Руководство системного администратора R-Service.

Обзор решения RR Tech Service Management.

R-Service

r-service.tech

СОДЕРЖАНИЕ

1. ОБЗОР ПРОДУКТА

RR Tech Service Management – это современная ITSM/ESM система с поддержкой SIAM.

Система работает по модель клиент-сервер. Для подключения к системе не требуется специальное ПО, ведь используются веб-браузеры.

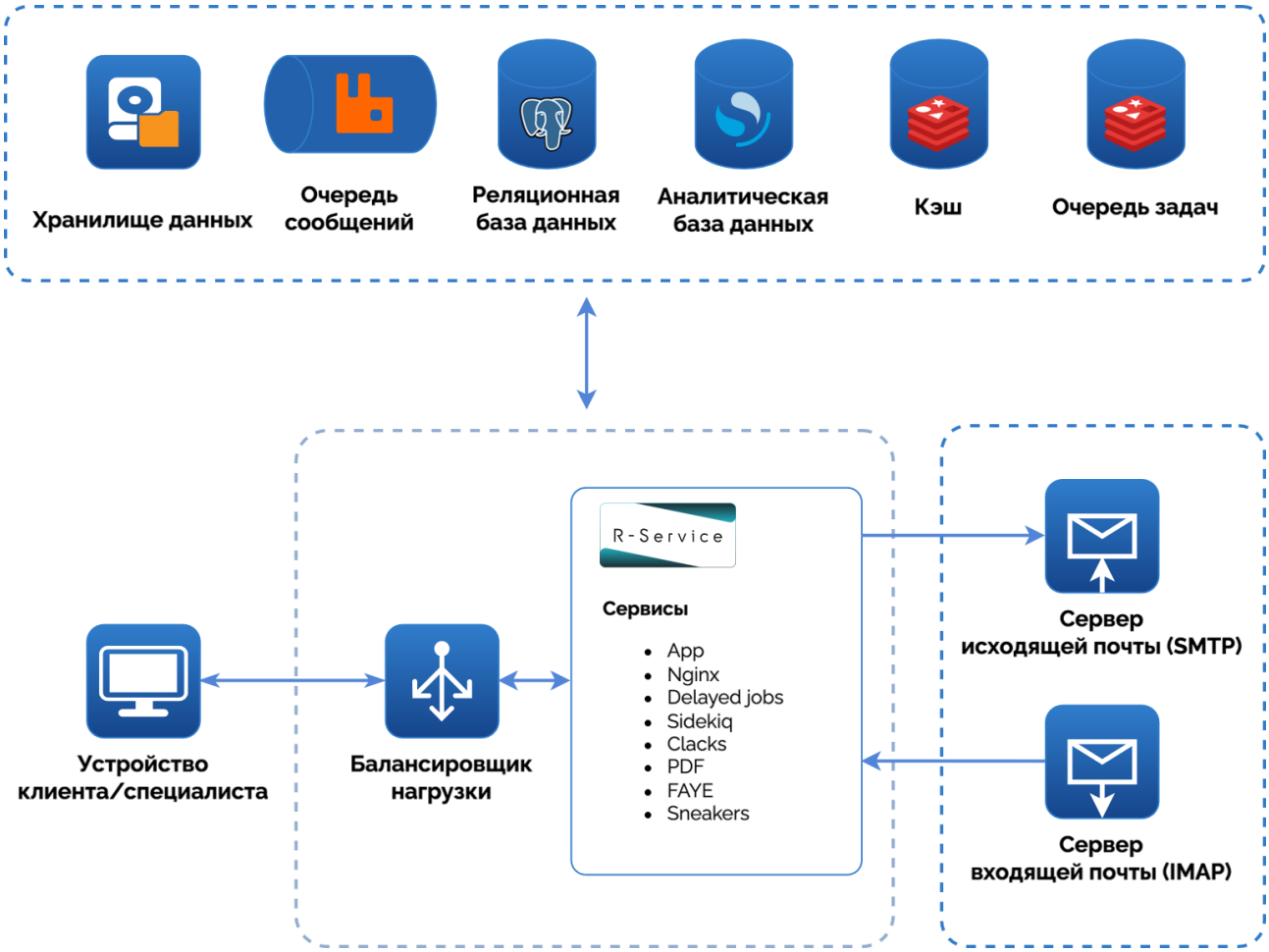
В основе концепции продукта лежит понятие мультиарендности (multitenancy) – возможности изоляции данных для разных клиентов в рамках одной инсталляции. Система RR Tech Service Management развивает данный концепт, позволяя устанавливать доверие между разными пространствами[1], делиться сущностями и перенаправлять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

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

2. ОБЗОР АРХИТЕКТУРЫ

Система RR Tech Service Management 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 репликами серверов ролей app и realtime, а также выполняет SSL-терминацию.

2.2. Реляционная база данных

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

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 job, которая реагирует на отложенные задачи, формируемые при каждом изменении данных в PostgreSQL. Роль 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, SMB, 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 app и job.

2.5. Сервер исходящей почты

Для отправки исходящих сообщений система интегрируется с почтовым сервером посредством протокола SMTP.

2.6. Сервер входящей почты

Для приема входящих писем система использует протокол IMAP.

2.7. Сервер развертывания

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

Теперь рассмотрим выделенные компоненты системы. Все компоненты системы можно поделить на три основных роли: app, realtime и job. Рассмотрим, что делает каждая из ролей и какие компоненты на ней выполняются.

2.8. 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 с сетевого хранилища.

Компонент	Описание
app	Основной компонент системы. Принимает запросы от компонента nginx, генерирует страницы и отдает их через nginx обратно клиенту.
nginx[2]	Sidecar для app. Всегда работает вкупе с app. Предоставляет статические файлы и вложения. Проксирует запрос в app, если это не запрос на статический файл или вложение

2.9. 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 QoS 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 IMAP.
memcached	Сервис кэширования.

2.10. realtime

На данной роли выполняются компоненты, необходимые для работы функциональности реального времени[3]. Данная роль отвечает за функцию отправки изменений в реальном времени клиентам, детекцию коллизий и отправку нотификаций.

Компонент	Описание
faye	Обработчик соединений реального времени. Данный сервис принимает запрос на установку сессии от браузера, и при возможности обновляет сессию до 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 app/job ролей информацию об изменении данных и уведомлениях.
redis	Сервис Key/Value хранилища. Используется realtime ролью для хранения сессий и app/job ролями для инкрементных значений и distributed locking

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

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

3. ДАННЫЕ

Следующие данные хранятся в разных компонентах системы.

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

4. ДОМЕНЫ И ПОДДОМЕНЫ

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 wildcard SSL сертификаты. Такой SSL сертификат выпускается сразу на все поддомены определенного домена;
- использовать wildcard DNS. Многие сервера системы доменных имен поддерживают 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 rr-tech и 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
graphql	Поддомен для GraphQL API
oauth	Поддомен для OAuth API
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[4]
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:

- Продуктив – r-service.example.com
- QA – r-service-qa.example.com
- Development – r-service-dev.example.com

[4] http/1.1 загружает файлы друг за другом в рамках одного поддомена. http/2.0 поддерживает мультиплексирование, 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.

Environment переменная	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 from и reply-to адрес
ITRP_NOREPLY_MAILBOX	noreply@example.com	Ящик, который будет использоваться как 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. Catch-all (общий ящик с маршрутизацией по домену)
2. Subaddressing (“один ящик, маршрутизация через +”)

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 Catch-all

Принцип работы:

Catch-all — это механизм, при котором **все письма, отправленные на любой адрес определённого домена**, перенаправляются в один общий ящик. Например, если настроен 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

Создаётся **один почтовый ящик**, к которому система подключается по **IMAP**.

На почтовом сервере настраивается **catch-all для поддомена** mail, который перенаправляет все письма на этот ящик.

Система, анализируя заголовок **To**, определяет конкретный сервис или пространство, куда относится письмо.

Преимущества:

1. Простота настройки для множества адресов.
2. Работает даже если почтовый сервер не поддерживает subaddressing.

Недостатки:

1. Не всегда подходит для On-Premise инсталляций

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

Принцип работы:

Subaddressing (или "плюсовая адресация") — 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 **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 **IMAP**. Пример: rs@r-service.tech.
2. Если почтовый сервер **поддерживает 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 subaddressing**:

- Нужно настроить алиасы для адресов с +, чтобы они перенаправлялись на основной ящик.

4. Для **On-Premise Exchange** рекомендуется:

- создать **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 On-Premise инсталляций).

Недостатки:

1. Требует поддержки subaddressing со стороны почтового сервера.
2. Для некоторых On-Premise решений (например, Exchange без modern mode) может потребоваться ручная настройка alias или транспортных правил.

6. НАСТРОЙКА СИСТЕМЫ

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

- .env – общий для всех серверов;
- .env.web – переменные для серверов роли app;
- .env.job – переменные для серверов роли job;
- .env.realtime – переменные для серверов роли realtime.

Список всех переменных доступен в Приложении 1.

7. МЕХАНИЗМЫ РАЗВЕРТЫВАНИЯ

Система R-Service 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 rolling updates.

7.1. Docker Swarm или Docker Compose

В данном механизме развертывания используется предопределенный набор серверов, четко поделенных на роли (как описано в **Обзор архитектуры**). Для оркестрации 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. 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.

8. САЙЗИНГ 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:

- Продуктив;
- QA.

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

Также возможно использовать больше сред (к примеру, продуктив, qa, test и development), но обычно это не требуется.

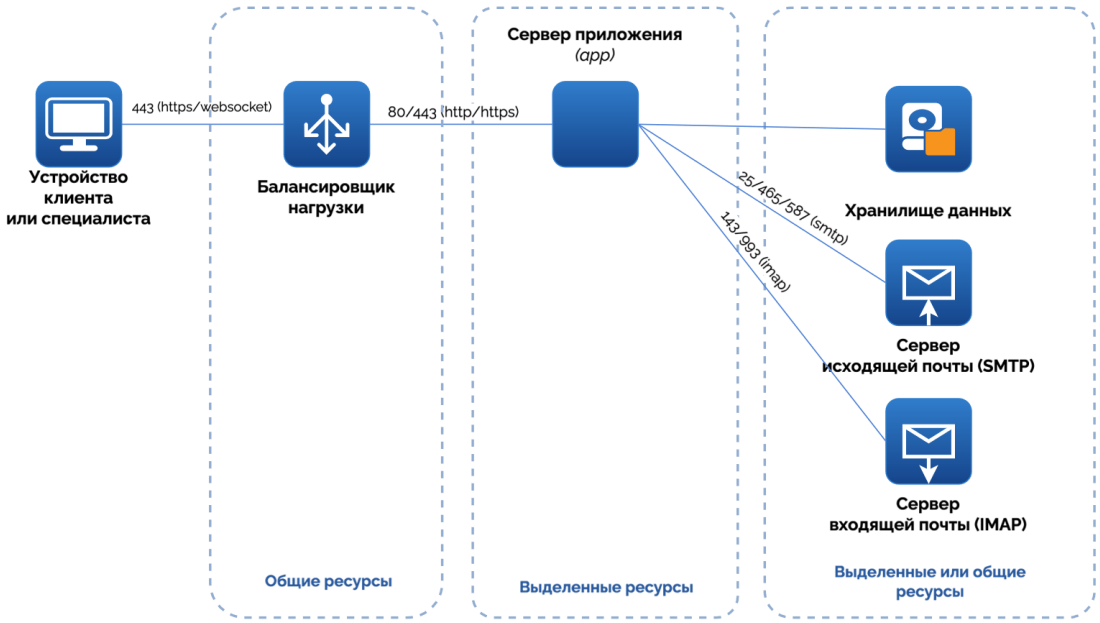
Данная секция по большей части применима для механизма развертывания Docker Swarm, но может являться отправной точкой для планирования требуемых ресурсов в кластере Kubernetes.

8.1. Пилотная/тестовая инсталляция

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

Роль	Имя	CPU	RAM	Storage
app	APP1	8 (100%)	32GB	64 GB (ssd), 80 GB (HDD)

⚠ Важно! Только для использования в рамках пилота/тестирования



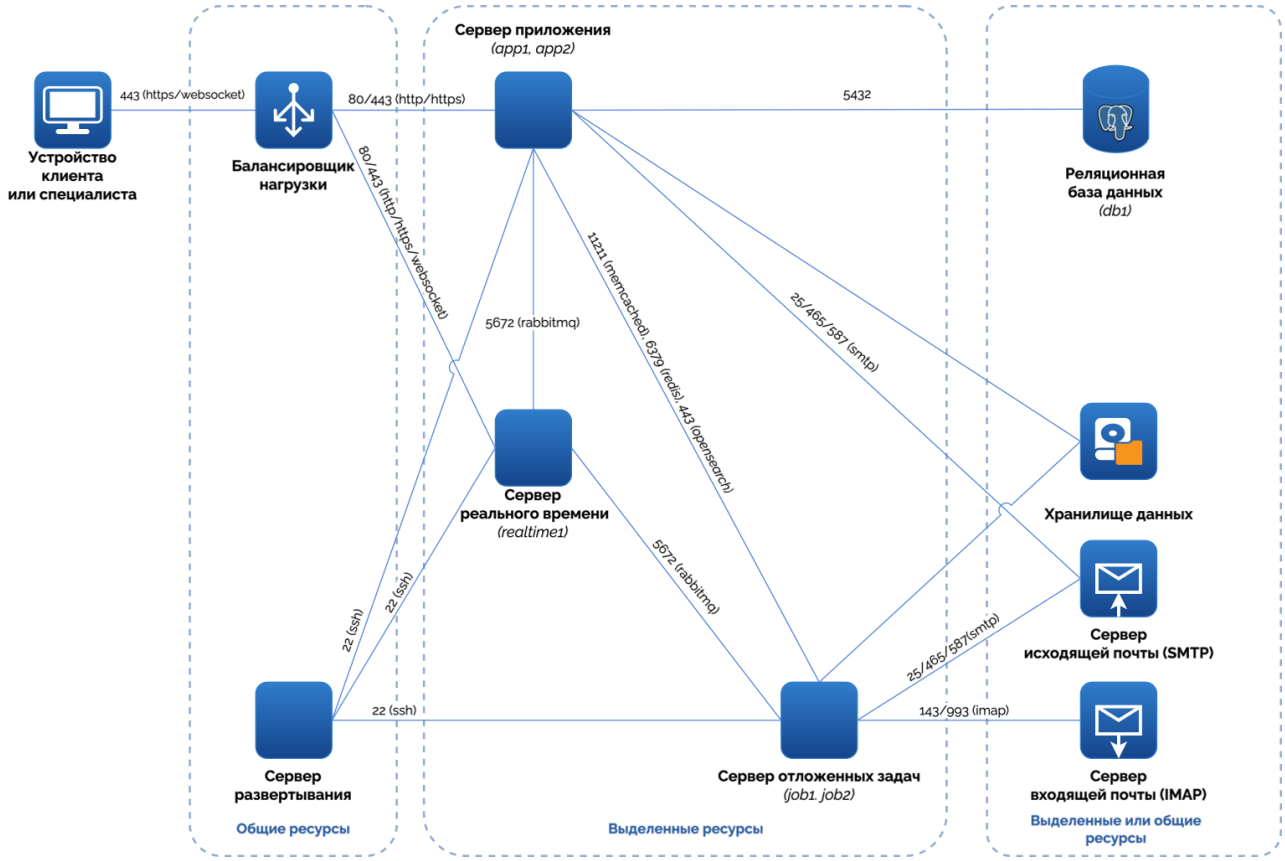
8.2. Маленькая инсталляция (примерно 500 специалистов, 8400 запросов в день)

В данной конфигурации обеспечивается отказоустойчивость ролей app и job. Для экономии ресурсов аналитическая база данных работает на одном из серверов роли job.

Не обеспечивается отказоустойчивость сервера реляционной базы данных. При необходимости, возможно построение кластера.

Подробнее про кластеризацию и отказоустойчивость (см. в Ошибка! Источник ссылки не найден.).

Роль	Имя	CPU	RAM	Storage
app	APP1	2 (100%)	4GB	30 GB (HDD)
app	APP2	2 (100%)	4GB	30 GB (HDD)
job	JOB1	4 (100%)	16GB	30 GB (HDD), 50 GB (SSD, OpenSearch storage)
job	JOB2	4 (100%)	16GB	30 GB (HDD)
realtime	REALTIME1	2 (100%)	4GB	30 GB (HDD)
	DB1	4 (100%)	16GB	30 GB (HDD, Boot) 50 GB (SSD, DB Storage)
	DEPLOYMENT[5]	1 (25%)	2GB	30 GB (HDD, Boot)



[5] Сервер deployment общий для всех сред

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

В данной конфигурации обеспечивается отказоустойчивость ролей app и job, а также реляционной базы данных.

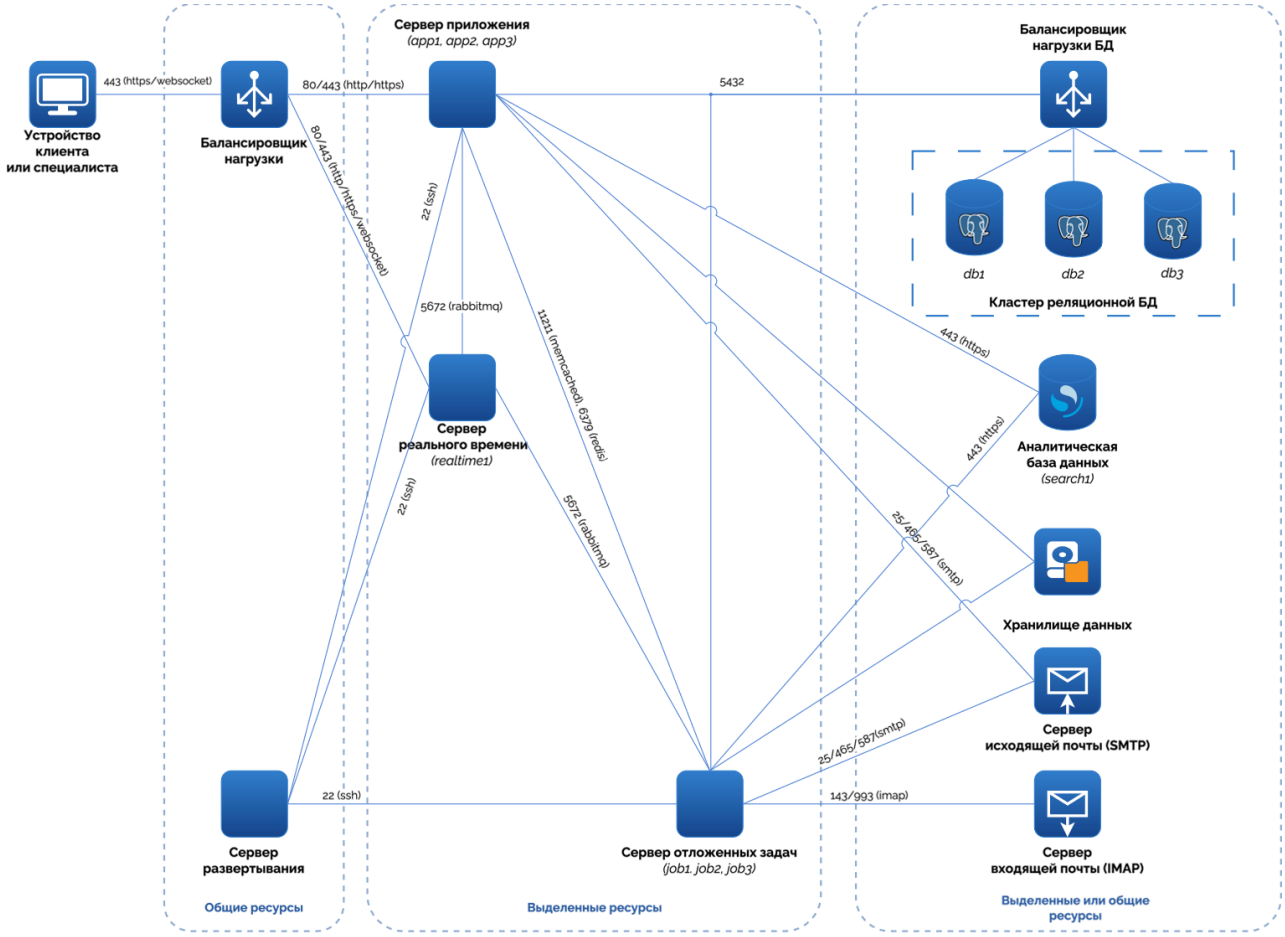
Кластер реляционной базы данных рекомендуется организовывать средствами patroni. Также рекомендуется использовать etcd, размещенный на каждой из нод сервера БД. Таким образом, при отказе любой из нод сработает механизм failover, ведь даже с двумя нодами соберется кворум в etcd.

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

Подробнее про кластеризацию и отказоустойчивость (см. в Ошибка! Источник ссылки не найден.).

Роль	Имя	CPU	RAM	Storage
app	APP1	2 (100%)	4GB	30 GB (HDD)
app	APP2	2 (100%)	4GB	30 GB (HDD)
app	APP3	2 (100%)	4GB	30 GB (HDD)
job	JOB1	4 (100%)	16GB	30 GB (HDD)
job	JOB2	4 (100%)	16GB	30 GB (HDD)
job	JOB3	4 (100%)	16GB	30 GB (HDD)
realtime	REALTIME1	2 (100%)	4GB	30 GB (HDD)
	DB1	4 (100%)	16GB	30 GB (HDD, Boot) 200 GB (SSD, DB Storage)
	DB2	4 (100%)	16GB	30 GB (HDD, Boot) 200 GB (SSD, DB Storage)
	DB3	4 (100%)	16GB	30 GB (HDD, Boot) 200 GB (SSD, DB Storage)
	SEARCH1	4 (100%)	16GB	30 GB (HDD, Boot) 200 GB (SSD, DB Storage)
	DEPLOYMENT[6]	1 (25%)	2GB	30 GB (HDD, Boot)

[6] Сервер deployment общий для всех сред



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

В данной конфигурации обеспечивается отказоустойчивость ролей app, job, realtime, а также реляционной и аналитической базы данных.

Для масштабирования серверов app роли используется 4 воркера в компоненте app, вместо стандартных двух.

Вспомогательные сервисы также собраны в кластеры:

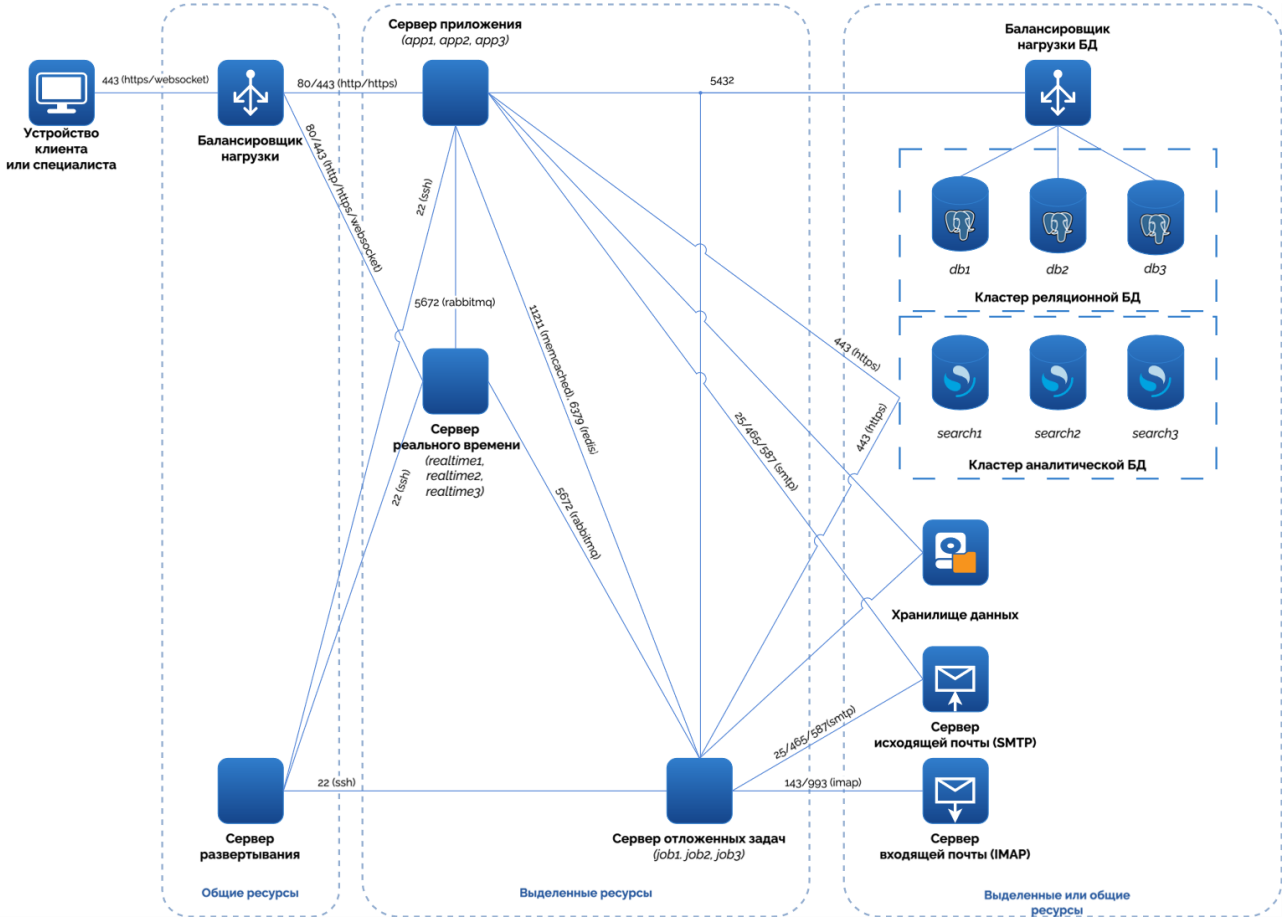
- RabbitMQ работает в режиме quorum queues на трех серверах realtime;
- Redis работает в sentinel режиме на трех серверах job.

Кластер реляционной базы данных рекомендуется организовывать средствами patroni. Также рекомендуется использовать etcd, размещенный на каждой из нод сервера БД. Таким образом, при отказе любой из нод сработает механизм failover, ведь с двумя нодами соберется кворум в etcd.

Подробнее про кластеризацию и отказоустойчивость (см. в Ошибка! Источник ссылки не найден.).

Роль	Имя	CPU	RAM	Storage
app	APP1	4 (100%)	6GB	30 GB (HDD)
app	APP2	4 (100%)	6GB	30 GB (HDD)
app	APP3	4 (100%)	6GB	30 GB (HDD)
job	JOB1	4 (100%)	16GB	30 GB (HDD)
job	JOB2	4 (100%)	16GB	30 GB (HDD)
job	JOB3	4 (100%)	16GB	30 GB (HDD)
realtime	REALTIME1	2 (100%)	4GB	30 GB (HDD)

	DB1	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	DB2	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	DB3	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	SEARCH1	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	SEARCH2	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	SEARCH3	4 (100%)	16GB	30 GB (HDD, Boot) 500 GB (SSD, DB Storage)
	DEPLOYMENT[7]	1 (25%)	2GB	30 GB (HDD, Boot)



[7] Сервер deployment общий для всех сред

8.5. Требования к оборудованию

1. Необходимо обеспечить каналы связи шириной не менее 1 Gbit/s между всеми хостами.
2. Время на всех хостах должно быть одинаковым и должно синхронизироваться из единого источника

9. ТРЕБОВАНИЯ К ПО

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 app, job и realtime должен быть установлен Docker Engine и Docker Compose v2.

Если вы используете стандартный 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 (soft limit)	8192
Все	Максимальное количество файловых дескрипторов на процесс (hard limit)	65536
Realtime	fs.file-max	2097152
Realtime	Максимальное количество файловых дескрипторов на процесс (hard limit)	100000
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 patroni и etcd. Рекомендуемая конфигурация - минимум три ноды. С тремя нодами 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 ноды. С тремя нодами обеспечивается кворум.

9.7. Балансировщик

Для правильной работы системы следующие http заголовки должны присутствовать в запросе, переданном от балансировщике в компонент роли 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 app возможно использовать следующие endpoint для active health check:

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

Используйте данный endpoint только тогда, когда ваш балансировщик умеет проверять тело ответа, а не только код ответа.

Для проверки здоровья компонентов роли realtime возможно использовать следующие endpoint для active health check:

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

Используйте данный endpoint только тогда, когда ваш балансировщик умеет проверять тело ответа, а не только код ответа.

10. ТРЕБОВАНИЯ К КВАЛИФИКАЦИИ СИСТЕМНОГО АДМИНИСТРАТОРА СИСТЕМЫ

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

1. Администрирование ОС 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, базовое понимание работы переменных в 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 R-Service или пройденный вводный курс специалиста.

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 ВМ и контейнеров;
- понимание работы протокола OpenTelemetry.

2. Базы данных:

- понимание работы механизмов репликации в БД PostgreSQL;
- опыт развертывания кластеров БД PostgreSQL;
- 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 R-Service".

11. РЕЗЕРВНОЕ КОПИРОВАНИЕ

Резервные копии необходимо хранить на внешних, не относящихся к инсталляции ресурсах. 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.

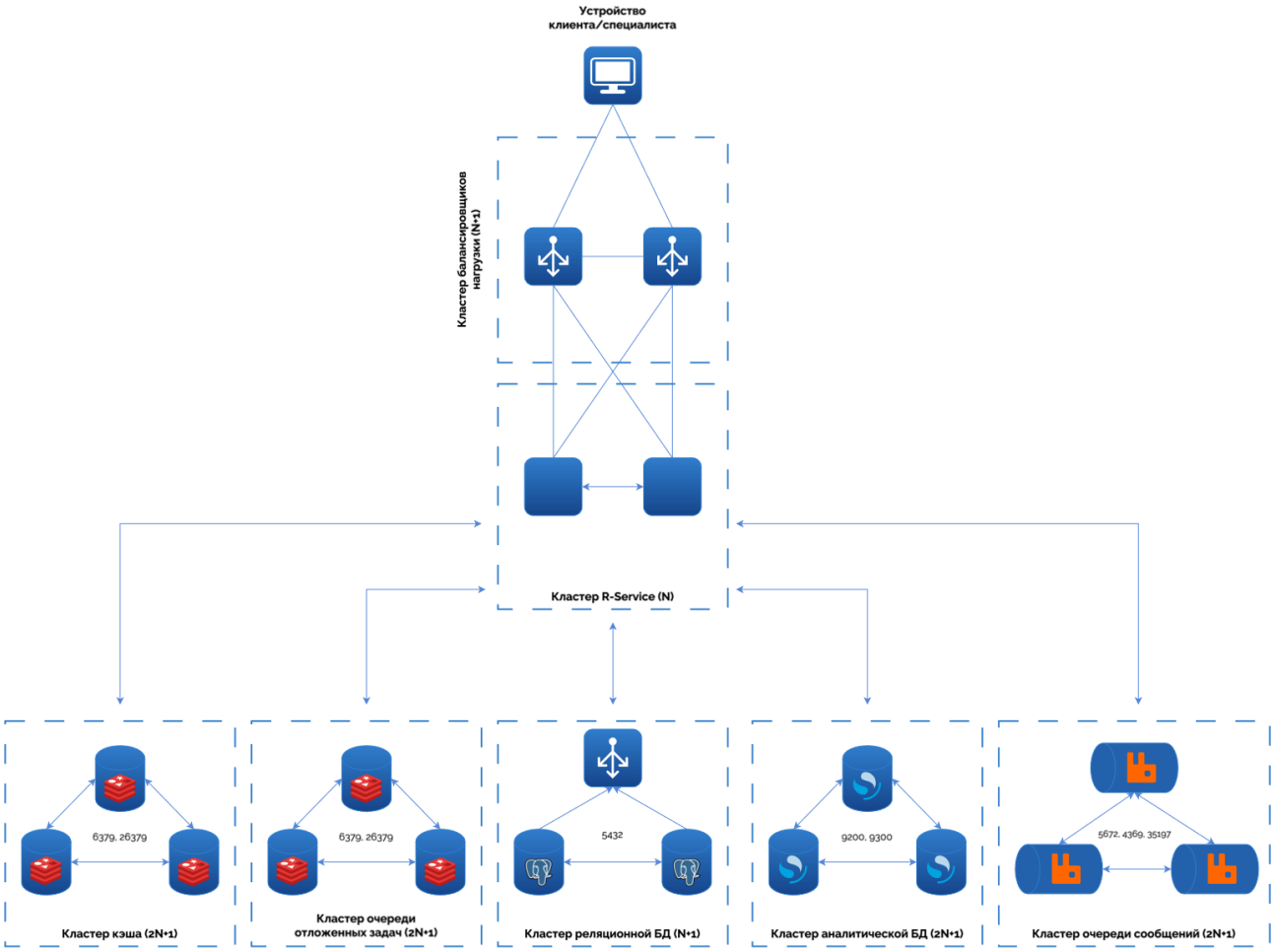
12. ОТКАЗОУСТОЙЧИВОСТЬ И МАСШТАБИРОВАНИЕ

Система разработана с учетом отказоустойчивости.

Система будет отвечать на запросы пользователей имея лишь один сервер компонента 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 Master-Replica(s). Один сервер избирается главным и отправляет все изменения на остальные сервера.

Для реализации кластера PostgreSQL рекомендуется использование решения patroni . В качестве хранилища рекомендуется использование etcd, установленного на каждой из нод БД.

Желательно использование connection pooler, установленных на каждой из нод, таких как pgbouncer.

В качестве балансировщика рекомендуется использование HAProxy.

Для обеспечения полноценного кворума рекомендуется использование нечетного количества нод.

Минимальное количество серверов для организации кластера: три сервера.

12.2. Аналитическая база данных

Аналитическая база данных OpenSearch является распределенной базой данных и использует шардирование. Каждый из серверов хранит свою копию данных, что обеспечивает распределение чтения/записи между разными нодами 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. quorum queues, где из сервера образуют кворум. Кворум – соглашение между большинством нод 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 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 Sentiniel. Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

12.5. Очередь отложенных задач

СУБД Redis возможно объединить в кластер 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 Sentiniel. Подробнее в официальной документации.

Минимальное количество серверов для организации кластера: три сервера.

12.6. Приложение R-Service

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

Исключением является 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 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 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 health check, пользователи не должны заметить ошибок.
Обеспечение отказоустойчивости	Возможна 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: • urgent – задачи для рабочих процессов не будут создаваться. • fast – изменения не будут переиндексированы в аналитической БД. Не будут отправлены вебхуки. Возможны расхождения данных, отсутствие новых сущностей в табличных представлениях. • normal – некоторые уведомления и письма не будут отправлены. Многие фоновые вычисления не будут выполнены. • slow – импорты и экспорты не будут выполнены • periodic – переодические задачи не будут выполнены. • notifications – уведомления и письма не будут отправлены.
Обеспечение отказоустойчивости	Возможна и рекомендуется использование нескольких реплик для каждой из очередей.

Восстановление	Каждый из воркеров автоматически начнет обрабатывать очередь.

12.10. Компонент clacks

Поведение при полном отказе	PDF файлы для согласований и экспорта дашбордов 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 дашбордов 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 websocket соединения от балансировщика не устанавливаются. Функциональность реального времени не работает.
Деградированное поведение	Производительность под нагрузкой снижена. Если балансировщик правильно обрабатывает 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 детекция 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 app.
Деградированное поведение	Уменьшение производительности компонента app.
Обеспечение отказоустойчивости	Невозможно.
Восстановление	После решения проблемы компонент 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 детекция 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 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 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 OAuth v2 endpoints в системе. По умолчанию: oauth
ITRP_SHOW_QA_MESSAGE	Используйте “true”, чтобы отображать баннер “Вы подключены к QA среде R-Service”. По умолчанию: false
ITRP_COOKIE_SECRET_TOKEN	Задайте как случайный токен достаточной длинны
ITRP_OTP_SECRET	Задайте как случайный токен достаточной длинны
ITRP_OTP_SALT	Задайте как случайный токен достаточной длинны
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 from адрес, если ответ на письмо не предусмотрен
ITRP_SUPPORT_MAILBOX	Используется в письмах регистрации как from и 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 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 true. Показывает то, что приложение доступно только по https.
ITRP_DEFAULT_SUPPORT_URL	URL, который будет указан в письмах регистрации
ITRP_SUPPORT_ACCOUNT_ID	ID Пространства, которое будет использоваться как support-пространство в среде. Именно с этого аккаунта будет доступна /support консоль
ITRP_INBOUND_FORWARDED_TO_ATTRIBUTE	Как описано в соответствующей статье БЗ, оригинальный 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 header
ITRP_INBOUND_ALLOW_BLANK_RETURN_PATH	Письма без return-path оцениваются как спам и игнорируются по умолчанию. Задайте это значение в “true” если return-path очищается при передаче писем.
ITRP_INBOUND_CATCH_ALL	Задайте в “true” когда используется catch-all ящик для обработки входящих писем. Если эта переменная false – то тогда приложение будет

	использовать “+sitename” для установки соответствия пространства к письму
ITRP_INBOUND_EMAIL_ERRORS_ACCOUNT	Sitename пространства, где будут хранится ошибки приема входящих писем
ITRP_SECURE_COOKIES	Задайте в «false» если вход не работает. Некоторые балансировщики не передают зашифрованный cookie клиентам.
ITRP_SAML_DESTINATION	Задайте в «false» чтобы удалить свойство Destination в samlp:AuthnRequest. Это иногда необходимо для такого, чтобы предовратить 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 проксироваться
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 длинны
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 read-требовательны 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 «true», если сервер базы данных поддерживает системную переменную max_execution_time. По умолчанию: true
DB_SSL_CA_CERT	Задайте в «» (пустая строка) когда сервер базы данных не работает с SSL
DB_SSL_CA_PATH	Задайте в «» (пустая строка) когда сервер базы данных не работает с SSL
DB_SSL_CIPHER	Укажите предпочитаемый алгоритм для SSL-шифрования. По умолчанию: DHE-RSA-AES256-SHA
DB_SSL_MODE	Используйте “DISABLED” для того, чтобы выключить проверку SSL-сертификата. Доступные значения: • DISABLED • PREFERRED • REQUIRED (не проверяет, но требует – используйте для самоподписанных сертификатов) • VERIFY_CA • VERIFY_IDENTITY По умолчанию: 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
DB_TLS_MAX_VERSION	Максимальная допустимая версия протокола TLS
Кэширование (Memcached)	
MEMCACHED_HOST	Адрес сервера Memcached; обычно – адрес job-сервера, на котором работает memcached
MEMCACHED_MEMORY	Максимальное количество памяти, которое можно использовать для хранения объектов. По умолчанию: 512
MEMCACHED_SECRET_TOKEN	Задайте как случайный токен достаточной длинны
Redis	
REDIS_HOST	Адрес сервера Redis; обычно – адрес realtime-сервера, 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 IMAP(S) сервера
CLACKS_PORT	Порт сервера; обычно 143 или 993
CLACKS_USERNAME	Имя пользователя
CLACKS_PASSWORD	Пароль
CLACKS_ENABLE_SSL	Используйте “false”, если используется IMAP вместо IMAPS

CLACKS_MAILBOX	Папка, который необходимо проверять на наличие входящих писем. По умолчанию: INBOX
CLACKS_ARCHIVEBOX	Папка, куда будут перемещаться обработанные сообщения, к примеру ARCHIVE. Важно сохранять размер папки INBOX маленьким – IMAP начинает тормозить когда в одной папке много писем.
CLACKS_DELETE_AFTER_FIND	Используйте “false” чтобы оставлять все письмо в INBOX. Только для отладки
Функциональность реального времени	
RABBIT_MQ_HOST	Адрес RabbitMQ; обычно адрес realtime-сервера
FAYE_URL	Публичный url для realtime сервера. По умолчанию: https://realtime.ITRP_DOMAIN
FAYE_SECRET_TOKEN	Задайте как случайный токен достаточной длинны
FAYE_REDIS_HOST	Адрес Redis; обычно – адрес realtime-сервера, на котором работает redis
Генерация PDF	
PDFGENERATOR_URL	Адрес сервиса генерации pdf. Обычно – https://pdf (используется внутренняя сеть docker). По умолчанию: https://pdf
Почта - DKIM	
MAIL_DKIM_ENABLED	Используйте “true” чтобы включить DKIM аутентификацию. По умолчанию: false
MAIL_DKIM_DOMAIN	Домен для DKIM. По умолчанию: ITRP_DOMAIN
MAIL_DKIM_SELECTOR	Selector, добавляемый к домену, используется для поиска информации о публичном ключе DKIM. 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 SMTP сервера
RSERVICE_SMTP_PORT	Порт SMTP сервера
RSERVICE_SMTP_DOMAIN	Домен, используемый в SMTP HELO
RSERVICE_SMTP_USERNAME	Username для авторизации в SMTP
RSERVICE_SMTP_PASSWORD	Password для авторизации в SMTP
RSERVICE_SMTP_AUTHENTICATION	Тип авторизации в 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 verify констант OpenSSL, none или peer.

ПРИЛОЖЕНИЕ 2. СЕТЕВЫЕ ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТОВ СИСТЕМЫ.

Источник	Назначение	Порт (протокол)	Примечание
Устройство клиента/специалиста	Loadbalancer	443 (https/websocket)	
Loadbalancer	nginx	80/443 (http/https)	Рекомендуется использование 443 (https)
Loadbalancer	faye	443 (https/websocket)	
app	Реляционная БД	5672	
app	Аналитическая БД	9200 (https)	
app	Кэш (redis)	6379	
app	Очередь отложенных задач (redis)	6379	
Loadbalancer	faye	443 (https/websocket)	
app	Реляционная БД	5672	
app	Аналитическая БД	9200 (https)	
app	Кэш (redis)	6379	
app	Очередь сообщений (rabbitmq)	5672	
app	Сетевое хранилище		
app	SMTP сервер	25/587/465 (SMTP)	
delayed-job	Реляционная БД	5672	
delayed-job	Аналитическая БД	9200 (https)	
delayed-job	Кэш (redis)	6379	
delayed-job	Очередь отложенных задач (redis)	6379	
delayed-job	Очередь сообщений (rabbitmq)	5672	
delayed-job	pdf	443	
delayed-job	SMTP сервер	25/587/465 (SMTP)	
faye	Кэш (redis)	6379	
faye	Сетевое хранилище		
sneakers	faye	443	
sneakers	Очередь сообщений (rabbitmq)	5672	
clacks	Реляционная БД	5672	
clacks	Сетевое хранилище		
clacks	IMAP сервер	143/993 (IMAP/IMAPS)	