Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики: Эксплуатационная практика

Название предприятия: ООО «ВК Цифровые Технологии»

Студент: группа ИУ8-104-2025 л.д. 20У633
Мильченко Иван Дмитриевич — 14.07.2025
(подпись, дата)

От профильного предприятия:
представитель по доверенности № ВКЦТ 2.1-25-41 от 13.03.2025 г. Евтухов Семен Николаевич — М. П. 14.07.2025
(подпись, дата)

Руководитель от предприятия:
руководитель команды Моисеев Георгий Николаевич — 14.07.2025
(подпись, дата)

Руководитель от кафедры:
доцент кафедры ИУ8 Зайцева Анастасия Владленовна _____
(подпись, дата)

Оценка: _____

Москва, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ

Название предприятия: ООО «ВК Цифровые Технологии»

Сроки практики: с 01.07.2025 по 14.07.2025

Специальность / направление: 10.05.01 Компьютерная безопасность

Специализация / профиль: 10.05.01_01 Математические методы защиты информации

За время прохождения практики студенту надлежит согласно программе практики:

Создать дашборд для мониторинга продукта Tarantool Clusters Federation.

Руководитель от кафедры:
доцент кафедры ИУ8 Зайцева Анастасия Владленовна — 14.07.2025 (подпись, дата)

От профильного предприятия:
представитель по доверенности № ВКЦТ 2.1-25-41 от 13.03.2025 г. Евтухов Семен Николаевич — М. П. 14.07.2025 (подпись, дата)

Руководитель от предприятия:
руководитель команды Моисеев Георгий Николаевич — 14.07.2025 (подпись, дата)

Студент: группа ИУ8-104-2025 л.д. 20У633
Мильченко Иван Дмитриевич — (подпись, дата)

Является обязательным листом отчёта по практике. Документ не должен содержать информацию, отнесённую в установленном порядке к государственной тайне РФ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Место прохождения практики – Общество с ограниченной ответственностью «ВК Цифровые технологии», команда Tarantool, подразделение Tarantool Clusters Federation. Период прохождения практики – с 1 июля 2025 г. по 14 июля 2025 г.

Цель прохождения практики – выявить необходимые метрики для мониторинга продукта, внедрить их в программный код приложения, а затем осуществить их вывод на дашборд.

Для выполнения поставленной задачи необходимо выполнить следующие ее подзадачи:

- Создать документ с анализом существующих метрик с четким описанием того, для чего нужна каждая из них.
- По результатам анализа внедрить недостающие метрики в программный код приложения.
- Написать дашборд в Grafana с использованием фреймворка Grafonnet.
- Подготовить понятную документацию как для разработчиков, так и для передачи задачи в команду документации.

ОСНОВНАЯ ЧАСТЬ

1 Характеристика компании

«VK Tech»– это технологическое подразделение крупнейшей российской социальной сети «ВКонтакте», занимающееся разработкой и поддержкой высоконагруженных систем, инструментов для разработчиков, а также различных продуктое, поддерживающих экосистему «ВКонтакте» и её партнёров. Одним из ключевых направлений работы «VK Tech» является развитие платформы Tarantool.

Tarantool [1] – это высокопроизводительная платформа, объединяющая в себе функциональность «in-memory»базы данных и сервера приложений, обеспечивающая низкую задержку и высокую масштабируемость. Tarantool активно используется как в инфраструктуре «ВКонтакте», так и в других крупных компаниях для обработки больших объемов данных в реальном времени.

Помимо базового ядра Tarantool, в Enterprise Edition доступны коробочные продукты и решения, которые закрывают задачи интеграции, масштабирования и эксплуатации:

- Tarantool DB – основной продукт для высоконагруженных OLTP-сценариев, in-memory база данных с репликацией и шардингом.
- Tarantool CDC (Change Data Capture) —- механизм потоковой передачи изменений из Tarantool во внешние системы (Kafka, ClickHouse, PostgreSQL и др.).
- Tarantool Queue Enterprise – система очередей сообщений и задач для надёжного асинхронного обмена, с расширенными возможностями мониторинга и отказоустойчивости.
- Tarantool Column Store – колоночное хранилище, ориентированное на аналитические нагрузки.
- Tarantool Clusters Federation – комплекта инструментов для межкластерной репликации данных с автоматическим переключением трафика при проблемах на одном кластере.

Далее вся информация и объем работы, выполненный в рамках практики, будет относиться к последнему представленному продукту.

2 Описание продукта «Tarantool Clusters Federation»

Tarantool Clusters Federation (TCF) [2] позволяет построить отказоустойчивую систему из двух независимых кластеров Tarantool. В такой системе один из кластеров является активным и принимает все запросы от приложения. Второй кластер является пассивным и содержит копию данных активного кластера.

TCF позволяет управлять переключением трафика между активным и пассивным кластерами. Например, размещение кластеров в разных центрах обработки данных позволяет минимизировать негативные последствия при техногенных или природных инцидентах. Также переключение трафика на другой кластер позволяет проводить технические работы и изменения логики работы приложения без простоя.

Сам продукт состоит из ролей

Далее будет представлена подробное описание архитектуры продукта.

На рисунке 1 представлена архитектурная схема для понимания внутреннего устройства системы и взаимосвязей между ключевыми элементами и участниками.

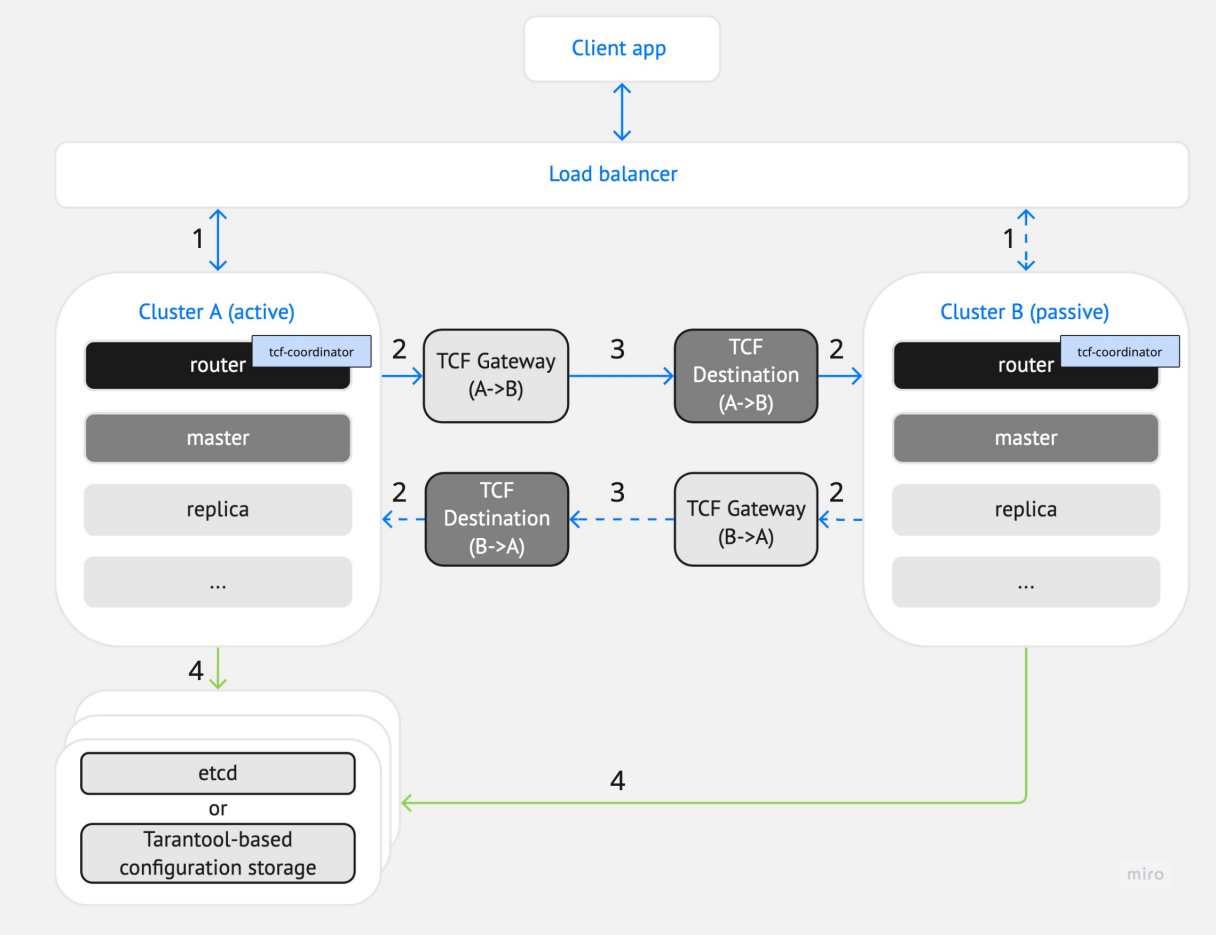

Рисунок 1 – Архитектура взаимодействия компонентов TCF между собой

TCF включает следующие компоненты:

– Приложение (**Client app**).
– Внешний балансировщик (**Load balancer**).
– Активный и пассивный кластеры (**Cluster A и Cluster B**).
– TCF-координатор в качестве роли на инстансе типа роутер кластерв.
– TCF-worker в качестве роли на каждом инстансе тарантула.
– Репликаторы данных.
– Хранилище состояния кластеров (etcd [3] или Tarantool-based configuration storage).

На рисунке 1 цифрами обозначены элементы и участники взаимодействия в TCF:

1. DML-пользователь. Выполняет DML-операции (чтение, запись, обновление, удаление данных). Когда кластер переводится в пассивное состояние, TCF блокирует доступ для всех DML-пользователей данного кластера. Таким образом организуется запрет изменения данных на пассивном кластере.
2. Служебный пользователь репликации данных, который нужен для непосредственной репликации между двумя класетрами.
3. Взаимодействие через gRPC-протокол. В gRPC-конфигурации нет пользователей: в конфигурации Gateway указывается адрес для приёма входящих gRPC-запросов от Destination. В конфигурации Destination указывается тот же адрес, что и в Gateway, на который компонент отправляет gRPC-запросы. Внешние подключения не предполагаются. Разграничение доступа можно обеспечить через TLS и выдачу компонентам сертификатов для безопасного взаимодействия друг с другом.
4. Пользователь хранилища состояния кластеров: etcd или Tarantool-based configuration storage.

Далее будет представлена более подробная информация по каждому из компонентов.

Client app – приложение, отправляющее запросы на активный кластер Tarantool через внешний балансировщик. Данное приложение устанавливает соединение с кластером от имени пользователя, у которого есть необходимые права на чтение и запись в кластер.

Load balancer – внешний балансировщик, выполняющий переключение трафика между приложением и кластерами Tarantool. Балансировщик отправляет все запросы на адреса роутеров активного кластера. При штатной работе трафик от приложения на роутеры пассивного кластера не направляется. Для определения состояния кластера необходимо отправить HTTP-запрос к любому из узлов кластера на заданный адрес обработчика запроса.

При активном кластере HTTP-ответ содержит код 200 и строку active. При пассивном — также код 200, но со строкой passive. При наличии неполадок возможны другие коды ответа из диапазонов 4xx или 5xx. Балансировщик должен направлять запросы только на активный кластер, то есть на узлы, возвращающие строку active, и не отправлять запросы на пассивные или неработоспособные кластеры, идентифицируемые по строке passive или по ошибочным HTTP-кодам.

TCF в типичной конфигурации содержит два идентичных кластера – активный и пассивный:

- **Cluster A** – активный кластер Tarantool, на который перенаправляются запросы от приложения.
- **Cluster B** – пассивный кластер Tarantool, который содержит копию данных активного кластера, но не принимает запросы от приложения.

Кластеры хранят свое состояние в централизованном хранилище. Состояние кластеров может быть изменено вручную или автоматически.

Технологическая роль — это программный модуль, реализующий определенные функции или логику. Это не те роли, которые могут быть назначены пользователям. Для работы TCF используются две технологические роли в конфигурации кластера:

- **TCF-координатор** – отвечает за автоматическое переключение состояния. Чтобы обеспечить отказоустойчивость, в каждом кластере должно быть запущено два и более экземпляра Tarantool с этой ролью. Назначается выборочным экземплярам типа storage и router в кластере Tarantool, которыми управляет TCF.
- **TCF-worker** – инициализирует и координирует переключение статусов кластеров (активный-пассивный) и предоставляет базовый HTTP API для управления TCF. Назначается всем экземплярам типа storage и router в кластере Tarantool, которыми управляет TCF.

Данные с активного кластера на пассивный реплицируются по протоколу gRPC в асинхронном режиме с помощью пары Gateway/Destination (шлюз/-приемник):

- Gateway подключается к активному кластеру в режиме анонимной реплики для получения потока транзакций. Поток транзакций сериализуется в gRPC.
- Destination подключается к Gateway по gRPC и запрашивает поток транзакций, затем десериализует поток транзакций и записывает их в пассивный кластер по протоколу iproto.

На рисунке 1 выше присутствуют две пары Gateway/Destination:

- TCF Gateway (A->B) и TCF Destination (A->B): Gateway и Destination, которые используются для репликации данных из Cluster A в Cluster B, когда Cluster A является активным;

– TCF Gateway (B->A) и TCF Destination (B->A): Gateway и Destination, которые используются для репликации из Cluster B в Cluster A, когда Cluster B становится активным.

Направление репликации определяется конфигурацией репликаторов данных. Если адреса Gateway/Destination не указаны в конфигурации, репликация выполняется в обе стороны. Если адреса заданы, переключение кластера в пассивный режим останавливает соответствующую пару. Односторонняя репликация снижает риск конфликтов, двусторонняя — компенсирует отставания асинхронной репликации.

Например, в случае сбоев на активном кластере (включая полную временную недоступность), активным становится текущий пассивный кластер, но из-за асинхронности не гарантируется, что он содержит все актуальные данные. После восстановления старый активный кластер возвращается в систему как пассивный. Если в его WAL остались непереданные данные, при двусторонней репликации они будут доставлены на новый активный кластер; при односторонней — нет, так как пассивный кластер не инициирует передачу данных.

Для работы TCF требуется внешний state provider – централизованное хранилище, которое хранит информацию о состояниях кластеров.

TCF поддерживает два типа такого хранилища:

– etcd – распределенное хранилище типа ключ-значение;
– хранилище конфигурации на основе Tarantool: хранилище, состоящее из набора реплик Tarantool.

3 Проектирование собираемых метрик

Так как TCF состоит из технологических ролей (tcf-worker и tcf-coordinator) и репликаторов данных (Gateway и Destination), которыми являются отдельные независимые сервисы, сбор метрик нужно интегрировать в каждый из этих компонентов. Использование этих метрик позволяет администраторам своевременно выявлять проблемы, анализировать производительность и обеспечивать стабильную работу кластеров и межкластерных репликаторов данных. Таким образом все собираемые метрики можно разделить на несколько категорий:

- метрики, собираемые с инстансов Tarantool;
- метрики, собираемые с сервисов репликаторов.

Метрики предоставляются в формате Prometheus. Используются следующие типы метрик Prometheus:

- counter – монотонно возрастающий счетчик;
- gauge – числовое значение, которое может как возрастать, так и убывать.

Так как TCF – продукт про репликацию, проектирование метрик сделаны с упором на эти параметры. В качестве определяющей метрики для ее отслеживания была выбрана метрика vclock-signature (сигнатура vclock). Сигнатура – это сумма всех элементов вклока, исключая нулевую компоненту. Vclock расшифровывается как vector clock (векторные часы). Они состоят из ассоциативного массива, где ключом является идентификатор сервера репликасета, а значением последний LSN (log sequence number), когда сервер был мастером. LSN – число (счетчик), инкрементирующийся каждый раз, когда в журнале WAL (write ahead log) добавляется новая подтвержденная запись.

Таким образом, если на каком-то из компонентов значения сигнатур не будут совпадать, администратор сможет найти место, где происходит отставание и своевременно обнаружить проблему.

Ниже указаны название метрик, относящиеся к каждому компоненту TCF, а также их назначение.

3.1 Метрики технологических ролей

Метрики позволяют отслеживать статус кластера, количество переданных и прочитанных данных, а также диагностировать ошибки при обмене данными между кластерами (в скобках будет указано название метрики в формате Prometheus [4]):

- Состояние кластера (*tcf_is_active*) – активность текущего кластера. Тип в prometheus: *gauge*. Принимает значения 1 (активный кластер) и 0 (пассивный кластер).
- tcf_dst_vclock_signature – сигнатура vclock, применённая на соседнем кластере. Тип: gauge. Отображает состояние репликации на стороне кластера-получателя (Destination). Может использоваться для сравнения с *tcf_src_vclock_signature*. Отгруппирована по uuid репликасета.
- tcf_source_vclock_signature – последняя записанная vclock-сигнатура на исходном кластере. Тип: gauge. Как говорилось ранее, разница между значениями vclock на кластерах может сигнализировать о задержках в репликации. Отгруппирована по uuid репликасета.

3.2 Метрики Gateway

Для того, чтобы включить экспорт метрик с сервиса по HTTP, нужно в поле *gateway.metrics_enabled* файла конфигурации выставить значение *true*. Были выделены следующие метрики:

1. *tcf_gateway_sent_total* – суммарное количество записей, отправленных на компонент Destination. Тип: counter. Отгруппированы по названию спейса (таблицы) и uuid репликасета.
2. *tcf_gateway_sent_errors_total* – суммарное количество ошибок, возникших при отправке данных на компонент Destination. Тип: counter. Отгруппированы по названию спейса (таблицы) и uuid репликасета.
3. *tcf_gateway_read_total* – суммарное количество записей, прочитанных с исходного кластера. Тип: counter. Отгруппированы по названию спейса (таблицы) и uuid репликасета.
4. *tcf_gateway_read_errors_total* – суммарное количество ошибок, возникших при чтении данных с исходного кластера. Тип: counter. Отгруппированы по названию спейса (таблицы) и uuid репликасета.

5. *tcf_gateway_limbo_vclock_signature* – сигнатура vclock из limbo. Тип: gauge. Показывает vclock signature, полученную из лимбо (структура данных, обрабатывающая транзакционный поток репликации), отгруппированная по набору реплик и Gateway, через который идет нагрузка. Отгруппированы по uuid репликасета.
6. *tcf_gateway_sent_vclock_signature* – сигнатура vclock, отправленная из Gateway в Destination. Тип: gauge. Фиксирует сигнатуру vclock, отправленную из Gateway в Destination. Используется для контроля синхронизации между кластерами. Группируется по наборам реплик. Отгруппированы по uuid репликасета.
7. *tcf_gateway_http_responses_total* – показывает количество HTTP-ответов от Gateway с конкретным методом, путем и статусом. Отгруппированы по типу запроса, пути запроса, статус-кода ответа.

3.3 Метрики Destination

Для того, чтобы включить экспорт метрик с сервиса по HTTP, нужно в поле *destination.metrics_enabled* файла конфигурации выставить значение *true*. Аналогичные метрики с минимальными отличиями созданы в Destination:

1. *tcf_destination_recv_total* – общее количество событий, полученных от компонента Gateway. Тип: counter. Отгруппированы по uuid репликасета и имени спейса (таблицы).
2. *tcf_destination_recv_errors_total* – общее количество ошибок при получении данных. Тип: counter. Отгруппированы по типу ошибки.
3. *tcf_destination_push_total* – суммарное количество событий, отправленных в Destination. Тип: counter.
4. *tcf_destination_push_errors_total* – количество ошибок, возникших в Destination при попытке отправить данные на целевой кластер. Тип: counter.
5. *tcf_destination_recv_vclock_signature* – сигнатура vclock, полученная от компонента Gateway. Тип: gauge. Содержит сумму всех ненулевых значений вектора vclock, полученного на Destination от Gateway. Используется для оценки состояния репликации на стороне Destination.

6. *tcf_destination_sent_vclock_signature* – сигнатура vclock, отправленная из Destination. Тип: gauge. Фиксирует сумму ненулевых значений vclock, отправленных Destination в его кластер. Позволяет отслеживать текущее состояние репликации.
7. *tcf_destination_http_responses_total* – показывает количество HTTP-ответов от Destination с конкретным методом, путем и статусом.

3.4 Системные метрики

Также были введены системные метрики для репликаторов для мониторинга системных параметров, таких как количество потоков, отданных системой приложению, объем потребляемой памяти, динамика работы сборщика мусора и многое другое. Все описанные метрики были интегрированы в код программы и покрыты тестами. Для того, чтобы экспортировать метрики из приложения, нужно обратиться по HTTP порту по адресу *http://<server_addr:server_port>/metrics*.

4 Панель мониторинга

Для того, чтобы отобразить интегрированные метрики на панель мониторинга, было принято решение использовать инструмент Graffonet [5], который представляет библиотеку на языке Jsonnet [6], предназначенную для программной генерации дашбордов в Grafana [7].

Обычно дашборды в Grafana собирают вручную через интерфейс: кликают, перетаскивают панели, настраивают фильтры. Но как только графиков становится достаточно много, поддерживать их в актуальном состоянии превращается в рутинную и затратную работу. Grafonnet решает эту проблему: дашборды описываются в виде кода, и их можно быстро менять, копировать, хранить в Git и разворачивать автоматически. Таким образом был изучен язык Jsonnet с фреймворком Grafonnet и были отображены метрики, представленные в предыдущих пунктах.

На рисунке 2 изображена архитектура сбора метрик. Каждый компонент оснащен одинаковым HTTP API, который отдает метрики в формате Prometheus по пути `/metrics`. БД Prometheus собирает эти метрики с определенным интервалом, который называется `scrape_interval`, а затем отдает их в Grafana, который занимается визуализацией.

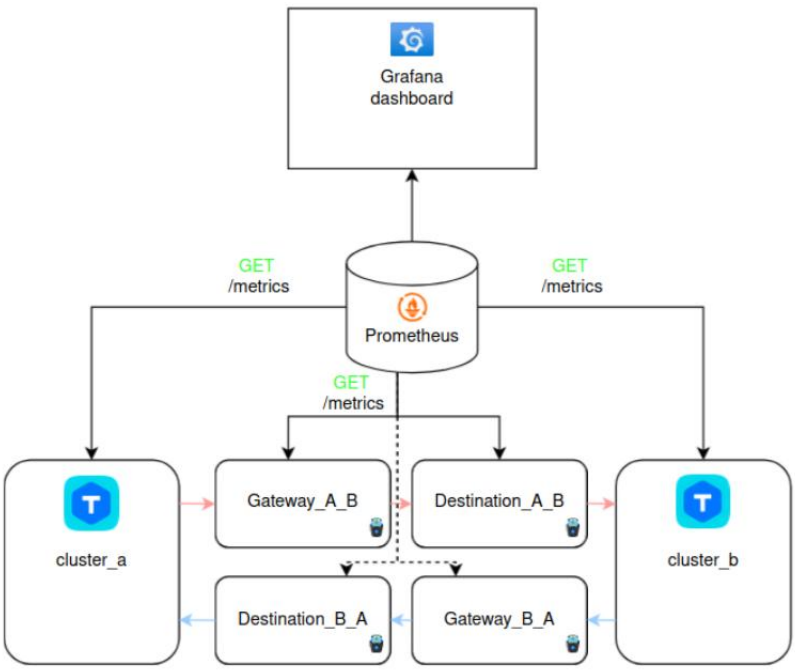

Рисунок 2 – Архитектура сбора метрик со всех компонентов TCF

4.1 Настройка Prometheus

Для сбора всех метрик с роли *tcf-worker*, *Gateway*, *Destination* необходимо написать конфиг с указанием джоб, в котором перечислить все необходимые адреса.

Листинг 1 – Пример конфигурации панели мониторинга для Prometheus

```
global:
  scrape_interval: 5s
  evaluation_interval: 5s

scrape_configs:
  - job_name: "cluster_a"
    static_configs:
      - targets:
          - localhost:8081
          - localhost:8082
          - localhost:8083
          - localhost:8084
          - localhost:8085
  - job_name: "cluster_b"
    static_configs:
      - targets:
          - localhost:18081
          - localhost:18082
          - localhost:18083
          - localhost:18084
          - localhost:18085
  - job_name: "replicators"
    static_configs:
      - targets:
          - localhost:10081
          - localhost:10082
          - localhost:10181
          - localhost:10182
```

Здесь, на листинге 1:

- `scrape_interval` – интервал, с которым Prometheus будет собирать метрики (каждые 5 секунд);
- `evaluation_interval` – интервал, с которым будет оцениваться состояние правил в Prometheus (5 секунд);
- `scrape_configs` – список задач (jobs) для сбора метрик с различных сервисов или кластеров;
- `targets` – список целей, каждая цель указывает на определённый сервис или экземпляр, с которого Prometheus должен запросить метрики;
- `cluster_a` – список адресов сервисов из исходного кластера;
- `cluster_b` – список адресов сервисов из целевого кластера;
- `replicators` – список адресов репликаторов Gateway и Destination.

Чтобы правильно отображать метрики репликаторов в Grafana, нужно задать псевдоним (alias) Gateway и Destination в конфигурации репликаторов данных, в секциях gateway.alias и destination.alias соответственно.

4.2 Доступные панели и их графики

В таблице 1 каждой метрике сопоставлены следующие характеристики: панель на дашборде Grafana, а также имя и описание графика.

Таблица 1 – Метрики Grafana для репликаторов и кластеров

№	**Панель Grafana**	**Имя графика**	**Метрика**	**Описание графика**
1	Replicators info	Events received from gateway	tcf_destination_recv_total	Общее количество событий, полученных от компонента Gateway
2	Replicators info	Events read from source cluster	tcf_gateway_read_total	Суммарное количество записей, прочитанных с исходного кластера
3	Replicators info	Events sent to destination cluster (от gateway)	tcf_gateway_sent_total	Суммарное количество записей, отправленных на компонент Destination
4	Replicators info	Events sent to destination cluster (or destination)	tcf_destination_push_total	Суммарное количество событий, отправленных в Destination

Продолжение таблицы 1

№	**Панель Grafana**	**Имя графика**	**Метрика**	**Описание графика**
5	Replicators info	Vclock signature received from gateway	tcf_destination_ recv_vclock_ signature	Сигнатура vclock, полученная от компонента Gateway
6	Replicators info	Vclock signature sent to destination cluster	tcf_destination_ sent_vclock_ signature	Сигнатура vclock, отправленная из Destination
7	Replicators info	Vclock signature received from limbo	tcf_gateway_ limbo_vclock_ signature	Сигнатура vclock из limbo (очереди транзакций)
8	Replicators info	Vclock signature sent from gateway	tcf_gateway_ sent_vclock_ signature	Сигнатура vclock, отправленная из Gateway в Destination
9	Replicators info	Reading from source cluster errors	tcf_gateway_ read_errors_ total	Суммарное количество ошибок, возникших при чтении данных с исходного кластера
10	Replicators info	Sending to destination errors	tcf_gateway_ sent_errors_ total	Суммарное количество ошибок, возникших при отправке данных на компонент Destination
11	Replicators info	Reading from gateway errors	tcf_destination_ recv_errors_ total	Общее количество ошибок при получении данных

Продолжение таблицы 1

№	**Панель Grafana**	**Имя графика**	**Метрика**	**Описание графика**
12	Replicators info	Sending to destination errors	tcf_destination_ push_errors_ total	Количество ошибок, возникших в Destination при попытке отправить данные на целевой кластер
13	Source cluster info / Destination cluster info	Is cluster active	tcf_is_active	Активность текущего кластера: зелёный — активный, жёлтый — пассивный
14	Source cluster info	Vclock signature sent from source cluster	tcf_src_vclock_ signature	Сигнатура vclock, отправленная на Gateway из текущего кластера
15	Destination cluster info	Vclock signature received from destination cluster	tcf_dst_vclock_ signature	Сигнатура vclock, применённая на целевом кластере

На рисунке 3 представлены графики №1-4 из таблицы 1. Все 4 графика повторяют друг друга, что означает отсутствие потерь на каждом из репликаторов при репликации данных.

Рисунок 3 – Графики репликаторов с количеством обработанных событий в секунду

На рисунке 4 представлены графики №5-8 из таблицы 1. Как видно из одинаковых графиков, сигнатуры повторяются, что является штатной ситуацией.

Рисунок 4 – Графики репликаторов с сигнатурой векторных часов на каждом этапе

На рисунке 5 представлены графики №9-12 из таблицы 1. На графиках ошибок наблюдается только одна, что говорит о закрытии репликационного потока. Так как запрос на репликацию повторился, поток переоткрылся, а значит ситуация не является аварийной. На графиках RPS HTTP-запросов видны запрос на старт *Gateway* и получение статуса *Destination*.

Рисунок 5 – Графики репликаторов с отображением ошибок и статусами HTTP-запросов

На рисунке 6 представлена панель №14 из таблицы 1. На графиках виден активный статус кластера, растущую сигнатуру, говорящую о реплицируемых данных. Также виден пустой график сигнатуры соседнего кластера, так как репликации в обратную сторону еще не было. HTTP-запросы на этот кластер не поступали, поэтому последний график пуст.

Рисунок 6 – Панель кластера-источника

На рисунке 7 представлена панель №15 из таблицы 1. На графиках виден пассивный статус кластера, растущую сигнатуру, говорящую о реплицируемых данных. Здесь график соседнего не пустой, так как с него идет нагрузка. На этот кластер HTTP-запросов также не поступало.

Рисунок 7 – Панель кластера назначения

На рисунке 8 представлены графики системных метрик репликаторов: количество выделенных горутин, использованной памяти, выделение, освобождение памяти и т. д.

Рисунок 8 – Панель системных метрик репликаторов

ЗАКЛЮЧЕНИЕ

В ходе прохождения практики была выполнена работа по созданию системы мониторинга для продукта Tarantool Clusters Federation (TCF).

В процессе практики были решены следующие задачи:

- проведён анализ архитектуры TCF и определены метрики, необходимые для контроля работы кластеров и репликаторов;
- реализована интеграция недостающих метрик в код приложения;
- разработана панель мониторинга в Grafana с использованием фреймворка Grafonnet;
- выполнена визуализация ключевых показателей работы системы, включая состояние репликации, активность кластеров, количество обработанных событий и возникающие ошибки.

Созданная система мониторинга обеспечивает своевременное выявление сбоев, контроль производительности и повышение надёжности эксплуатации продукта. Полученные результаты обладают практической значимостью для сопровождения и развития TCF.

Все поставленные задачи практики были выполнены в полном объёме. В ходе работы были закреплены знания и получены навыки в области мониторинга распределённых систем, применения инструментов Prometheus и Grafana, а также автоматизации развёртывания дашбордов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Tarantool Developers. Документация Tarantool. — 2025. — URL: https://www.tarantool.io/en/doc/latest/ ; [Электронный ресурс]. Дата обращения: 01.09.2025.

2. TCF Developers. Документация TCF. — 2025. — URL: https://www.tarantool.io/en/clustersfederation/doc/latest/ ; [Электронный ресурс]. Дата обращения: 01.09.2025.

3. etcd-io. Документация etcd. — 2025. — URL: https://etcd.io/docs/ ; [Электронный ресурс]. Дата обращения: 01.09.2025.

4. Prometheus Authors. Prometheus. — 2025. — URL: https://prometheus.io/docs/ ; [Электронный ресурс]. Дата обращения: 01.09.2025.

5. Grafana Labs. Grafonnet. — 2025. — URL: https://github.com/grafana/grafonnet-lib ; [Электронный ресурс]. Дата обращения: 01.09.2025.

6. Google. Jsonnet. — 2025. — URL: https://jsonnet.org ; [Электронный ресурс]. Дата обращения: 01.09.2025.

7. Grafana Labs. Grafana. — 2025. — URL: https://grafana.com/docs/ ; [Электронный ресурс]. Дата обращения: 01.09.2025.