Эксплуатационная документация SimpleOne

Содержание

1. Примечания к обновлению . . . 4
2. Требования к квалификации специалиста . . . 12
3. Состав дистрибутива . . . 13
4. Описание типов экземпляров . . . 13
5. Архитектура . . . 14
   5.1. Высокодоступная архитектура . . . 14
   5.2. Простая (single-node) архитектура . . . 17
6. Подготовка инфраструктуры . . . 18
   6.1. Требования к серверам . . . 18
   6.2. Разбивка дисков . . . 20
   6.3. Требования к программной части . . . 22
   6.4. Требования к каналам связи . . . 23
   6.5. Требования к балансировщику . . . 25
   6.6. Таблица сетевых взаимодействий . . . 25
   6.7. Требования к внешней БД . . . 26
      6.7.1. Настройка внешней БД . . . 26
      6.7.2. Обновление внешней БД . . . 28
         Этап 1: Подготовка . . . 28
         Этап 2: Обновление PostgreSQL (выполняется заказчиком) . . . 29
         Этап 3: Восстановление . . . 29
         Этап 4: Запуск и валидация . . . 29
7. Установка ПО . . . 30
   7.1. Перед установкой . . . 30
   7.2. Установка платформы . . . 30
      7.2.1. Новый модуль SLA . . . 31
      7.2.2. Настройка конфигурационных файлов . . . 32
         Для высокодоступной архитектуры . . . 32
         Для простой архитектуры . . . 33
      7.2.3. Проверка корректности заполненного inventory . . . 33
      7.2.4. Запуск плейбука . . . 34
      7.2.5. Проверка работоспособности экземпляра . . . 34
   7.3. После установки . . . 35
      7.3.1. Сбор статистики БД и анализ нагрузки . . . 35
8. Обновление ПО . . . 37

8.1. Очередность обновления . . . 37
8.2. Обновление платформы . . . 37
8.3. Загрузка пакетов обновлений бизнес-решений . . . 38
8.3.1. Загрузка пакетов . . . 38
8.3.2. Подготовка пакета . . . 39
8.3.3. Импорт пакета . . . 39
8.3.4. Устранение коллизий . . . 40
9. Настройка поддержки ПО . . . 44
9.1. Контроль состояния экземпляра . . . 44
9.1.1. Мониторинг . . . 44
Мониторинг инфраструктуры . . . 44
Дашборды инфраструктурных компонентов . . . 45
Мониторинг приложения . . . 46
9.1.2. Трассировка . . . 46
9.1.3. Настройка системы сбора и анализа логов . . . 48
9.2. Резервное копирование и восстановление данных . . . 48
9.2.1. Рекомендации по резервному копированию . . . 48
Настройка резервного копирования . . . 48
Восстановление данных . . . 49
9.3. Миграция баз данных . . . 49
9.4. Очистка данных Apache Kafka . . . 50
9.4.1. Запуск очистки . . . 51
9.4.2. Запуск приложения после очистки . . . 52
9.5. Замена SSL-сертификатов . . . 52
9.6. Изменение URL-адреса . . . 52
9.7. Расширение ресурсов . . . 52
9.7.1. Увеличение ресурсов хостов . . . 53
9.7.2. Добавление новых воркеров . . . 53
9.8. Предоставление доступа внешним специалистам . . . 53
10. Приложения . . . 54
10.1. Приложение 1. Пример списка предустановленного ПО . . . 54
10.1.1. Рекомендуемые средства диагностики . . . 55
10.2. Приложение 2. Настройка переменных плейбука . . . 56
10.2.1. Заполнение переменных backend . . . 56
Настройка SSO . . . 56
Настройка воркеров . . . 56
10.2.2. Заполнение переменных monitoring . . . 60
10.2.3. Заполнение переменных system . . . 61
10.2.4. Заполнение переменных vars . . . 61
Настройка приложения . . . 61
Настройка хранилища S3 . . . 62

Настройка БД . . . 62
Настройка внешней БД . . . 62
Настройка RabbitMQ . . . 63
Настройка MinIO . . . 63
Настройка Kafka . . . 63
Настройка Jaeger . . . 64
Настройка backend . . . 64
Настройка PgBouncer . . . 65
Настройка экспорта . . . 65
Настройка SSL-сертификатов . . . 65
10.3. Приложение 3. Таблица сетевых взаимодействий компонентов с внешними системами . . . 66
10.4. Приложение 4. Высокодоступная схема, контур PROD . . . 78

1. Примечания к обновлению

Изменения, привносимые с новыми версиями платформы, должны отображаться в **inventory** обновляемых экземпляров.

Версия 1.32.0

- В этой версии добавлен механизм плавного завершения работы сервиса при получении сигналов `SIGTERM` или `SIGINT`, позволяющий завершить текущие задачи до остановки сервиса. Для корректной реализации плавного завершения в файле `docker-compose` должен быть указан параметр `stop_grace_period`, который определяет максимальное время, выделяемое на завершение работы контейнеров. Данный параметр должен быть больше 10 секунд и по умолчанию составляет 15 секунд.
- В описание воркеров по умолчанию добавлено описание воркера **worker_process_by_vcs_pack**.

Версия 1.31.0

- PostgreSQL обновлена до 17 версии.
  - В новой версии изменена структура таблицы **pg_stat_statements**, а именно удалено тело запроса, колонка query. В результате данного изменения нарушается работа метрики **pg_stat_statements_calls_total**. Для ее восстановления выполните действия, указанные ниже.
  - При использовании внешней СУБД, ее необходимо обновить до 17 версии. Инструкции по обновлению приведены в соответствующем разделе.

Обновления документации

- В разделе Обновление платформы:
  - Добавлено примечание о возможности обновления платформы без последовательного обновления через все промежуточные версии.
  - Добавлено примечание о необходимости убедиться в работоспособности экземпляра перед обновлением.
- В примечания по обновлению для версии 1.27.0 добавлено предупреждение о необходимости проверки очередей Kafka при обновлении.
- В примечания по обновлению для версии 1.29.0 добавлено уточнение по изменению числа узлов Redis.
- В разделе Настройка внешней БД обновлена требуемая версия PostgreSQL.
- Обновлены ссылки на списки доступных метрик в разделе Мониторинг инфраструктуры.

Восстановление метрик

Для возвращения к штатному отображению метрик, в частности метрики количества запросов в минуту, отсортированных по признаку типа запроса, после обновления на версию платформы 1.31.0 необходимо в Grafana настроить promql-запросы (касается только запросов с фильтрацией метрик по типу запроса к БД), поддерживающие новую структуру метрик (на примере фильтрации по признаку типа SELECT):

```
(sum(increase(pg_stat_statements_calls_total{instance=~"$instance"} and on(queryid) pg_stat_statements_query_id{query=~".SELECT."}[1m])) by (datname)
```



В случае, если на одном дашборде имеется возможность выводить метрики нескольких экземпляров, имеющих в составе комплекса PostgreSQL версии как <17, так и >=17, требуется выбрать подход к оформлению запросов promql, поддерживающих обратную совместимость при изменении структуры таблиц метрик в зависимости от установленной версии СУБД:

```
(sum(increase(pg_stat_statements_calls_total{instance=~"$instance"} and on(queryid) pg_stat_statements_query_id{query=~".SELECT."}[1m])) by (datname) and on() pg_static{short_version=~"1[7-9]."}) or (sum(increase(pg_stat_statements_calls_total{query=~".*SELECT.", instance=~"$instance"}[1m])) by (datname) and on() pg_static{short_version=~"1[0-6].*"})
```

Версия 1.30.0

- В микросервис **Rule Engine** добавлены:
  - автоматическая настройка `GOMAXPROCS`, которая задает ограничение по количеству параллельных выполняемых процессов. Значение `GOMAXPROCS` определяется значением CPU-quota, установленным для сервиса в файле `docker-compose`.
  - переменная `MAX_PROCESSES`, через которую можно указать количество используемых vCPU.
- nginx обновлен до последней стабильной версии 1.28.0.
- добавили описание микросервисов и обновили список воркеров в разделе Настройка воркеров.
- обновлена ссылка на миграционный скрипт в разделе Очередность обновления.
- исправлена ошибка в названии хоста с etcd_03 на etcd_01, для перемещения хоста **redis_auth_sentinel** необходимо запустить плейбук.

Версия 1.29.0

- Оптимизация микросервиса **Indication Calculator**
  - Для уменьшения нагрузки на микросервис в случае наличия большого количества активных индикаторов на одной таблице, в него была добавлена предварительная фильтрация индикаторов. Теперь индикаторы обрабатываются только в случае соответствия RE-модели записи используемой в условии запуска или если были

изменены поля, указанные в условиях индикатора. Подробнее читайте в пользовательской документации.

    - Для отключения механизма предварительной фильтрации, добавьте в контейнер **indication-calculator** переменную окружения `ENABLE_CONDITION_PRECHECK` со значением `0`.

- Пакет **kafka-ui**, входящий в дистрибутив, обновлен до версии 0.7.2.
- Уточнена версия пакета cryptography, необходимая для виртуальных машин на Oracle Linux в списке предустановленного ПО.
- В раздел Настройка внешней БД добавлен пункт о необходимости создания баз данных при подготовке кластера внешней СУБД.
- Дополнено описание изменений, связанных с переносом авторизации на API-шлюз в версии Платформы 1.28.2.
- В раздел Установка платформы добавлено примечание о настройке пути к файлу лога установки продукта.
- Для распределенной архитектуры из хост-группы Redis удален один узел Redis и теперь их 2. Перед обновлением на версию 1.29.0 необходимо убедиться, что в подгруппе **redis_node** в inventory числится два узла.

Версия 1.28.2

- Реализовано горизонтальное масштабирование микросервиса **Rule Engine**, запускающего бизнес-правила для индикаций.
    - С этой версии число партиций для распределенной архитектуры не задается напрямую, а передается из переменной `rule_engine_replicas`, находящейся в файле `/vars/simple_backend/rule_engine.yml` (по умолчанию 4). Число реплик сервиса **Rule Engine** рассчитывается из числа партиций по коэффициенту 0,5. Таким образом, на каждой из зон доступности распределенного экземпляра, за исключением зоны 3, разворачивается по 2 реплики сервиса **Rule Engine**. К каждому экземпляру сервиса **Rule Engine** задается одна партиция топика `indication_calc.public.sys_indication`.
    - На простом экземпляре разворачивается один экземпляр сервиса **Rule Engine**, для которого задается одна партиция топика `indication_calc.public.sys_indication`.
- В раздел Настройка приложения добавлено описание переменной `vendor_id` и уточнено описание переменной `client_prefix`.
- Дополнена недостающая информация о добавленных компонентах и открытых портах, вошедших в версию 1.26.0.

Перенос авторизации на API-шлюз

- Начиная с версии 1.28.2, логика создания, валидации и пролонгации bearer-токенов перенесена из контейнера **backend-api** в API-шлюз (компонент в контейнере **proxy**) в качестве единой точки авторизации для монолита и микросервисов. URL-адрес для подключения к API-шлюзу формируется на основе переменных окружения `API_GATEWAY_HOST` и `API_GATEWAY_PORT`;

- Валидация и пролонгация bearer-токенов производится при помощи Lua-скрипта.
  - Создание токена инициируется при помощи обращения к Lua-эндпоинту `/create`.
    - В случае успешного логина пользователя через эндпоинт `/auth/login`, выполняется подзапрос на Lua-эндпоинт `/create`, в результате чего токен сохраняется в key-value базы данных.
    - Сохраненный токен возвращается в теле ответа в монолит и клиенту в поле `auth_key` в ответе на запрос `/auth/login`.
  - Деактивация (разлогинивание) пользователя с агентского интерфейса производится посредством обращения к Lua-эндпоинту `/delete` (`/logout`).
- Созданный токен хранится в контейнере **cache**.
- Срок жизни токена, предоставляющего доступ пользователя к системе (access_token), задается в секундах через переменную окружения `TOKEN_TTL` и по умолчанию составляет 259200 секунд (3 дня).
- В связи с переносом логики, увеличены требования к ресурсам для серверов FRONTEND-01 и FRONTEND-02 распределенной схемы в Таблице 3. Требования к контуру PROD.

Версия 1.28.0

- В Таблице 3. Требования к контуру PROD увеличены минимальные требования к объему оперативной памяти для серверов CACHE-01 и CACHE-02. Теперь требуемый объем памяти состявляет 4 Гб даже при числе агентов ≤ 100.
- Уточнено описание изменений, внесенных в плейбук в версии 1.26.0.
- В файле плейбука `complex-offline.yml` для хостов `kafka_server` и `kafka_connect` добавлена роль `itglobal_com.docker_apps_monitoring`.

Версия 1.27.0

- Для корректной работы SSO в плейбук добавлены переменные `backend_saml_private_key` и `backend_saml_private_key_passphrase`. Подробнее о них в разделе Настройка SSO.
- В инструкцию по настройке доступа к экземпляру по нескольким доменным именам добавлено описание дополнительных настроек, необходимых для работы функциональности.
- Внесены уточнения в Требования к программной части: поддерживаемая версия ALT Server – 10.4 (Mendelevium). Также в Приложение 1. Пример списка предустановленного ПО добавлены требования к версиям пакетов, устанавливаемых на ALT Server.
- В этой версии коннектор **monolith-source** заменен на коннекторы **dynamic-monolith** и **static-monolith**. При этом не удаляется связанный с коннектором **monolith-source** слот репликации в СУБД.



Для предотвращения потери необработанных консьюмерами сообщений, перед обновлением на версию 1.27.0 или более новую и удалением слота репликации, необходимо проверить очередь в Kafka на наличие лага в

топиках коннектора **monolith-source**. При наличии лага, необходимо дождаться обработки скопившихся сообщений.

Для удаления слота репликации используйте команду:

```
SELECT pg_drop_replication_slot('debezium_monolith');
```

Версия 1.26.0

- Начиная с этой версии доступно использование внешней СУБД с экземплярами на простой и распределенной схемах размещения компонентов. Требования к внешней СУБД и ее настройка описаны в разделе Требования к внешней БД.
- Для обеспечения надежного хранилища токенов и сессий пользователей без потери авторизации реализовано распределенное хранилище с двумя узлами Redis (master и replica). Реализована Sentinel-схема переключения узла replica в режим master в случае отказа текущего узла master.
  - Добавлен дополнительный Redis-кэш для хранения данных аутентификации в простой и распределенной схемах развертывания.
  - Для распределенной схемы реализован кластер Redis с Redis Sentinel, данные реплицируются между узлами master и replica без шардирования.
  - Реализована поддержка Redis Sentinel на HAProxy. Все запросы к Redis проксируются через HAProxy до актуального master-узла.
  - При отключении master-узла replica-узел становится новым master-узлом и HAProxy переключается на запись в него.
  - При восстановлении выключенного узла он становится новым replica-узлом и восстанавливает состояние репликации (загружает все недостающие/новые записи из нового master-узла).
  - Узлы работают в режиме постоянного хранения данных (persistence).
  - Узлы Redis Sentinel расположены рядом с экземплярами самого Redis (то есть на трех хостах рядом с узлами Redis (web-cache, auth-master и auth-replica) также расположены 3 узла Sentinel).
  - Опционально допускается вариант, когда в инфраструктуре присутствует выделенный хост **CACHE03**. В таком случае третий экземпляр **redis_auth_sentinel** необходимо разместить на нем, вместо хоста **etcd**. Если вышеперечисленные хосты отсутствуют, контейнер будет размещен на хосте **etcd_01** (в этом случае требуется открыть порт 8379).
  - В Приложение 4. Высокодоступная схема, контур PROD добавлены компоненты **redis_auth** и **redis_auth_sentinel**.
  - Контейнеры Redis остаются на хостах **CACHE01** и **CACHE02** с открытым портом 6379 на обоих узлах.
  - Существенного увеличения нагрузки на хосты не предполагается, требования к хостам не меняются.
- Обновлена схема хостов Redis. Вместо одной группы redis с 2 хостами появились 2

группы: redis (состоящая из 2 хостов) и redis_cluster (которая включает в себя 2 подгруппы: redis_node, состоящую из 2 хостов, и redis_sentinel, которая включает в себя 3 хоста).

- Для сетевого взаимодействия контейнер **redis_auth** использует открытый порт 7379, а для **redis_auth_sentinel** - открытый порт 8379.

Версия 1.25.0

- Добавлен компонент PgBouncer на стороне серверов Backend и БД, который разворачивается только на распределенной схеме при включенной переменной `pgbouncer_enabled=true`. Компонент нужен для лучшей утилизации ресурсов СУБД под высокой нагрузкой. Подробнее см. Настройка PgBouncer.
- Добавлена переменная окружения `backend_api_origins`, которая позволяет настроить доступ к экземпляру по нескольким доменным именам. Также изменились правила Cross-origin resource sharing (CORS): теперь доступ к endpoint-ам возможен только с указанных в переменной хостов. Подробнее о настройке смотрите в разделе Заполнение переменных vars.

Версия 1.24.2

- Небольшие правки документации: уточнили формулировку про конкурентных пользователей-агентов и неизменность UID пользователей Linux.

Версия 1.23.3

- Увеличен размер heap памяти, выделяемой для Kafka и Kafka Connect. Он определяется переменной окружения `KAFKA_HEAP_OPTS` и устанавливается исходя из объема оперативной памяти. Для минимального и максимального размера heap (**-Xms** и **-Xmx**) устанавливаются одинаковые значения.
  - Для простой архитектуры, размер heap определяется заранее заданными значениями:
    - если объем оперативной памяти меньше 15 Гб – для heap используется 2 Гб;
    - если больше 15, но меньше 30 Гб то – для heap используется 4 гб;
    - если 30 Гб и больше – для heap используется 6 Гб.
  - Для высокодоступной архитектуры, размер heap рассчитывается следующим образом:
    - если объем оперативной памяти меньше 15 Гб – для heap используется 75% от ее объема;
    - если объем оперативной памяти 30 Гб и больше – для heap используется 85% от ее объема.
- Появилась возможность создать дополнительный хост с холодной репликой БД. Подробнее о настройке в разделе Для высокодоступной архитектуры.

- В плейбук добавлены переменные окружения:
  - `CDC_CONSUMER_CREATE_TOPICS_REPLICATION_FACTOR` со значением, равным количеству kafka-брокеров;
  - `CDC_CONSUMER_CREATE_TOPICS_NUM_PARTITIONS` со значением 2.

В распределенном режиме топики, созданные сервисом **indication-calc**, реплицируются.

- С новой версии запросы SELECT на опрашивание состояния задач на экспорт выполняются на реплике.

Версия 1.22.3

- Добавлен инструмент трассировки запросов Jaeger для анализа работы приложения и выявления проблем, связанных с производительностью. Подробнее в разделе Трассировка.
- Для работы Jaeger с хоста ETCD в таблице сетевых взаимодействий компонентов добавлено 4 новых связи.

Версия 1.21.3

- Добавлены переменные для настройки экспорта:
  - `list_exporter_worker_count` – количество воркеров экспорта.
  - `list_exporter_data_request_batch_size` – размер батч-файлов запросов данных.

Версия 1.21.2

- Роль **BACKEND** разделена на **BACKEND-API**, **BACKEND-MICROSERVICES**, **BACKEND-WORKER**, **BACKEND-SCHEDULER**. Подробнее о компонентах в таблице раздела Высокодоступная архитектура.
  - Из-за разделения роли **BACKEND** изменены требования к конфигурациям серверов 4.1 Требования к серверам.
  - Изменены данные разбивки дисков 4.2 Разбивка дисков.
  - Добавлены новые группы хостов и порты в таблице сетевых взаимодействий.
  - Обновлена таблица сетевых взаимодействий Приложение 4. Таблица сетевых взаимодействий.
- С новой версии новый модуль SLA включен по умолчанию. Миграцию необходимо выполнить в случае, если она не была проведена раньше. Новый модуль SLA
- Изменены настройки логирования. Для сбора логов теперь необходимо настраивать Docker daemon. Подробнее о настройке в разделе Настройка системы сбора и анализа логов.
- Теперь необходимо установить пакет **passlib** для простой архитектуры, а также для **FRONTEND** хостов распределенной архитектуры.

- Обновлены команды для выключения и включения приложения для увеличения ресурсов хостов.

Версия 1.20.4

- Добавлена переменная окружения для отключения создания JSON-файла записей до обновления и обрезки в 2,5 Мбайт. Подробнее в разделе Заполнение переменных vars.

Версия 1.20.1

- Добавлен новый модуль SLA и миграция существующих индикаторов и индикаций на него.
- Добавлен новый воркер **event_sla**. Подробнее о формуле вычисления количества воркеров в разделе Заполнение переменных monitoring.
- Обновился список предустановленного ПО. Подробнее см. Приложение 1. Пример списка предустановленного ПО.
- Добавлен новый раздел 7.4 Очистка данных Apache Kafka с инструкцией очистки для простой и высокодоступной схем.

Версия 1.19.1

- Изменен размер и разбивка диска для каждого kafka-сервера. Подробнее в разделах 4.1 Требования к серверам и 4.2 Разбивка дисков.
- Не рекомендуется обновление до 1.19.1 экземпляров с настроенной внешней БД.
- Изменились требования к версии docker-ce. Подробнее см. Приложение 1. Пример списка предустановленного ПО.
- Ограничение по размеру записей составляет 2,5 Мбайт. Подробнее в разделе Импорт пакета.

Версия 1.18.3

- Для высокодоступной схемы необходимо добавление хостов Kafka и Connect. Для single-node архитектуры перед обновлением нужно добавить дополнительные вычислительные ресурсы. Количество ресурсов для новых хостов зависит от конфигурации экземпляра. Подробнее в разделе Подготовка инфраструктуры.
- Необходимо заполнить переменные `postgres_su_pass`, `postgres_user_ro`, `postgres_pass_ro`. См. раздел Заполнение переменных vars.
- Добавились новые переменные ansible: `minio_region`, `kafka_ui_user`, `kafka_ui_password`, `kafka_ui_deploy`. Подробнее в разделе Заполнение переменных vars.
- Для RHEL-based ОС убедиться, что pip3 пакет requests версии <2.30.
- При необходимости ручного запуска контейнеров single-node архитектуры или на серверах распределенной схемы с ролью backend необходимо добавлять профиль simpleone в команду docker-compose. Например, `docker-compose --profile simpleone up -d`

2. Требования к квалификации специалиста

Специалист, который будет разворачивать инфраструктуру и выполнять рекомендации данной документации, должен обладать следующими профессиональными навыками:

- **Администрирование операционных систем на базе Linux**
  - знание системы управления контейнерами Docker (запуск/остановка контейнеров, понимание работы docker-compose, анализ логов работы контейнеров);
  - опыт использования инструмента управления конфигурацией Ansible (понимание механизмов работы плейбуков, базовое понимание работы с переменными Ansible);
  - понимание сетевых настроек на уровне операционной системы и умение работать с ними.
- **Брокеры сообщений**
  - опыт настройки брокера сообщений RabbitMQ и управления им;
  - опыт настройки брокера Kafka и управления им;
- **Базы данных**
  - опыт администрирования баз данных Redis и PostgreSQL;
  - опыт работы с распределенным хранилищем etcd;
- **Прокси и веб-серверы**
  - понимание принципов работы и умение настраивать L4 и L7 балансировщики, а также реверсивные прокси и веб-серверы – HAProxy, nginx;
- **Мониторинг**
  - опыт работы с комплексом Elastic Stack для экспорта, анализа и визуализации технических логов (Elasticsearch, Logstash и Kibana) или Graylog, Fluentd;
  - опыт работы со сбором метрик в формате Prometheus (VMAgent) и системами их хранения (Prometheus, VictoriaMetrics);
- **Хранилище данных**
  - опыт работы с объектным хранилищем MinIO;
- **Дополнительно**
  - знание платформы SimpleOne: умение работать в агентском интерфейсе, понимание работы системы контроля версий SimpleOne (VCS), умение работать с конфигурационными пакетами и разрешать конфликты записей VCS. Рекомендуется пройти курс Разработчик решений с использованием Low-/No-Code инструментов на платформе SimpleOne.

3. Состав дистрибутива

Архив с дистрибутивом содержит в себе следующие файлы и директории:

- `/files` – конфигурационные файлы, необходимые для установки SimpleOne;
- `/inventory` – конфигурационные файлы плейбука;
- `/roles` – роли, которые запускаются плейбуком;
- `/templates` – шаблоны, из которых создаются конфигурационные файлы;
- `/vars` – информация о версиях сборок;
- `ansible.cfg` – стандартная конфигурация Ansible;
- `complex-offline.yml` – плейбук;
- `images.tar.gz` – набор образов Docker, которые будут устанавливаться;
- `README.md` – актуальная инструкция и описание скриптов, необходимых для запуска плейбука;
- `sla_migration.yml` – мигратор sla.

Не рекомендуется вносить изменения в какие-либо файлы, если иное не описано в настройке конфигурационных файлов. Также в директориях распакованного дистрибутива не должно находиться никаких посторонних файлов с расширениями *.yaml* и *.yml* и директорий.

4. Описание типов экземпляров

Решение SimpleOne предоставляет три типа экземпляров, которые рекомендуется устанавливать комплектом:

PROD

Основной продуктивный набор программных средств, предназначенный для установки и работы текущего стабильного и протестированного релиза продуктов SimpleOne. Рекомендован для использования конечными пользователями.

TEST

Набор программных средств решения SimpleOne, предназначенный для установки текущего релиз-кандидата и тестирования взаимодействия со сторонними информационными системами. Рекомендуется использовать тестировщикам.

DEV

Набор программных средств, предназначенный для разработки новой функциональности продуктов SimpleOne. Рекомендуется использовать администраторам и разработчикам.

5. Архитектура

Решение SimpleOne предлагает два типа размещения компонентов:

- высокодоступная архитектура;
- простая архитектура (single-node).

Ниже приведены описания предлагаемых типов архитектур, а также схемы расположения компонентов и их описание.

5.1. Высокодоступная архитектура

Высокодоступная архитектура – это совокупность компонентов, расположенных на нескольких узлах, объединенных в свою очередь в зоны доступности. Такая архитектура наиболее оптимальна для высоконагруженных продуктивных экземпляров. Подробная схема компонентов высокодоступной архитектуры находится в Приложении 4.

Для высокодоступной архитектуры доступно два варианта реализации:

- минимальная распределенная схема (с уменьшенным набором компонентов QUEUE и STORAGE);
- полная схема (рис. 1 и 2).



На данный момент обновление экземпляра с минимальной на полную схему на нем самом невозможно.

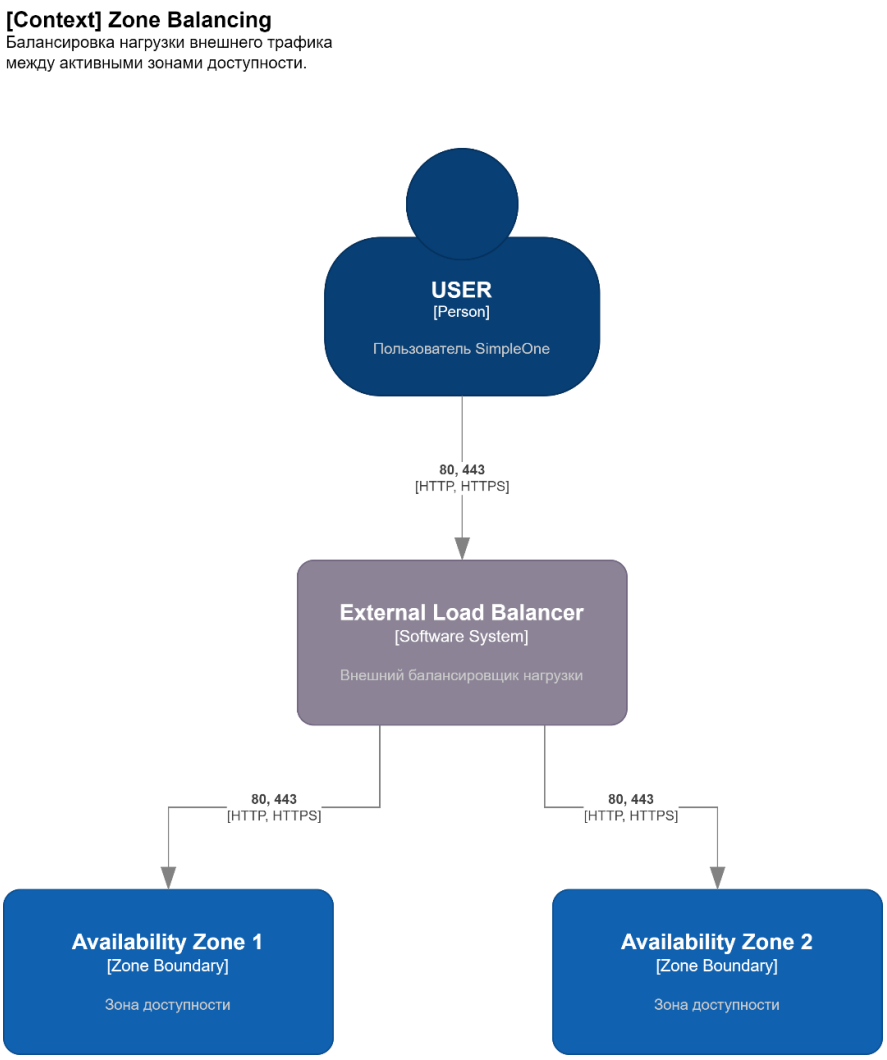


Рисунок 1. Внешний балансировщик нагрузки, контур PROD

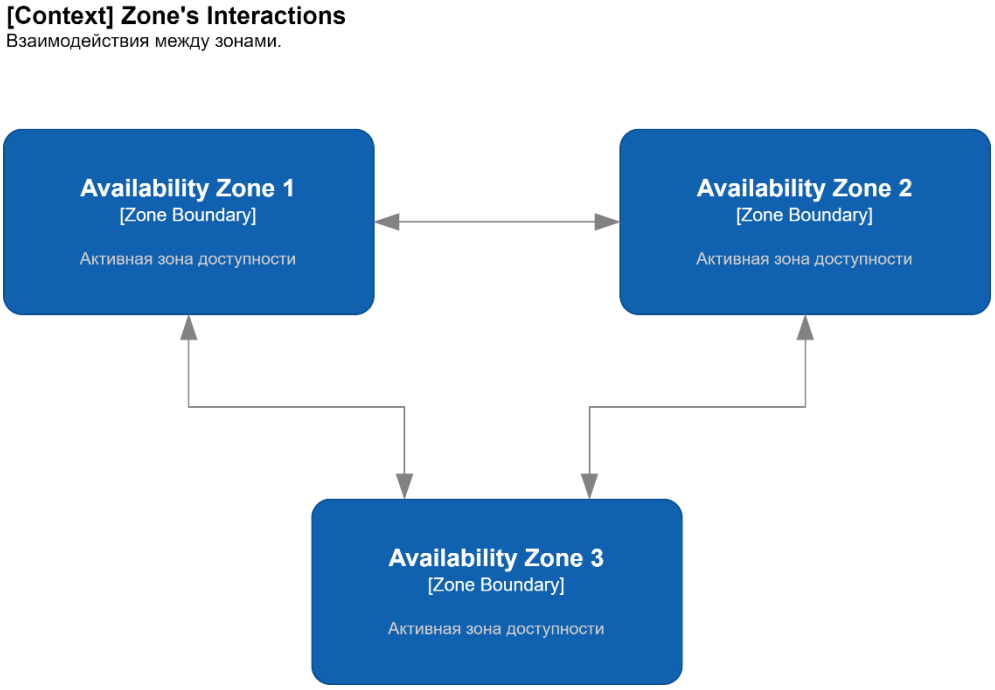


Рисунок 2. Взаимодействие зон высокодоступной схемы

Описание компонентов контура **PROD** представлено в таблице 1.

Таблица 1. Описание компонентов контура PROD

№	Название	Расположение	Назначение сервера
1	FRONTEND-01	PROD:FRONTEND-01	Веб-приложение для взаимодействия пользователя с системой
2	FRONTEND-02	PROD:FRONTEND-02	Веб-приложение для взаимодействия пользователя с системой
3	STORAGE-01	PROD:STORAGE-01	ПО, реализующее функции отказоустойчивого распределенного объектного хранилища
4	STORAGE-02	PROD:STORAGE-02	ПО, реализующее функции отказоустойчивого распределенного объектного хранилища
5	STORAGE-03	PROD:STORAGE-03	ПО, реализующее функции отказоустойчивого распределенного объектного хранилища
6	STORAGE-04	PROD:STORAGE-04	ПО, реализующее функции отказоустойчивого распределенного объектного хранилища
7	BACKEND-API-01	PROD:BACKEND-API-01	Серверное приложение, отвечающее за принятие внешнего трафика
8	BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	Серверное приложение, отвечающее за выполнение задач
9	BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	Серверное приложение, содержащее логику работы микросервисов
10	BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	Серверное приложение, отвечающее за выполнение задач по расписанию
11	BACKEND-API-02	PROD:BACKEND-API -02	Серверное приложение, отвечающее за принятие внешнего трафика
12	BACKEND-WORKER-02	PROD:BACKEND-WORKER-2	Серверное приложение, отвечающее за выполнение задач
13	BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	Серверное приложение, содержащее логику работы микросервисов
14	BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	Серверное приложение, отвечающее за выполнение задач по расписанию
15	QUEUE-01	PROD:QUEUE-01	ПО для асинхронной работы элементов приложения
16	QUEUE-02	PROD:QUEUE-02	ПО для асинхронной работы элементов приложения
17	QUEUE-03	PROD:QUEUE-03	ПО для асинхронной работы элементов приложения
18	QUEUE-04	PROD:QUEUE-04	ПО для асинхронной работы элементов приложения
19	CACHE-01	PROD:CACHE-01	Локальный кэш данных
20	CACHE-02	PROD:CACHE-02	Локальный кэш данных
21	DATABASE-01	PROD:DATABASE-01	Постоянное хранилище информации
22	DATABASE-02	PROD:DATABASE-02	Постоянное хранилище информации
23	ETCD-01	PROD:ETCD-01	Репликация БД и обеспечение консистентности данных

№	Название	Расположение	Назначение сервера
24	KAFKA-01	PROD:KAFKA-01	Потоковая передача данных
25	KAFKA-02	PROD:KAFKA-02	Потоковая передача данных
26	KAFKA-03	PROD:KAFKA-03	Потоковая передача данных
27	CONNECT-01	PROD:CONNECT-01	Перемещение данных между Kafka и БД
28	CONNECT-02	PROD:CONNECT-02	Перемещение данных между Kafka и БД



Все зоны кластера (Zone 1, Zone 2 и Zone 3) должны быть одинаковы по аппаратному обеспечению, но каждая с учетом необходимых ресурсов. Сетевые соединения должны быть равноценны по пропускной способности и задержке. Политика виртуализации CPU должна быть одинаковая на всех узлах распределенной схемы. Каждая из зон (в том числе и Zone 3) должна находиться в отдельном помещении/дата-центре для обеспечения отказоустойчивости.

В случае, если высокодоступная схема используется на продуктивных экземплярах, рекомендуется использование похожей схемы и на экземплярах для тестирования. Однако допустимо, чтобы на экземплярах для тестирования и разработки использовалась Простая (single-node) архитектура.

5.2. Простая (single-node) архитектура

Простая архитектура представляет собой размещение всех компонентов SimpleOne на одном узле. Она менее надежна по сравнению с высокодоступной схемой, однако менее ресурсоемкая.

Простая архитектура подходит для разработки и тестирования. Также возможно размещение продуктивного экземпляра в случае, если на нем будет работать небольшое количество пользователей.

Таблица 2. Описание элементов контура DEV

№	Название	Расположение	Назначение сервера
1	DEV-01	DEV:DEV-01	Содержит все компоненты системы, необходимые для функционирования

6. Подготовка инфраструктуры

6.1. Требования к серверам

В зависимости от количества агентов, которые будут использовать экземпляр, отличаются и требования, которым серверы должны соответствовать. В случае, если количество агентов, использующих экземпляр, меньше 100, некоторые серверы могут отсутствовать.

В таблицах ниже отражены три варианта минимальных конфигураций сервера в зависимости от количества конкурентных пользователей – агентов, работающих одновременно (10% от общего количества агентов). При таких конфигурациях гарантирована бесперебойная работа ПО SimpleOne в случае отказоустойчивой архитектуры.

Приведенные ниже параметры определены исходя из показателей скорости дисков SSD не менее 30 iops на 1 Гб и частоты процессоров от 3.00 GHz.

Необходимо добавить возможность расширения ресурсов для всех виртуальных машин. В зависимости от специфики экземпляра и его нагруженности может потребоваться увеличение ресурсов. В параметры не входит нагрузка от резервного копирования и других регламентных задач инфраструктуры заказчика.



- Объем RAM на 2 Гб указан без учета ресурсов, затраченных на виртуализацию.
- Значения vCPU для серверов BACKEND-* и DATABASE указаны в следующем варианте: **значение без PgBouncer → значение с установленным PgBouncer**.

Таблица 3. Требования к контуру PROD

Зона / Физ. сервер	**Название сервера**	**≤ 100 агентов (10-20 конкурентных пользователей)**			**200 агентов (20 конкурентных пользователей)**			**500 агентов (50 конкурентных пользователей)**		
		vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб
Zone 1	FRONTEND-01	2	4	60	2	4	60	3	6	60
	BACKEND-API-01	4→6	4	70	6→8	4	70	14→16	8	70
	BACKEND-WORKER-01*	2→4	4	70	2→4	4	70	4→6	4	70
	BACKEND-MICROSERVICES-01	2→4	4	70	2→4	4	70	2→4	4	70
	BACKEND-SCHEDULER-01	2→4	4	70	2→4	4	70	2→4	4	70
	DATABASE-01	8→10	16	SSD, 250	8→10	16	SSD, 250	16→18	32	SSD, 250
	STORAGE-01	2	2	SAS, 512	2	2	SAS, 512	2	2	SAS, 512

Зона / Физ. сервер	Название сервера	≤ 100 агентов (10-20 конкурентных пользователей)			200 агентов (20 конкурентных пользователей)			500 агентов (50 конкурентных пользователей)		
	STORAGE-02	-	-	-	2	2	SAS, 512	2	2	SAS, 512
	QUEUE-01	2	2	100	2	2	100	2	2	100
	QUEUE-02	-	-	-	2	2	100	2	2	100
	CACHE-01	2	4	50	2	4	50	2	4	50
	KAFKA-01	4	8	SSD, 250	6	12	SSD, 250	6	12	SSD, 250
	CONNECT-01	2	4	50	2	4	50	2	4	50
	Всего:	**32→42**	**56**	**1552**	**40→50**	**64**	**2164**	**59→69**	**86**	**2164**
Zone 2	FRONTEND-02	2	4	60	2	4	60	3	6	60
	BACKEND-API-02	4→6	4	70	6→8	4	70	14→16	8	70
	BACKEND-WORKER-02*	2→4	4	70	2→4	4	70	4→6	4	70
	BACKEND-MICROSERVICES-02	2→4	4	70	2→4	4	70	2→4	4	70
	BACKEND-SCHEDULER-02	2→4	4	70	2→4	4	70	2→4	4	70
	DATABASE-02	8→10	16	SSD, 250	8→10	16	SSD, 250	16→18	32	SSD, 250
	STORAGE-03	2	2	SAS, 512	2	2	SAS, 512	2	2	SAS, 512
	STORAGE-04	-	-	-	2	2	SAS, 512	2	2	SAS, 512
	QUEUE-03	2	2	100	2	2	100	2	2	100
	QUEUE-04	-	-	-	2	2	100	2	2	100
	CACHE-02	2	4	50	2	4	50	2	4	50
	KAFKA-02	4	8	SSD, 250	6	12	SSD, 250	6	12	SSD, 250
	CONNECT-02	2	4	50	2	4	50	2	4	50
	Всего:	**32→42**	**56**	**1552**	**40→50**	**64**	**2164**	**59→69**	**86**	**2164**
Zone 3	ETCD-01	4	8	SSD, 60	4	8	SSD, 80	4	8	SSD, 100
	KAFKA-03	4	8	SSD, 250	6	12	SSD, 250	6	12	SSD, 250
	Всего:	**8**	**16**	**310**	**10**	**20**	**330**	**10**	**20**	**350**

* В зависимости от количества настроенных воркеров может потребоваться увеличение ресурсов по vCPU, RAM.

Для некоторых серверов рекомендуется использовать тип накопителя, производительность которого не ниже указанного в таблице.



Необходимо предусмотреть возможность увеличения ресурсов по vCPU, RAM и SSD/SAS для серверов Kafka и Connect. Размер дискового пространства на

серверах Kafka зависит от размера БД на экземпляре.

Для обновления на хостах БД необходимо предусмотреть в два раза больше места, чем требуется для БД.

Таблица 4. Требования к контуру DEV

Название сервера	**≤ 100 агентов (10-20 конкурентных пользователей)**			**200 агентов (20 конкурентных пользователей)**			**500 агентов (50 конкурентных пользователей)**		
	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб
DEV-01	8	18	500	8	32	500	8	32	500

Требования для простой архитектуры (single-node) контуров **PROD** и **TEST** представлены в таблицах 5 и 6.

Таблица 5. Требования к контуру PROD (single-node)

Название сервера	**≤ 100 агентов (10-20 конкурентных пользователей)**			**200 агентов (20 конкурентных пользователей)**			**500 агентов (50 конкурентных пользователей)**		
	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб
PROD-01	46	80	2064	54	92	3288	88	128	3288

Таблица 6. Требования к контуру TEST (single-node)

Название сервера	**≤ 100 агентов (10-20 конкурентных пользователей)**			**200 агентов (20 конкурентных пользователей)**			**500 агентов (50 конкурентных пользователей)**		
	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб	vCPU, шт.	RAM, Гб	SSD / SAS, Гб
TEST-01	8	18	500	8	32	500	8	32	500

6.2. Разбивка дисков

Для установки платформы разбивка дисков и отдельное монтирование каких-либо директорий не требуется – достаточно монолитного раздела на диске.

Однако, если существует необходимость в разбиении дискового пространства на директории, то рекомендуется создать следующие разделы:

- `/` – корневая директория хоста, системная директория;
- `/var` – скачанные образы Docker;
- `/data` – пользовательская информация, данные из БД и S3;
- `/opt` – хранилище конфигураций для развертывания проекта;
- `/home` – временное хранилище информации и логов, необходимое для расследования возникших инцидентов;

- `/tmp` – временные файлы;
- `swap` – раздел для файла подкачки. Его величина зависит от объема RAM.

В таблицах 7 и 8 представлены примеры схем для высокодоступной и простой (single-node) архитектуры соответственно при минимальных конфигурациях серверов. В случае увеличения дискового пространства потребуется увеличение размера директорий.

Таблица 7. Разбивка диска для высокодоступной архитектуры

№	Имя	Всего, Гб	/, Гб	/var, Гб	/data, Гб	/opt, Гб	/home, Гб	/tmp, Гб	swap, Гб
1	FRONTEND	65	16	45	–	1.0	1.0	1.0	1.0
2	BACKEND-API	55	20	27**	–	1.0	1.0	2.0	4.0*
3	BACKEND-WORKERS	55	20	35	–	1.0	1.0	2.0	4.0*
4	BACKEND-MICROSERVICES	55	20	35	–	1.0	1.0	2.0	4.0*
5	BACKEND-SCHEDULER	55	20	35	–	1.0	1.0	2.0	4.0*
6	DATABASE	255	14	35	194	1.0	1.0	2.0	8.0
7	STORAGE	517	15	55	440**	1.0	2.0	2.0	2.0
8	QUEUE	105	14	85	–	1.0	1.0	2.0	2.0
9	CACHE	55	14	35	–	1.0	1.0	2.0	2.0
10	ETCD	65	16	45	–	1.0	1.0	1.0	1.0
11	KAFKA	255	13	35	200	1.0	1.0	1.0	4.0
12	CONNECT	55	14	35	–	1.0	1.0	2.0	2.0

Таблица 8. Разбивка диска для простой (single-node) архитектуры

№	Имя	Всего, Гб	/, Гб	/var, Гб	/data, Гб	/opt, Гб	/home, Гб	/tmp, Гб	swap, Гб
1	dev	505	14	55**	422**	2.0	2.0	2.0	8.0

* Величина раздела `swap` зависит от объема RAM.

** При обновлении нужно предусмотреть дополнительное место на дисках, так как размер JSON-файла с полной версией данных, которые формируются в результате обрезки записей, превышающих размер в 2,5 Мбайт, зависит от БД экземпляра. Записи архивируются во временную директорию внутри контейнера **backend-api**, затем архив попадает в **MinIO** и удаляется из контейнера **backend-api**. Чтобы отключить создание файла, настройте переменную окружения **backend_migration_create_bigdata_log**.



В процессе эксплуатации системы может возникнуть нехватка пространства в разделе **/var**. Это связано с ошибкой в работе Docker, из-за которой папка `/var/lib/docker/overlay2` не очищается. До устранения ошибки рекомендуем использовать обходное решение:

1. Определить контейнер, который приводит к высокой утилизации диска:

a. Вывести сортировку по содержимому директории `/var/lib/docker/overlay2/`:
`du -sh /var/lib/docker/overlay2/* | sort -hr | head -n 10`
Определить ID контейнера/слоя, который имеет аномальный размер, значительно больше других.

b. Выявить с каким контейнером связан ID:
`docker ps -q | xargs docker inspect -f '{{.Id}} {{.Name}} {{.GraphDriver.Data.MergedDir}}' | grep <ID>`
Должно появиться наименование контейнера.

2. Остановить и удалить контейнер, определенный в пункте 1 (вызывает простой):
`docker rm -fv <_container_name_>`

3. Пересоздать контейнеры:
`bash up.sh`

6.3. Требования к программной части

1. Поддерживаемая операционная система – RHEL-совместимый дистрибутив версии не ниже 8.5 или не ниже 9 (рекомендован вендором – Oracle Linux, протестирована работа на AlmaLinux). Также поддерживается ALT Server 10.4 (Mendelevium).
2. Должны быть предоставлены доменные имена и SSL-сертификаты в формате .pem (требуется 2 доменных имени для каждого типа экземпляров: для экземпляра SimpleOne и для работы объектного хранилища), а также корневой сертификат удостоверяющего центра.
3. Если используется доменное имя не из зоны ***.simpleone.ru**, то необходимо предоставить SSL-сертификаты в формате .pem для основного доменного имени и для доменного имени хранилища S3 или Wildcard-сертификат в формате .pem для домена.



Закрытый ключ сертификата в формате .pem должен иметь следующие свойства:

- Алгоритм: **RSA**
- Формат: **PKCS#8**
- Размер: **2048 bits**
- Без пароля

4. Если используется доменное имя не из зоны ***.simpleone.ru**, то на DNS-серверах должны быть созданы записи, ссылающиеся на IP-адрес сервера:
   - <***имя_экземпляра***>.<***домен***>
   - **s3-**<***имя_экземпляра***>.<***домен***>
5. На хосте с доступом ко всем виртуальным машинам, на которых будет установлено ПО SimpleOne, необходимо установить ПО Ansible версии не ниже 4 и не выше 6.
6. На виртуальных машинах экземпляров не должно быть посторонних сервисов,

антивирусов, графических подсистем и других подобных программ.

6.4. Требования к каналам связи

1. Необходимо обеспечить сетевую доступность следующих ресурсов (в случае оффлайн-установки или обновления, выполнение данного требования необязательно):
   - при разворачивании системы:
     - https://docker.io
     - https://download.docker.com
     - https://registry-1.docker.io
     - https://pypi.python.org
     - https://www.iana.org/time-zones
   - постоянный доступ:
     - https://reg01.simpleone.ru – используется для обновления системы на новые версии
2. Необходимо обеспечить доступность следующих входящих портов и протоколов: **tcp/80**, **tcp/443** до хостов с ролью **FRONTEND** либо до балансировщика, который должен удовлетворять требованиям к балансировщику.
3. Все хосты (прокси) между клиентом и сервером должны свободно пропускать соединения по протоколу Websocket. Сам клиент тоже должен его поддерживать.
4. Необходимо обеспечить каналы связи шириной не менее 1 Gbit/s между всеми виртуальными машинами или серверами.
5. Связность между компонентами платформы необходимо обеспечить в соответствии с Таблицей 9.



- Каждый из хостов с ролью **STORAGE** (**STORAGE-01–04**) должен иметь доступ по порту **9000/tcp** ко всем остальным хостам **STORAGE** в обоих направлениях.
- При размещении **redis_auth_sentinel** на хосте **etcd_01**, необходимо открыть порт 8379.

6. Для экземпляра, доменные имена которого указывают на внешний IP-адрес, имеющего проброс порта **443** на внешнем IP, на сетевом оборудовании должен быть включен Nat Hairpin (NAT Loopback) для обеспечения возможности обращения самого экземпляра к себе на порт **443** по доменным именам **s3-<*имя_экземпляра*>.<*домен*>** и **<*имя_экземпляра*>.<*домен*>**.
7. Время на всех хостах должно быть точным и синхронизировано из одного источника.
8. Зоны доступности должны находиться в переделах одного региона. Регион – это физическое местоположение, где находятся центры обработки и передачи данных в одной географической области.

Таблица 9. Внутреннее сетевое взаимодействие компонентов

№	Группа хостов источника	Группа хостов назначения	Порты назначения
1	VPS: FRONTEND	VPS: BACKEND-API	80, 82, 443, 8443, 7000/tcp
2	VPS: FRONTEND	VPS: BACKEND-MICROSRVICES	18090/tcp
3	VPS: FRONTEND	VPS: QUEUE	15672/tcp
4	VPS: FRONTEND	VPS: KAFKA	18080/tcp
5	VPS: FRONTEND	VPS: STORAGE	9000/tcp
6	VPS: FRONTEND	VPS: ETCD	16686/tcp
7	VPS: FRONTEND	VPS: CACHE	7379/tcp
8	VPS: BACKEND-API	VPS: KAFKA	8083/tcp
9	VPS: BACKEND-API	VPS: DATABASE	5432, 5433, 6432*, 8008/tcp
10	VPS: BACKEND-API	VPS: QUEUE	5672/tcp
11	VPS: BACKEND-API	VPS: CACHE	6379/tcp
12	VPS: BACKEND-API	VPS: ETCD	14268, 4318/tcp
13	VPS: BACKEND-API	VPS: FRONTEND	80/tcp
14	VPS: BACKEND-WORKER	VPS: BACKEND-MICROSRVICES	18080, 18083, 18084,18082, 19000/tcp
15	VPS: BACKEND-WORKER	VPS: DATABASE	5432, 5433, 6432*, 8008/tcp
16	VPS: BACKEND-WORKER	VPS: QUEUE	5672/tcp
17	VPS: BACKEND-WORKER	VPS: CACHE	6379/tcp
18	VPS: BACKEND-WORKER	VPS: ETCD	14268, 4318/tcp
19	VPS: BACKEND-MICROSRVICES	VPS: BACKEND-API	18091/tcp
20	VPS: BACKEND-MICROSRVICES	VPS: BACKEND-MICROSRVICES	18088, 18080–18086, 18090, 19000/tcp
21	VPS: BACKEND-MICROSRVICES	VPS: KAFKA	29092, 8081/tcp
22	VPS: BACKEND-MICROSRVICES	VPS: DATABASE	5432, 5433, 6432*, 8008/tcp
23	VPS: BACKEND-MICROSRVICES	VPS: QUEUE	5672/tcp
24	VPS: BACKEND-MICROSRVICES	VPS: CACHE	6379/tcp
25	VPS: BACKEND-MICROSERVICES	VPS: ETCD	14268, 4318/tcp
26	VPS: BACKEND-SCHEDULER	VPS: BACKEND-MICROSRVICES	18080, 18083, 18084,18082, 19000/tcp
27	VPS: BACKEND-SCHEDULER	VPS: DATABASE	5432, 5433, 6432*, 8008/tcp
28	VPS: BACKEND-SCHEDULER	VPS: QUEUE	5672/tcp
29	VPS: BACKEND-SCHEDULER	VPS: CACHE	6379/tcp

№	Группа хостов источника	Группа хостов назначения	Порты назначения
30	VPS: DATABASE	VPS: DATABASE	2379, 2380, 5432/tcp
31	VPS: DATABASE	VPS: ETCD	2379/tcp
32	VPS: STORAGE-01–04	VPS: STORAGE-01–04	9000/tcp
33	VPS: ETCD	VPS: ETCD	2379, 2380/tcp
34	VPS: QUEUE	VPS: QUEUE	5672, 4369, 15672, 25672/tcp
35	VPS: KAFKA	VPS: KAFKA	9092, 29092, 9093/tcp
36	VPS: CONNECT	VPS: DATABASE	5432, 5433, 8008/tcp
37	VPS: CONNECT	VPS: KAFKA	29092, 8081/tcp
38	VPS: CACHE	VPS: CACHE	7379/tcp, 8379/tcp

* Порт используется, когда включен **PgBouncer**.

6.5. Требования к балансировщику

1. Для продуктивного контура необходимо обеспечить единую точку входа в приложение с помощью балансировщика на 4 или 7 уровне модели OSI, размещенного и обслуживаемого Клиентом.
2. Балансировщик должен свободно пропускать соединения по протоколу Websocket.
3. Балансировка происходит на уровне TCP с проверкой доступности порта, на который происходит подключение.
4. Созданные ранее DNS-записи должны указывать на балансировщик.
5. Для обеих DNS-записей, в набор хостов, на которые осуществляется балансировка, должны входить серверы с ролью **FRONTEND**, входящие и исходящие порты одинаковые: **tcp/80**, **tcp/443**.
6. Балансировщик должен равномерно загружать серверы **FRONTEND** и следить за их доступностью для своевременного переключения запросов на рабочий хост.
7. У балансировщика должно быть снято ограничение на размер запроса, а также для балансировщика типа L7 должен быть установлен сертификат.
8. У балансировщика должно быть настроено разрешение на передачу имени сервера через расширение Server Name Indication (SNI) протокола TLS при соединении с HTTPS-сервером (директория **/proxy_ssl_server_name**).

6.6. Таблица сетевых взаимодействий

Правила сетевого доступа для решения SimpleOne в контурах **PROD**, **DEV** и **TEST** представлены в Приложении 3. Таблица сетевых взаимодействий. Данная таблица составлена для случая, когда архитектура контура **TEST** повторяет архитектуру контура **PROD** и является высокодоступной.

В случае если архитектура контура **TEST** будет являться простой (single-node), набор правил будет такой же, как и у контура **DEV**.

6.7. Требования к внешней БД



Внешняя БД должна соответствовать следующим требованиям:

- СУБД, версия СУБД и конфигурация аппаратного обеспечения, на котором установлена СУБД, должны соответствовать указанным в эксплуатационной документации;
- В СУБД не должно быть БД, которые не входят в список, приведенный в пункте 6 настоящего раздела;
- В базах данных ПО SimpleOne не должно быть схем, которые не входят в список, приведенный в пункте 7 настоящего раздела;
- Не допускается модификация схемы БД Клиентом в обход интерфейса ПО без согласования с Лицензиаром.
- В СУБД не должны быть установлены расширения, не предусмотренные настоящим разделом.
- Все узлы внешней СУБД должны находиться в одном ЦОД.

6.7.1. Настройка внешней БД

1. Версия PostgreSQL – 17. Рекомендуется использовать последнюю минорную версию.
2. Необходимо установить в БД часовой пояс UTC:
   - При создании БД указать часовой пояс в `postgresql.conf`.
   - Для действующей БД необходимо выполнить команду `SELECT pg_reload_conf()`.
   - При использовании **patroni** – через **patroniclt edit-config**.
3. Включить расширение **pg_trgm** для БД – `CREATE EXTENSION pg_trgm`; задать параметру конфигурации `wal_level` значение `logical`.
4. В файле `/inventory/[single,distributed]/group_vars/all/vars.yml` заполнить переменные в соответствии с пунктом Настройка внешней БД раздела 8.2.2 Заполнение переменных.
5. В СУБД должны быть созданы пользователи с правами:

№	Имя пользователя	Права	Переменные плейбука
1	simple_user	*LOGIN, CREATEDB, REPLICATION, SUPERUSER* Команды: - `GRANT USAGE on schema public to simple_user;` - `GRANT CREATE on schema public to simple_user;`	Логин: `postgres_user` Пароль: `postgres_pass` Файл: `inventory/distributed/group_vars/all/vars.yml`

№	Имя пользователя	Права	Переменные плейбука
2	readonly	*NOLOGIN* Команды: - `GRANT USAGE ON SCHEMA public TO readonly;` - `GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;` - `GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO readonly;` - `ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO readonly;`	
3	simple_user_ro	readonly	Логин: `postgres_user_ro` Пароль: `postgres_pass_ro` Файл: `inventory/distributed/group_vars/all/vars.yml`

6. Перед подключением кластера к приложению, в удаленной СУБД необходимо создать базы данных, перечисленные в подпунктах **a** и **b** ниже.

   Вы можете применить дамп любой внутренней базы данных к соответствующей базе данных внешнего кластера перед его подключением по процедуре, описанной в разделе Миграция баз данных.

   a. Базы данных SimpleOne:

№	Имя БД	Владелец и другие права	Переменная плейбука

1	postgres_simpleone	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE simpleone TO simple_user`
			`postgres_simpleone_db_name`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

2	simple_dq_app_db	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE simple_dq_app_db TO simple_user`
			`backend_dq_app_pg_db_name`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

3	simple_schedule_job_db	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE simple_schedule_job_db TO simple_user`
			`backend_schedule_job_pg_master_database`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

4	simple_scheduler_db	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE simple_scheduler_db TO simple_user`
			`backend_scheduler_pg_db_name`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

№	Имя БД	Владелец и другие права	Переменная плейбука

5	simple_indc_db	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE simple_indc_db TO simple_user`
			`indication_calculator_source_pg_db_name`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

6	list_exporter	simple_user
		
		Команда:
		`GRANT ALL PRIVILEGES ON DATABASE list_exporter TO simple_user`
			`list_exporter_pg_db_name`
			
			Файл:
			`inventory/[single,distributed]/group_vars/all/vars.yml`

b. Служебные БД:

   - postgres
   - template0
   - template1

7. В каждой из БД должны быть только перечисленные схемы:
   - information_schema – служебная
   - metric_helpers – служебная
   - pg_catalog – служебная
   - pg_toast – служебная
   - public – схема, в которой расположены таблицы SimpleOne
   - user_management – служебная

6.7.2. Обновление внешней БД

В связи с переходом на PostgreSQL 17 версии, перед обновлением на версию платформы 1.31.0, необходимо выполнить следующие действия для настройки внешней СУБД:

Этап 1: Подготовка

1. Согласовать время простоя.
2. Если внешняя СУБД PostgreSQL используется не только для SimpleOne, проверить совместимость расширений (postgis, timescaledb и т.д.) и функций с новой версией.
3. Создать бэкап пользовательских баз данных:

```
# Создание дампов для каждой пользовательской базы данных
# Получение списка пользовательских баз (исключая системные)
psql -U postgres -t -c "SELECT datname FROM pg_database WHERE datname NOT IN
('postgres', 'template0', 'template1', 'rdsadmin') AND datistemplate = false;" | \
while read dbname; do
  if [ -n "$dbname" ]; then
    pg_dump -U postgres -d "$dbname" -Fc -f "backup_${dbname}.dump"
  fi
```

```
done
# Отдельно дамп ролей и привилегий
pg_dumpall -U postgres --roles-only > roles_backup.sql
```

Этап 2: Обновление PostgreSQL (выполняется заказчиком)

1. Остановить PostgreSQL 14
2. Обновить PostgreSQL до 17 версии (метод зависит от используемой ОС):
   Пример для Ubuntu/Debian:
   `sudo systemctl stop postgresql`
   `sudo apt update && sudo apt install postgresql-17`

Этап 3: Восстановление

1. Восстановить роли и пользовательские базы данных.
   Восстановление ролей:
   `psql -U postgres -f roles_backup.sql`
   Восстановление пользовательских баз данных:

```
for dumpfile in backup_*.dump; do
  dbname=$(echo "$dumpfile" | sed 's/backup_\(.*\)\.dump/\1/')
  createdb -U postgres "$dbname"
  pg_restore -U postgres -d "$dbname" -v "$dumpfile"
done
```

Этап 4: Запуск и валидация

1. Если PostgreSQL 17 не запустился автоматически, запустить его либо вручную, либо с выполнением обновления до версии платформы 1.31.0.
2. Провести валидацию данных и подключения.
3. Провести тестирование приложения (connection strings остаются прежними).

7. Установка ПО

7.1. Перед установкой

Перед установкой платформы необходимо выполнить следующие действия:

1. Создать нужные хосты и настроить их параметры в соответствии с требованиями. На хостах рекомендуется установить Oracle Linux версии выше 8.5.
2. Выпустить SSL-сертификаты.
3. В случае высокодоступной архитектуры необходимо провести настройку балансировщика в соответствии с требованиями к балансировщику при помощи добавления созданных виртуальных машин и настройки для них правил, и указать DNS-записи для этого балансировщика.
4. Установить на хосты необходимое ПО из списка предустановленного ПО.
5. Убедиться, что служба **docker** из списка предустановленного ПО находится в автоматическом запуске при запуске ОС, а также находится во включенном состоянии на всех ОС во время установки платформы.
6. Убедиться, что на хост, с которого будет осуществляться конфигурация и запуск плейбука, установлены следующие пакеты:
   - установленный через pip **ansible** ≥4, но <6;
   - установленный через pip **cryptography** <36;
   - установленный через pip **jinja2** >3;
   - **sshpass**.
7. Убедиться, что на экземпляре СУБД нет схем таблиц, не входящих в поставку. В противном случае при установке, обновлении с оффлайн дистрибутива и последующей работе системы могут возникнуть ошибки.

Когда выполнение всех этапов завершено, необходимо убедиться, что все требования из перечня требований выполнены, после чего можно приступать к установке платформы.

7.2. Установка платформы



Для получения лога установки продукта в отдельном файле, необходимо задать имя файла в параметре `log_path` в разделе `[defaults]` файла `ansible.cfg`, который находится в корневой директории распакованного дистрибутива. Например:

```
log_path = my_log.log
```

Файл лога будет создан в корневой директории распакованного дистрибутива. Чтобы файл был создан в другой директории, укажите требуемый путь, например:

```
log_path = /installation_logs/my_log.log
```

Установка платформы происходит в несколько этапов:

1. Скачивание и распаковка архива с дистрибутивом.
    - С версии 1.21.3 появились длинные пути и имена файлов. Для правильной распаковки необходимо использовать команду `tar -xvf`.
2. Создание **inventory** – нужно скопировать директорию `/group_vars` и файл `inventory.yml` с примерами, в зависимости от выбранной архитектуры, поместить их в директорию, созданную вне структуры распакованного дистрибутива (пример названий папки: `distributed-prod` или `single-dev`).
    - `/inventory/single` – для простой архитектуры;
    - `/inventory/distributed` – для высокодоступной архитектуры.
3. Настройка конфигурационных файлов плейбука.
4. Запуск плейбука.
5. Проверка работоспособности экземпляра.

Если после выполнения всех шагов ошибок не появилось и переход на портал нового экземпляра происходит корректно, установку платформы можно считать завершенной.

Для установки бизнес-решений необходимо импортировать соответствующие пакеты.



Не меняйте UID пользователей Linux на экземпляре, все UID должны быть по умолчанию.

7.2.1. Новый модуль SLA

С версии 1.21.2 новый модуль SLA включен по умолчанию. Дополнительно необходимо выполнить следующие настройки:

1. При настройке переменных плейбука необходимо заполнить переменные Kafka.
2. Выполнить миграцию индикаторов и индикаций, если она не была выполнена ранее.
3. Произвести ребалансировку воркеров, если она не была выполнена ранее.

Миграция индикаций и индикаторов

При миграции индикаций и индикаторов на новый модуль SLA будут удалены таблицы **slm_timepoint_indicator** и **slm_timepoint_indication** и связанные с ними скрипты, бизнес-правила и другое. Записи из таблиц перенесутся в таблицы **sys_indicator** и **sys_indication** соответственно.



Все пользовательские поля таблиц **slm_timepoint_***, созданные на экземпляре, в новые таблицы не мигрируют.

1. Во время обновления и миграции индикаторов и индикаций экземпляр будет недоступен. Перед миграциями необходимо импортировать SOP-файл Simple Application.
2. Чтобы запустить мигратор, необходимо выполнить следующую команду:

```
ansible-playbook sla_migration.yml -vi "<path-to-inventory>" -e '{sla_migration: true}'
```

При миграции данных на новый SLA, если для индикации не существует индикатора, после переноса ее статус изменится на "Отменено".

Миграция индикаторов и индикаций не включает в себя миграцию связанных отчетов. Для корректной работы уже созданных отчетов, необходимо вручную изменить их настройки и указать таблицы нового модуля SLA.

В случае, если на экземпляре используется приложение ITSM:

1. Необходимо установить ITSM версии 1.8.1 и выше или дополнительный пакет **[ITSM] build.1026+new sla widgets**.
2. Изменить значение системного свойства **itsm.slm.new_indication_enabled** на **true**.

В результате на ITSM сущностях будет работать виджет SLA с индикациями нового модуля.

7.2.2. Настройка конфигурационных файлов

Настройка конфигурационных файлов плейбука производится в директории `/inventory`. Для каждого нового экземпляра необходимо будет производить отдельную настройку конфигурационных файлов.

Для высокодоступной архитектуры

Файлы конфигурации для высокодоступной архитектуры находятся в директории `/inventory/distributed`:

- в корне директории находится файл `inventory.yml`, в котором необходимо заполнить адреса созданных хостов в зависимости от их роли;
- в субдиректории `/group_vars/all` хранятся файлы, необходимые для настройки созданных хостов;
- в субдиректории `/group_vars/kafka_server` хранится файл `main.yml`, необходимый для настройки серверов Kafka.

Сначала необходимо настроить файл `inventory.yml`:

- по шаблону добавить необходимые хосты для каждой группы (группы хостов, объединенных одной ролью) или удалить из файла те хосты, которые не будут использоваться;



Для добавления холодного slave-узла БД укажите в файле дополнительный хост и настройте для него следующие переменные:

- `postgres_patroni_nofailover` – если указать значение `true`, slave-узел никогда не станет master-узлом целевой базы данных в случае аварийной ситуации со сменой лидера.
- `postgres_patroni_noloadbalance` – если указать значение `true`, slave-узел отключается от балансировки нагрузки. Backend-приложения SimpleOne

не будут отправлять запросы на эту реплику.

Пример заполнения:

```
pg_node02-prod:
    ansible_host: 10.172.0.XXX
    postgres_patroni_nofailover: true
    postgres_patroni_noloadbalance: true
```

Для дополнительного хоста требуются такие же ресурсы, настройки ОС и ПО, как и для двух хостов master- и slave-узлов БД.

- вместо шаблонов указать адреса хостов, на которые будет производиться установка.

После настройки `inventory.yml` нужно заполнить переменные плейбука в директории `/inventory/distributed/group_vars/all` в соответствии с Приложением 2. Настройка переменных плейбука.

Для простой архитектуры

Файлы конфигурации для простой архитектуры находятся в директории `/inventory/single`.

- в корне директории находится файл `inventory.yml`, в котором необходимо вместо шаблона указать адрес хоста, на котором будет производится установка платформы. Например: `ansible_host: 10.172.0.0`;
- в субдиректории `/group_vars/all` находятся файлы, необходимые для настройки созданных хостов.

После настройки файла `inventory.yml` нужно заполнить переменные плейбука в директории `/inventory/single/group_vars/all` в соответствии с Приложением 2. Настройка переменных плейбука.

7.2.3. Проверка корректности заполненного inventory

После того как конфигурационные файлы настроены, на хосте, с которого будет производиться установка, необходимо из директории, в которой находится файл `complex-offline.yml`, выполнить команду:

```
ansible-playbook complex-offline.yml -i <path-to-inventory> -CD
```

где `<path-to-inventory>` – относительный путь до директории, содержащей файл `inventory.yml`.

В результате будет проверена корректность заполнения **inventory**, переменных и доступов до хостов, где будет разворачиваться платформа SimpleOne.

В случае обнаружения ошибок, рекомендуется повторно проверить доступы к хостам и корректность заполнения переменных в **inventory**. До устранения ошибок к следующему пункту не переходить.

7.2.4. Запуск плейбука

После того как была произведена проверка корректности заполнения **inventory**, необходимо из директории, в которой находится файл `complex-offline.yml`, выполнить команду:

```
ansible-playbook complex-offline.yml -i <path-to-inventory>
```

где `<path-to-inventory>` – относительный путь до директории, содержащей файл `inventory.yml`.

В файле `complex-offline.yml` указан порядок разворачивания инфраструктурных компонентов, который можно использовать в случае возникновения проблем на каких-либо этапах установки.

Для запуска плейбука рекомендуется использовать отдельный хост, имеющий связность со средой выполнения приложения.

Если для простой архитектуры запуск будет происходить на хосте, с которого настраивались конфигурационные файлы, то необходимо использовать следующую команду для запуска плейбука:

```
ansible-playbook complex-offline.yml -i <path-to-inventory> -c local
```

где `<path-to-inventory>` – относительный путь до директории, содержащей файл `inventory.yml`, а аргумент `-c local` определяет исполнение на локальном хосте.

Для запуска плейбука под **sudo**, а не **root** нужно указать соответствующего пользователя (`ansible_user: sudo`), и добавить следующие переменные в раздел `all:vars` файла `inventory/[single,distributed]/inventory.yml`:

- `ansible_become: true`
- `ansible_become_password: <user_password>` (опционально).

Для указания паролей непосредственно при выполнении плейбука можно дописать к команде выполнения аргумент `-kK` (вторая K необходима, так как пользователь не **root**).

Если ключ пользователя находится в директории `.ssh/` и имеет стандартное имя (`id_rsa`, `id_ed25519` и т.д.) или если ключ добавлен в **ssh-agent**, то пароль можно не указывать.

7.2.5. Проверка работоспособности экземпляра

Когда работа плейбука завершится, необходимо зайти на адрес экземпляра, который был установлен, и убедиться в том, что открывается страница авторизации. Данные экземпляра и данные для авторизации были указаны в конфигурационных файлах плейбука (файл `/inventory/[single,distributed]/group_vars/all/vars.yml`).

Также для того, чтобы проверить корректность установки экземпляра, необходимо выполнить на всех хостах команду `docker ps -a`. В результате не должно быть незапущенных контейнеров, находящихся в статусах, отличных от статуса **Running (Healthy)**.

В случае, если установка завершилась с ошибкой, необходимо проанализировать информацию об ошибке и самостоятельно ее устранить. Если самостоятельно ошибку устранить не удалось, необходимо направить информацию об ошибке и директорию `/inventory` в службу Customer Care через портал самообслуживания ServiceDesk.

7.3. После установки

7.3.1. Сбор статистики БД и анализ нагрузки

Чтобы понять, на какие пользовательские таблицы и колонки происходит большая нагрузка можно использовать статистику, содержащую сведения о выполненных SQL-запросах. Для этого необходимо использовать инструмент БД **pg_stat_statements**. По умолчанию опция включена.

Если опция **pg_stat_statements** отключена, для ее подключения необходимо иметь доступ в контейнер **postgres-<*hostname*>**. Нужно изменить следующую строку в файле `/home/postgres/pgdata/pgroot/data/postgresql.conf`, которая добавит **pg_stat_statements** в список расширений: `shared_preload_libraries = 'pg_stat_statements'`, и перезапустить PostgreSQL.

По умолчанию создано представление, необходимое для доступа к данным. Чтобы создать его, подключившись к исследованной БД, нужно выполнить команду:

```
CREATE EXTENSION pg_stat_statements
```

Если ранее опция была включена, таблица **pg_stat_statements** будет наполнена данными. Для того, чтобы проанализировать запросы в рамках какой-то группы REST-запросов, предварительно можно инициировать очистку с помощью команды:

```
SELECT * FROM pg_stat_statements ORDER BY total_exec_time/calls DESC;
```

Наиболее трудоемкие запросы будут отображаться в верхней части списка.

Для удобства анализа можно экспортировать данные в csv-файл с помощью следующей команды:

```
docker exec --tty postgres-<hostname> psql -U <db_user_name> <db_name> -c "\COPY (select * from pg_stat_statements ORDER BY total_exec_time/calls DESC) TO '/pg_stat.csv' DELIMITER ',' CSV HEADER;"
```

Для завершения работы с **pg_stat_statements** необходимо:

1. Очистить статистику при необходимости, если опция ранее была отключена: `pg_stat_statements: SELECT pg_stat_statements_reset()`
2. Удалить расширение: `DROP EXTENSION pg_stat_statements`
3. Возвратить строки с `shared_preload_libraries` в файле `postgresql.conf` в исходное состояние.
4. Перезапустить PostgreSQL.

Анализ SQL-запросов.

Для определения медленно выполняющихся запросов необходимо проанализировать колонки **total_exec_time** и **calls**. Отношение этих колонок показывает среднее время выполнения запроса. Настройку индексов базы необходимо проводить только по согласованию с вендором. Самостоятельное создание индексов приведет к нарушению лицензионного договора.



Для поддержания стабильной и быстрой работы необходимо производить периодическое обслуживание БД и таблиц в соответствии с документацией PostgreSQL. Например, средствами VACUUM (вызывает простой).

8. Обновление ПО

8.1. Очередность обновления

Обновление системы рекомендуется производить в следующем порядке:

1. обновление платформы;
2. обновление бизнес-решений;
3. (если не выполнялось ранее) миграция индикаторов и индикаций в новый модуль SLA.

Это обусловлено тем, что некоторые пакеты обновлений бизнес-решений зависят от версии платформы, поэтому в некоторых случаях неверная последовательность обновления продуктов может порождать большое количество коллизий.



Если на экземпляре используются колонки типа **Rich Text**, то при обновлении с версии ниже 1.24.2 на версию 1.24.2 или выше необходимо подготовить данные к миграции в колонки типа **Wysiwyg**, предусмотренной в плейбуке обновления. Для этого выполните следующие действия **перед** запуском плейбука обновления:

1. Перейдите в контейнер BACKEND-API:
   `docker exec -it backend-api bash`
2. Скачайте универсальный (подходящий для любых старых версий) скрипт в директорию `/app/console/controllers`. Для этого выполните команду (после копирования команды из pfd-файла, перед ее вводом в консоль необходимо удалить переносы строк в текстовом редакторе):
   `curl -o /app/console/controllers/RichTextController.php "https://s3-home.simpleone.ru/public-attachment/8/c1/pqlyh2gma73gqfhdtvtkwrzy6ryd5w8?response-content-disposition=inline%3B%20filename%3D%22RichTextController.php%22&response-content-type=application%2Foctet-stream%3B"`
3. Внутри контейнера выполните команду: `php yii rich-text/repair`
4. Дождитесь сообщения `Repair finished`.

После этого вы можете запустить плейбук обновления.

При последующих обновлениях платформы повторно выполнять эти действия не требуется.

8.2. Обновление платформы



Обновление платформы возможно без последовательного обновления через все промежуточные версии. Например, можно обновить платформу с версии 1.23.3 на версию 1.30.0 напрямую, без необходимости запрашивать все промежуточные дистрибутивы и разворачивать их по очереди.

> При этом необходимо обязательно ознакомиться с примечаниями к обновлению для всех промежуточных версий и учитывать их.

Для самостоятельного обновления необходимо запросить у службы Customer Care версию дистрибутива, на которую планируется обновить платформу, и локальные пакеты для обновления Simple Application.

После получения архива с дистрибутивом новой версии необходимо следовать инструкции по установке платформы. Далее, нужно в порядке возрастания загрузить конфигурационные пакеты (SOP-файлы) публичных версий Simple Application, которые являются «дельтой» между текущей версией целевого экземпляра и новой версией.



- Перед обновлением необходимо удостовериться в работоспособности кластеров на экземпляре и самого экземпляра.
- Для обновления платформы на новую версию необходимо заново настроить конфигурационные файлы плейбука новой версии.
- Для обновления на хостах БД нужно в 2 раза больше места, чем занимает сама БД.

8.3. Загрузка пакетов обновлений бизнес-решений

Загрузка пакетов обновлений производится при помощи импорта конфигурационных пакетов, которые представляют собой файлы с расширением **.sop** (SOP-файлы). Последнюю версию любого бизнес-решения можно найти в разделе «*Business Applications*» на портале Service Desk.

Для импорта подготовленного пакета обновлений используются следующие элементы:

- **Retrieved Pack** — содержит SOP-файл для обновления бизнес-решения. В нем содержатся все необходимые записи для реализации пакета конфигурации.
- **Retrieved Records** – содержит записи из SOP-файла, которые являются временными, но равными записям VCS из загруженного конфигурационного пакета. После завершения импорта эти записи будут удалены.
- **VCS Preview Log** — предоставляет сведения о результатах загрузки пакета перед его импортом на экземпляр. Предварительный просмотр записей журнала предупреждает о возможных коллизиях и помогает их исправить.

8.3.1. Загрузка пакетов

Чтобы загрузить пакет обновлений, выполните следующие действия:

1. Перейдите в раздел **Configuration** → **Retrieved Packs**.
2. Нажмите кнопку **New**, чтобы создать новый конфигурационный пакет.
3. Загрузите SOP-файл, нажав значок вложений 📎.
4. Нажмите кнопку **Save** или **Save and exit**, чтобы применить изменения.

5. Нажмите кнопку **Load pack**, чтобы загрузить записи из прикрепленного файла во временную таблицу **Retrieved Records**.

В случае, если вы прикрепили более одного файла, то будут импортированы записи только из последнего пакета. Пакеты, прикрепленные ранее, будут проигнорированы.

Если вы загрузили записи из неправильного пакета, и вам нужно повторно загрузить файл без создания новой записи **Retrieved Pack**, выполните следующие действия:

1. Нажмите на значок вложений 📎.
2. Нажмите **Upload from your device** и загрузите нужный файл.
3. Нажмите **Reload pack**. В результате будут удалены ранее загруженные записи и загружены записи из нового файла.
4. Завершите импорт, нажав **Import pack**.

После корректной загрузки пакета обновления в связном списке можно посмотреть информацию о вставляемых записях.

8.3.2. Подготовка пакета

Когда пакет обновления загружен, вы можете проверить записи, которые будут применены на экземпляре. На вкладке **Retrieved Records** можно найти все записи, загруженные из SOP-файла.

Чтобы просмотреть загруженные записи, выполните следующие действия:

1. Убедитесь, что статус нужной записи **Retrieved Pack** — **Previewed**.
2. Прокрутите вниз до области **Related Lists** и перейдите на вкладку **Retrieved Records**.

После загрузки пакета обновления, некоторые записи VCS могут вызвать ошибки. Чтобы проверить, есть ли какие-либо коллизии или ошибки, выполните следующие действия:

1. Откройте запись **Retrieved Pack**.
2. Прокрутите вниз до области **Related Lists** и откройте вкладку **VCS Preview Log**.
3. Отфильтруйте записи в статусе **Good** с помощью конструктора условий.
4. Если в выборке есть какие-либо записи, обработайте их (см. Устранение коллизий), пока все записи не окажутся в статусе **Good**. Если пакет содержит записи, которые не следует импортировать, измените статус записи на **Skipped**.

8.3.3. Импорт пакета

Чтобы применить пакет обновления в системе, выполните следующие действия:

1. Проверьте состояния записей в связанном списке **VCS Preview Log**:
   - будут применены записи в статусах **Good** и **Warning**;
   - записи в статусах **Collision** и **Skipped** будут игнорироваться системой.

2. Нажмите кнопку **Import pack**.

В результате импорта:

- В текущем экземпляре создается новая запись **VCS Local Pack**, в который копируются записи, созданные на предыдущих этапах.
- Для всех записей в этом локальном пакете установлен флажок **Is current**, который означает, что эта версия является текущей на данный момент.
- Все записи из этого пакета обновлений создаются на целевом экземпляре в соответствующих таблицах.
- Временные записи из таблицы **Retrieved Records** удаляются.
- Состояние записи **Retrieved Pack** изменяется на **Committed**.

После завершения импорта процесс обновления можно считать успешно завершенным.



С версии 1.19.1 размер записей ограничен до 2,5 Мбайт, поэтому большие записи после обновления будут обрезаны до 2,5 Мбайт.

Для скачивания JSON-файла с записями до обновления и обрезки, необходимо использовать ссылку следующего вида:

`https://<домен_хранилища>/public-attachment/0/0/archive_cut_record.json`

Чтобы отключить создание JSON-файла с записями до обновления и обрезки, необходимо указать значение переменной `backend_migration_create_bigdata_log`. Подробнее в разделе Заполнение переменных vars.

8.3.4. Устранение коллизий

При загрузке пакетов могут возникать коллизии – ошибки, которые не позволяют импортировать определенные части пакета. В случае, если они не решаются, а пакет все равно загружается, то те записи, которые создают эти коллизии, переходят в статус **Skipped** и не применяются в системе.

Запись, которая содержит ошибки, можно найти в связанном в списке **VCS Preview Log**. Эти записи содержат причину ошибки в поле **Message** и имеют следующие статусы:

- **Warning** – запись имеет незначительные ошибки, которые будут пропущены или исправлены автоматически;
- **Skipped** – запись будет пропущена при импорте пакета;
- **Collision** – запись не может быть импортирована без дополнительных действий.

Некоторые коллизии можно устранить, выполнив следующую последовательность шагов:

1. В импортируемом пакете конфигурации пролистайте вниз до области **Related Lists** и откройте вкладку **VCS Preview Log**.
2. Отфильтруйте записи в статусе **Good** при помощи конструктора условий.

3. Откройте из выборки запись, с которой возникла проблема.
4. Прочитайте текст в поле **Message**, чтобы найти причину ошибки.
5. Откройте соответствующую запись в списке **Retrieved Records**.
6. Внесите необходимые изменения в полученную запись в поле **JSON Copy**.



В некоторых случаях, особенно когда записи версий нуждаются в сложных исправлениях, предпочтительно настроить ошибочные записи на исходном экземпляре и попытаться импортировать конфигурационный пакет еще раз.

7. Нажмите **Save** или **Save and exit**, чтобы применить изменения.
8. Вернитесь к записи конфигурационного пакета и нажмите кнопку **Prepare changes**.
9. Убедитесь, что статус записи **VCS Preview Log** изменился на **Good**.
10. Повторяйте эти шаги до тех пор, пока все записи во вкладке **VCS Preview Log** не перейдут в статус **Good**.
11. Когда все записи во вкладке **VCS Preview Log** находятся в статусе **Good**, нажмите кнопку **Import pack**.

Ниже рассмотрены виды коллизий и примеры их решений.

В системе и данном конфигурационном пакете нет записи, на которую ссылается добавляемая запись.

Возможное сообщение об ошибке: *Reference {sys_id} in {column_name} does not exist in referenced table {reference_table_name}*.

Решение коллизии: собрать конфигурационный пакет заново на исходном экземпляре, добавив в него элемент, на который ссылается добавляемая запись.

В системе есть ссылка на запись, которая удаляется.

Возможное сообщение об ошибке: *{column_name} column in {reference_table_name} table referenced on this record.*

Решение коллизии: изменить статус коллизии на *Skipped* (данная запись будет пропущена при импорте).

В системе нет записи, которая удаляется.

Решение коллизии: собрать конфигурационный пакет заново на исходном экземпляре или оставить в статусе коллизии (данная запись будет пропущена при импорте).

Не совпадают атрибуты объекта.

Возможное сообщение об ошибке: *Column «manager» related to «user» does not exist*.

Пример:

1. Создаем объект, у которого N-количество полей.
2. Переносим его на экземпляр, где у этого же объекта N-1 количество полей.
3. Получается коллизия и считается, что объект не валидный, обновление/создание такого объекта не будет произведено.

Решение коллизии:

1. Создать/обновить объект без тех атрибутов, которых не существует на экземпляре.
2. Продолжить импорт в состоянии **Warning**. В этом случае колонка не будет создана, а импортированные записи будут иметь набор полей N-1.

Целевой объект обновления на целевом экземпляре новее записи в загружаемом конфигурационном пакете.

Возможное сообщение об ошибке: *The system has a newer record.*

Решение коллизии: при превью такие записи проставляются в статус *Skipped,* на этапе применения записи в статусе *Skipped* пропускаются и остаются как были на экземпляре, если статус был изменен на *Good,* то применяется запись из конфигурационного пакета, затирая существующую на экземпляре.

Удаляемая таблица содержит данные (если таблица без записей, коллизии нет).

Возможное сообщение об ошибке: *Table is not empty.*

Решение коллизии:

1. Измените состояние на **Good**. В этом случае таблица и все записи будут удалены.
2. Сохраните состояние **Skipped**. Таблица останется такой, какая она и была.

Конфигурационный пакет содержит объекты, относящиеся к приложению, которого нет на экземпляре.

Возможное сообщение об ошибке: *Related application "{application_id}" does not exist. Please repack this pack on the source instance.*

Решение коллизии: собрать конфигурационный пакет заново, создав новое приложение на экземпляре.

Нарушение уникальности ключа.

Возможное сообщение об ошибке: *This record has a duplication of a unique key. DETAIL: Key (column_id, record_id, language_id)=(156941403909472422, 159170696512371798, 156628684306200767).*

Решение коллизии:

1. Удалить запись на целевом экземпляре и нажать кнопку **Prepare changes**.
2. Пропустить импорт записи.

Вставляемая колонка существует на некотором уровне наследования.

Возможное сообщение об ошибке: *A record for column A already created in the table X.*

Решение коллизии: изменить статус коллизии на **Skipped** (данная запись будет пропущена при импорте).

9. Настройка поддержки ПО

9.1. Контроль состояния экземпляра

Контроль состояния экземпляра – проверка реальной работоспособности системы в данный момент времени для выявления различных нарушений в ее работе.

Основными направлениями контроля состояния экземпляра являются мониторинг инфраструктуры, мониторинг приложения, а также сбор и анализ логов.

9.1.1. Мониторинг

Осуществляйте мониторинг запрашиваемых ресурсов ВМ у гипервизора. При возникновении инцидентов по производительности будьте готовы предоставить метрики.

Мониторинг инфраструктуры

Мониторинг платформы производится с помощью агентов VictoriaMetrics и Prometheus-совместимых экспортеров, которые входят в состав платформы. Сервер для хранения метрик в состав платформы не входит, необходимо подключить его самостоятельно. Метрики отдаются в формате, совместимом с Prometheus remote write protocol.

Настройка подключения происходит при помощи переменных плейбука. В случае, если настройка производится для уже установленного экземпляра, необходимо заново запустить плейбук, изменив значения переменных для мониторинга.

В состав платформы входят следующие экспортеры:

- **rabbitmq-exporter** (kbudde/rabbitmq-exporter) – сбор метрик брокера сообщений RabbitMQ (список доступных метрик).
- **nginx-exporter** (nginx/nginx-prometheus-exporter) – сбор метрик контейнера **proxy** (список доступных метрик).
- **redis-exporter** (oliver006/redis_exporter) – сбор метрик контейнера **redis** (список доступных метрик).
- **postgres-exporter** (prometheuscommunity/postgres-exporter) – сбор метрик контейнера **postgres** (список доступных метрик).
- **cadvisor** (gcr.io/cadvisor/cadvisor) – сбор метрик контейнеров **docker** (список доступных метрик).
- **node-exporter** (prom/node-exporter) – сбор метрик виртуальной машины (список доступных метрик).
- **jmx-exporter** (prometheus/jmx_exporter) – установлен внутри контейнеров Kafka, Kafka Connect и Kafka Schema Registry. Списки доступных метрик:
  - для Kafka
  - для Kafka Connect

- для Kafka Schema Registry

Схема экспортетов для распределенной архитектуры показана на рисунке ниже.

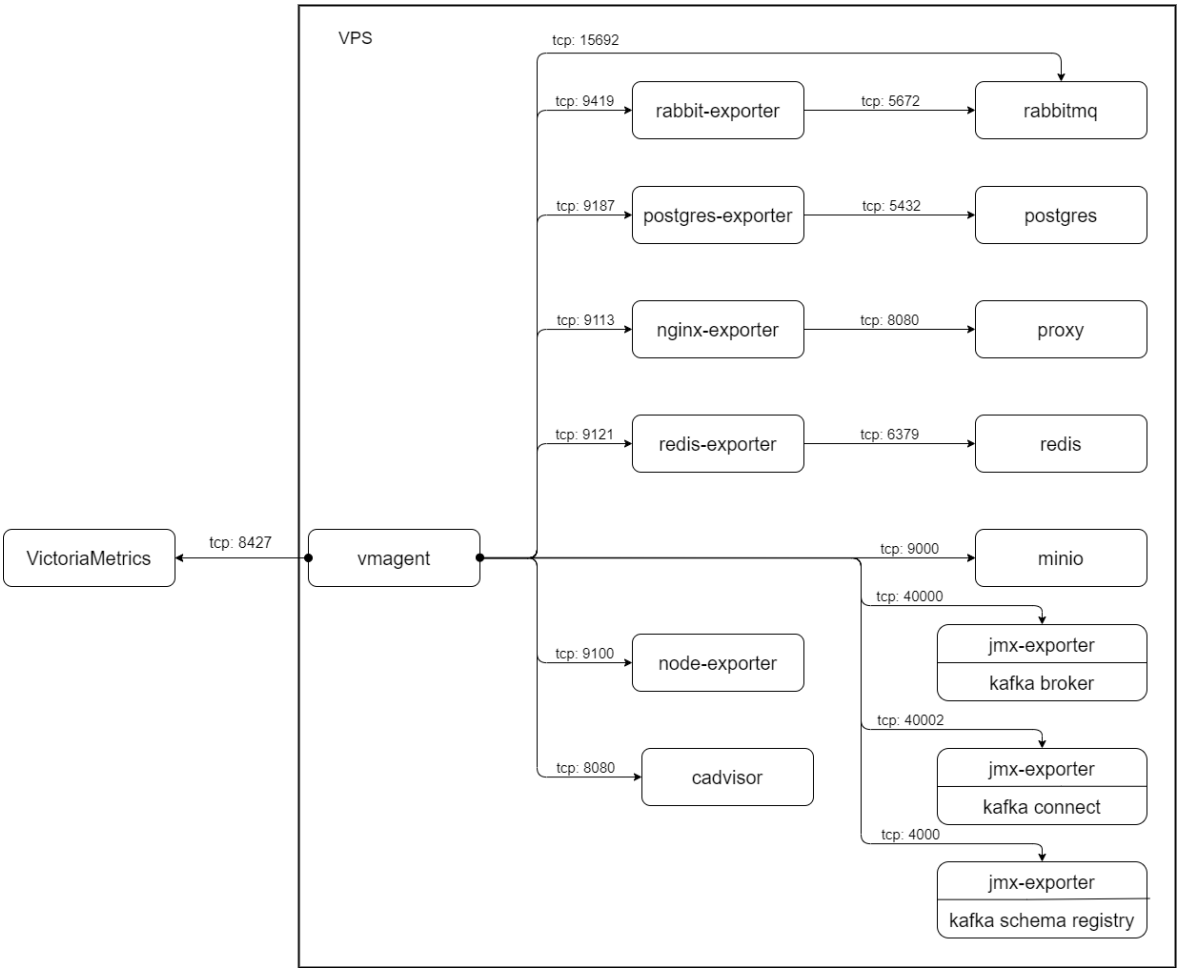


Рисунок 4. Схема экспортеров

Для того, чтобы настроить VictoriaMetrics, Prometheus и Grafana, можно использовать публичную документацию этих продуктов:

- VictoriaMetrics: https://docs.victoriametrics.com
- Prometheus: https://prometheus.io/docs/introduction/overview
- Grafana: https://grafana.com/docs/grafana/latest

У компонентов MinIO метрики отдаются на уровне приложения.

Дашборды инфраструктурных компонентов

- `https://<simple_instance_uri>/tools/backend-haproxy-ui` – показывает доступность компонентов, к которым обращается backend.
- `https://<simple_instance_uri>/tools/rabbitmq-ui` – веб-панель управления **rabbit-mq**, используется для диагностики проблем сервера очереди.
- `https://<simple_instance_uri>/tools/kafka-ui` – веб-панель управления кластером компонентов Kafka, используется для диагностики проблем потоковой передачи данных.

Для открытия интерфейсов диагностики своей подсети необходимо в плейбуке в файле `/inventory/[single,distributed]/group_vars/all/vars.yml` задать переменную по следующему примеру:

```
proxy_infrastructure_dashboards_allowed_networks:

    - "1.1.1.0/24"
    - "18.12.44.55"
```

В ней нужно указать разрешенные подсети или адреса, с которых необходимо получить доступ.

Мониторинг приложения

На данный момент метрики для мониторинга приложения находятся в разработке. Основные метрики будут доступны через http-endpoint-ы и будут передавать данные в формате JSON. Инструмент для обработки метрик выбирается самостоятельно.

9.1.2. Трассировка

Платформа SimpleOne включает в себя инструмент трассировки запросов Jaeger для анализа работы приложения и выявления проблем, связанных с производительностью. Jaeger работает на основе стандарта распределенного трейсинга и мониторинга OpenTelemetry. Установить Jaeger необходимо на хост **ETCD-01**. При этом может потребоваться увеличение ресурсов указанного хоста.



Настройка трассировки запросов во внешнем Jaeger или Prometheus **не поддерживается**.

Трассировка состоит из **спанов** – отдельных действий в системе, которые могут быть вложены друг в друга. Спаны могут содержать:

- Последовательность выполнения запросов.
- Время выполнения запросов.
- Метаинформацию о выполняемом действии: входные параметры, текущую запись, запросы к хранилищам.

Спаны не объединяются в трейсы, они могут быть вложены в родительские спаны. Вложенные спаны доступны при межпроцессном взаимодействии.



Трассируемый запрос выполняется в два раза дольше обычного. Это связано с необходимостью дополнительно отправлять информацию по сети в Jaeger. Поэтому рекомендуется включать трассировку точечно.

Для активации Jaeger на экземпляре, необходимо выполнить следующие шаги:

1. Убедиться, что браузер поддерживает базовую аутентификацию.

2. Установить расширение браузера для настройки Cookie, например, Cookie-Editor.
3. Создать Cookie с названием **APM_ENABLED** и значением **1**. Рекомендуется включать заголовок для выполнения одного запроса в связи с тем, что трассировка существенно влияет на скорость отклика.
4. Открыть Jaeger по ссылке **https://<*simple_instance_uri*>/tools/jaeger-ui**
5. Ввести данные для входа в Jaeger. Эти данные задаются на этапе развертывания приложения в переменных **jaeger_web_login** и **jaeger_web_password** в файле `/inventory/[single,distributed]/group_vars/all/vars.yaml`.

С помощью интеграции Jaeger можно получить информацию о выполнении запросов:

- **Backend API** – запросы на сервер из клиентского приложения. Эти запросы отправляются при использовании агентского интерфейса и пользовательского портала. При включенном Cookie, трассировка этих действий будет сохраняться в Jaeger.
- REST API

Отслеживаемые типы спанов:

- **db** – выполнение запроса к базе данных.
- **redis** – выполнение запроса к Redis.
- **grpc_request** – выполнение запроса по протоколу gRPC.
- **run_engines** – запуск механизмов в соответствии с порядком выполнения бизнес-правил, рабочих процессов, уведомлений, механизмов.
- **run_business_rules** – выполнение блока бизнес-правил.
- **run_async_business_rules** – запуск асинхронных бизнес-правил.
- **run_approval_rules** – запуск правил согласования.
- **run_workflows** – запуск рабочих процессов.
- **run_notifications** – запуск уведомлений.
- **run_business_rule** – выполнение определенного бизнес-правила (можно посмотреть его **sys_id** в метаинформации).
- **trigger_event** – запуск платформенных событий.
- **run_handler** – запуск обработчика события платформы.
- **save_entity** – сохранение записи.
- **delete_entity** – удаление записи.
- **delete_multiple_entities** – множественное удаление записей.
- **batch_insert_entities** – множественное добавление записей.
- **run_script** – общий блок выполнения скрипта, который включает в себя экспорт и импорт переменных и классов, логирование, запуск скрипта.
- **run_script_internal** – запуск скрипта и импорт переменных результата.
- **worker_entrypoint** – выполнение запросов к очередям в RabbitMQ, из которых воркеры

забирают работу.

9.1.3. Настройка системы сбора и анализа логов



Процедура сбора технических логов копии **не входит в оффлайн поставку** и выполняется вручную специалистами заказчика.

Для настройки сбора логов необходимо настроить Docker daemon и настроить логирование.

9.2. Резервное копирование и восстановление данных

Для того чтобы защитить данные от аппаратных сбоев, человеческих ошибок, вирусов и кибератак, необходимо использовать резервные копии. Системы резервного копирования и восстановления данных должны предусматривать простое развертывание и администрирование, удобное управление процессами резервного копирования и восстановления, оперативное восстановление данных.

Ниже приведены некоторые рекомендации по настройке резервного копирования, а также применению резервной копии на экземпляре.

9.2.1. Рекомендации по резервному копированию

1. Резервные копии необходимо хранить на внешних, по отношению к платформе SimpleOne, ресурсах.
2. Резервное копирование на уровне виртуальных хостов рекомендуется производить раз в сутки, а полное резервное копирование, включая хранилище S3 – не реже, чем раз в неделю.
3. Рекомендуемый срок хранения резервных копий – не менее 7 дней.

Настройка резервного копирования

Снятие резервных копий с компонентов баз данных и объектного хранилища, используемых в платформе, будет производиться автоматически в соответствии с параметрами, установленными при конфигурации экземпляра в плейбуке.

Для того чтобы настроить параметры резервного копирования, необходимо:

1. Перейти к файлу, где хранятся значения переменных по умолчанию:
   - `/roles/distributed/postgres/defaults/main.yml` – для высокодоступной архитектуры;
   - `/roles/single/defaults/main.yml` – для простой архитектуры.
2. Для настройки резервного копирования БД в открытом файле указать значения для следующих переменных:
   - `postgres_backup_enabled: true` – активирует резервное копирование;
   - `postgres_backup_s3_endpoint` – адрес S3-хранилища, где будут храниться резервные

копии;

- `postgres_backup_s3_bucket` – имя бакета, в который будет осуществляться резервное копирование БД;
- `postgres_backup_s3_access_key` – ключ доступа к хранилищу;
- `postgres_backup_s3_secret_key` – секретный ключ хранилища;
- `postgres_backup_walg_restore_enabled` – служебная переменная, не рекомендуется изменять;
- `postgres_backup_schedule` – расписание сохранения копий;
- `postgres_backup_days_to_retain` – количество дней для хранения копии;
- `postgres_backup_encryption_enabled` – включение шифрования копии;
- `postgres_backup_encryption_key` – ключ шифрования (опционально).

3. Для осуществления резервного копирования данных, хранящихся в MinIO, рекомендуется использовать rclone или mc (MinIO Client).



Резервное копирование RabbitMQ и перенос данных между экземплярами не предусмотрены из-за технических ограничений. Как следствие, при смене архитектуры с простой на высокодоступную, все необработанные сообщения в очереди будут утеряны.

Восстановление данных



Процедура восстановления БД из резервной копии **не входит в оффлайн поставку** и выполняется вручную специалистами заказчика.

Перед восстановлением данных необходимо выполнить очистку данных Kafka по инструкции, приведенной в пункте Запуск очистки.

После восстановления данных необходимо запустить плейбук деплоя в соответствии с пунктом Запуск приложения после очистки.

9.3. Миграция баз данных



Процедура миграции баз данных **не входит в оффлайн поставку** и выполняется вручную специалистами заказчика.

Возможна миграция БД с версии платформы на ту же самую (рекомендуется) или версию платформы выше. При переносе БД на целевую версию платформы выше текущей после восстановления данных необходимо произвести миграции, запустив тот же плейбук, что используется для установки или обновления.

Перед миграцией необходимо выполнить очистку данных Kafka на целевом экземпляре по инструкции, приведенной в пункте Запуск очистки.

После миграции данных необходимо запустить плейбук деплоя в соответствии с пунктом

Запуск приложения после очистки.

Для миграции данных необходимо:

1. Скопировать данные БД Postgres при помощи утилиты pg_dump. Она находится внутри контейнера с БД PostgreSQL (при использовании PostgreSQL из поставки). Необходимо снять дампы каждой базы внутри PostgreSQL, кроме стандартной базы с именем “**postgres**” и других служебных БД PostgreSQL. Снятие дампов данных для высокодоступной архитектуры рекомендуется осуществлять с узла, имеющего роль **slave** (**replica**).
2. В целевой среде перед влитием данных из дампов необходимо очистить имеющиеся в базах данные, без удаления самих баз. Для этого нужно использовать команду `DROP SCHEMA public CASCADE; CREATE SCHEMA public;` во всех БД, в которые будут вливаться данные из снятых дампов.
3. Если возникает ошибка о том, что нет подходящего пользователя для импорта БД, необходимо создать пользователя командой `CREATE USER simple_user WITH PASSWORD 'postgresrandompassword'`:
   - `simple_user` – имя пользователя.
   - `postgresrandompassword` – пароль пользователя в соответствии с принятой в компании парольной политикой.
4. Скопировать данные MinIO через утилиту rclone. Для этого необходимо подключить к утилите как мигрируемый стенд, так и целевой (подробнее в статье rclone), и запустить синхронизацию с мигрируемого на целевой с помощью rclone sync.
5. Запустить плейбук, используемый для установки или обновления платформы на целевом стенде при необходимости.
6. После клонирования БД, рекомендуется очистить кэш **redis**. Пример команды:
   `docker exec -it redis redis-cli flushall`

На экземпляре с распределенной схемой два контейнера: **active** и **backup**. Необходимо сбросить содержимое **active**, используя команду `redis-cli flushall`. Если подключена другая база, необходимо узнать значение переменной окружения **REDIS_DATABASE** внутри любого backend-контейнера. Для переключения нужно зайти в контейнер **redis**: `docker exec -it redis redis-cli`, переключится в нужную базу `SELECT N` и очистить ее `FLUSHALL`.



Создание снимка (snapshot) виртуальной машины и его выгрузка не является аналогом дампа БД при копировании экземпляра.

9.4. Очистка данных Apache Kafka

Перед восстановлением базы данных, необходимо выполнить остановку контейнеров и очистку данных Kafka.

9.4.1. Запуск очистки

Для простой схемы:

1. Остановить и удалить контейнеры приложений:
   `docker rm -f $(docker ps --format "{{ .Names }}" --filter "label=part-of=backend")`
2. Остановить и удалить все контейнеры, связанные с Kafka:
   `docker rm -f $(docker ps --format "{{ .Names }}" --filter "label=part-of=kafka")`
3. Очистить директории, которые были примонтированы к Kafka, по стандарту это `/data/kafka` и `/data/kafka-logs` (переменные `kafka_datadir`, `kafka_logdir`).

После очистки необходимо проверить доступность базы данных и изменить пароль пользователя базы данных, потому что он мог поменяться. Для этого нужно:

1. Перейти в контейнер **postgresql**:
   `docker exec -it -u postgres postgres /bin/bash` (имя контейнера может отличаться).
2. Выполнить команду `psql`, затем `\password simpleuser` (переменная `postgres_user`) и ввести пароль из переменной `postgres_pass`.

Для высокодоступной схемы:

1. На хостах из группы **backend** остановить и удалить контейнеры приложений:
   `docker rm -f $(docker ps --format "{{ .Names }}" --filter "label=part-of=backend")`
2. На хостах из группы **kafka_cluster**, остановить и удалить все контейнеры, связанные с Kafka:
   `docker rm -f $(docker ps --format "{{ .Names }}" --filter "label=part-of=kafka")`
3. На хостах из подгруппы **kafka_server** очистить директории, которые были примонтированы к Kafka. По стандарту это `/data/kafka` и `/data/kafka-logs`. Переменные (`kafka_datadir`, `kafka_logdir`).

После очистки необходимо проверить доступность базы данных и изменить пароль пользователя базы данных, потому что он мог поменяться. Для этого нужно:

1. Перейти в контейнер **postgresql**:
   `docker exec -it -u postgres postgres /bin/bash` (имя контейнера может отличаться).
2. Выполнить команду `psql`, затем `\password simpleuser` (переменная `postgres_user`) и ввести пароль из переменной `postgres_pass`.

Изменения на master-node целевой базы данных:

1. Удалить слоты репликации. Пример команд:
   `SELECT * FROM pg_replication_slots WHERE slot_type='logical' SELECT pg_drop_replication_slot ('<имя слота>')`
2. Удалить публикации **dbz_publication** для БД: **postgres_simpleone**, **simple_indc_db**, **simple_schedule_job_db**. Внутри контейнера БД выполнить команды:
   ```
   \c postgres_simpleone
   DROP PUBLICATION dbz_publication;
   ```

```
\c simple_indc_db
DROP PUBLICATION dbz_publication;
\c simple_schedule_job_db
DROP PUBLICATION dbz_publication;
```

9.4.2. Запуск приложения после очистки

Запустить плейбук деплоя заново, с тегом **single**:

```
ansible-playbook complex-offline.yml -i <path-to-inventory> -t single
```

где `<path-to-inventory>` – относительный путь до директории, содержащей файл `inventory.yml`.

Для высокодоступной схемы

Запустить плейбук деплоя заново, с тегами **kafka**, **backend**:

```
ansible-playbook complex-offline.yml -i <path-to-inventory> -t kafka, backend
```

где `<path-to-inventory>` – относительный путь до директории, содержащей файл `inventory.yml`.

9.5. Замена SSL-сертификатов

Замена SSL-сертификатов на экземпляре SimpleOne происходит через переменные плейбука (файл `/inventory/[single,distributed]/group_vars/all/vars.yml`), которые настраивались при установке платформы:

- `vault_cert_crt;`
- `vault_cert_key;`
- `vault_additional_ca_cert` (при использовании самоподписанных сертификатов).

После замены сертификатов необходимо запустить плейбук.

9.6. Изменение URL-адреса

Для того, чтобы изменить URL-адрес экземпляра, необходимо в файле `/inventory/[single,distributed]/group_vars/all/vars.yml` изменить значение переменной `simple_instance_url` на требуемый URL-адрес, после – запустить плейбук.



Нельзя менять значение переменной `simple_instance_name` в файле `/inventory/[single,distributed]/group_vars/all/vars.yml`.

9.7. Расширение ресурсов

9.7.1. Увеличение ресурсов хостов

Для того чтобы увеличить мощности хостов, на которых располагается ПО SimpleOne, необходимо:

1. выключить один из узлов кластера;
2. произвести расширение ресурсов (команды для расширения выбираются специалистами самостоятельно в зависимости от выбранной ОС);
3. включить узел.

Если нагрузка по CPU или памяти на все узлы **BACKEND-*** превышает 50% от максимальной, рекомендуется полностью останавливать приложение перед выполнением операции: команда `./down.sh` (для выключения) и `./up.sh` (для включения).

Для увеличения ресурсов узла, на котором располагается БД, необходимо:

1. отключить **slave**-узел и увеличить на нем ресурсы;
2. включить **slave**-узел и сделать его **master**-узлом;
3. повторить операцию для второго узла.

Для расширения ресурсов в случае простой архитектуры необходимо полностью выключить приложение и только после этого выполнять нужные операции.

9.7.2. Добавление новых воркеров

В случае, если количество необработанных сообщений в очереди растет, может потребоваться добавление новых воркеров.

Для этого необходимо в файл `/inventory/[single,distributed]/group_vars/all/backend.yml` плейбука добавить нужное количество воркеров (см. Заполнение переменных backend), после чего заново запустить плейбук.

9.8. Предоставление доступа внешним специалистам

В случае, если для установки, обновления или настройки поддержки ПО привлекаются внешние специалисты, необходимо обеспечить этим специалистам (как 1-й, так и 2-й линии) прямой root доступ до сервера по SSH для проведения установки, либо у учетных записей пользователей должны быть sudo права без ограничений. Допустимо использование VPN или промежуточного сервера с ОС Linux.

Сотрудники 1-й и 2-й линий будут подключаться со следующих IP-адресов:

- 78.140.234.98 (Office SPB)
- 46.243.181.35 (VPN)

10. Приложения

10.1. Приложение 1. Пример списка предустановленного ПО

На всех хостах должно быть предустановлено ПО из списка ниже:

- ca-certificates
- containerd.io
- curl
- docker-ce-cli=24.0.6
- docker-ce=24.0.6
- docker-compose-plugin=2.18.1
- jq
- python3
- python3-pip
- tar
- unzip

На хостах **FRONTEND** необходимо установить **passlib** (через пакетный менеджер pip). Также его нужно установить и для простой архитектуры.

Также на всех хостах должен быть установлен **python3** пакет (через пакетный менеджер pip) docker.

1. Через pip docker-compose необходимо удалить старые пакеты:
   `pip3 uninstall docker-compose`
2. Добавить alias для старой версии docker-compose. Для этого в `/usr/local/bin/docker-compose` нужно разместить скрипт следующего содержания:
   ```
   #!/usr/bin/env bash
   echo -e "\033[1;35m[WARNING] This is aliased command \"docker compose\"\033[0m"
   docker compose $@
   ```

Список пакетов для установки может меняться в зависимости от установленной ОС.

Для RHEL пакеты **containerd**, **docker-ce-cli** и **docker-ce** необходимо устанавливать из репозитория Docker: https://download.docker.com/linux/centos/docker-ce.repo. Указанный файл репозитория необходимо разместить в директории `/etc/yum.repos.d`.

Для ALT Server должны быть установлены следующие версии пакетов:

- Docker 27.1.1, build 59a4d52
- Docker Compose 2.29.7

- Python 3.9.20
- Cryptography 41.0.7

Для Oracle Linux должны быть установлены следующие версии пакетов:

- Cryptography 35.x.x

10.1.1. Рекомендуемые средства диагностики

На хостах для проверки соответствия требованиям можно установить следующие средства диагностики:

- Nload – генератор TCP и UDP трафика.
- Netcat – Unix генератор TCP и UDP трафика.
- iPerf – кросс платформенный инструмент для генерации TCP и UDP трафика.
- sysbench – тестирование производительности для систем Linux.
- atop – полноэкранный монитор производительности.
- htop – вывод на терминал списка запущенных процессов и информации о них.
- bind-utils – набор утилит отправки запросов DNS для получения информации об интернет-хостах.
- curl – кросс-платформенная программа командной строки для взаимодействия с множеством серверов по различным протоколам.
- ncdu – утилита для работы с переполненными дисковыми пространствами.
- lnav – инструмент командной строки для просмотра системных журналов.
- mtr или mtr-tiny – программа для диагностики сети.
- psmisc – пакет утилит для мониторинга процессов и данных.
- tcpdump – утилита, позволяющая перехватывать и анализировать сетевой трафик.

10.2. Приложение 2. Настройка переменных плейбука

10.2.1. Заполнение переменных backend

В файле `/inventory/[single,distributed]/group_vars/all/backend.yml` необходимо настроить следующие параметры:

- `backend_deploy_yii_env` – среда, которая будет разворачиваться. Можно указать `prod`, `dev` или `test`;
- `backend_tls_enabled` – включение шифрования между прокси сервером и серверами с ролью **backend**. По умолчанию выполняется автоматическая единовременная генерация самоподписного сертификата.

При необходимости использовать собственный сертификат, нужно заполнить соответствующие переменные в этом же файле:

- `backend_tls_cert`
- `backend_tls_key`
- `backend_tls_ca`
- `backend_tls_ca_key`
- `backend_tls_ca_passphrase`

В ином случае оставить переменные необъявленными.

Настройка SSO

Для корректной работы SSO с ADFS необходимо заполнить следующие переменные:

- `backend_saml_private_key` – приватный ключ сертификата для подписи запросов к ADFS-серверу. Если значение не указано, переменная автоматически заполняется значением переменной `vault_cert_key`. Используемый сертификат необходимо загрузить на ADFS-сервер.
- `backend_saml_private_key_passphrase` – пароль для приватного ключа сертификата, если он зашифрован.

Настройка воркеров

- `backend_worker_svcs` – переопределяет количество воркеров.

Перечень всех воркеров, которые можно добавить, указан в файле `roles/single/backend/defaults/main/backend_worker.yml` для простой архитектуры и `roles/distributed/backend/worker/defaults/main/backend_worker.yml` для распределенной архитектуры (переменная `backend_worker_svcs_default`).

Пример шаблона для воркеров:

```
backend_worker_svcs:
  echo:
    replica_count: 1
    resources:
      limits: {}
      reservations: {}
```

По умолчанию настроены следующие воркеры:

№	Воркер	Масштабируемый	Описание
1	**worker_attachment_index**	да	Парсинг вложения (и его содержимого при возможности), а также обеспечение полнотекстового поиска.
2	**worker_batch_insert_entities** (устаревший)	да	Обработка очереди с набором данных записей для вставки.
3	**worker_batch_insert_records** (устаревший)	да	Множественное создание записей по данным. Воркер может создавать оба типа записей.
4	**worker_batch_insert_system_entities**	да	Обработка очереди с набором данных записей для вставки.
5	**worker_echo** (устаревший)	да	Вывод в консоль полученного сообщения (используется только для тестирования).
6	**worker_email_parse**	да	Выполнение **sys_inbound_action** для писем, попавших в систему. В сообщениях содержится `sys_id` писем из **sys_email** с типом Incoming.
7	**worker_email_send**	да	Отправка писем из **sys_email** с типом Outgoing.
8	**worker_event** *	да	Запуск событий от имени пользователя, который указан в поле `user_id` записи `sys_event`.
9	**worker_event_sla** *	да	Обработка событий по пересчету индикаций. Воркер используется для поддержки функциональности устаревшего механизма SLA, реализованного в приложении SA. Он перехватывает события **sys_event** с названиями "createIndication", "cancelIndication", "pauseIndication", "resumeIndication", "completeIndication", "resetIndication", "recalculateIndication" и выполнет только их. Периодические пересчеты выполняются через механизм System Events. Когда количество событий на пересчет SLA преобладает над остальными событиями, создается существенный временной лаг. Чтобы исключить влияние событий SLA на другие события, их вынесли в отдельную очередь, которая обрабатывается другим воркером.
10	**worker_handle_external_rabbitmq_message**	да	Используется для выполнения скриптов из записей sys_rabbitmq_consumer при получении сообщения от external-rabbitmq-consumer.

№	Воркер	Масштабируемый	Описание
11	**worker_log_archive_create**	да	Используется для архивирования и очищения записей больших таблиц. Воркер создает архив в таблице **sys_log_archive** из записей в таблицах логов.
12	**worker_log_archive_unpack**	да	Используется для распаковки архивов из таблицы **sys_log_archive** в таблицы логов.
13	**worker_manage_delegation**	да	Запускает и останавливает делегирование по временным меткам.
14	**worker_recreate_search_indices**	да	Пересоздает поисковые индексы в таблице **sys_search_index**. Инициатором процесса является API SimpleSearch в серверных скриптах.
15	**worker_run_business_rule**	да	Запускает асинхронные бизнес правила. Для продуктивных сред необходимо задать не менее 10 воркеров, для остальных – не менее 3.
16	**worker_search_index_record**	да	Используется для обновления поисковых индексов записи в таблице **sys_search_index**, при ее вставке, обновлении и удалении.
17	**worker_to_file** (устаревший)	да	Воркер записи полученного сообщения в файл (используется только для тестирования).
18	**worker_transform_import_set**	нет	Переносит записи из временной таблицы **c_imp_*** в целевую таблицу и запускает скрипты трансформации. Для любой архитектуры нельзя задавать в сумме больше воркеров, чем по умолчанию задано системой, так как это спровоцирует дублирование записей при импорте.
19	**worker_process_by_vcs_pack** *	да	Обеспечивает асинхронную загрузку конфигурационных пакетов (SOP).

* Настройки воркера **process_by_vcs_pack** задаются следующими переменными:

- `rabbitmq_single_active_consumer` – позволяет управлять аргументом очереди `x-single-active-consumer`.
- `rabbitmq_queue_type` – позволяет определять тип очереди, который должен быть создан в RabbitMQ (`classic`, `quorum`, `stream`).

Микросервисы и их назначение:

№	Микросервис	Масштабируемый	Описание
1	**indication-calculator**	да	Основной микросервис, реализующий механизм SLA. Выполняет проверку записей по условиям индикаторов. Создает, управляет и пересчитывает индикации.
2	**list-exporter**	да	Обеспечивает быстрый экспорт списка записей в форматы JSON и XLSX. Экспорт выполняется асинхронно.

№	Микросервис	Масштабируемый	Описание
3	**backend-condition-api**	да	Используется микросервисом **indication-calculator** для проверки соответствия проверяемой записи условиям индикаторов, когда применяются неподдерживаемые операторы, JS-инъекции, опции конструктора условий или REM в условии.
4	**monolith-mediator**	да	Обеспечивает взаимодействие микросервисов с базой данных монолита.
5	**external-rabbitmq-consumer**	да	Получает и обрабатывает сообщения из очередей внешней системы RabbitMQ. Микросервис использует конфигурацию из **sys_rabbitmq_consumer** для подключения и указания целевой очереди. При получении сообщения из вне он отправляет сообщение в очередь внутреннего RabbitMQ, откуда это сообщение на обработку забирает воркер **worker_handle_external_rabbitmq_message**.
6	**lexical-converter**	да	Конвертирует данные в формат Wysiwyg и обратно.
7	**dot-walking**	да	Используется микросервисами (например, **indication-calculator**) для получения значений через dot-walking. Микросервис имеет прямой доступ к базе данных монолита.
8	**delay-queue**	да	Отправляет в очередь сообщение по истечению отложенного времени. Сообщение `routing_key / user_ID / impersonator_ID / queue_message_ID / delay / data`.
9	**external-rabbitmq-producer**	да	Отправляет сообщения во внешнюю очередь RabbitMQ. Его функционирование обеспечивается серверным классом SimpleExternalRabbitMQ и конфигурациями из записей **sys_rabbitmq_producer**.
10	**schedule-job**	да	Запуск отложенных задач, таких как отложенный импорт и отложенный скрипт.
11	**push-notifications**	да	Устанавливает соединение по протоколу Websocket и уведомляет фронтенд о присутствии других пользователей на форме редактирования записи (модуль индикации присутствия).
12	**rule-engine**	да	Запускает механизмы обработки записей после их вставки в базу данных монолита. Сервис является надстройкой над монолитом и выполняет внутри себя нужную часть его функциональности. Он запускается только при CUD операциях в записях таблицы **sys_indication**. Выполняемые механизмы: бизнес-правила с типом "After" и "Async after", рабочие процессы, правила уведомлений и правила согласований.

* При обновлении на версию 1.20.1 и выше, необходимо произвести ребалансировку

воркеров, если она еще не выполнялась:

- При использовании старого модуля SLA
  - Текущее количество воркеров `event` умножить на 0,6 и округлить в большую сторону. Полученное значение задать в переменной `event`;
  - Текущее количество воркеров `event` умножить на 0,4 и округлить в большую сторону. Полученное значение задать в переменной `event_sla`;
- При использовании нового модуля SLA
  - Текущее количество воркеров `event` умножить на 0,6 и округлить в большую сторону. Полученное значение задать в переменной `event`;
  - Установить в переменной `event_sla` значение `0`.

Параметры `reservations` и `limits` должны иметь значение `{ }`.

В случае заполненности очередей при выполнении операций, описанных в перечне, рекомендуется добавить дополнительный воркер в файле `/inventory/[single,distributed]/group_vars/all/backend.yml`.

10.2.2. Заполнение переменных monitoring

По умолчанию мониторинг выключен. Чтобы его включить, необходимо в файле `/inventory/[single,distributed]/group_vars/all/monitoring.yml` изменить значение переменной `monitoring_enabled` на `true`.

При этом заполнить следующие переменные:

- `monitoring_external_labels` – дополнительные метки для метрик мониторинга, принимает в качестве значения list;
- `vmagent_remote_write_address_list` – адрес, на который будут передаваться метрики, принимает в качестве значения list (по шаблону, указанному в файле);
- `vmagent_remote_write_port` – порт, по которому будут передаваться метрики;
- `vmagent_auth_enabled` – необходимость авторизации в prometheus;
- `vmagent_auth_ssl_enabled` – использование ssl;
- `vmagent_basic_auth_user_insert` – пользователь для prometheus;
- `vmagent_basic_auth_pass_insert` – пароль для Prometheus, заполняется вручную в соответствии с принятой в компании парольной политикой;
- `rabbitmq_exporter_user` – имя пользователя для сервиса RabbitMQ; значение должно совпадать со значением переменной `rabbitmq_default_user` в файле `/inventory/[single,distributed]/group_vars/all/vars`.
- `rabbitmq_exporter_password` – пароль для сервиса **rabbitmq**; значение должно совпадать со значением переменной `rabbitmq_default_pass` в файле `/inventory/[single,distributed]/group_vars/all/vars.yml`.

10.2.3. Заполнение переменных system

Для высокодоступной архитектуры в файле `/inventory/[single,distributed]/group_vars/all/system.yml` можно настроить название сети и постфиксы контейнеров, однако лучше оставить значения по умолчанию:

- `docker_network_name` – название сети, используемое в команде `docker network ls`;
- `docker_network_dc_name` – название сети внутри файла `docker-compose`;
- `docker_container_name_postfix_mask` – постфикс, добавляемый к ресурсам контейнеров и сети.

Для простой архитектуры только в переменной `docker_compose_path` указать путь до конфигураций приложения.

10.2.4. Заполнение переменных vars

В файле `/inventory/[single,distributed]/group_vars/all/vars.yml` необходимо произвести следующие настройки:

Настройка приложения

- `simple_instance_name` – краткое название экземпляра. При обновлении не рекомендуется менять значение существующего экземпляра;
- `simple_instance_url` – URL экземпляра;
- `simple_admin_password` – пароль администратора, заполняется вручную значением в соответствии с принятой в компании парольной политикой;
- `client_prefix` – клиентский префикс экземпляра, который отвечает за формат системных названий таблиц, создаваемых в `sys_db_table`. При создании таблиц в наименование таблицы будет добавляться префикс в формате: `{префикс приложения}_{клиентский префикс экземпляра}_`. Например, `hrm_ross_meeting`, где `hrm` – префикс таблиц, указанный в записи приложения `sys_application`, `ross` – клиентский префикс экземпляра, `meeting` – часть наименования таблицы, заданная пользователем. Не рекомендуем использовать клиентские префиксы длиннее **10** символов т.к. длина префикса влияет на длину системных названий создаваемых таблиц, которая имеет ограничение в **63** символа.
- `vendor_id` – идентификатор вендора, который определяет значение одиннадцатой цифры в идентификаторах ID для записей, которые будут создаваться на экземпляре не через импорт конфигурационных пакетов. По умолчанию значение `vendor_id` равно **0**. Данное значение может использоваться только на экземплярах, которые принадлежат компании SimpleOne. На партнерских экземплярах в качестве `vendor_id` должна быть указана цифра в диапазоне **2–8**. Для экземпляров клиента `vendor_id` должен равняться **9**.



Уточнить значения `client_prefix` и `vendor_id` для вашей компании вы можете у вашего менеджера по работе с партнерами (PAM) или через письмо на mailto:servicedesk@simpleone.ru.

При первом развертывании экземпляра возможно запросить изменение `client_prefix` для сокращения его длины, чтобы названия создаваемых

> таблиц были короче. Если вы уже создавали таблицы на вашем экземпляре, клиентский префикс лучше не менять, чтобы сохранить единый формат именования клиентских таблиц.

Настройка хранилища S3

- `s3_url` – url хранилища S3;
- `s3_console_url` – URL консоли хранилища S3; значение заполнять необязательно, но в случае заполнения, оно должно отличаться от значения в переменной `s3_url`;
- `s3_access_key` – ключ доступа для хранилища S3 – заполняется вручную значением в соответствии с принятой в компании парольной политикой;
- `s3_secret_key` – секретный ключ для хранилища S3 – заполняется вручную значением в соответствии с принятой в компании парольной политикой;

Настройка БД

- `postgres_datadir` – директория, где будут находиться данные БД;
- `postgres_su_pass` – пароль суперпользователя БД, заполняется вручную значением в соответствии с принятой в компании парольной политикой;
- `postgres_user` – имя пользователя БД;
- `postgres_pass` – пароль пользователя БД, заполняется вручную значением в соответствии с принятой в компании парольной политикой;
- `postgres_user_ro` – имя пользователя БД только для чтения;
- `postgres_pass_ro` –пароль пользователя БД только для чтения, заполняется вручную значением в соответствии с принятой в компании парольной политикой;
- `postgres_db_name` – название БД;
- `postgres_allow_nossl` – позволяет подключаться к БД без SSL-соединения (по умолчанию `1` – включен).

> 
> Не рекомендуется использовать специальные символы для логина и пароля к БД. Также для имени пользователя зарезервированы следующие значения: **admin**, **cron_admin**, **postgres**, **robot_zmon**, **standby**, **user**, **zalandos** – их использование может привести к ошибкам при разворачивании БД.

Настройка внешней БД

- **postgres_internal** – при использовании внешней БД необходимо установить значение **false**.
- **postgres_external_host** – адрес удаленной БД:
  - '109.145.54.xxx'
    Если не используется **haproxy**, а БД несколько, указывается IP-адрес master-узла.
- **postgres_external_hosts** – адреса удаленных БД, если их несколько:
  - '109.145.54.xxx'

    - '10.121.81.xxx'

- **postgres_external_port** – порт удаленной БД.
- **postgres_external_port_ro** – переменную необходимо заполнить, если у БД доступен отдельный порт с ролью Replica (read-only). Если не используется **haproxy**, значение этой переменной должно совпадать с переменной `postgres_external_port`.
- **postgres_haproxy_enabled** – установить значение `false`, если не требуется использовать **haproxy**. Значение по умолчанию: `true`.

Настройка RabbitMQ

- `rabbitmq_default_vhost` – название хоста по умолчанию;
- `rabbitmq_default_user` – пользователь RabbitMQ;
- `rabbitmq_default_pass` – пароль пользователя, заполняется вручную значением в соответствии с принятой в компании парольной политикой;



Не рекомендуется использовать специальные символы для логина и пароля.

Настройка MinIO

- `minio_tmp_dir` – директория хранения временных файлов MinIO;
- `minio_data_dir` – директория хранения данных MinIO;
- `minio_region` – регион MinIO.

Настройка Kafka

- `kafka_ui_user` – имя пользователя панели управления **kafka-ui**;
- `kafka_ui_password` – пароль пользователя панели управления **kafka-ui**, заполняется вручную значением в соответствии с принятой в компании парольной политикой.
- Переменная `kafka_ui_deploy` со значением `true` должна быть задана на одном из хостов группы **kafka_server** (на одном уровне с переменной `ansible_host`). Переменная заполняется в файле `inventory.yml` только для распределенного экземпляра, на котором должна быть установлена панель управления **kafka-ui**, с соблюдением форматирования, показанного в примере ниже:

```
kafka_cluster:
    children:
        kafka_server:
            hosts:
                kafka0-prod:
                    ansible_host: 10.170.0.230
                    kafka_ui_deploy: true
                kafka1-prod:
                    ansible_host: 10.170.0.229
                kafka2-prod:
                    ansible_host: 10.170.0.234
```



Не рекомендуется использовать специальные символы для логина и пароля.

Настройка Jaeger

- `jaeger_web_login` – имя пользователя панели управления Jaeger;
- `jaeger_web_password` – пароль пользователя панели управления Jaeger, заполняется вручную значением в соответствии с принятой в компании парольной политикой.

Настройка backend

Настройка доступа по нескольким доменным именам

- `backend_api_origins` – укажите список доменных имен, по которым должен быть доступен экземпляр, в следующем формате:

```
backend_api_origins:
    - https://domain1.simpleone.ru
    - https://domain2.mycompany.ru
    - https://domain3.ru
```

По умолчанию значение переменной пустое. При этом кроссдоменные запросы недоступны.

Установите значение *, чтобы разрешить запросы с любого домена.



Из соображений безопасности не рекомендуется разрешать запросы с любого домена.

Чтобы обеспечить доступ к экземпляру по дополнительному домену, необходимо создать отдельный сервер с балансировщиком, на IP которого будет разрешаться дополнительное доменное имя. Балансировщик должен принимать запросы по дополнительному домену, обеспечивать SSL-соединение с соответствующими этому домену сертификатами и выполнять проксирование на основной домен.

Пример конфигурации дополнительного балансировщика с использованием **Nginx**:

```
server {
    listen 443 ssl;
    http2 on;

    server_name second-domain.example.ru;

    ssl_certificate /etc/nginx/certs/second-domain.example.ru.crt;
    ssl_certificate_key /etc/nginx/certs/second-domain.example.ru.key;

    location / {
        proxy_pass https://main-domain.example.ru;
        proxy_ssl_server_name on;
```

```
    }
}
```

Настройка PgBouncer

Добавьте в файл `/inventory/distributed/group_vars/all/vars.yml` переменную `pgbouncer_enabled` со значением `true`, чтобы включить компонент **PgBouncer** на распределенной архитектуре на стороне серверов бэкенда и БД. При включенном **PgBouncer**, запросы к БД будут проходить по цепочке **HAProxy** → **PgBouncer** → **PostgreSQL**.

При повторном развертывании со значением переменной `false`, запросы к БД будут проходить без участия **PgBouncer**.

Настройка экспорта

- `list_exporter_data_request_batch_size` – размер пакетного файла с запросами данных. Значение по умолчанию: `200`.
- `list_exporter_worker_count` – количество воркеров экспорта. Значение по умолчанию: `20` – для простой архитектуры, `10` – для каждого экземпляра микросервиса распределенной архитектуры.

Настройка SSL-сертификатов

- `vault_cert_crt` – полная цепочка сертификатов, в следующем порядке: основной сертификат на покрытие домена, затем промежуточный сертификат и в конце удостоверяющий корневой сертификат;
- `vault_cert_key` – приватный ключ;
- `vault_additional_ca_cert` – собственная цепочка сертификатов, в следующем порядке: промежуточные сертификаты, затем сертификат центра авторизации. Заполняется при использовании самоподписанных сертификатов.



Генерация самоподписного сертификата не входит в функциональность плейбука по развертыванию платформы и выполняется вручную специалистами заказчика.

- В случае, если для S3 нужно указать отдельный сертификат, необходимо использовать переменные `vault_cert_s3_crt` и `vault_cert_s3_key`.

10.3. Приложение 3. Таблица сетевых взаимодействий компонентов с внешними системами

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
PROD							
1	VPS: BACKEND-API-01	PROD:BACKEND-API-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
2	VPS: BACKEND-API-02	PROD:BACKEND-API-02	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
3	VPS: BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
4	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
5	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
6	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
7	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
8	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
9	VPS: BACKEND-API-01	PROD:BACKEND-MICROSERVICES-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
10	VPS: BACKEND-API-02	PROD:BACKEND-MICROSERVICES-02	MS Exchange	MS Exchange	imap, smtp, pop4	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
11	VPS: BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
12	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	MS Exchange	MS Exchange	imap, smtp, pop4	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
13	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
14	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	MS Exchange	MS Exchange	imap, smtp, pop4	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
15	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
16	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	MS Exchange	MS Exchange	imap, smtp, pop4	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
17	VPS: BACKEND-API-01	PROD:BACKEND-API-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
18	VPS: BACKEND-API-02	PROD:BACKEND-API-02	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
19	VPS: BACKEND-MICROSERVICES01	PROD:BACKEND-MICROSERVICES-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
20	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
21	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
22	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
23	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
24	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
25	VPS: FRONTEND-01	PROD:FRONTEND-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
26	VPS: FRONTEND-02	PROD:FRONTEND-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
27	VPS: STORAGE-01	PROD:STORAGE-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
28	VPS: STORAGE-02	PROD:STORAGE-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
29	VPS: STORAGE-03	PROD:STORAGE-03	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
30	VPS: STORAGE-04	PROD:STORAGE-04	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
31	VPS: BACKEND-API-01	PROD:BACKEND-API-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
32	VPS: BACKEND-API-02	PROD:BACKEND-API-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
33	VPS: BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
34	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
35	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
36	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
37	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
38	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
39	VPS: QUEUE-01	PROD:QUEUE-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
40	VPS: QUEUE-02	PROD:QUEUE-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
41	VPS: QUEUE-03	PROD:QUEUE-03	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
42	VPS: QUEUE-04	PROD:QUEUE-04	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
43	VPS: CACHE-01	PROD:CACHE-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
44	VPS: CACHE-02	PROD:CACHE-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
45	VPS: DATABASE-01	PROD:DATABASE-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
46	VPS: DATABASE-02	PROD:DATABASE-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
47	VPS: ETCD-01	PROD:ETCD-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
48	VPS: KAFKA-01	PROD:KAFKA-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
49	VPS: KAFKA-02	PROD:KAFKA-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
50	VPS: KAFKA-03	PROD:KAFKA-03	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
51	VPS: CONNECT-01	PROD:CONNECT-01	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
52	VPS: CONNECT-02	PROD:CONNECT-02	Log Aggregator	Log Aggregator	gelf	12201/udp	Сбор логов контейнеров docker
53	Vendor	vpn gateway	VPS: FRONTEND-01	PROD:FRONTEND-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
54	Vendor	vpn gateway	VPS: FRONTEND-02	PROD:FRONTEND-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
55	Vendor	vpn gateway	VPS: STORAGE-01	PROD:STORAGE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
56	Vendor	vpn gateway	VPS: STORAGE-02	PROD:STORAGE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
57	Vendor	vpn gateway	VPS: STORAGE-03	PROD:STORAGE-03	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
58	Vendor	vpn gateway	VPS: STORAGE-04	PROD:STORAGE-04	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
59	Vendor	vpn gateway	VPS: BACKEND-API-01	PROD:BACKEND-API-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
60	Vendor	vpn gateway	VPS: BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
61	Vendor	vpn gateway	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
62	Vendor	vpn gateway	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
63	Vendor	vpn gateway	VPS: BACKEND-API-02	PROD:BACKEND-API-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
64	Vendor	vpn gateway	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
65	Vendor	vpn gateway	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
66	Vendor	vpn gateway	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
67	Vendor	vpn gateway	VPS: QUEUE-01	PROD:QUEUE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
68	Vendor	vpn gateway	VPS: QUEUE-02	PROD:QUEUE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
69	Vendor	vpn gateway	VPS: QUEUE-03	PROD:QUEUE-03	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
70	Vendor	vpn gateway	VPS: QUEUE-04	PROD:QUEUE-04	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
71	Vendor	vpn gateway	VPS: CACHE-01	PROD:CACHE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
72	Vendor	vpn gateway	VPS: CACHE-02	PROD:CACHE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
73	Vendor	vpn gateway	VPS: DATABASE-01	PROD:DATABASE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
74	Vendor	vpn gateway	VPS: DATABASE-02	PROD:DATABASE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
75	Vendor	vpn gateway	VPS: ETCD-01	PROD:ETCD-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
76	Vendor	vpn gateway	VPS: KAFKA-01	PROD:KAFKA-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
77	Vendor	vpn gateway	VPS: KAFKA-02	PROD:KAFKA-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
78	Vendor	vpn gateway	VPS: KAFKA-03	PROD:KAFKA-03	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
79	Vendor	vpn gateway	VPS: CONNECT-01	PROD:CONNECT-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
80	Vendor	vpn gateway	VPS: CONNECT-02	PROD:CONNECT-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
81	VPN User	vpn gateway	VPS: FRONTEND-01	PROD:FRONTEND-01	http, https	80, 443/tcp	Доступ к экземпляру через web-интерфейс

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
82	VPN User	vpn gateway	VPS: FRONTEND-02	PROD:FRONTEND-02	http, https	80, 444/tcp	Доступ к экземпляру через web-интерфейс
83	VPS: FRONTEND-01	PROD:FRONTEND-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
84	VPS: FRONTEND-02	PROD:FRONTEND-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
85	VPS: STORAGE-01	PROD:STORAGE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
86	VPS: STORAGE-02	PROD:STORAGE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
87	VPS: STORAGE-03	PROD:STORAGE-03	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
88	VPS: STORAGE-04	PROD:STORAGE-04	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
89	VPS: BACKEND-API-01	PROD:BACKEND-API-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
90	VPS: BACKEND-API-02	PROD:BACKEND-API-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
91	VPS: BACKEND-MICROSERVICES-01	PROD:BACKEND-MICROSERVICES-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
92	VPS: BACKEND-MICROSERVICES-02	PROD:BACKEND-MICROSERVICES-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
93	VPS: BACKEND-WORKER-01	PROD:BACKEND-WORKER-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
94	VPS: BACKEND-WORKER-02	PROD:BACKEND-WORKER-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
95	VPS: BACKEND-SCHEDULER-01	PROD:BACKEND-SCHEDULER-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
96	VPS: BACKEND-SCHEDULER-02	PROD:BACKEND-SCHEDULER-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
97	VPS: QUEUE-01	PROD:QUEUE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
98	VPS: QUEUE-02	PROD:QUEUE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
99	VPS: QUEUE-03	PROD:QUEUE-03	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
100	VPS: QUEUE-04	PROD:QUEUE-04	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
101	VPS: CACHE-01	PROD:CACHE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
102	VPS: CACHE-02	PROD:CACHE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
103	VPS: DATABASE-01	PROD:DATABASE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
104	VPS: DATABASE-02	PROD:DATABASE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
105	VPS: ETCD-01	PROD:ETCD-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
106	VPS: KAFKA-01	PROD:KAFKA-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
107	VPS: KAFKA-02	PROD:KAFKA-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
108	VPS: KAFKA-03	PROD:KAFKA-03	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker

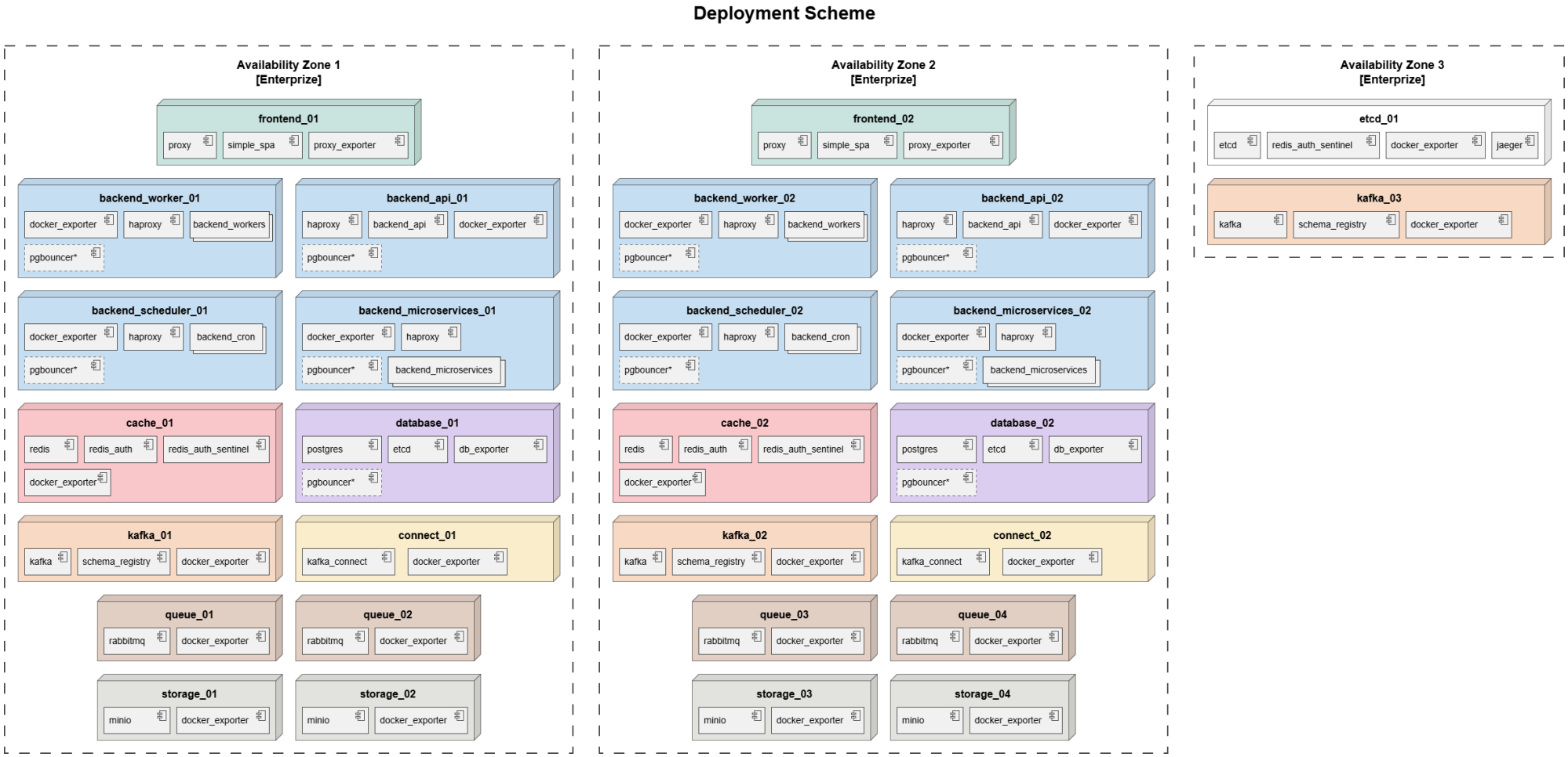
№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
109	VPS: CONNECT-01	PROD:CONNECT-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
110	VPS: CONNECT-02	PROD:CONNECT-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
TEST							
111	VPS: BACKEND-API-01	TEST:BACKEND-API-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
112	VPS: BACKEND-MICROSERVICES-01	TEST:BACKEND-MICROSERVICES-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
113	VPS: BACKEND-WORKER-01	TEST:BACKEND-WORKER-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
114	VPS: BACKEND-SCHEDULER-01	TEST:BACKEND-SCHEDULER-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация
115	VPS: BACKEND-API-01	TEST:BACKEND-API-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
116	VPS: BACKEND-MICROSERVICES-01	TEST:BACKEND-MICROSERVICES-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
117	VPS: BACKEND-WORKER-01	TEST:BACKEND-WORKER-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
118	VPS: BACKEND-SCHEDULER-01	TEST:BACKEND—SCHEDULER-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
119	VPS: BACKEND-API-01	TEST:BACKEND-API-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
120	VPS: BACKEND-MICROSERVICES-01	TEST:BACKEND-MICROSERVICES-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
121	VPS: BACKEND-WORKER-01	TEST:BACKEND-WORKER-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
122	VPS: BACKEND-SCHEDULER-01	TEST:BACKEND-SCHEDULER-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
123	Vendor	vpn gateway	VPS: FRONTEND-01	TEST:FRONTEND-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
124	Vendor	vpn gateway	VPS: STORAGE-01	TEST:STORAGE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
125	Vendor	vpn gateway	VPS: STORAGE-02	TEST:STORAGE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
126	Vendor	vpn gateway	VPS: BACKEND-API-01	TEST:BACKEND-API-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
127	Vendor	vpn gateway	VPS: BACKEND-MICROSERVICES-01	TEST:BACKEND-MICROSERVICES-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
128	Vendor	vpn gateway	VPS: BACKEND-WORKER-01	TEST:BACKEND-WORKER-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
129	Vendor	vpn gateway	VPS: BACKEND-SCHEDULER-01	TEST:BACKEND-SCHEDULER-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
130	Vendor	vpn gateway	VPS: QUEUE-01	TEST:QUEUE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
131	Vendor	vpn gateway	VPS: QUEUE-02	TEST:QUEUE-02	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
132	Vendor	vpn gateway	VPS: CACHE-01	TEST:CACHE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
133	Vendor	vpn gateway	VPS: DATABASE-01	TEST:DATABASE-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
134	VPN User	vpn gateway	VPS: FRONTEND-01	TEST:FRONTEND-01	http, https	80, 443/tcp	Доступ к экземпляру через web-интерфейс

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
135	VPS: FRONTEND-01	TEST:FRONTEND-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
136	VPS: STORAGE-01	TEST:STORAGE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
137	VPS: STORAGE-02	TEST:STORAGE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
138	VPS: BACKEND-API-01	TEST:BACKEND-API-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
139	VPS: BACKEND-MICROSERVICES-01	TEST:BACKEND-MICROSERVICES-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
140	VPS: BACKEND-WORKER-01	TEST:BACKEND-WORKER-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
141	VPS: BACKEND-SCHEDULER-01	TEST:BACKEND-SCHEDULER-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
142	VPS: QUEUE-01	TEST:QUEUE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
143	VPS: QUEUE-02	TEST:QUEUE-02	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
144	VPS: CACHE-01	TEST:CACHE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
145	VPS: DATABASE-01	TEST:DATABASE-01	VM Server	Metrics Aggregator	http, https	8427/tcp	Сбор метрик контейнеров docker
146	VPS: DATABASE-01	TEST:DATABASE-01	SimpleOne Container Registry	reg01.simpleone.ru	http, https	80, 443/tcp	Хранилище контейнеров SimpleOne
DEV							
147	VPS: DEV-01	DEV:DEV-01	ADFS	ADFS	http, https	80, 443/tcp	SSO аутентификация

№	Описание источника	IP-адрес источника	Описание адреса назначения	IP-адрес назначения	Протокол	Порт назначения	Описание
148	VPS: DEV-01	DEV:DEV-01	MS Exchange	MS Exchange	imap, smtp, pop3	25, 587, 465, 110, 995, 143, 993/tcp	Входящая/исходящая почта
149	VPS: DEV-01	DEV:DEV-01	Active Directory	AD	ldap, ldaps	389, 636/tcp	Импорт данных из AD, аутентификация через AD
150	Vendor	vpn gateway	VPS: DEV-01	DEV:DEV-01	ssh	22/tcp	ssh доступ к инфраструктуре для установки обновлений
151	VPN User	vpn gateway	VPS: DEV-01	DEV:DEV-01	http, https	80, 443/tcp	Доступ к экземпляру через web-интерфейс

10.4. Приложение 4. Высокодоступная схема, контур PROD



* Компонент **PgBouncer** присутствует только если при развертывании экземпляра была задана соответствующая переменная.

Рисунок 5. Высокодоступная схема развертывания