB Storage)
	DEPLOYMENT	1 (25%)	2 Гб	30 Гб (HDD, Boot)



8.4 Крупная инсталляция (более 10000 специалистов, 201600 запросов в день)

В данной конфигурации обеспечивается отказоустойчивость ролей приложения (app), «заданий/работы» (job) и реального времени (realtime), а также реляционной и аналитической базы данных.

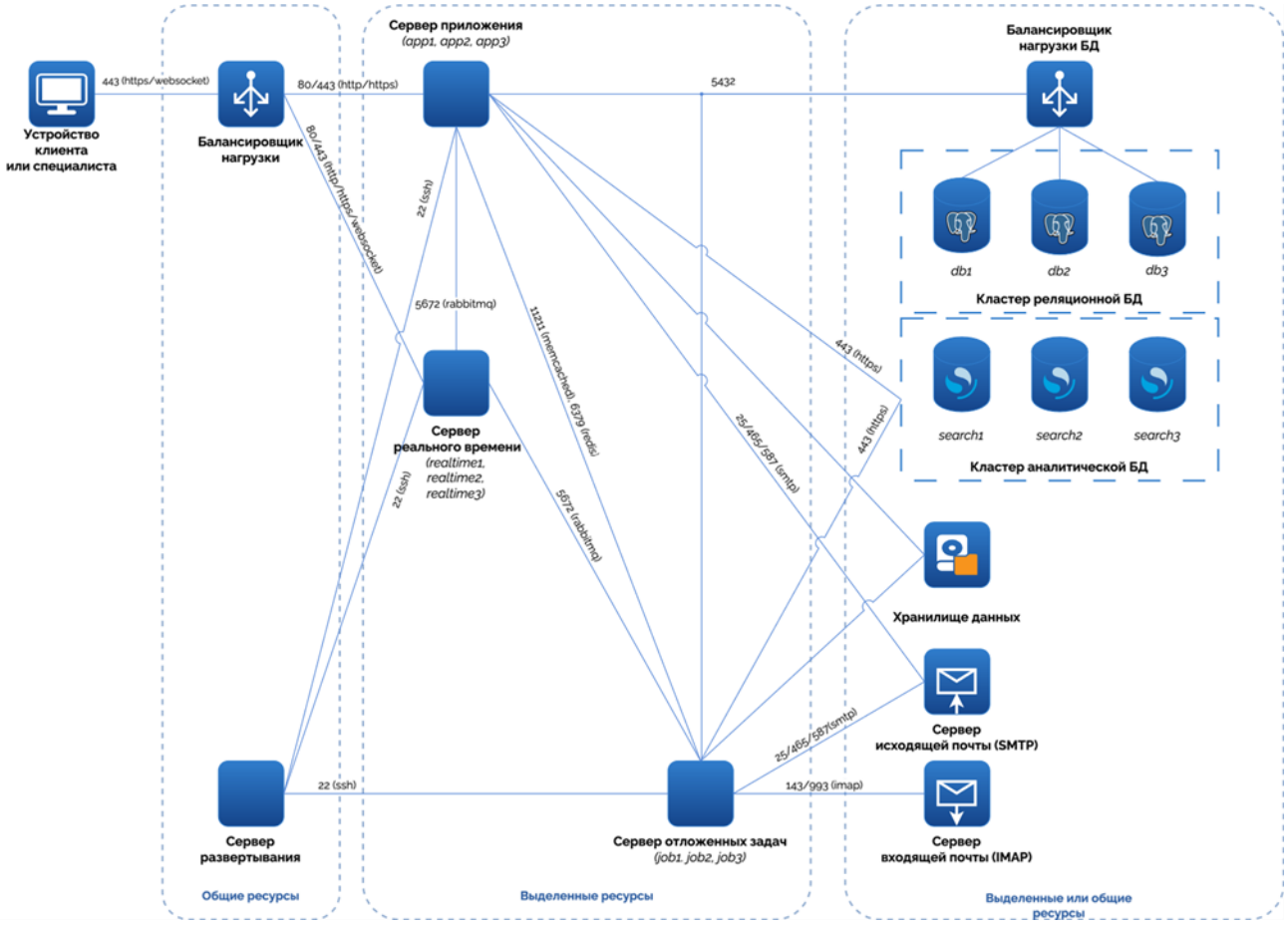
Для масштабирования серверов роли приложения (app) используется 4 воркера (фоновый процесс, сценарий или отдельный сервер, предназначенный для выполнения задач, независимых от основного приложения) в компоненте приложения (app) вместо стандартных двух.

Вспомогательные сервисы также собраны в кластеры:

- RabbitMQ работает в режиме кворумных очередей (quorum queues) на трех серверах реального времени (realtime);
- Redis работает в режиме мониторинга и обеспечения высокой доступности (sentinel) на трех серверах «заданий/работы» (job).

Кластер реляционной базы данных рекомендуется организовывать средствами инструмента Patroni. Также рекомендуется использовать etcd (распределенное, отказоустойчивое хранилище данных типа «ключ-значение» (key-value), написанное на языке Go), размещенный на каждом из узлов сервера БД. Таким образом, при отказе любого из узлов сработает механизм аварийного переключения (failover), ведь с двумя узлами соберется кворум в etcd.

Роль	Имя	Процессор (CPU)	Память (RAM)	Хранилище (Storage)
Прилолжение (app)	APP1	4 (100%)	6 Гб	30 Гб (HDD)
Прилолжение (app)	APP2	4 (100%)	6 Гб	30 Гб (HDD)
Прилолжение (app)	APP3	4 (100%)	6 Гб	30 Гб (HDD)
Задание/работа (job)	JOB1	4 (100%)	16 Гб	30 Гб (HDD)
Задание/работа (job)	JOB2	4 (100%)	16 Гб	30 Гб (HDD)
Задание/работа (job)	JOB3	4 (100%)	16 Гб	30 Гб (HDD)
Реальное время (realtime)	REALTIME1	2 (100%)	4 Гб	30 Гб (HDD)
	DB1	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	DB2	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	DB3	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	SEARCH1	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	SEARCH2	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	SEARCH3	4 (100%)	16 Гб	30 Гб (HDD, Boot) 500 Гб (SSD, DB Storage)
	DEPLOYMENT	1 (25%)	2 Гб	30 Гб (HDD, Boot)



8.5 Требования к оборудованию

1. Необходимо обеспечить каналы связи шириной не менее 1 Гб/сек между всеми хостами[14].

2. Времяна всех хостах должно быть одинаковыми должно синхронизироваться из единого источника.

[14] Хосты - устройства, работающие в режиме «клиент-сервер».

9 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

9.1 Браузер клиента

Для правильного функционирования системы необходимо использовать современные браузеры Chrome, Яндекс Браузер, Firefox, Safari актуальных версий не старше одного года.

9.2 Операционная система

На виртуальных машинах не должно быть посторонних сервисов, антивирусов, графических подсистем и других подобных программ.

Имя	Версия
Ubuntu	>=24.04
Red Hat Enterprise Linux	9 мажорная
Astra Linux	>=1.8

9.3 Docker

На серверах ролей приложения (app), «заданий/работы» (job) и реального времени (realtime) должен быть установлен Docker Engine и Docker Compose v2.

Если вы используете стандартный драйвер создания журналов логов (logging driver) (json), то убедитесь, что вы настроили ротацию логов, как описано тут. Рекомендуется использовать следующие настройки в файле daemon.json:

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "100m",
        "max-file": "10"
    }
}
```

9.4 Системные переменные

Следующие системные переменные требуются для работы системы:

Роль сервера	Название переменной	Значение
Все	Максимальное количество файловых дескрипторов на процесс (мягкий лимит - soft limit)	8192
Все	Максимальное количество файловых дескрипторов на процесс (жесткий лимит - hard limit)	65536
Реальное время (Realtime)	fs.file-max	2097152
Реальное время (Realtime)	Максимальное количество файловых дескрипторов на процесс (жесткий лимит - hard limit)	100000
Поиск (Search)	vm.max_map_count	262144

Значения данных системных переменных автоматически устанавливаются во время этапа инсталляции системы скриптами развертывания.

9.5 Реляционная база данных

В качестве реляционной базы данных используется PostgreSQL версии 18 (также поддерживается версия 17). Система не требует установки каких-либо расширений PostgreSQL.

Для реализации высокой доступности и отказоустойчивости рекомендуется использовать кластер PostgreSQL, управляемые инструментом Patroni и etcd. Рекомендуемая конфигурация - минимум три узла. С тремя узлами обеспечивается кворум.

9.6 Аналитическая база данных

В качестве аналитической базы данных используется OpenSearch мажорной версии 3. Система использует следующие расширения:

- analysis-icu
- analysis-phonenumber

Образ docker с предустановленными расширениями входит в стандартный комплект поставки.

Для реализации высокой доступности и отказоустойчивости рекомендуется использовать кластер OpenSearch. Рекомендуемая конфигурация - минимум три узла. С тремя узлами обеспечивается кворум.

9.7 Балансировщик

Для правильной работы системы следующие http[15] заголовки должны присутствовать в запросе, переданном от балансировщика в компонент роли приложения (app) системы:

- Host – убедитесь, что он исходный домен, так как данный заголовок используется для определения пространства
- X-Forwarded-For – используется для определения исходного IP-адреса
- X-Forwarded-Proto – всегда https

Для проверки здоровья компонентов роли приложения (app) возможно использовать следующие конечные точки (endpoint) для активной проверки работоспособности (active health check):

- /_.gif – проверка здоровья компонента nginx
- /teapot – проверка здоровья компонента приложения (app). Отвечает с телом ответа “I am a teapot”.

Используйте данную конечную точку (endpoint) только тогда, когда ваш балансировщик умеет проверять тело ответа, а не только код ответа.

Для проверки здоровья компонентов роли реального времени (realtime) возможно использовать следующие конечные точки (endpoint) для активной проверки работоспособности (active health check):

- /teapot – проверка здоровья компонента faye. Отвечает с телом ответа “I am a teapot”.

Используйте данную конечную точку (endpoint) только тогда, когда ваш балансировщик умеет проверять тело ответа, а не только код ответа.

[15] HTTP - протокол передачи гипертекста

10 ТРЕБОВАНИЯ К КВАЛИФИКАЦИИ СИСТЕМНОГО АДМИНИСТРАТОРА СИСТЕМЫ

Системный администратор, который будет разворачивать и обслуживать систему, должен обладать следующими профессиональными навыками:

1. **Администрирование операционных систем на базе Linux:**

- опыт работы с системой контейнеризации Docker (запуск/остановка контейнеров, открытие логов, работа с Docker Compose);
- понимание работы инструмента Ansible (понимание механизма работы плейбуков[16], базовое понимание работы переменных в инструменте Ansible);
- понимание сетевых настроек системы и умение работать с ними.

2. **Базы данных:**

- понимание работы реляционной БД PostgreSQL. Опыт развертывания и администрирования данной БД;
- понимание механизмов создания резервных копий БД.

3. **Балансировщики нагрузки**

- понимание работы механизмов балансировки нагрузки;
- понимание работы механизмов проверки состояния загрузки;
- опыт настройки L4/L7 балансировщиков и реверсивных прокси, таких как nginx или HAProxy.

4. **Дополнительно:**

- знание системы Р-Сервис или пройденный вводный курс специалиста системы Р-Сервис.

Для обслуживания крупных инсталляций также рекомендовано обладать следующими профессиональными навыками:

1. **Мониторинг:**

- опыт работы и настройки систем экспорта логов, таких как Elastic Stack или Graylog;
- опыт работы и настройки систем сбора метрик, таких как Prometheus;
- опыт работы и настройки систем сбора трассировок, таких как Jaeger;
- опыт работы и настройки системы визуализации данных, таких как Grafana;
- опыт работы с механизмами инструментации[17] ВМ и контейнеров;
- понимание работы протокола OpenTelemetry.

2. **Базы данных:**

- понимание работы механизмов репликации в БД PostgreSQL;
- опыт развертывания кластеров БД PostgreSQL;

[16] Плейбуки - файлы сценариев, описывающие последовательность автоматических задач для настройки серверов, развертывания приложений или управления инфраструктурой

[17] Инструментация - процесс внедрения в программный код специальных средств для мониторинга производительности, отладки ошибок и сбора метрик в реальном времени.

- опыт развертывания кластеров БД OpenSearch;
- понимание работы БД Redis и опыт её настройки;
- понимание работы БД Memcached и опыт её настройки.

3. **Брокер сообщений:**

- опыт работы с брокером сообщений RabbitMQ;
- понимание механизмов создания кластера брокера сообщений RabbitMQ.

4. **Дополнительно:**

- пройденный курс "Системный администратор Р-Сервис".

11 РЕЗЕРВНОЕ КОПИРОВАНИЕ

Резервные копии необходимо хранить на внешних ресурсах, не относящихся к инсталляции. Рекомендуемый срок хранения резервных копий – не менее 7 дней.

11.1 Реляционная база данных

Резервные копии данных из реляционной базы данных должны производиться на регулярной основе. Администрирование реляционной базы данных — это ответственность заказчика — механизм создания резервных копий входит в процесс администрирования.

11.2 Сетевое хранилище

Резервные копии файлов из сетевого хранилище должны производиться на регулярной основе.

Резервное копирование данных других компонентов системы не имеет практического смысла, так как или они несут временных характер, или их воссоздание возможно из резервных копий других данных.

12 ОТКАЗОУСТОЙЧИВОСТЬ И МАСШТАБИРОВАНИЕ

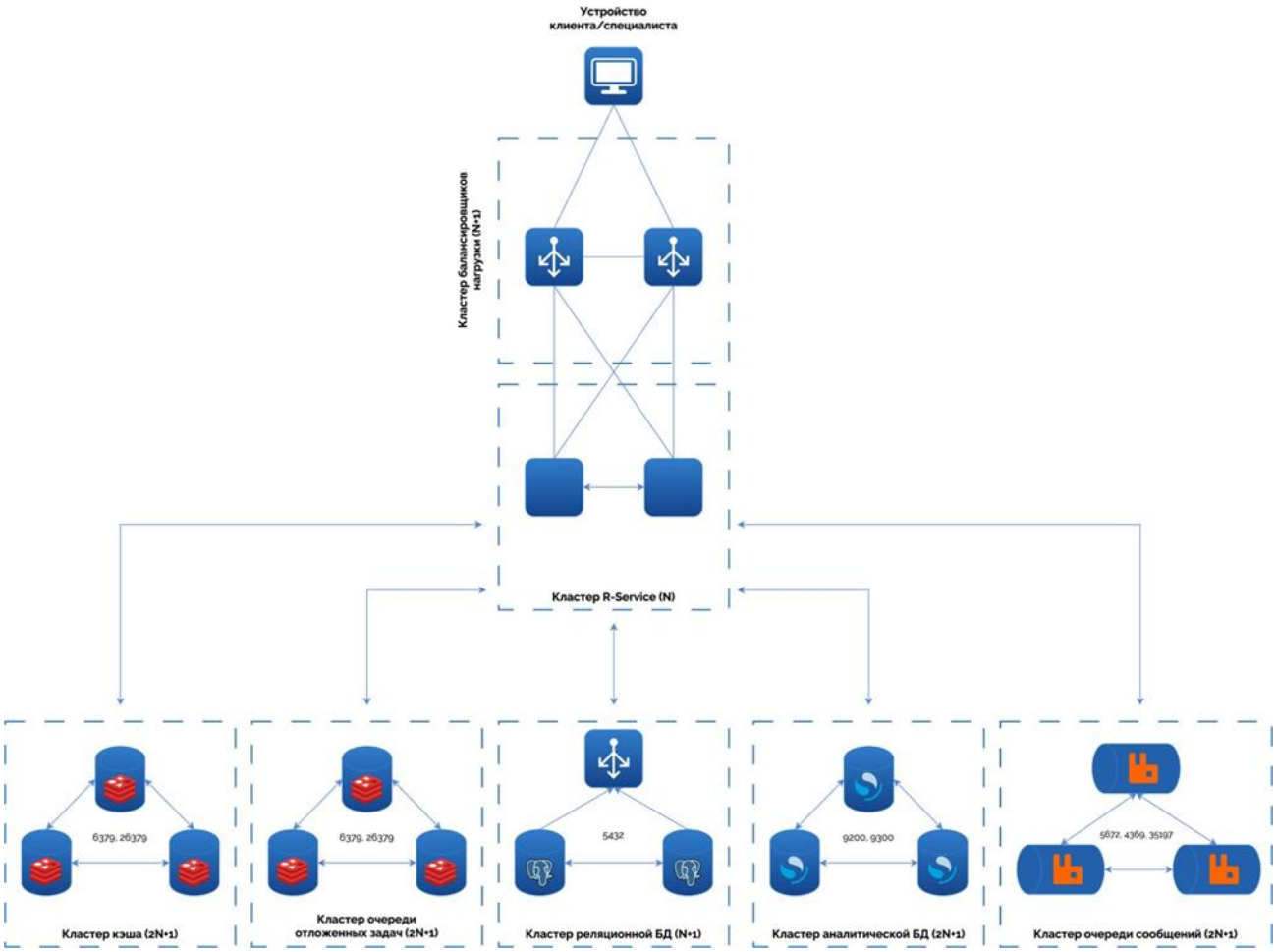
Система разработана с учетом отказоустойчивости.

Система будет отвечать на запросы пользователей, имея лишь один сервер компонента приложения (app) и реляционную базу данных.

Полный отказ многих компонентов не несет за собой недоступность системы.

Система продолжит работать, но некоторые функции будут деградированы.

Почти все компоненты системы возможно запустить в кластерном режиме или с использованием нескольких реплик.



12.1 Реляционная база данных

Реляционную базу данных под управлением СУБД PostgreSQL возможно объединить в кластер мастер-реплик(и) (Master-Replica(s)). Один сервер избирается главным и отправляет все изменения на остальные сервера.

Для реализации кластера PostgreSQL рекомендуется использование инструмента Patroni. В качестве хранилища рекомендуется использование etcd, установленного на каждом из узлов БД.

Желательно использование пула соединений (connection pooler), установленных на каждом из узлов, таких как Pgbouncer.

В качестве балансировщика рекомендуется использование HAProxy.

Для обеспечения полноценного кворума рекомендуется использование нечетного количества узлов.

Минимальное количество серверов для организации кластера: три сервера.

12.2 Аналитическая база данных

Аналитическая база данных OpenSearch является распределенной базой данных и использует шардирование[18].

Каждый из серверов хранит свою копию данных, что обеспечивает распределение чтения/записи между разными узлами и обеспечение отказоустойчивости и сохранности данных.

Для реализации кластера OpenSearch используется встроенный функционал.

Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

12.3 Очередь сообщений

Очередь сообщений RabbitMQ использует т.н. «кворумную очередь» (quorum queues), где из сервера образуют кворум. Кворум – соглашение между большинством узлов о состоянии и содержании очереди.

Для реализации кластера RabbitMQ используется встроенный функционал.

Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

12.4 Кэш

СУБД Redis возможно объединить в кластер мастер-реплик(и) (Master-Replica(s)). Один сервер избирается главным и отправляет все изменения на остальные сервера.

Для реализации кластера СУБД Redis в целях использования под кэш рекомендуется использование функционала системы мониторинга и автоматического обеспечения высокой доступности (Sentiniel). Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

12.5 Очередь отложенных задач

СУБД Redis возможно объединить в кластер мастер-реплик(и) (Master-Replica(s)). Один сервер избирается главным и отправляет все изменения на остальные сервера.

Для реализации кластера СУБД Redis в целях использования под кэш рекомендуется использование функционала системы мониторинга и автоматического обеспечения высокой доступности (Sentiniel). Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

[18] Шардирование - метод горизонтального масштабирования баз данных (БД), при котором огромная таблица или вся база разбивается на мелкие независимые части, распределяемые по разным серверам.

12.6 Приложение Р-Сервис

Почти каждый компонент приложения Р-Сервис возможно запустить с использованием нескольких реплик, что обеспечивает масштабирование и отказоустойчивость при отказе одного из серверов.

Исключением являются компоненты:

- delayed-job-periodic
- clacks

Данные компоненты должны быть в единичном экземпляре на всю систему.

Минимальное количество серверов для организации отказоустойчивости: два сервера.

12.7 Балансировщик нагрузки

Для полного резервирования каждого из компонентов возможно использование двух серверов балансировки. Механизм реализации будет зависеть от структуры сети и существующих ресурсов.

Как один из вариантов – использование сетевого протокола для обеспечения отказоустойчивости шлюза по умолчанию (VRRP) и построенных на базе данного протокола решений, таких как Keepalived.

Минимальное количество серверов для организации отказоустойчивости: два сервера.

Следующая секция рассматривает поведение системы при отказе компонента, а именно:

- поведение при полном отказе;
- поведение при деградации;
- возможно ли обеспечение отказоустойчивости данного компонента;
- как данный компонент восстанавливается после отказа.

12.8 Компоненты приложения (app) / nginx

Поведение при полном отказе	http/https соединения от балансировщика не устанавливаются. Пользователи не могут открыть никакую страницу системы.
Деградированное поведение	Производительность под нагрузкой снижена. Если балансировщик правильно обрабатывает проверку работоспособности (health check), пользователи не должны заметить ошибок.
Обеспечение отказоустойчивости	Возможно и рекомендуется использование нескольких реплик данных компонентов.
Восстановление	Балансировщик нагрузки после восстановления должен включить реплику в пул балансировки и начать передавать трафик после решения проблемы.

12.9 Компонент delayed-job

Поведение при полном отказе	Отложенные задачи не выполняются.
Деградированное поведение	Так как используется несколько очередей, деградация каждой из очереди будет нести немного разное поведение: Срочный (urgent) – задачи для рабочих процессов не будут создаваться. Быстрый (fast) – изменения не будут переиндексированы в аналитической БД. Не будут отправлены вебхуки*. Возможны расхождения данных, отсутствие новых сущностей (записей) в табличных представлениях. **Вебхук - механизм взаимодействия между веб-сервисами, который позволяет автоматически отправлять данные в реальном времени при возникновении определенного события.* Нормальный (normal) – некоторые уведомления и письма не будут отправлены. Многие фоновые вычисления не будут выполнены. Медленный (slow) – импорты и экспорты не будут выполнены Периодический (periodic) – периодические задачи не будут выполнены. Уведомления (notifications) – уведомления и письма не будут отправлены.
Обеспечение отказоустойчивости	Возможно и рекомендуется использование нескольких реплик для каждой из очередей.

12.10 Компонент clacks

Поведение при полном отказе	PDF-файлы для согласований и экспорта панелей мониторинга не генерируются.
Деградированное поведение	
Обеспечение отказоустойчивости	Невозможно.
Восстановление	После решения проблемы не обработанные письма обработаются.

12.11 Компонент pdf

Поведение при полном отказе	PDF-файлы для согласований и экспорта панелей мониторинга не генерируются.
Деградированное поведение	
Обеспечение отказоустойчивости	Возможно использование нескольких реплик.
Восстановление	PDF-файлы генерируются компонентом delayed-job, который повторит попытку генерации 10 раз, каждый раз увеличивая задержку перед повторной попыткой. Как только проблема будет решена, отложенная задача на генерацию PDF будет успешно выполнена.

12.12 Компонент faye

Поведение при полном отказе	http/https и веб-сокет* (websocket) соединения от балансировщика не устанавливаются. Функциональность реального времени не работает.
Деградированное поведение	Производительность под нагрузкой снижена. Если балансировщик правильно обрабатывает проверку работоспособности (health check), пользователи не

	должны заметить ошибок.
Обеспечение отказоустойчивости	Возможно, использование нескольких реплик. Необходимо использование одного и того же сервера компонента redis.
Восстановление	Браузеры постоянно пробуют повторить соединение с сервером реального времени (данным компонентом). Как только проблема будет решена, соединение будет восстановлено.

12.13 Компонент sneakers

Поведение при полном отказе	Не отправляются уведомления и изменения через функционал реального времени. Если компонент faye работает, то при отказе sneakers будет работать только распознавание коллизий.
Деградированное поведение	Производительность под нагрузкой снижена. Возможна задержка отправки уведомлений и изменений через функционал реального времени.
Обеспечение отказоустойчивости	Возможно использование нескольких реплик. Необходимо использование одного и того же сервера компонента rabbitmq.
Восстановление	Очередь сообщений будет разобрана со времени после решения проблемы.

12.14 Компонент redis

Поведение при полном отказе	Компонент faye не позволяет устанавливать сессии. Время ответа основного приложения увеличено.
Деградированное поведение	Производительность под нагрузкой снижена. Возможна задержка отправки уведомлений и изменений через функционал реального времени.
Обеспечение отказоустойчивости	Возможно использование кластера.
Восстановление	После решения проблемы все компоненты попробуют вновь установить соединение с компонентом Redis.

12.15 Компонент Memcached

Поведение при полном отказе	Уменьшение производительности компонента приложения (app).
Деградированное поведение	Уменьшение производительности компонента приложения (app).
Обеспечение отказоустойчивости	Невозможно.
Восстановление	После решения проблемы компонент приложения (app) попробуют вновь установить соединение с компонентом memcached.

12.16 Компонент rabbitmq

Поведение при полном отказе	Не отправляются уведомления и изменения через функционал реального времени. Если компонент faye работает, то при отказе rabbitmq будет работать только распознавание коллизий.
Деградированное поведение	Производительность под нагрузкой снижена. Возможна задержка отправки уведомлений и изменений через функционал реального времени.
Обеспечение отказоустойчивости	Возможно использование кластера.
Восстановление	После решения проблемы все компоненты реального времени/приложения/заданий/работы (realtime/app/job) попробуют вновь установить соединение с компонентом rabbitmq.

12.17 Аналитическая база данных

Поведение при полном отказе	Не работает поиск, аналитика, фильтры.
Деградированное поведение	Деградирована производительность многих функций, ведь используются более медленные запросы к реляционной БД.
Обеспечение отказоустойчивости	Возможно использование кластера.
Восстановление	После решения проблемы все компоненты приложения/заданий/работы (app/job) попробуют вновь установить соединение с Opensearch. Так как некоторые данные могут не быть реиндексированы после отказа, рекомендуется проводить полную реиндексацию после проблем.

ПРИЛОЖЕНИЕ 1. СПИСОК ПЕРЕМЕННЫХ ДЛЯ НАСТРОЙКИ ПРИЛОЖЕНИЯ

Название	Описание
Основные переменные	
ITRP_DOMAIN	Доменное имя, по которому будет доступна система
ITRP_API_SUBDOMAIN	Поддомен, по которому будет доступно REST API в системе. По умолчанию: api
ITRP_GRAPHQL_SUBDOMAIN	Поддомен, по которому будет доступно GraphQL API в системе. По умолчанию: graphql
ITRP_OAUTH_SUBDOMAIN	Поддомен, по которому будет доступны конечные точки (endpoints) OAuth v2 в системе. По умолчанию: oauth
ITRP_SHOW_QA_MESSAGE	Используйте «истина» (“true”), чтобы отображать баннер “Вы подключены к тестовой среде Р-Сервис”. По умолчанию: «ложь» (false)
ITRP_COOKIE_SECRET_TOKEN	Задайте как случайный токен* достаточной длины. **Токен - уникальный цифровой знак, идентификатор или единица учета.*
ITRP_OTP_SECRET	Задайте как случайный токен достаточной длины.
ITRP_OTP_SALT	Задайте как случайный токен достаточной длины.
ITRP_ERRORS_MAILBOX	Почтовый ящик, с которого будут отправляться уведомления об ошибках (HTTP 500)
ITRP_INFO_MAILBOX	Почтовый ящик, который будет получать отчеты об использовании каждую неделю и месяц
ITRP_NOREPLY_MAILBOX	Почтовый ящик, который будет использоваться как «от» (from) адрес, если ответ на письмо не предусмотрен.
ITRP_SUPPORT_MAILBOX	Используется в письмах регистрации как «от» (from) и «ответить-кому» (reply-to) адрес.
ITRP_INBOUND_MAILBOX	Ящик входящей почты. Используется для добавления комментариев к запросам, проблемам, релизам, изменениям, задачам и многому другому в системе .mailto:noreply@example.com
ITRP_BOUNCED_MAILBOX	Используется как обратный путь (return-path) во всех письмах.
ITRP_ASSESTS_HOSTS	Список поддоменов, разделенный запятой, по адресам которых возможно получить ресурсы приложения (статические файлы – CSS, JavaScript, изображения и шрифты). Несколько поддоменов позволяют браузерам получать несколько ресурсов одновременно.
ITRP_SSL	Всегда «истина» (true). Показывает то, что приложение доступно только по https.
ITRP_DEFAULT_SUPPORT_URL	URL, который будет указан в письмах регистрации.
ITRP_SUPPORT_ACCOUNT_ID	Идентификатор Пространства, которое будет использоваться как саппорт*-пространство в среде. Именно с этого аккаунта будет доступна

	/саппорт консоль. *Саппорт, от англ. Support – обеспечение помощи в решении проблем.
ITRP_INBOUND_FORWARDED_TO_ATTRIBUT E	Как описано в соответствующей статье базы знаний, оригинальный «ответить-кому» (reply-to) ящик, который включает директиву +<account_name>, должен быть доступен, если письмо переслано в ящик входящей почты. Используйте эту переменную для того, чтобы задать имя нового заголовка (header).
ITRP_INBOUND_ALLOW_BLANK_RETURN_P ATH	Письма без обратного пути (return-path) оцениваются как спам и игнорируются по умолчанию. Задайте это значение в «истина» (“true”) если обратный путь (return-path) очищается при передаче писем.
ITRP_INBOUND_CATCH_ALL	Задайте «истина» (“true”), если используется общий ящик с маршрутизацией по домену (catch-all) для обработки входящих писем. Если эта переменная «ложь» (false), то тогда приложение будет использовать имя сайта (“+sitename”) для установки соответствия пространства к письму.
ITRP_INBOUND_EMAIL_ERRORS_ACCOUNT	Имя сайта (Sitename) пространства, где будут храниться ошибки приема входящих писем.
ITRP_SECURE_COOKIES	Задайте «ложь» (false), если вход не работает. Некоторые балансировщики не передают зашифрованные куки* клиентам. *Куки, от англ. Cookie - небольшие файлы, которые веб-сайты сохраняют на устройстве пользователя (в браузере) для запоминания информации о нем.
ITRP_SAML_DESTINATION	Задайте «ложь» (false), чтобы удалить свойство Destination в samlp:AuthnRequest. Это иногда необходимо для того, чтобы предотвратить циклическое перенаправление (redirect-loop) в AD FS 2.1.
ITRP_TRUSTED_PROXIES	Укажите через запятую список доверенных прокси-клиентов, которые получают доступ к приложению из приватной подсети (к примеру, 172.16.0.0/12)
ITRP_PROXY_HOST	Адрес прокси-сервера
ITRP_PROXY_PORT	Порт прокси-сервера
ITRP_NO_PROXY_HOSTS	Разделенный запятой список адресов, которые не должны использовать прокси.
ITRP_ALLOWED_VIDEO_DOMAINS	Разделенный запятой список доменов, с которых возможно добавлять видео. По умолчанию: www.youtube-nocookie.com,player.vimeo.com
Короткие ссылки	
ITRP_SHORT_URL_DOMAIN	Поддомен для сервиса коротких ссылок. По умолчанию: io.ITRP_DOMAIN
ITRP_SHORT_URL_SCHEME	Протокол, который используется в коротких ссылках. По умолчанию: https://

ITRP_SHORT_URL_MAX_TOKENS	Максимальное количество коротких ссылок, которое может быть создано на пространство. По умолчанию: 1000000
ITRP_SHORT_URL_MAX_RESERVED	Максимальное количество коротких ссылок, которое может быть зарезервировано на пространство. По умолчанию: 10000
Хранение файлов	
STORAGE_MAX_FILESIZE	Максимальный размер файла, который может быть загружен в мегабайтах. По умолчанию: 20
STORAGE_LOCAL_SECRET_KEY	Задайте как случайный токен достаточной длины.
База данных	
DB_WRITER_HOST	Адрес основного сервера базы данных.
DB_WRITER_PORT	Порт основного сервера базы данных.
DB_WRITER_USERNAME	Имя пользователя сервера базы данных, под которым будет пытаться авторизоваться приложение. Убедитесь, что у данного пользователя есть права на создание базы данных/схемы при первичной инсталляции.
DB_WRITER_PASSWORD	Пароль сервера базы данных.
DB_WRITER_DATABASE	Имя базы данных/схемы.
DB_READER_HOST	Адрес вторичного сервера базы данных. Используется для более требовательных к чтению (read-требовательных) задач: аналитики, экспортов и многого другого.
DB_READER_PORT	Порт вторичного сервера базы данных.
DB_READER_USERNAME	Имя пользователя сервера базы данных.
DB_READER_PASSWORD	Пароль сервера базы данных.
DB_READER_DATABASE	Имя базы данных/схемы.
DB_MAX_EXECUTION_TIME_SUPPORTED	Используйте «истина» («true»), если сервер базы данных поддерживает системную переменную max_execution_time. По умолчанию: истина (true).
DB_SSL_CA_CERT	Задайте в «» (пустая строка) когда сервер базы данных не работает с уровнем защищенных слоев (SSL).
DB_SSL_CA_PATH	Задайте в «» (пустая строка) когда сервер базы данных не работает с уровнем защищенных слоев (SSL).
DB_SSL_CIPHER	Укажите предпочитаемый алгоритм для шифрования уровня защищенных слоев (SSL-шифрования). По умолчанию: DHE-RSA-AES256-SHA

DB_SSL_MODE	Используйте «Отключено» (“DISABLED”) для того, чтобы выключить проверку сертификата уровня защищенных слоев (SSL-сертификата). Доступные значения: • Отключено (DISABLED) • Предпочтительно (PREFERRED) • Требуется (REQUIRED) (не проверяет, но требует – используйте для самоподписанных сертификатов) • Проверка центра сертификации (VERIFY_CA) • Проверка идентификации (VERIFY_IDENTITY) По умолчанию: Проверка идентификации (VERIFY_IDENTITY)
DB_TLS_CIPHERSUITES	Наборы шифров, которые допустимы для зашифрованных соединений, использующих TLSv1.3
DB_TLS_MIN_VERSION	Минимальная допустимая версия протокола TLS*. Доступные значения: • 1 для TLSv1 • 2 для TLSv1.1 • 3 для TLSv1.2 • 4 для TLSv1.3 По умолчанию: 3 ** TLS (от англ.Transport Layer Security — безопасность транспортного уровня) — это криптографический протокол, обеспечивающий конфиденциальность, целостность и аутентификацию данных при обмене между узлами в сети Интернет.*
DB_TLS_MAX_VERSION	Максимальная допустимая версия протокола TLS.
Кэширование (Memcached)	
MEMCACHED_HOST	Адрес сервера кэширования (Memcached); обычно – адрес сервера заданий/работы (job), на котором работает кэширование (memcached)
MEMCACHED_MEMORY	Максимальное количество памяти, которое можно использовать для хранения объектов. По умолчанию: 512
MEMCACHED_SECRET_TOKEN	Задайте как случайный токен достаточной длины.
Redis	
REDIS_HOST	Адрес сервера Redis; обычно – адрес Сервера реального времени (realtime), на котором работает redis.
REDIS_PORT	Порт для подключения к redis; обычно 6379
REDIS_PASSWORD	Задайте пароль, если ваш сервер Redis требует его для подключения.
Поиск (OpenSearch)	
ELASTICSEARCH_URLS	Адреса поисковых серверов, разделенных запятой. Могут включать протокол и порт (как пример https://search1.internal:1234,https://search2.internal:1234)

Clacks (сервис обработки входящей почты)	
CLACKS_ADDRESS	Адрес сервера сетевых протоколов IMAP(S)
CLACKS_PORT	Порт сервера; обычно 143 или 993
CLACKS_USERNAME	Имя пользователя.
CLACKS_PASSWORD	Пароль.
CLACKS_ENABLE_SSL	Используйте «ложь» (“false”), если используется сетевой протокол IMAP вместо сетевого протокола IMAPS.
CLACKS_MAILBOX	Папка, которую необходимо проверять на наличие входящих писем. По умолчанию: входящие (INBOX)
CLACKS_ARCHIVEBOX	Папка, куда будут перемещаться обработанные сообщения, к примеру архивированные (ARCHIVE). Важно сохранять размер папки входящих (INBOX) маленьким - сетевой протокол IMAP начинает тормозить, если в одной папке много писем.
CLACKS_DELETE_AFTER_FIND	Используйте «ложь» (“false”), чтобы оставлять все письма во входящих (INBOX). Только для отладки.
Функциональность реального времени	
RABBIT_MQ_HOST	Адрес RabbitMQ; обычно адрес сервера реального времени (realtime)
FAYE_URL	Публичный унифицированный указатель ресурса (url) для сервера реального времени (realtime). По умолчанию: https://realtime.ITRP_DOMAIN
FAYE_SECRET_TOKEN	Задайте как случайный токен достаточной длины.
FAYE_REDIS_HOST	Адрес Redis; обычно - адрес сервера реального времени (realtime), на котором работает redis.
Генерация PDF	
PDFGENERATOR_URL	Адрес сервиса генерации pdf. Обычно - https://pdf (используется внутренняя сеть docker). По умолчанию: https://pdf
Почта - метод аутентификации электронной почты с добавлением цифровой подписи к письму (DKIM)	
MAIL_DKIM_ENABLED	Используйте «истина» (“true”), чтобы включить DKIM аутентификацию. По умолчанию: «ложь» (false)
MAIL_DKIM_DOMAIN	Домен для DKIM. По умолчанию: ITRP_DOMAIN
MAIL_DKIM_SELECTOR	Переключатель (Selector), добавляемый к домену, используется для поиска информации о публичном ключе DKIM. По умолчанию: по умолчанию (default)

MAIL_DKIM_PRIVATE_KEY	Приватный ключ DKIM, который используется для генерации TXT-записей.
	Чтобы сгенерировать пару публичный/приватный ключ, которая может быть использована как значение MAIL_DKIM_PRIVATE_KEY:
	$ openssl genrsa -out dkim.private.key 2048
	$ openssl rsa -in dkim.private.key -out dkim.public.key -pubout -outform PEM

Остальное	
GOOGLE_MAPS_JAVASCRIPT_API_KEY	Используйте ваш API-ключ для Google Maps. Требуется для использования приложения Google Maps из магазина приложений.
RSERVICE_SMTP_HOST	Адрес сервера простого протокола передачи почты (SMTP).
RSERVICE_SMTP_PORT	Порт сервера простого протокола передачи почты (SMTP).
RSERVICE_SMTP_DOMAIN	Домен, используемый в команде-приветствии простого протокола передачи почты (SMTP HELO).
RSERVICE_SMTP_USERNAME	Имя пользователя для авторизации в простом протоколе передачи почты (SMTP).
RSERVICE_SMTP_PASSWORD	Пароль для авторизации в простом протоколе передачи почты (SMTP).
RSERVICE_SMTP_AUTHENTICATION	Тип авторизации в простом протоколе передачи почты (SMTP) (plain, login (по умолчанию) или cram_md5).
RSERVICE_SMTP_OPENSSL_VERIFY_MODE	При использовании TLS определяет, как OpenSSL будет верифицировать сертификаты. Может быть одной из проверочных констант OpenSSL, нет (none) или пир (peer).