Программа «Управление балансами пользователей “Forward BMS”»

Правообладатель ООО «Форвард-Телеком»

Forward BMS (Forward Balance Management System) – решение, выполняющее функции управления балансами пользователей.

Основное назначение - организация эффективного управления балансами в многобалансовой системе, когда ресурсы, доступные пользователю, могут быть выражены не только денежными средствами, но и в виде баллов, единиц товаров, SMS, байтов, секунд и так далее. При этом операции над балансами могут осуществляться несколькими внешними системами одновременно. Управление балансами происходит в режиме реального времени.

Областью применения системы является обслуживание компаний, относящихся к разным областям деятельности, предоставляющих услуги, включая услуги связи, или поставляющих товары по предоплатной / постоплатной схеме, при этом решение может применяться совместно с расчетными и платежными системами. Решение предоставляет широкие возможности, если компания ставит перед собой задачи контроля расходов в режиме реального времени, построения оптимального подхода в предоставлении услуг с ограничивающими условиями.

Контактная информация: support@fw-t.ru

1 Распространение программного обеспечения

Дистрибутив комплекта программ распространяется на физических носителях (USB-flash накопителе) или по ссылке на облачное хранилище.

Дистрибутив состоит из набора компонентов, дополнительных модулей расширения функционала, распространяемых библиотек.

Дистрибутив формируется для каждого заказчика с учетом технологической экосистемы и необходимых модулей расширения функционала

Подробная документация поставляется после адаптации системы в процессе внедрения с учетом особенностей кастомизации.

2 Содержание дистрибутива программного обеспечения

Перечень устанавливаемых дополнительных компонентов осуществляется в соответствии со спецификацией поставки. Дистрибутивы в комплекте поставки содержат компоненты:

1. библиотеки и зависимости для tarantool-cartridge;
2. yaml файл топологии кластера;
3. модули расширений функционала, включая описание внутренней структуры;

Дистрибутивы могут поставляться как архив, пакет или docker-образ и обеспечивают в дальнейшем корректную установку Forward BMS.

3 Порядок установки программного обеспечения

Ввиду вариативности используемого технологического стека на стороне заказчика и, как следствие, – невозможности создания единой инструкции по установке – установку и первичную настройку программного обеспечения комплекса выполняют специалисты Forward.

Оборудование (сервер) должен удовлетворять требованиям в соответствии с приложением к договору (требования к аппаратной части сервера зависят от предполагаемой нагрузки сервера).

Программное обеспечение системы непрерывно развивается и с каждой версией дополняется новым функционалом. Обновление всех или отдельных программных компонентов системы, а также добавление дополнительных модулей расширения функционала производится специалистами Forward.

Специалистами Forward проводится групповое обучение специалистов Заказчика по развертыванию и настройкам системы в целях обеспечения её дальнейшей корректной эксплуатации.

3.1 Подготовка к установке

Forward BMS использует для работы решение Kubernetes.

1. Перед установкой модулей Forward BMS необходимо убедиться, что на целевом сервере установлены tarantool-operator и tarantool-cartridge;

2. Наличие доступа к неймспейсу в кластере, наличие нод для работы.

3.2 Установка Forward BMS

Порядок инсталляции Forward BMS представляет собой последовательное выполнение действий по:

1. Подготовке сервера к использованию Tarantool;

2. Развороту кластера. Топология кластера описывается в yaml файле.

Образ контейнера docker с решением Forward BMS интегрируется в среду тарантул tarantool-cartridge.

3.3 Базовая проверка корректности установки Forward BMS

В /opt/tarantool/test-script/ расположены скрипты для минимальной проверки работы Forward BMS по нативному протоколу:

1. owner_test.py - позволяет проверить функции создания/получения/удаления владельца
2. balance_test.py - позволяет проверить функции создания/получения/удаления/редактирования баланса

3. income_test.py - позволяет проверить функции создания/получения/усечения пополнения
4. owner_reserve_test.py - позволяет проверить функции создания/получения/отмены/подтверждения резервов владельца.

Любой из скриптов запускается из консоли следующей командой

```
python owner_test.py
python balance_test.py
python income_test.py
python owner_reserve_test.py
```

При запуске скрипта в вывод консоли будет написан результат выполнения. По факту выполнения будет создана тестируемая сущность, получена информация по ней, проведены манипуляции и в конечном итоге удалена. Ниже приведен результат выполнения.

```
forward@BMS-DEMO:/opt/tarantool/test-script$ python owner_test.py
Start test add_owner with ident new_owner_1698253781.283734
Result:  {'bucket_id': 27215, 'balances': [], 'owner_ident':
'new_owner_1698253781.283734', 'owner_id': UUID('cf11a9cb-e94c-4929-9eb4-
de503d99d68d')}
Start test get_owner with ident new_owner_1698253781.283734
Result:  {'bucket_id': 27215, 'balances': [], 'owner_ident':
'new_owner_1698253781.283734', 'owner_id': UUID('cf11a9cb-e94c-4929-9eb4-
de503d99d68d')}
Start test delete_owner with ident new_owner_1698253781.283734
Result:  {'bucket_id': 27215, 'owner_ident': 'new_owner_1698253781.283734',
'owner_id': UUID('cf11a9cb-e94c-4929-9eb4-de503d99d68d')}
Start test get_owner with ident new_owner_1698253781.283734
Result:  {'code': -1, 'reason': 'Owner new_owner_1698253781.283734 not found'}
```

4 Эксплуатация системы

4.1 Общие сведения

В период пуско-наладки специалистам Forward требуется предоставление доступа на сервер системы.

Специалистами Forward может быть проведено групповое обучение специалистов Заказчика по администрированию и эксплуатации системы с предоставлением необходимых инструкций в условиях реализации конкретного проекта.

Администрирование Forward BMS производится эксплуатирующей организацией с привлечением, при необходимости, специалистов технической поддержки и разработчиков Forward.

Данная документация может не отражать описания некоторых последних модификаций Forward BMS.

4.2 Архитектура

Все компоненты системы Forward BMS разнесены и зарезервированы, что обусловлено необходимостью обеспечить высокую отказоустойчивость для систем данного класса с использованием резервирования как на программном, так и на аппаратном уровне. По сравнению с тарификатором у BMS более простая логика, системе необходимо быстро и надежно выполнять простые операции над большим количеством данных. Для этих целей используются возможности Tarantool Cartridge с применением Lua модулей (ролей).

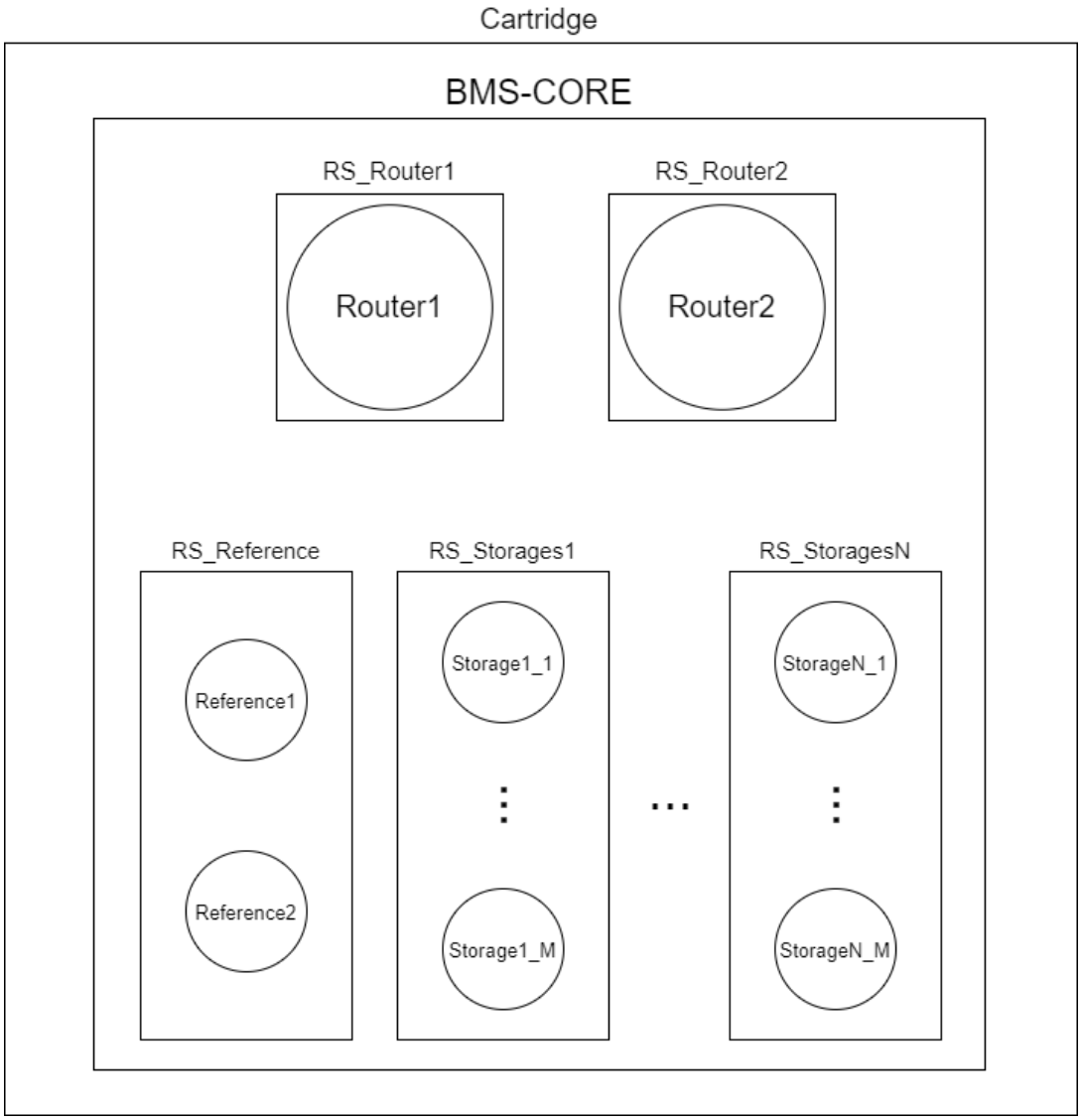


Рисунок 1 – Схема решения BMS

Типичный кластер Tarantool Cartridge для Forward BMS содержит следующие составные части:

- два набора реплик Router, реализующих API для работы модулей BMS-ON-I и BMS-OFF-I.
- N наборов реплик Storage, для хранения данных.
- Структура Кластера может упрощаться или усложнятся в зависимости от поставленных задач и предъявляемых требований заказчика.

Подразумевается, что BMS работает с некоторым количеством внешних систем, которые наполняют ее данными, в частности, создают пользователей, их балансы, пополнения и резервы. Данное взаимодействие осуществляется двумя способами через API:

- Нативный протокол (обращение с помощью скриптов на Lua при подключении к ноде с ролью router) – см. Приложение А;
- REST (более универсальный протокол) – см.Приложение Б.

Все запросы поступают на одну из нод с ролью router. Запрос маршрутизируется на один из шардов по ключу шардирования в master ноду. Master обрабатывает полученный запрос и возвращает ответ ноде router, которая в свою очередь дает ответ клиенту. После успешной обработки данных master нодой происходит репликация данных на slave. Репликация может проходить асинхронно или синхронно, в зависимости от настроек space.

Каждая нода знает текущую топологию всего кластера.

За актуальным состоянием master нод каждого шарда следит один router нод, которая наделена ролью failover-coordinator. Если один из шардов теряет master ноду, то router оперативно переключит одну из slave нод в режим master. После переключения изменение топологии кластера будет распространенно по всем нодам.

4.3 Интерфейс BMS

Интерфейс Forward BMS позволяет отслеживать работоспособность нод кластера, а также выполнять операции переключения, распределения данных между нодами и т.д.

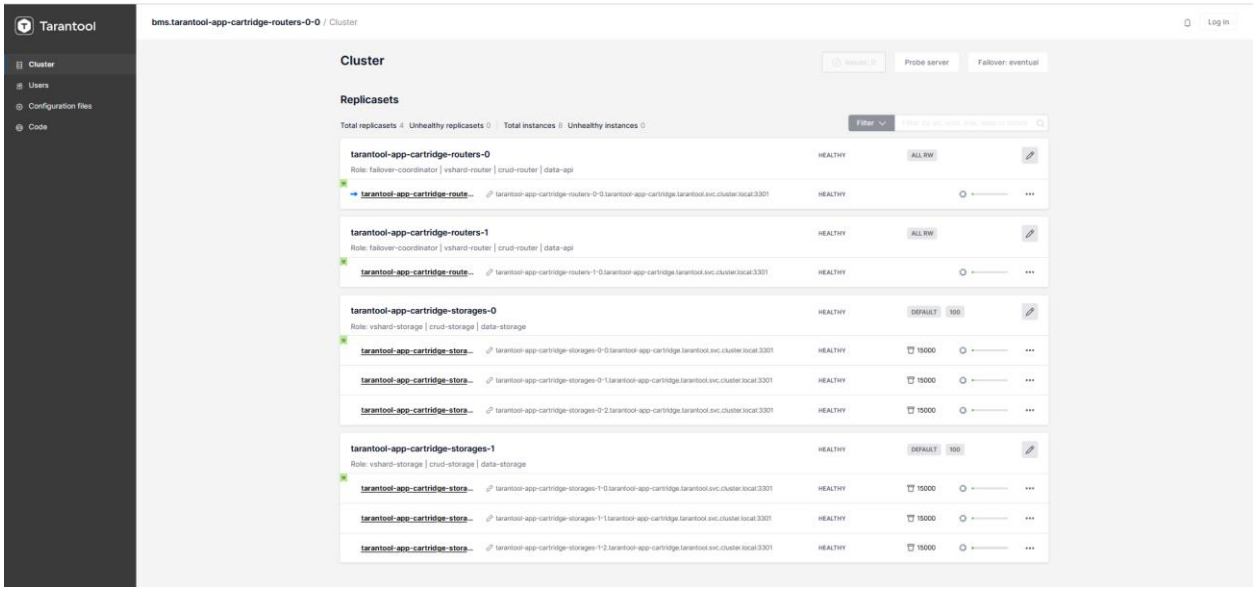


Рисунок 2 – Интерфейс управления кластером BMS

Синяя стрелочка на уровне ноды сообщает о том, что мы зашли в Web-интерфейс через этот элемент (см. Рисунок 3):

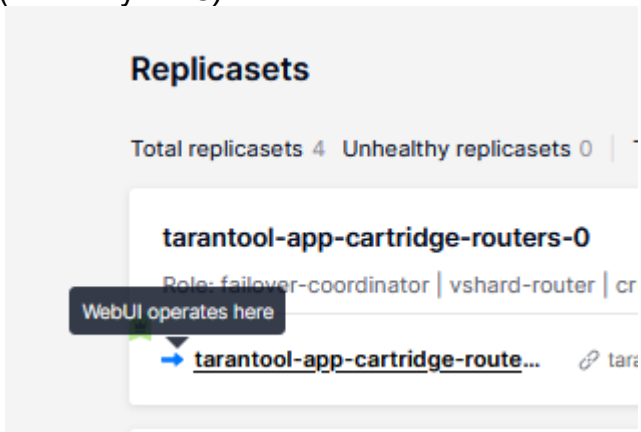


Рисунок 3 – Информация о входе в WebUI

4.3.1 Роутеры

Первые два блока, изображенные на Рисунок 2 – это роутеры (см.Рисунок 4), которые могут маршрутизировать запросы от внешних систем к шардам с ролью сторадж (см.Рисунок 5)

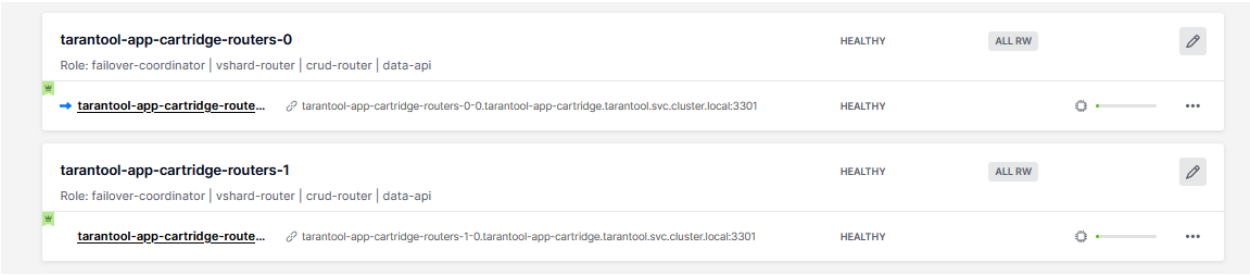


Рисунок 4 – Роутеры

Роутеров, как и стораджей может быть произвольное количество.

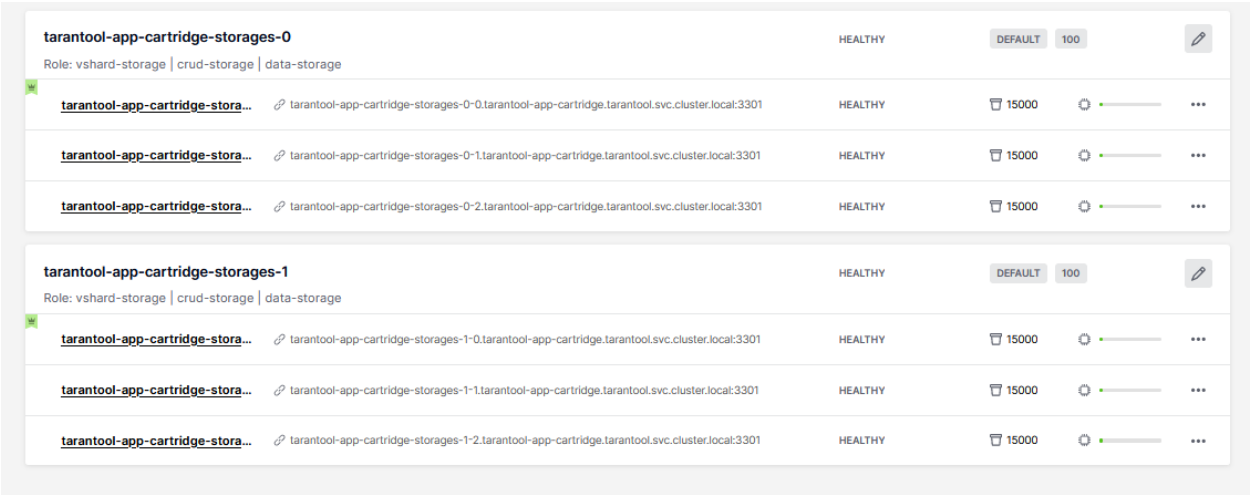


Рисунок 5 – Стораджи

Под названием каждого блока перечислены роли, которые назначены этому блоку (Рисунок 6):

Рисунок 6 – Роли

Роли для роутеров:

- failover-coordinator – отвечает за поддержание всех нод «здоровыми» (Healthy) и за переназначение мастеров;
- vshard-router – обеспечивает маршрутизацию из роутера к стораджам;
- crud-router – надстройка над vshard-router (нужна для внутренних задач системы);
- data-api – пользовательская роль Forward BMS, реализующая функции API.

У каждой ноды есть обозначение, является ли она мастером (зеленая «корона», при наведении на которую раскрывается слово Leader) в своем реплика-сете – см.Рисунок 7.

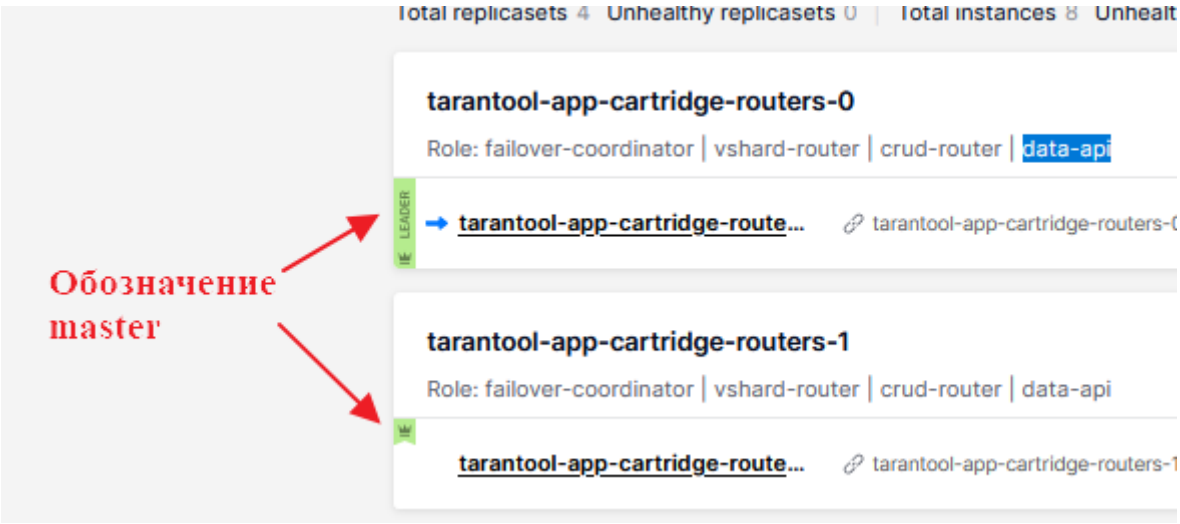


Рисунок 7 – Признак мастер-ноды

Для каждой ноды указываются такие дополнительные параметры (см.Рисунок 8), как:

1) Адрес
2) Состояние
3) Режим работы
4) Заполненность оперативной памяти

Рисунок 8 – Дополнительные параметры ноды

При раскрытии ноды на редактирование  (см.Рисунок 9) есть возможность изменения наименования ноды и ряда дополнительных параметров.

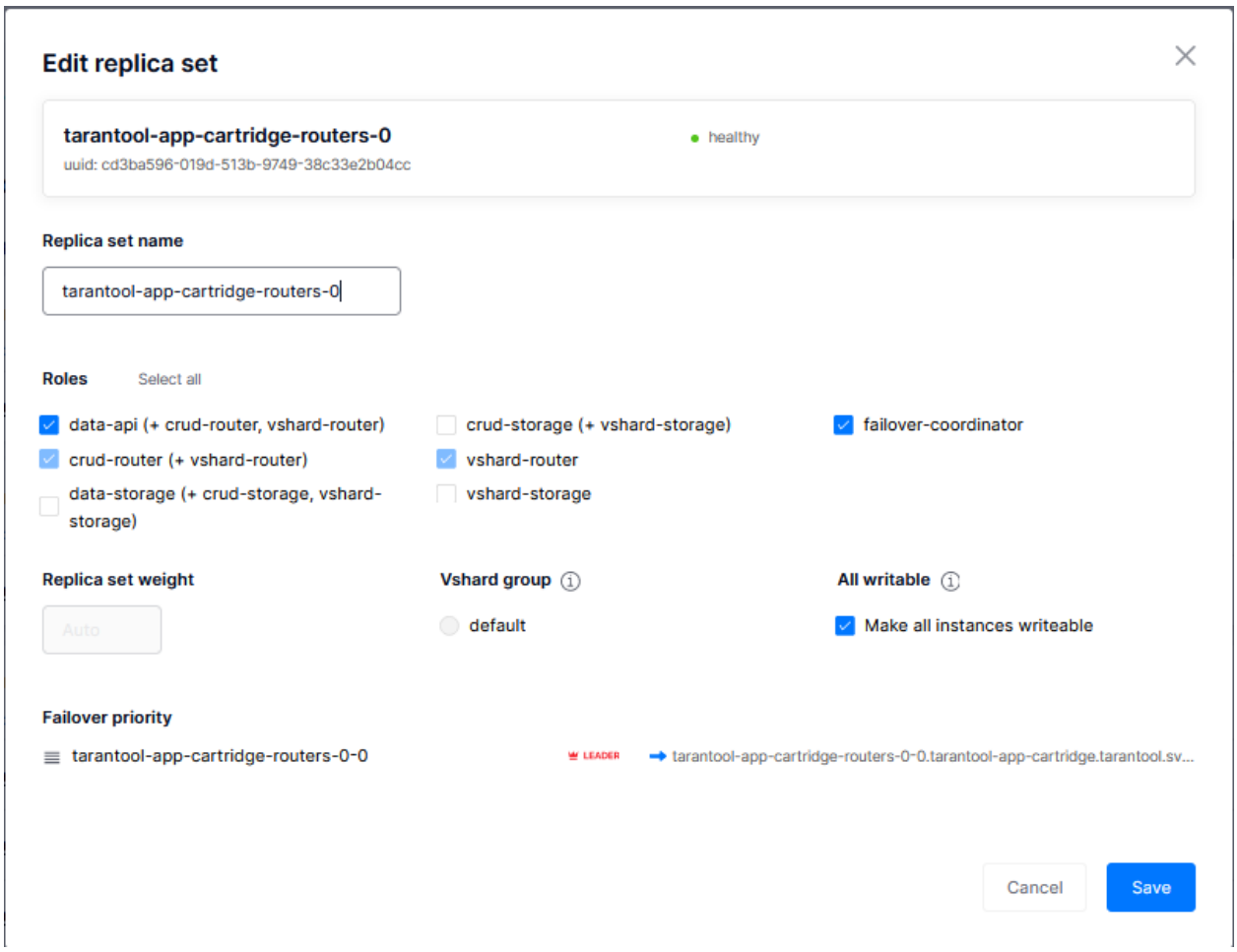


Рисунок 9 – Интерфейс редактирования ноды

Также для каждой ноды можно вызвать контекстное меню для дополнительных операций (Рисунок 10):

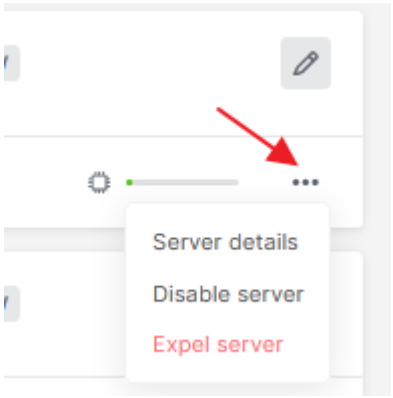


Рисунок 10 – Дополнительные операции

Server details – это детальная информация по серверу с настройками репликации и т.д.(см. Рисунок 11):
Disable server – временно отключить сервер
Expel server – полностью исключить сервер, так что его нельзя будет восстановить.

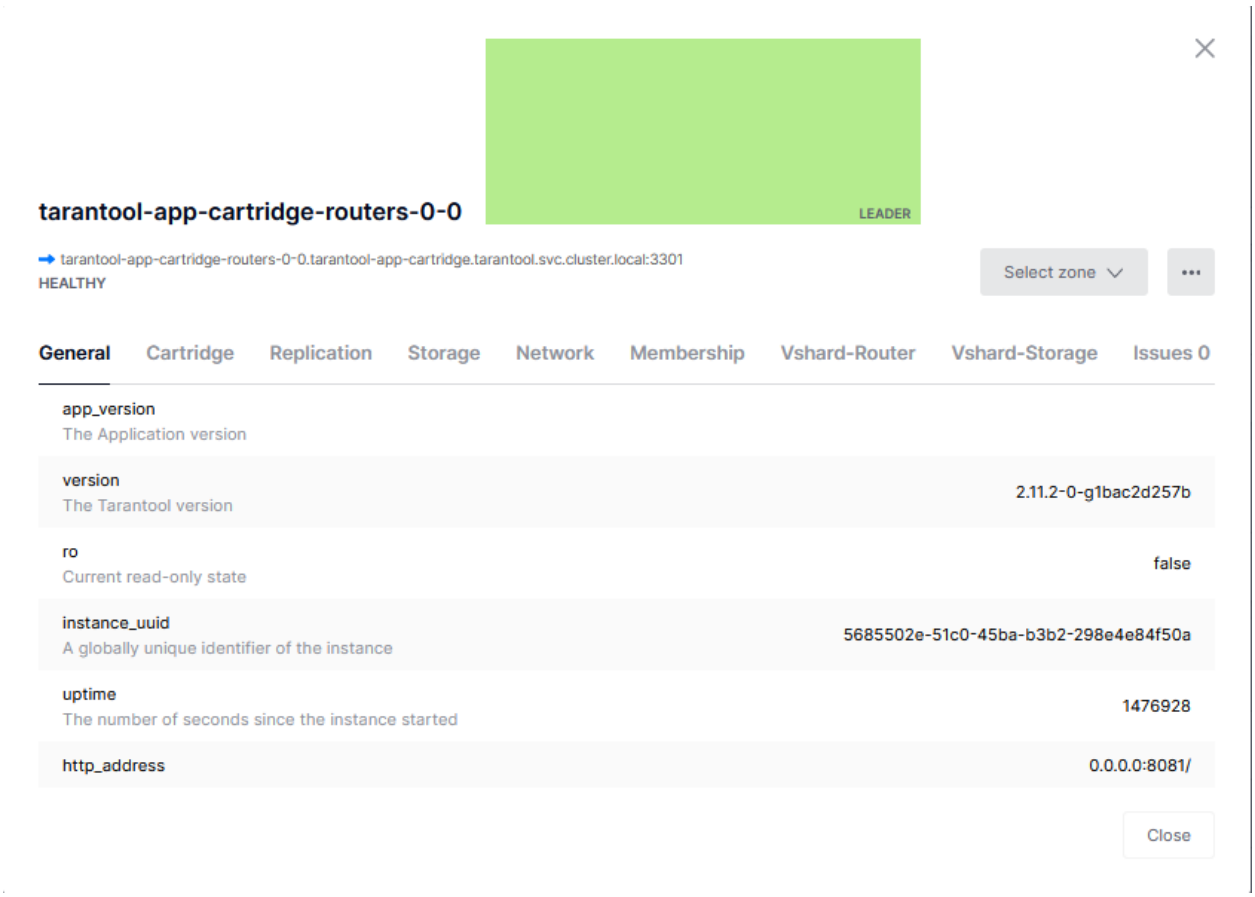


Рисунок 11 – Параметры server details

4.3.2 Стораджи

Стораджей, как и роутеров может быть произвольное количество.

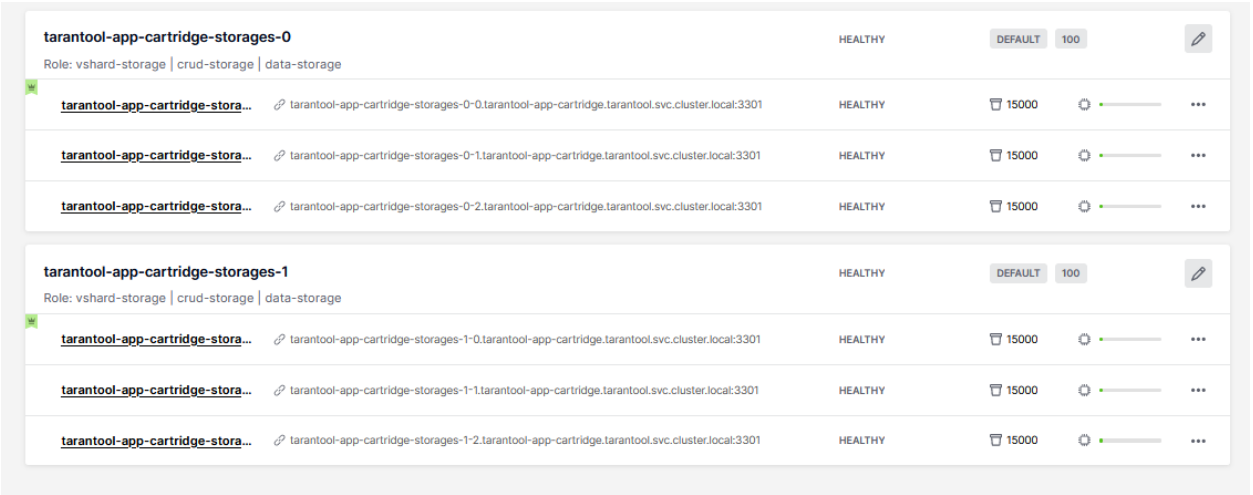


Рисунок 12 – Стораджи

Месторасположение элементов в описании стораджей совпадает с месторасположением этих элементов в описании роутеров (см. 4.3.1), поэтому в данном разделе будут рассмотрены отличительные особенности.

Роли для стораджей:

- vshard-storage – основная роль, которая занимается шардированием данных внутри набора реплик

- crud-storage - надстройка над vshard-storage
- data-storage - пользовательская роль Forward BMS, которая реализует функции по работе с полученными данными.

Параметр Total bucket (см.Рисунок 13): по умолчанию данные в картридже в кластере делятся на 30 000 корзин (bucket), которые по умолчанию равномерно распределяются между реплика-сетами (по 15 000 в случае 2 реплик-сетов (шардов)) – см.Рисунок 14 .

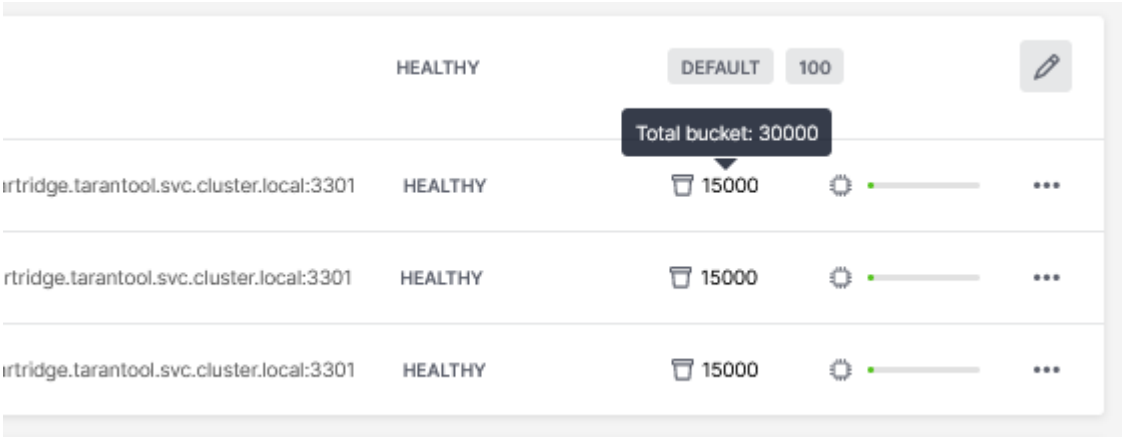


Рисунок 13 – Total bucket

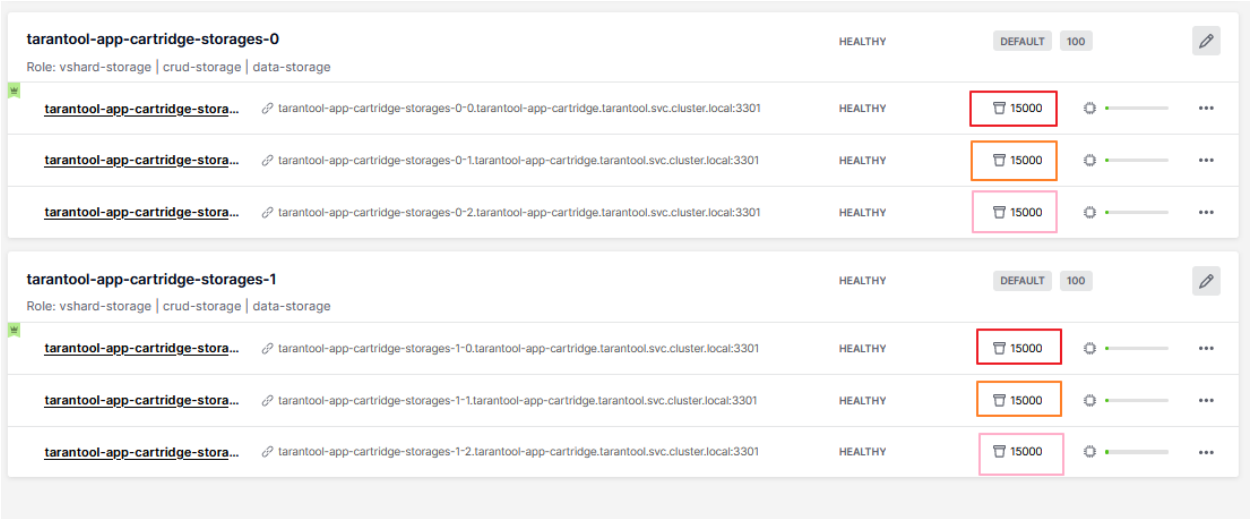


Рисунок 14 – Распределение данных по 15 000 по replicasets

Для каких-то данных, например, для владельца – выбирается поле или набор полей, которые будут представлять собой ключ шардирования (в простейшем случае – это хэш от значений, который может принимать значение от 0 до 30 000). Когда роутер рассчитает это значение, он может извлечь данные из нужного шарда и, также, отправить данные туда по этому ключу.

Параметр Storage group (см.Рисунок 15) в рамках решения использует значение Default.

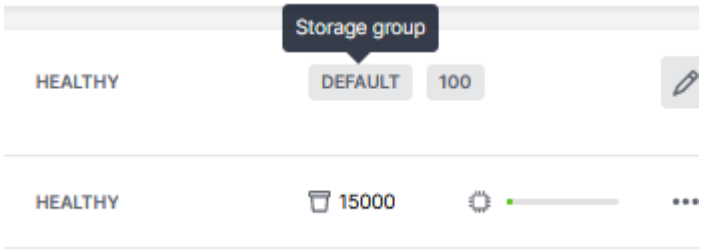


Рисунок 15 – Параметр группа стораджа

Параметр Replica set weight (см.Рисунок 16) – это соотношение того, сколько корзин по отношению к другим шардам будет храниться в этом шарде. Расчет выполняется следующим образом: надо сложить все веса, разделить количество корзин на общий вес и единица веса будет отвечать за получившееся кол-во корзин.

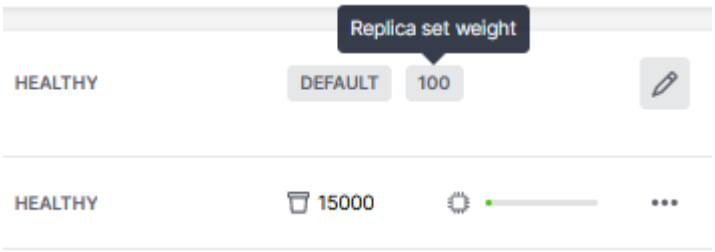


Рисунок 16 – Параметр вес реплика-сета

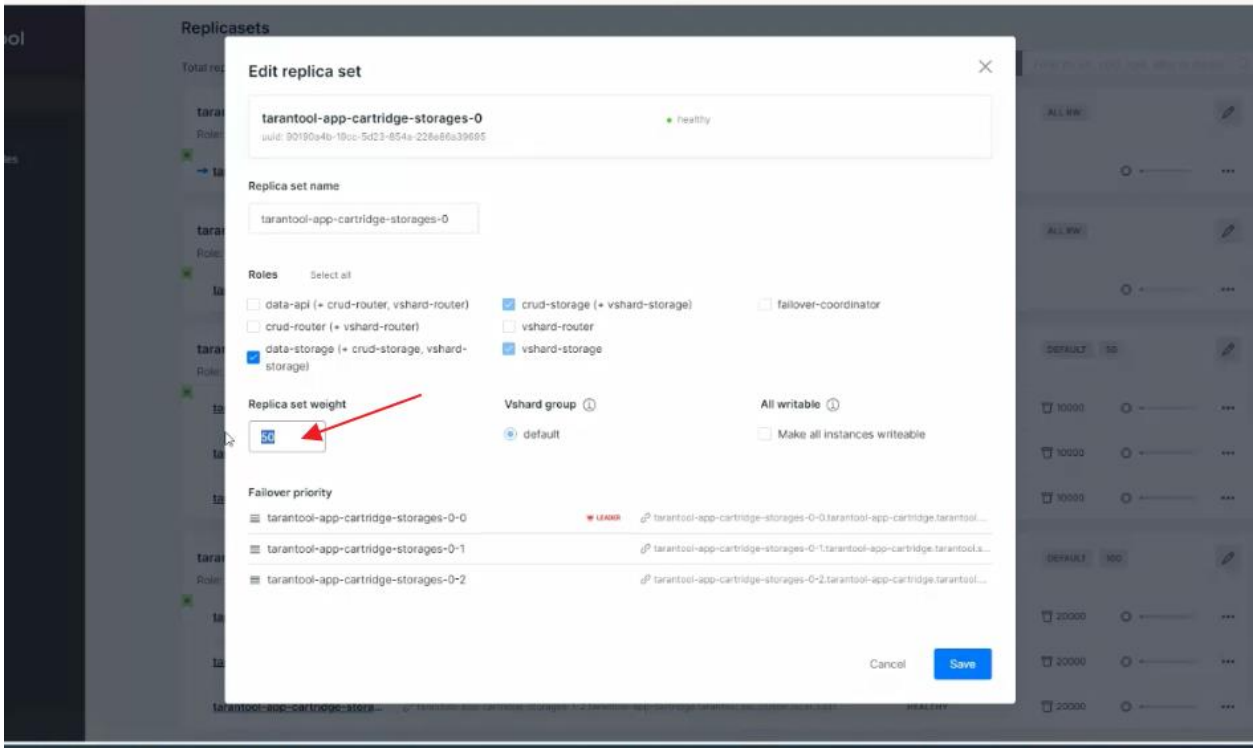


Рисунок 17 – Изменение параметра вес реплика-сета

Так, например, при весах 100/100, мы получим деление по 15 000 корзин.
При весах 50/100 деление корзин будет в соотношении 10 000/ 20 000 (см. Рисунок 18):
При весах 0/100 деление корзин будет в соотношении 0/30 000

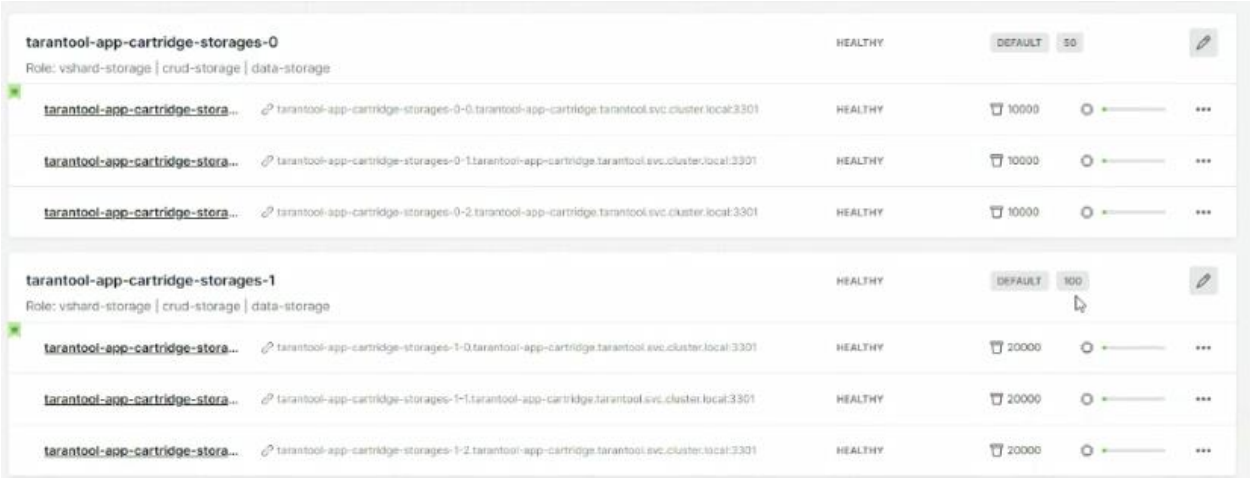


Рисунок 18 – Влияние веса реплика-сета на кол-во корзин

Изменение Failover priority – при открытии на редактирование параметров реплика-сета стораджа в нижней части интерфейса будут видны все стораджи этого реплика-сета. Перетаскиванием мышки друг относительно друга, любой из стораджей можно сделать мастером в пределах шарда, переместив его на самый верх списка (см. Рисунок 19):

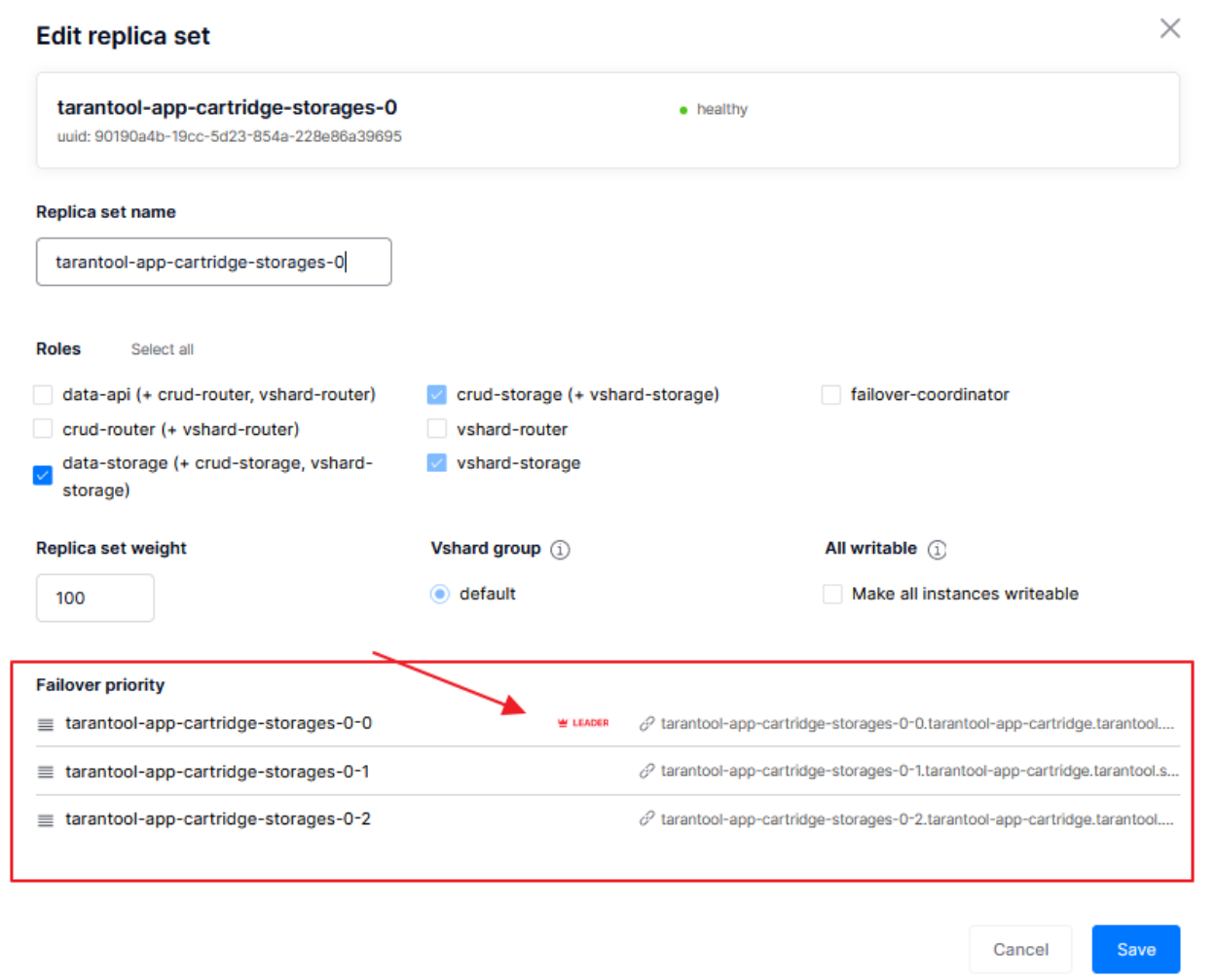


Рисунок 19 – Область Failover priority при редактировании replica set

4.3.3 Пользователи

По умолчанию Tarantool предоставляет модуль пользователей, который имеет смысл при включении авторизации. Если авторизация включена, доступ в веб-интерфейс Forward BMS будет осуществляться по логину/паролю для каждого пользователя (см. Рисунок 20)

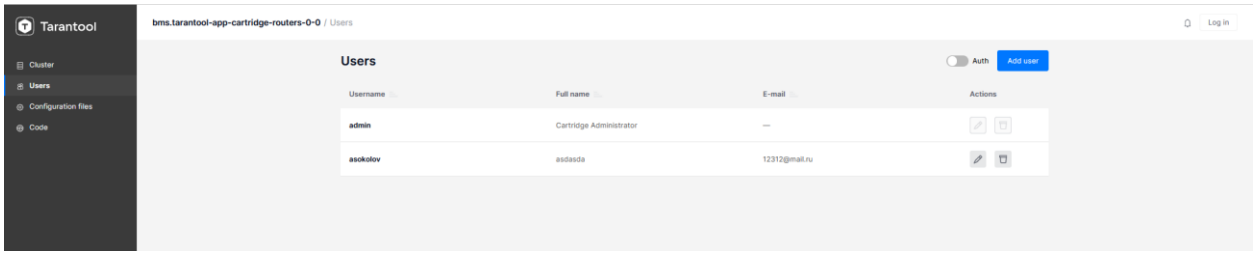


Рисунок 20 – Вкладка Users

4.3.4 Конфигурационные файлы

Данный раздел позволяет загрузить топологию кластера в виде YAML-файла (см.Рисунок 22) и выгрузить во вне топологию настроенного существующего кластера в том же формате.

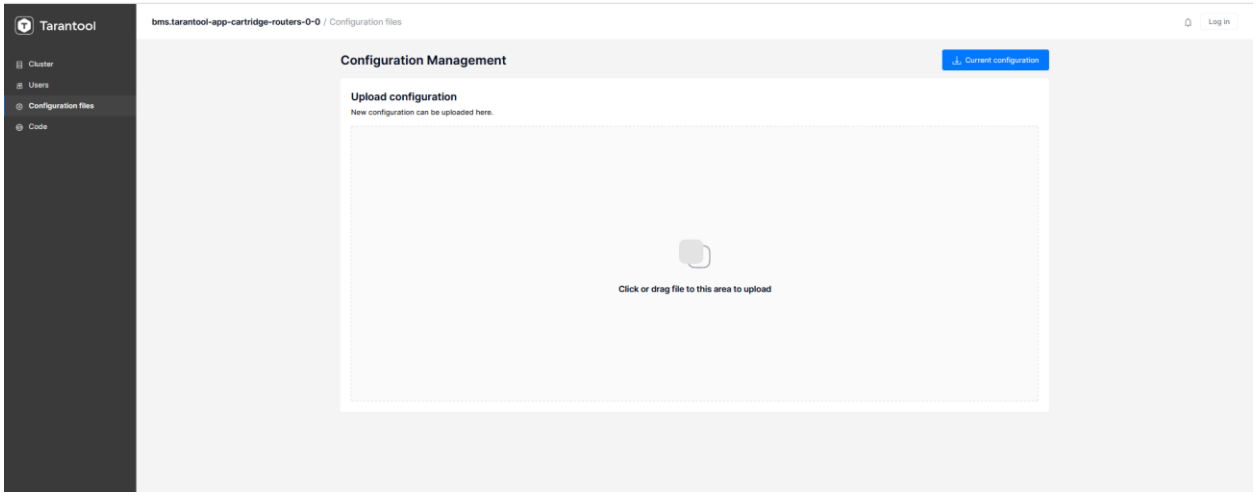


Рисунок 21 – Вкладка Configuration files

Рисунок 22 – Пример YAML-файла для загрузки топологии кластера

4.3.5 Код

Tarantool-Cartridge предоставляет возможность на данной вкладке подгружать файлы с кодом, которые «на лету» могут быть применены на всех нодах. В Forward BMS данная возможность не практикуется.

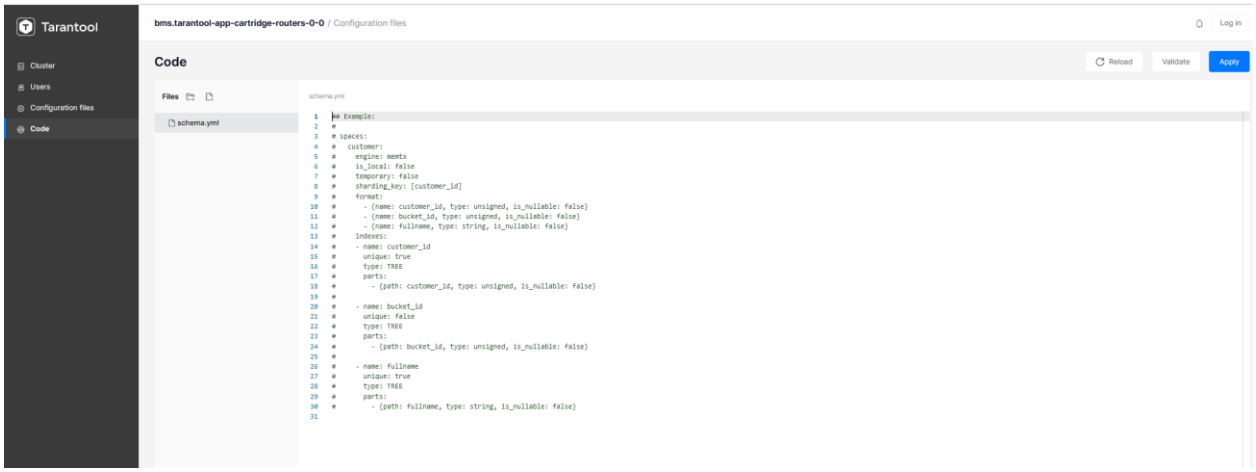


Рисунок 23 – Вкладка Code

4.4 Архивация логов

Практически на каждую операцию, выполняемую внутри BMS, выполняется логирование: вставка данных, редактирование, удаление данных. Данная информация хранится в оперативной памяти, что предполагает ее периодическую чистку и архивацию.

Администратор может запускать архивацию логов, что позволит выгрузить их из памяти на диск. Процесс очистки логов достаточно запустить один раз, после чего

он будет постоянно работать (пока его снова не остановят) и периодически выполнять архивацию при необходимости.

4.5 Архивация истекших пополнений

Пополнения имеют срок действия. Если у пополнения закончился срок действия, либо пополнение полностью израсходовано, то такие пополнения тоже сбрасываются из оперативной памяти на диск.

Участие администратора в запуске данного процесса аналогично описанному в п.4.4.

4.6 Очистка невостребованных резервов

При работе с резервами, среди них можно выделить те, которые не обязательно подтверждать (это отдельный признак). Если дата подтверждения резерва истекла, то процесс очистки невостребованных резервов либо подтверждает такой резерв (при выставленном признаке), либо сбрасывает в архив, если резерв не был подтвержден. Средства, которые были зарезервированы резервом, при этом, возвращаются или списываются с баланса.

Участие администратора в запуске данного процесса аналогично описанному в п.4.4.

4.7 Мониторинг BMS

4.7.1 Оценка состояния кластера по маркерам в интерфейсе Forward BMS

С помощью мониторинга Forward BMS администратору предоставляется возможность отслеживания нагрузки на кластер и конкретно на каждую ноду по-отдельности.

Ключевыми параметрами в интерфейсе, на которые стоит обращать внимание являются:

- Healthy – должно быть Healthy для всех реплика-сетов;
- Состояние памяти – memory usage – должно быть ниже максимального заданного значения.

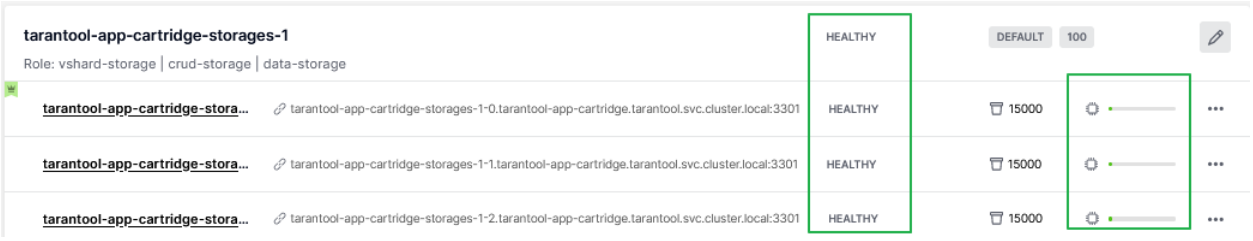


Рисунок 24 – Параметры Healthy и состояние памяти

4.7.2 Оценка состояния кластера по JMX-метрикам в Grafana

Forward BMS предоставляет встроенный набор JMX-метрик, значения которых могут быть выведены во внешнюю систему мониторинга (например, Grafana) для обеспечения наглядного и всестороннего наблюдения за работой кластера (см.Рисунок 25):

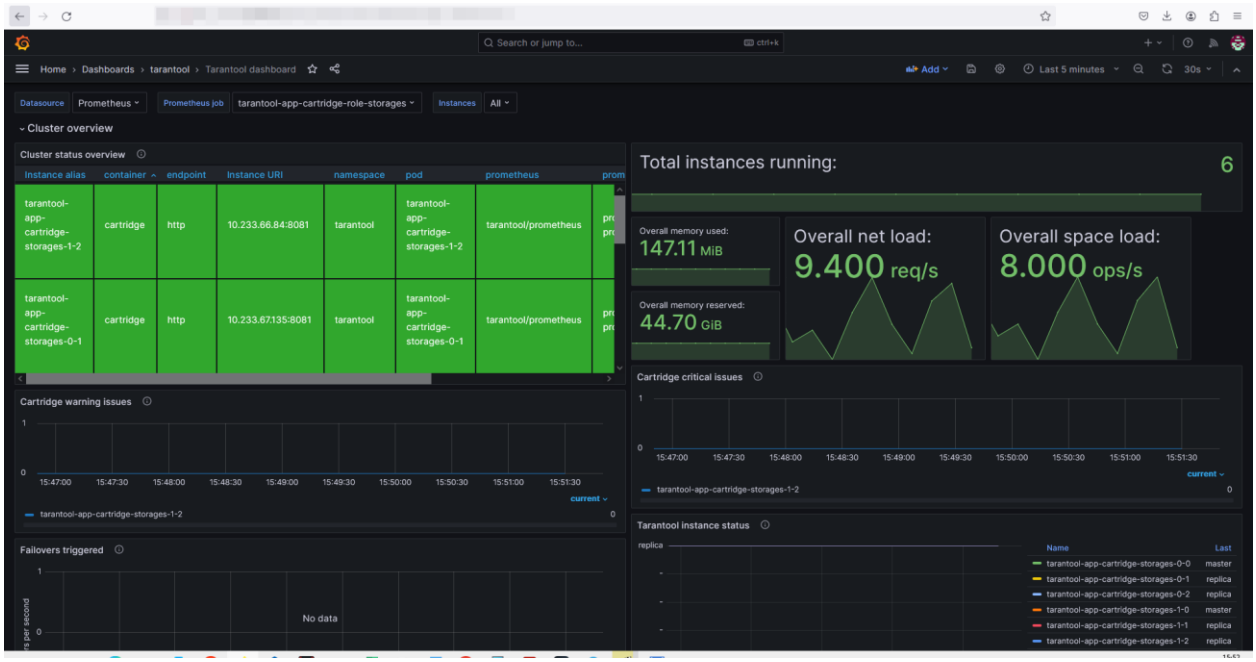
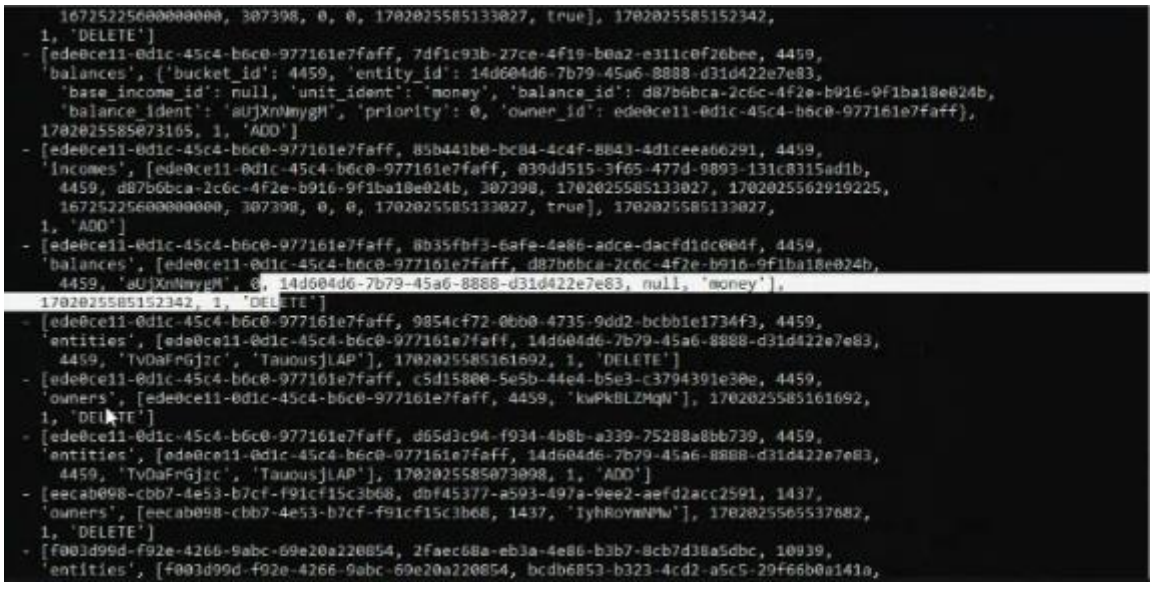


Рисунок 25 – Мониторинг кластера по JMX-метрикам

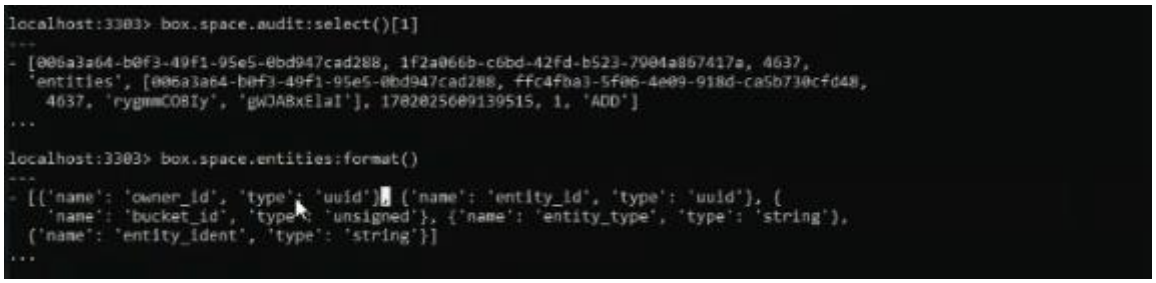
4.8 Аудит

При необходимости получить информацию о текущей деятельности ноды кластера (вставка, удаление записей и т.д.), можно подключиться к ней напрямую:

```
$tarantulctl connect login:password@IP:port
> box.space.audit:select()
```



Далее можно точнее рассмотреть интересующий фрагмент, вызвав команду с пояснением форматом записи.



Owner_id – владелец

Entity_id – код сущности

Bucket_id – корзина

Entity_type – тип сущности

Entity_ident – экземпляр сущности

4.9 Обновление BMS

4.9.1 Автоматическое обновление

Поскольку инсталляция и разворот Forward BMS выполняется с помощью Kubernetes, в системе есть скрипт, который посредством RollingUpdate обновляет BMS.

При таком обновлении происходит поочередное отключение реплика-сетов с перемещением данных на соседнюю реплику, происходит обновление реплики (выключение/включение), на нее перемещаются данные, реплика становится master, далее выполняется переход к следующей реплике.

4.9.2 Ручное обновление стораджей

При невозможности выполнения автоматического обновления, обновление можно установить вручную, введя предварительно все данные с обновляемого стораджа с помощью параметра Replica set weight, установив его в значение 0 на обновляемом сторадже и дождавшись миграции всех данных по bucket на второй сторадж. При достижении кол-ва bucket = 0, выполнить обновление, затем повторить действие на втором сторадже, выполнить обновление там, затем вернуть значение параметра Replica set weight в рабочее значение.

4.10 Возможные нештатные ситуации и способы их решения

4.10.1 Сетевая недоступность

При недоступности одной из нод необходимо:

1) Проверить доступность физического сервера, на котором расположена нода;
2) Восстановить сервер, чтобы он снова мог быть доступен в cartridge;
3) При невозможности восстановления сервера, добавить новый сервер и включить его вручную внутрь кластера.

4.10.2 Потеря master-ноды

В кластере есть ноды типа router (занимаются маршрутизацией запросов) и ноды типа storage (хранилища). Storage-ноды имеют репликацию (1 master и 2 slave ноды). Если теряется доступ к master-ноде, то внутренними средствами tarantool можно переназначить master, чтобы не потерялся доступ к данным. Это можно настроить как автоматически через интерфейс cartridge, либо в том же интерфейсе – вручную.

4.10.3 Повышенная нагрузка на одну из нод

В случае нештатной ситуации, которая вызывает повышенную нагрузку на одну из нод стораджа (например, проблемы с сервером или сетью), что влияет на скорость доступности для внешних сервисов, можно воспользоваться параметром Replica set weight, изменив его значение так, чтобы большее число bucket переместилось бы на другой сторадж, расположенный на доступном сервере кластера.

5 Приложение А: Методы нативного протокола

Описание в виде описания аргументов и результатов методов передается на основании договора на внедрение или обслуживание продукта.

В данном приложении представлен общий список методов нативного протокола для взаимодействия с BMS.

1) Владелец (Owner)
   a) Создание владельца (add_owner)
   b) Поиск владельца (get_owner)
   c) Удаление владельца (delete_owner)
2) Балансы (Balances)
   a) Добавление баланса (add_balance)
   b) Удаление баланса (delete_balance)
   c) Поиск баланса по идентификатору (get_balance)
   d) Редактирование баланса (edit_balance)
   e) Получение списка балансов (get_balance_list)
3) Пополнение
   a) Добавить пополнение (add_income)
4) Резервы - Резервы подразделяются на 2 типа - связанные (linked_reserve) и владельца (owner_reserve).
   a) Резервы владельца (owner_reserve)
      i) Создание (add_owner_reserve)
      ii) Подтверждение (confirm_owner_reserve)
      iii) Отмена (cancel_owner_reserve)
      iv) Редактирование резерва
   b) Связанные резервы
      i) Создание (add_linked_reserve)
      ii) Подтверждение (confirm_linked_reserve)
      iii) Отмена (cancel_linked_reserve)

6 Приложение Б: REST API

Описание в виде yaml-файла передается на основании договора на внедрение или обслуживание продукта.

В данном приложении представлен общий список методов REST для взаимодействия с BMS.