

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА/GRADUATION THESIS

**Разработка автоматизированных средств мониторинга и реагирования для систем
требующих оркестрации**

Автор/ Author

Титанов Владислав Михайлович

Направленность (профиль) образовательной программы/Major

Компьютерные системы и технологии 2019

Квалификация/ Degree level

Магистр

Руководитель ВКР/ Thesis supervisor

Пенской Александр Владимирович, кандидат технических наук, Университет ИТМО,
факультет программной инженерии и компьютерной техники, доцент (квалификационная
категория "ординарный доцент")

Группа/Group

P42191

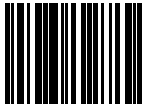
Факультет/институт/кластер/ Faculty/Institute/Cluster

факультет программной инженерии и компьютерной техники

Направление подготовки/ Subject area

09.04.01 Информатика и вычислительная техника

Обучающийся/Student

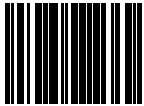
Документ подписан	
Титанов Владислав Михайлович	
01.06.2021	

(эл. подпись/ signature)

Титанов
Владислав
Михайлович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Пенской Александр Владимирович	
01.06.2021	

(эл. подпись/ signature)

Пенской
Александр
Владимирович

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Титанов Владислав Михайлович

Группа/Group P42191

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники

Квалификация/ Degree level Магистр

Направление подготовки/ Subject area 09.04.01 Информатика и вычислительная техника

Направленность (профиль) образовательной программы/Major Компьютерные системы и технологии 2019

Специализация/ Specialization Встраиваемые и киберфизические системы

Тема ВКР/ Thesis topic Разработка автоматизированных средств мониторинга и реагирования для систем требующих оркестрации

Руководитель ВКР/ Thesis supervisor Пенской Александр Владимирович, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

Срок сдачи студентом законченной работы до / Deadline for submission of complete thesis 31.05.2021

Техническое задание и исходные данные к работе/ Requirements and premise for the thesis

Основной целью работы является уменьшение затрат на разработку средств мониторинга и реагирования для систем требующих оркестрации. За счет предоставления конфигурируемого, с элементами автоматизированной настройки, средства мониторинга и реагирования для систем требующих оркестрации. В рамках этой задачи необходимо: 1) Провести исследование аналогичных проектов предоставляющих функции мониторинга и реагирования для оркеструемых систем. 2) Полностью описать требования к системе мониторинга и реагирования для систем требующих оркестрационных методов; 3) Спроектировать систему мониторинга за показателями системы с возможностью применения сценариев реакции на критическое изменение показателей, которое может перевести системы в нестабильный режим работы. 4) Разработать механизм реагирования на изменение показателей системы; 5) Разработать и продемонстрировать прототип системы мониторинга и реагирования; 6) Провести оценку полученных результатов.

Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)/ Content of the thesis (list of key issues)

- 1) Обзор и сравнительный анализ существующих средств мониторинга и реагирования систем требующих оркестрации;
- 2) Проектирование системы мониторинга и реагирования для проектов требующих

оркестрации;

3) Разработка механизма реагирования на критические состояния системы;

4) Прототипирование системы мониторинга и реагирования;

5) Оценка полученных результатов.

Перечень графического материала (с указанием обязательного материала) / List of graphic materials (with a list of required material)

Исходные материалы и пособия / Source materials and publications

1) "И. С. Лебедев, В. В. Семенов, М. Е. Сухопаров «Идентификация состояния отдельных элементов киберфизических систем на основе внешних поведенческих характеристик» 2018";

2) "James Turnbull. «TheDockerBook» 2016";

3) "Nigel Poulton. «TheKubernetesBook» 2019";

4) "Сергей Озеров, Александр Карабуто «Технологии виртуализации: вчера, сегодня, завтра»";

5) "Keith Adams, Ole Agesen A Comparison of Software and Hardware Techniques for x86 Virtualization";

6) ""Virtualize Your IT Infrastructure". VMWare. 2011";

7) "Nathan Liefting and Brian van Baekel. Zabbix 5 IT Infrastructure Monitoring Cookbook. 2021";

8) "Joel Bastos and Pedro Araujo. Hands-On Infrastructure Monitoring with Prometheus/ 2019";

9) "Navin Sabharwal and Piyush Pandey. Monitoring Microservices and Containerized Applications: Deployment, Configuration, and Best Practices for Prometheus and Alert Manager. 2020";

10) "Brian Brazil. Prometheus: Up & Running. 2018";

11) "Jason Dixon. Monitoring with Graphite. 2017";

12) "James Turnbull. The Art of Monitoring. 2016";

13) "Rob Ewaschuk and Betsy Beyer. Monitoring Distributed Systems. 2016";

14) "Patrik Uytterhoeven and Rihards Olups. Zabbix 4 Network Monitoring - Third Edition. 2019";

15) "Rihards Olups, Andrea Vacche and Patrik Uytterhoeven. Zabbix: Enterprise Network Monitoring Made Easy. 2017";

16) "Rihards Olups. Zabbix Network Monitoring - Second Edition"

Дата выдачи задания/ Objectives issued on 30.04.2021

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/
Thesis supervisor

Документ
подписан

Пенской
Александр
Владимирович

30.04.2021



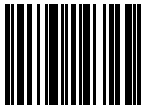
Пенской
Александр
Владимирович

(эл. подпись)

Задание принял к
исполнению/ Objectives
assumed by

Документ
подписан

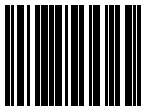
--

		
Титанов Владислав Михайлович		
30.04.2021		

(эл. подпись)

Титанов
Владислав
Михайлович

Руководитель ОП/ Head
of educational program

Документ подписан		
Платунов Алексей Евгеньевич		
19.05.2021		

(эл. подпись)

Платунов
Алексей
Евгеньевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ /
SUMMARY OF A GRADUATION THESIS**

Обучающийся/ Student

Титанов Владислав Михайлович

Наименование темы ВКР / Title of the thesis

Разработка автоматизированных средств мониторинга и реагирования для систем требующих оркестрации

Наименование организации, где выполнена ВКР/ Name of organization

Университет ИТМО

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ/
DESCRIPTION OF THE GRADUATION THESIS**

1. Цель исследования / Research objective

Уменьшение затрат на разработку средств мониторинга и реагирования для систем требующих оркестрации, за счет предоставления конфигурируемого средства мониторинга и реагирования для систем требующих оркестрации с элементами автоматизированной настройки.

2. Задачи, решаемые в ВКР / Research tasks

1) Проектирование средств отслеживания основных метрик систем оркестрации влияющих на стабильность работы системы (метрики кластера рассчитываются агрегируя метрики контейнеров, находящихся в нем), включая: количество контейнеров в кластере; загрузка CPU кластера на интервале; загрузка RAM кластера на интервале; загрузка SSD кластера на интервале. 2) Разработка механизмов идентификации критических состояний кластеров: автоматически (система накапливает конфигурации значений метрик, при которых происходили сбои, и выделяет из них критические диапазоны); вручную, то есть системе заранее вводятся критические диапазоны для каждой метрики. 3) Разработка механизмов реагирования на критическое состояние. 4) Проведение анализа полученных результатов, формулирование возможных точек для улучшения системы.

3. Краткая характеристика полученных результатов / Short summary of results/conclusions

1) На основе проведённого анализа существующих систем мониторинга и реагирования было разработано ТЗ учитывающее недостатки существующих систем, включающее как функциональные, так и нефункциональные требования. 2) Была предложена архитектура автоматизированной системы мониторинга и реагирования. 3) В качестве финального этапа был создан прототип системы мониторинга и реагирования, реализованный с учетом использования современных технологических трендов.

4. Наличие публикаций по теме выпускной работы/ Have you produced any

publications on the topic of the thesis

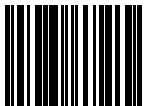
- 1 Рудзейт О.Ю., Титанов В.М., Петров С.Ю., Шневель В.С. Управление бизнес-процессами предприятия с использованием BPM-систем // Системный администратор -2019. - № 6(199). - С. 77-79 (Статья; ВАК, РИНЦ)

5. Наличие выступлений на конференциях по теме выпускной работы/ Have you produced any conference reports on the topic of the thesis

6. Полученные гранты, при выполнении работы/ Grants received while working on the thesis

7. Дополнительные сведения/ Additional information

Обучающийся/Student

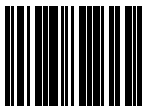
Документ подписан	
Титанов Владислав Михайлович	
01.06.2021	

(эл. подпись/ signature)

Титанов
Владислав
Михайлович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Пенской Александр Владимирович	
01.06.2021	

(эл. подпись/ signature)

Пенской
Александр
Владимирович

(Фамилия И.О./ name
and surname)

Оглавление

Список сокращений и условных обозначений.....	9
Термины и определения	10
Введение.....	11
1. Концепция оркестрации	13
2. Существующие средства мониторинга оркестрируемых систем .	16
2.1. Система Prometheus	16
2.2. Система Sentry.....	19
2.3. Система Zabbix.....	20
2.4. Система Munin.....	24
2.5. Сравнительные анализ.....	27
2.6. Постановка задачи	28
2.7. Вывод к главе	31
3. Проектирование системы мониторинга и реагирования	32
3.1. Построение модели сценариев использования	32
3.2. Основные состояния системы мониторинга и реагирования..	34
3.3. Подробной описание основных сценариев использования.....	34
3.4. Построение модели данных	49
3.5. Построение компонентной модели	50
3.6. Построение модели потоков данных	52
3.7. Построение диаграмм последовательности	52
3.8. Построение модели классов.....	54
3.9. Вывод к главе	55
4. Разработка механизма реагирования на критические состояния..	56
4.1. Концепция и формат описания сценариев реагирования	56
4.2. Обработка сценариев системой мониторинга и реагирования	60
4.3. Вывод к главе	61

5.	Протипирование системы мониторинга и реагирования	62
5.1.	Описание инструментов реализации прототипа системы	62
5.2.	Конфигурация прототипа	64
5.3.	Интеграционное описание	69
5.4.	Демонстрация работы системы	69
5.5.	Вывод к главе	71
6.	Оценка полученных результатов	72
	Заключение	73
	Список используемых источников	74
	Приложение 1 – Реализация API интерфейса прототипа АСМР	76
	Приложение 2 – Реализация веб-интерфейса прототипа АСМР	83
	Приложение 3 – Конфигурация КАФКА для прототипа АСМР	87
	Приложение 4 – Пример файла метрик, образованного кластером	93
	Приложение 5 – Конфигурация СУБД Tarantool	104

Список сокращений и условных обозначений

АСМР – автоматизированные средства мониторинга и реагирования.

Бэкенд – программно-аппаратная часть сервиса.

Исследуемая система – совокупность взаимосвязанных кластеров отслеживание состояния которых, является основной задачей АСМР.

СУБД – Система управления базой данных.

Фронтенд – клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

Термины и определения

Инстанс — это экземпляр объекта мониторинга, на практике представляет собой контейнер, содержащийся в кластере.

Контейнерная оркестрация — это автоматизация большей части операционных усилий, необходимых для запуска контейнерных рабочих нагрузок и сервисов. Это включает в себя широкий спектр вещей, которые необходимы программным командам для управления жизненным циклом контейнера, включая подготовку, развертывание, масштабирование (вверх и вниз), сетевое взаимодействие, балансировку нагрузки и многое другое.

Защищаемая автоматизированная информационная система — Автоматизированная информационная система, предназначенная для сбора, хранения, обработки, передачи и использования защищаемой информации с требуемым уровнем ее защищенности [1].

Защищаемые информационные ресурсы (автоматизированной информационной системы) - Информационные ресурсы автоматизированной информационной системы, для которых должен быть обеспечен требуемый уровень их защищенности.

Объект мониторинга — элемент ИТ-инфраструктуры с неделимой доступностью: сервер, операционная система, сетевой элемент, СУБД и пр., которым необходимо обеспечить штатную и стабильную работоспособность. Объект мониторинга может быть любым: локальным, виртуальным, сетевым и т. д.

Отказ — событие, заключающееся в полной или частичной утрате работоспособности [2][3].

Сбой — ненормальный режим, который может вызвать снижение или потерю способности функционального блока выполнять требуемую функцию. Сбой представляет собой состояние, характеризующееся неспособностью выполнить необходимую функцию [1].

Введение

Сейчас в мире использование распределенных архитектур так же, как использование концепции микросервисов или облачных сервисов, активно набирает популярность. Дальнейшим закономерным этапом является внедрение оркестрирующих компонентов Docker Swarm, Kubernetes, Apache Mesos и др.

Исходя из этого такие сложноорганизованные системы, состоящие из множества взаимосвязанных конвейеризированных компонентов или облачных сервисов, в данной работе, буду считать системами требующими оркестрации. Такие системы чаще всего сталкиваются со следующими проблемами, связанными с контролем за показателями системы [1]:

- Значение одного из критически значимых показателей системы вышло за пределы допустимого, при котором поведение системы не предсказуемо.
- Один из компонентов системы перешел в нестабильный режим или генерирует паразитические не корректные данные, и тем самым негативно влияет на корректную работы системы в целом.
- Связь с одним или несколькими компонентами системы потеряна, и система не может нормально функционировать.
- Один или несколько компонентов системы не согласовано поменяли свою конфигурацию.

Мониторинг и реагирование - одни из важнейших задач систем обеспечивающих бесперебойную работу множества приложений [4][5], в том числе и для которых, требуется оркестрация. Для многих задач, решаемых в контексте мониторинга и реагирования требуется историческая информация, которая помогает в планировании мощностей или расследовании инцидентов, также может понадобиться более высокая детализация для более детального изучения проблемы и более высокая степень свежести для того. Тем самым

можно рассматривать мониторинг, как источник информации для поддержания здоровья системы с точки зрения производства и бизнеса [6][7].

Также необходимо отметить, что основной целью автоматизированных систем мониторинга и реагирования (АСМР) для систем требующих оркестрации и работы в целом является - уменьшение затрат на разработку средств мониторинга систем оркестрации.

На основе вышеописанного можно сформировать следующие задачи данной работы:

- Исследование существующих систем мониторинга и реагирования для оркестрированных систем.
- Формирование технического задания для системы мониторинга и реагирования.
- Проектирование системы мониторинга и реагирования.
- Разработка механизма реагирования на нештатные ситуации.
- Создание прототипа системы мониторинга и реагирования.

1. Концепция оркестрации

Оркестрация контейнеров - представляет из себя процесс управления или планирования работы отдельных контейнеров для приложений на основе микросервисов в нескольких кластерах. В целом оркестрация это автоматизация всех аспектов координации и управления контейнерами. Оркестровка контейнеров ориентирована на управление жизненным циклом контейнеров и их динамических сред.

Инструменты для управления, масштабирования и обслуживания контейнерных приложений называются оркестраторами, и наиболее распространенными примерами из них являются Kubernetes и Docker Swarm. Развертывание в среде разработки обоих этих оркестраторов обеспечивается Docker Desktop, являющийся основным инструментом контейнеризации в данной работе

Оркестрация контейнеров осуществляется следующим образом. Файлы конфигурации сообщают инструменту оркестровки контейнеров, как подключиться к сети между контейнерами и где хранить их системные журналы. Инструмент оркестрации также организует развертывание контейнеров в кластеры и определяет оптимальную конфигурацию сети контейнеров. После того, как конфигурация сети определена, инструмент оркестрации управляет жизненным циклом контейнера на основе заранее определенных спецификаций. Инструменты оркестрации контейнеров работают в любой среде, в которой работают контейнеры.

Контейнер представляет из себя приложение вместе со всем его окружением и зависимостями. Контейнеры могут быть созданы, запущены, остановлены, перенесены или удалены. Каждый контейнер изолирован и является безопасной платформой для приложения.

Кластер является основным элементом оркестрации и формируются из множества контейнеров, каждый из которых выполняет определенную функцию — узла (node) или ведущего узла (master). На каждом узле также

хранятся группы из одного или нескольких контейнеров, а ведущий узел сообщает остальным узлам, когда создавать и удалять контейнеры и как изменить маршрут трафика в зависимости от нового места размещения контейнера. Типичная организация кластеров указана на рисунке 1.

К примеру, инструменты оркестрации для Docker следующие [6]:

- Docker Machine – подготавливает конфигурацию сети и устанавливает Docker Engine.
- Docker Swarm – кластеризует несколько сетей Docker под одной. Он также может интегрироваться с любым инструментом, который работает с одним хостом Docker.
- Docker Compose – позволяет развертывать многоконтейнерные приложения.

А инструменты оркестрации Kubernetes выполняют следующие функции:

- Автоматическое развертывание и репликация контейнеров.
- Оперативное масштабирование или горизонтальное масштабирование кластеров контейнеров.
- Балансировка нагрузки на группы контейнеров.
- Автоматическое изменение расписания отказавших контейнеров.
- Контролируемое открытие сетевых портов для систем за пределами кластера.

В сравнении с классическими инструментами управления частями системы оркестрация имеет следующие преимущества:

- Повышенная портативность - масштабирование приложения с помощью одной команды и масштабирования только определенных частей, не затрагивая всё приложение [3].
- Простое и быстрое развертывание.

- Повышенная производительность - Упрощенный процесс установки и уменьшение количества ошибок зависимостей.
- Повышенная безопасность - распределение определенных ресурсов без риска для внутренней или внешней безопасности. Изоляция приложений повышает безопасность веб-приложений, разделяя процессы каждого приложения на разные контейнеры [4][5].

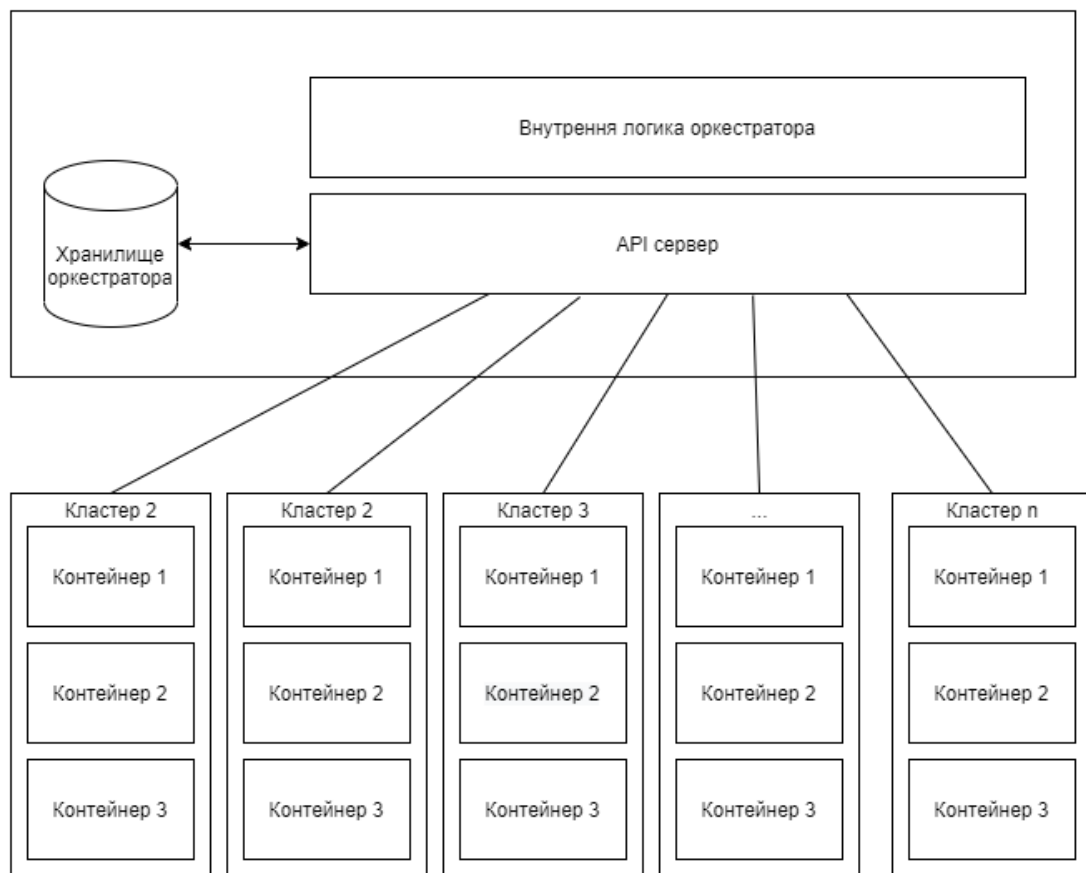


Рисунок 1 – Типичная организация оркестрированного приложения.

2. Существующие средства мониторинга оркестрируемых систем

В процессе анализа существующих продуктов необходимо было определить варьируемые параметры и измеряемые характеристики; список исследуемых сценариев нештатных ситуаций и причин их возникновения; список исследуемых методов и средств идентификации нештатных ситуаций; критерии оценки методов и средств идентификации нештатных ситуаций.

В качестве подобных проектов, выполняющих подобные функции были выбраны следующие решения с открытым кодом:

- Prometheus.
- Sentry.
- Zabbix.
- Munin.
- Grafana.

2.1. Система Prometheus

Prometheus — это набор инструментов для мониторинга и оповещения систем с открытым исходным кодом, изначально созданный в SoundCloud. С момента его создания в 2012 году многие компании и организации приняли Prometheus, и у проекта очень активное сообщество разработчиков и пользователей. Теперь это отдельный проект с открытым исходным кодом, который поддерживается независимо от какой-либо компании. Чтобы подчеркнуть это и прояснить структуру управления проектом, Prometheus присоединился к Cloud Native Computing Foundation в 2016 году в качестве второго размещенного проекта после Kubernetes [6][12].

2.1.1. Возможности

Основные возможности Prometheus:

- Использование многомерных моделей данных с данными временных рядов, идентифицированными по названию метрики и параметрами ключ / значение.
- PromQL, гибкий язык запросов для использования этой размерности.
- Отсутствие зависимости от распределенного хранилища; отдельные серверные узлы автономны.
- Сбор временных рядов происходит с помощью модели pull по HTTP.
- Передача временных рядов поддерживается через промежуточный шлюз.
- Цели обнаруживаются через обнаружение служб или статическую конфигурацию.
- Поддержка нескольких режимов построения графиков и дашбордов.

2.1.2. Составные части

Экосистема Prometheus состоит из нескольких компонентов, многие из которых не являются обязательными:

- Главный сервер Prometheus, который собирает и хранит данные временных рядов.
- Клиентские библиотеки для инструментального кода приложения.
- Push gateway's для поддержки короткоживущих рабочих мест.
- Специализированные экспортеры для таких сервисов, как HAProxy, StatsD, Graphite и т.д.
- Alertmanager для предупреждения ручки.
- Различные инструменты поддержки.

Большинство компонентов Prometheus написано на Go , что упрощает их сборку и развертывание в виде статических двоичных файлов [6][9].

2.1.3. Архитектура

Диаграмма на Рисунке 2 иллюстрирует архитектуру Prometheus и некоторых компонентов его экосистемы:

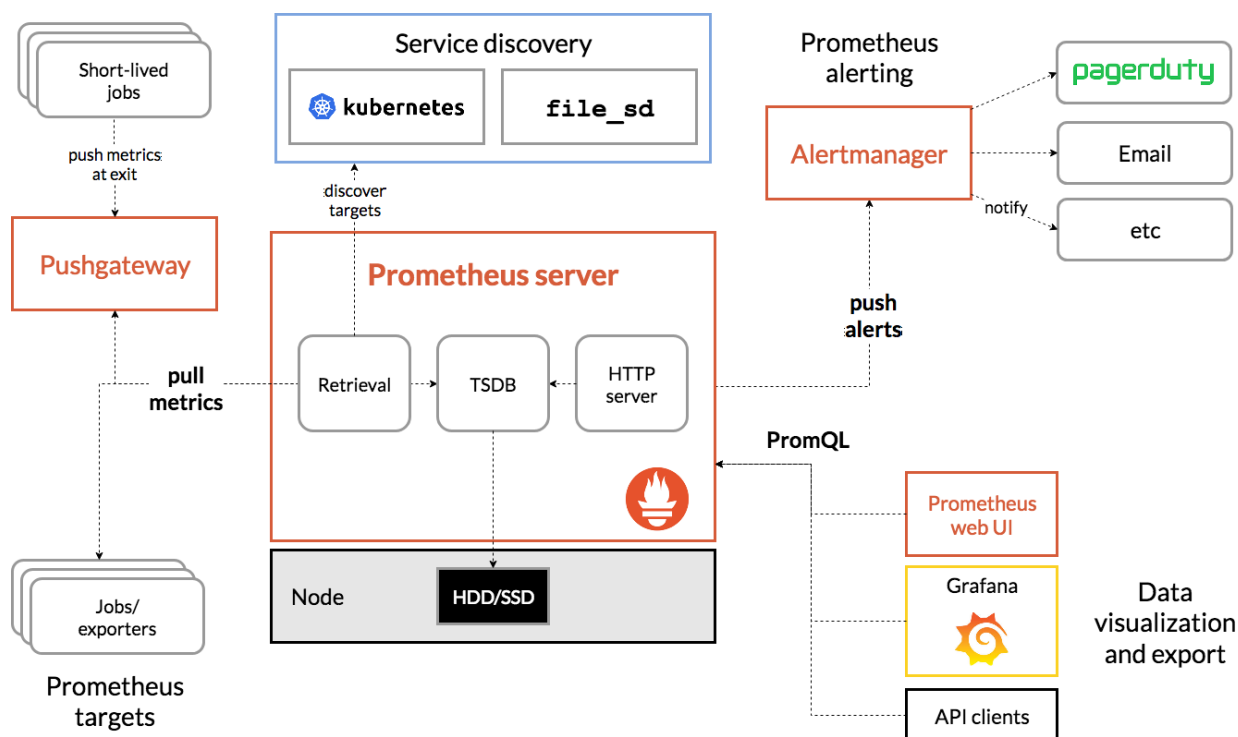


Рисунок 2 – Архитектура Prometheus [10]

Prometheus извлекает метрики из инструментальных заданий напрямую или через промежуточный push-шлюз для краткосрочных заданий. Он хранит все очищенные выборки локально и запускает правила для этих данных, чтобы либо агрегировать и записывать новые временные ряды из существующих данных, либо генерировать предупреждения. Grafana или другие потребители API могут использоваться для визуализации собранных данных [6].

Основными недостатками Prometheus можно считать отсутствие шифрования и стандартную модель данных с метрикой на основе ключа, которая может не совпадать с моделью сторонней системы. В этом случае нужно использовать экспортеры для преобразования метрик. Prometheus работает с данными по модели Pull, то есть для получения данных он сам опрашивает конечные точки.

2.2. Система Sentry

Sentry.io — это полнофункциональная система отслеживания ошибок с открытым исходным кодом, которая поддерживает широкий спектр серверных, браузерных, настольных и собственных мобильных языков и фреймворков, включая PHP, Node.js, Python, Ruby, C #, Java, Go, React, Angular, Vue, JavaScript и другие. Система используется в таких компаниях:

- Dropbox.
- AirBnB.
- PayPal.
- Uber.
- Reddit.
- Mozilla.
- MailChimp.
- Microsoft.

2.2.1. Возможности

Система Sentry обладает следующим функционалом [6]:

- Обновление списка ошибок в режиме реального времени;
- Группировка и сортировка полученных ошибок, например по частоте появления;
- Фильтрация ошибок по статусу, уровню логирования, источнику и другим параметрам;
- Возможность реинкарнации ошибки. Если ошибка была помечена как решенная и появилась снова, то она снова вносится в список и учитывается в отдельном потоке;
- Отправка e-mail, sms или чат-сообщений, в случае получения новой ошибки;
- Возможность запроса Feedback'а пользователя;

- Возможность интеграции с такими системами как JIRA, GitHub, Bitbucket и другими;

Основными недостатками Sentry является, то расширенный функционал Sentry осуществляется через платный доступ, однако можно попробовать его основные возможности на бесплатном тестовом аккаунте [6].

2.2.2. Архитектура

На рисунке 3 отображена структурная архитектура сервиса Sentry.

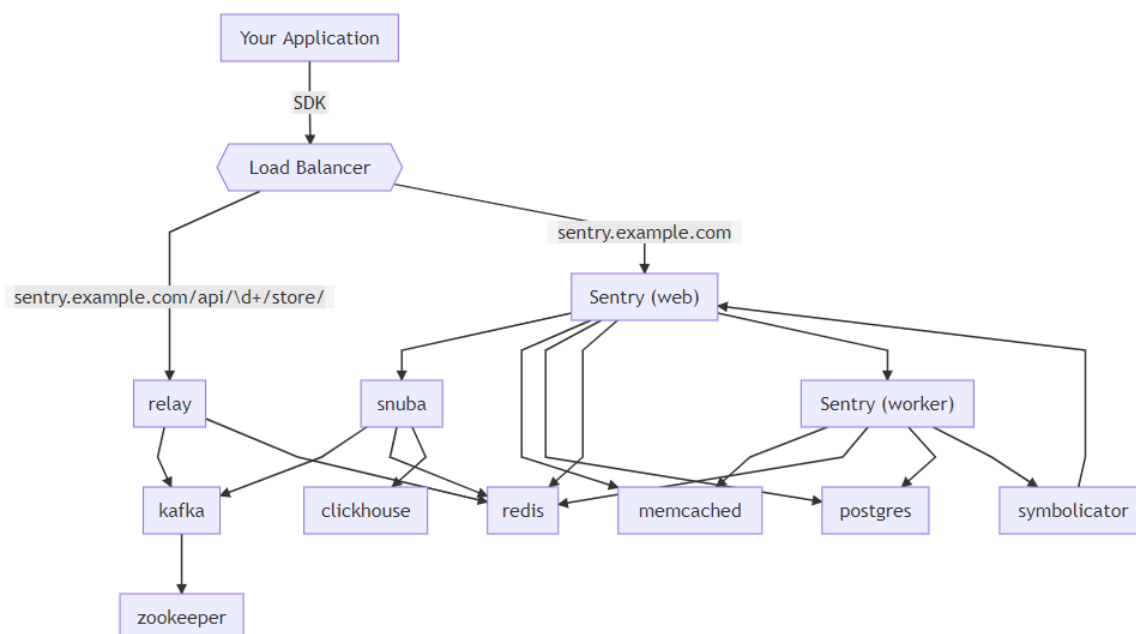


Рисунок 3 структурная архитектура мониторинга с Sentry [10]

2.3. Система Zabbix

Zabbix — свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования и приложений в целом.

Для хранения данных используется MySQL, PostgreSQL, SQLite или Oracle Database, веб-интерфейс написан на PHP. Поддерживает несколько видов мониторинга [7]:

- Simple checks — может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP, без установки какого-либо программного обеспечения на наблюдаемом хосте.
- Zabbix agent — может быть установлен на UNIX-подобных или Windows-хостах для получения данных о нагрузке процессора, использования сети, дисковом пространстве и так далее.
- External check — выполнение внешних программ, также поддерживается мониторинг через SNMP.

2.3.1. Возможности

Сервис Zabbix предоставляет следующие возможности:

- Распределённый мониторинг — до нескольких тысяч узлов.

Конфигурация младших узлов полностью контролируется старшими узлами, находящимися на более высоком уровне иерархии [7].

- Сценарии на основе мониторинга.
- Автоматическое обнаружение.
- Централизованный мониторинг журналов.
- Веб-интерфейс для администрирования и настройки.
- Отчётность и тенденции.
- SLA-мониторинг.
- Поддержка высокопроизводительных агентов (zabbix-agent).

практически для всех платформ

- Комплексная реакция на события.
- Поддержка SNMP v1, 2, 3.
- Поддержка SNMP-ловушек.
- Поддержка IPMI.
- Поддержка мониторинга JMX-приложений.
- Поддержка выполнения запросов в различные базы данных без необходимости использования сценарной обвязки.

- Расширение за счёт выполнения внешних скриптов.
- Гибкая система шаблонов и групп.
- Возможность создавать карты сетей.
- Интеграция с внешними системами с помощью плагинов.

Отдельный блок возможностей связан с автоматическим обнаружением: устройств по диапазону IP-адресов, доступных на них сервисах, также реализована SNMP-проверка. Обеспечивается автоматический мониторинг обнаруженных устройств, автоматическое удаление отсутствующих узлов, распределение по группам и шаблонам в зависимости от возвращаемого результата.

Основными недостатками Zabbix являются:

- Отсутствие встроенных средств управления агентами мониторинга. В коммерческих продуктах такие средства имеются. В Zabbix этого пока нет. Нет даже обновления инструментария на агентов.
- Скромная функциональность встроенной подсистемы визуализации. Grafana решает эту проблему.
- Встроенный мониторинг СУБД (ODBC). Дело в том, что такой мониторинг открывает у Zabbix отдельное подключение при каждом опросе метрики. И если база у вас большая (с большим количеством собираемых метрик), то пул соединений в ней может переполниться и база перестанет отвечать на запросы, в том числе для целевых систем.

2.3.2. Архитектура

Архитектура Zabbix включает в себя следующие элементы:

- Zabbix-сервер — ядро системы, которое дистанционно контролирует сетевые сервисы и является хранилищем, в котором содержатся все конфигурационные, статистические и оперативные данные. Он является тем субъектом в программном

обеспечении Zabbix, который оповещает администраторов о проблемах с контролируемым оборудованием.

- Zabbix-прокси собирает данные о производительности и доступности от имени Zabbix-сервера. Все собранные данные заносятся в буфер на локальном уровне и передаются Zabbix-серверу, к которому принадлежит прокси-сервер. Zabbix-прокси является идеальным решением для дистанционного контроля филиалов и других точек, в т.ч. сетей, не имеющих местных администраторов. Он может быть также использован для распределения нагрузки одного Zabbix-сервера. В этом случае, прокси только собирает данные, тем самым на сервер ложится меньшая нагрузка на ЦПУ и на устройства ввода/вывода.
- Zabbix-агент — программа контроля локальных ресурсов и приложений (таких как накопители, оперативная память, статистика процессора и т. д.) на сетевых системах, эти системы должны работать с запущенным Zabbix-агентом.
- Zabbix-агенты являются чрезвычайно эффективными из-за использования специфических системных вызовов для сбора информации и подготовки статистики.
- Веб-интерфейс — часть Zabbix-сервера, и, как правило (но не обязательно), запускается на том же физическом узле, что и Zabbix-сервер. Работает на PHP, требует веб-сервер (например: NGINX, Apache httpd).

Наиболее эффективная архитектура мониторинга с помощью Zabbix отображена на рисунке 4.

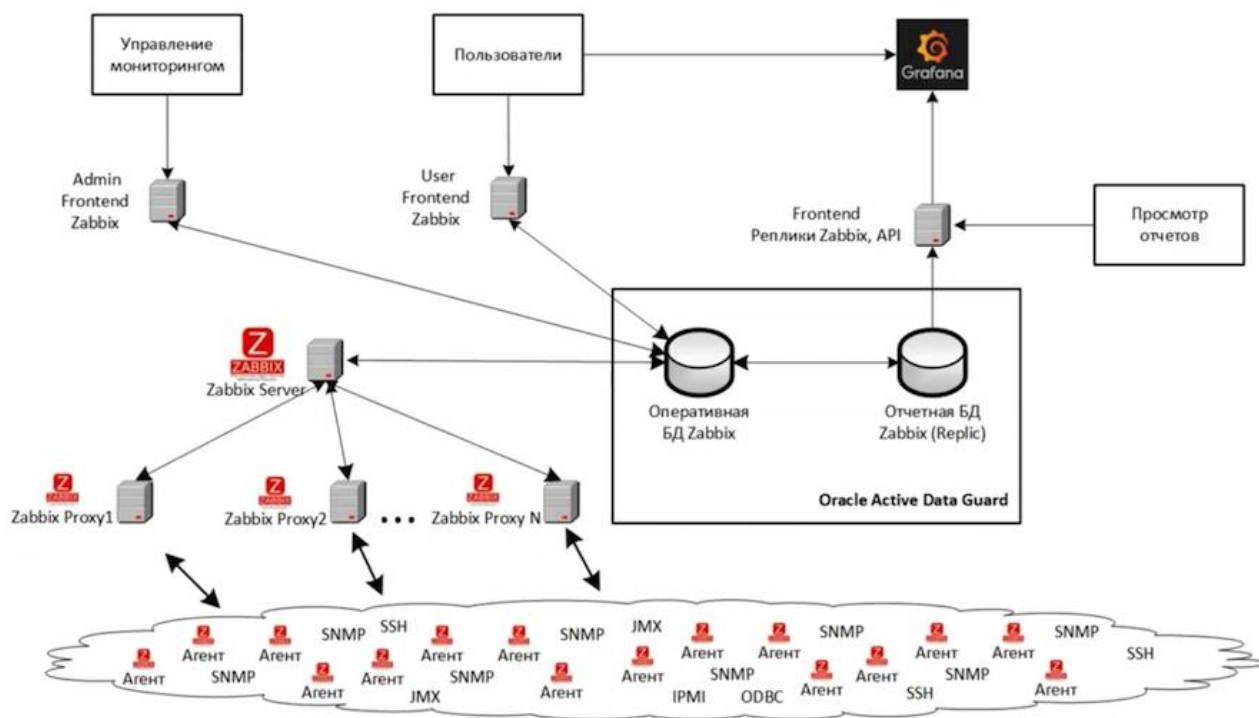


Рисунок 4 - архитектура мониторинга с помощью Zabbix [11]

2.4. Система Munin

Munin является свободным, с открытым исходным кодом компьютерной системы мониторинга, мониторинга сети и мониторинга инфраструктуры прикладного программного обеспечения.

Munin написан на Perl и использует RRDtool для создания графиков, которые доступны через веб-интерфейс. Основное внимание уделяется возможностям plug and play. В настоящее время доступно около 500 плагинов для мониторинга. Он предназначен для упрощения определения того, «что изменилось сегодня» при возникновении проблем с производительностью, а также для обеспечения прозрачности емкости и использования ресурсов [8].

Сама система состоит из двух независимых частей: сервера (сам munin), устанавливается на одну машину, куда собираются данные, и демона munin-node, который устанавливается на машины, которые мы будем мониторить. Сам этот демон представляет собой Perl-скрипт, который слушает 4949 порт с помощью Net::Server. При своём запуске он инициализирует плагины, установленные в /etc/munin/plugins. Раз в 5 минут сервер munin подключается

ко всем нодам, получает информацию от всех плагинов и сохраняет себе в базы rrdtool. Таким образом, для работы Munin'a не нужен даже MySQL.

2.4.1. Возможности

Сервис Munin представляет следующий набор возможностей [8]:

- Проведение логических группировок;
- Определение тенденций;
- Прогнозирование тенденций;
- Поддержка SNMP - стандартный интернет-протокол для управления устройствами в IP-сетях на основе архитектур TCP/UDP;
- Поддержка плагинов, триггеров / оповещений;
- Веб интерфейс;
- Поддержка распределенного мониторинга ;
- Поддержка Perl;
- Метод хранения данных RRDtool;
- Лицензия GNU GPL;
- Поддержка IPv6.

2.4.2. Архитектура

Архитектура мастер-узлов Munin отображена на рисунке 5.

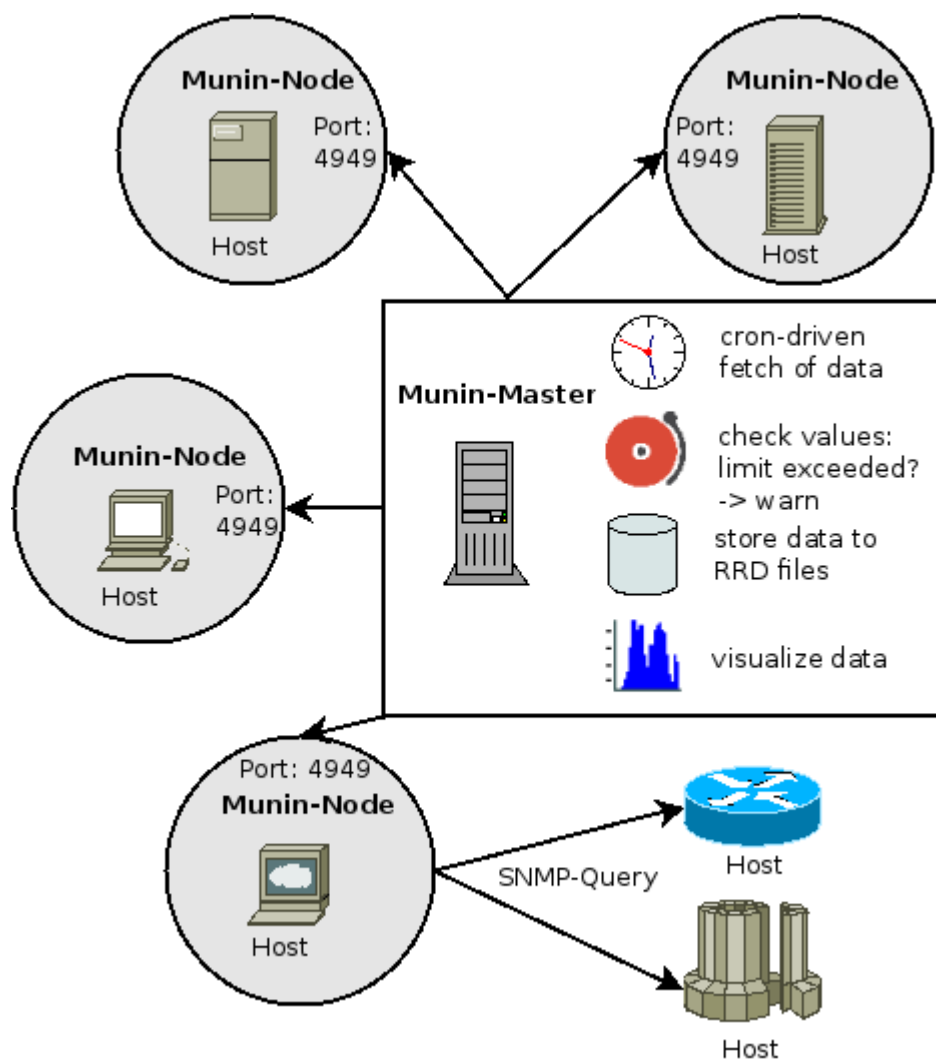


Рисунок 5 – Архитектура Munin [11]

Составные части Munin:

- **Мунин-Мастер.**
 - Мастер отвечает за все центральные задачи, связанные с Мунином. Он регулярно подключается к различным узлам, а затем синхронно запрашивает конфигурацию и значения различных показателей и сохраняет данные в файлах RRD.
- **Мунин-Узел.**
 - Узел — это небольшой агент, работающий на каждом отслеживаемом хосте. Также может быть безагентный мониторинг, но это особый случай. На машинах без встроенной поддержки сценариев Perl можно использовать

mupin-с, который является переписанным С компонентов узла mupin.

- Мунин-Плагин.
 - Плагин mupin — это исполняемый файл, роль которого заключается в сборе одного набора фактов о локальном сервере (или получении данных с удаленного компьютера через SNMP). Плагин вызывается с аргументом «config» для получения метаданных и без аргументов для получения значений. Это обязательные аргументы для каждого плагина.

2.5. Сравнительные анализ

2.5.1. Критерии сравнения

В качестве основных бинарных критериев сравнения применительно к задаче мониторинга и реагирования для систем требующих оркестрации выделены следующие [9]:

- Распределённый сбор метрик.
- Наличие API.
- Наличие графической визуализации.
- Наличие предсказательного механизма.
- Наличие механизма реагирования нештатной ситуации в работе системы.
- Задание сценариев реагирования на нештатную ситуацию.

2.5.2. Проведение сравнения

Таблица 1 – Сравнение систем мониторинга и реагирования

Средство мониторинга/Критерий сравнения	Распределённый сбор метрик	Наличие API	Наличие графической визуализации	Наличие предсказательного механизма	Наличие механизма реагирования на нештатной ситуации в работе системы	Задание сценариев реагирования на нештатную ситуацию
Prometheus	+	+	+	-	+	+
Sentry	+	+	+	+	+	-
Zabbix	+	-	+	+	+	-
Munin	+	+	+	-	+	-
Разрабатываемая система мониторинга и реагирования систем	+	+	+	+	+	+

2.6. Постановка задачи

Основным назначением системы мониторинга и реагирования является предупредить критическое состояние системы, которое может привести к переводу системы в нестабильный режим работы, или спровоцировать сбой [10]. А также решать следующие задачи:

1. Отслеживание основных метрик систем оркестрации влияющих на стабильность работы системы (метрики кластера рассчитываются агрегируя метрики контейнеров, находящихся в нем):
 - a. Количество контейнеров в кластере.
 - b. Загрузка CPU кластера на интервале.
 - c. Загрузка RAM кластера на интервале.
 - d. Загрузка SSD кластера на интервале.
2. Определение критических показателей кластеров

- a. Автоматически, система накапливает конфигурации значений метрик, при которых происходили сбои, и выделяет из них критические диапазоны.
 - b. Вручную, то есть системе заранее вводятся критические диапазоны для каждой метрики.
- 3. Реагирование на изменение метрик кластера (попадание их в критический диапазон). Предусмотрены следующие реакции.
 - a. Выполнение подготовленного сценария реагирования.
 - b. Оповещение ответственного лица.

2.6.1. Основные исследуемые показатели системы мониторинга и реагирования

Основными исследуемыми показателями системы являются:

- a. Количество контейнеров в кластере.
- b. Загрузка CPU кластера на интервале.
- c. Загрузка RAM кластера на интервале.
- d. Загрузка SSD кластера на интервале.

2.6.2. Формулировка требований

2.6.2.1. Функциональные требования

В качестве функциональных требований к АСМР в данной работе будем считать следующий список сценариев использования:

- 1. Конфигурирование АСМР.
- 2. Сбор метрик исследуемой системы.
- 3. Задание сценария реагирования системы.
- 4. Проверка совпадения метрик значениям в соответствующем сценарии.
- 5. Применение реагирования к исследуемой системе.
 - a. Модернизация оркестрируемой системы.
 - b. Оповещение ответственного лица.

6. Обработка реакций АСМР.

Подробная проработка и описание данных сценариев использования описано в разделе 3.3.

2.6.2.2. Нефункциональные требования

Основные нефункциональные требования к АСМР описаны в таблице 2.

Таблица 2 – Нефункциональные требования

Требование	Описание требования	Значение
Повторное использование	Требования к повторному использованию реализации или компонентов приложения, а также реализация приложения с возможностью повторного его использования для различных задач	Компоненты системы мониторинга и реагирования должны иметь возможность использоваться для выполнения похожих задач, без создания новых компонентов
Расширяемость	Требования к расширяемости приложения в связи с появлением новых функциональных требований	Система мониторинга и реагирования должна поддерживать внедрение новых функциональных требований без перепроектирования системы в целом
Портативность	Требования к портируемости (переносимости) приложения на различные платформы	Система мониторинга и реагирования должна поддерживаться такими платформами как: Windows 10, Linux, MacOS 10 - 11
Совместимость	Требования к взаимодействию между компонентами решения, между внешними компонентами, использование стандартных протоколов и технологий взаимодействия	Взаимодействие между частями системы мониторинга и реагирования реализовано по протоколу TCP IP в архитектуре REST API

Продолжение таблицы 2 – Нефункциональные требования

Требование	Описание требования	Значение
Модульность	Требования к разделению приложения на модули	Система мониторинга и реагирования должна состоять из обособленных, по смыслу назначенных на нее задач, модулей
Совместимость	Особые требования к совместимости между версиями приложений, между различными приложениями и внешними подсистемами	Система мониторинга и реагирования должна иметь возможность взаимодействовать с контейнеризированными системами Docker, а также с системами оркестрации Kubernetes, Apache Mesos, Docker Swarm

2.7. Вывод к главе

Из того, что описано в главе можно выделить то, что не одна из исследуемых систем не соответствует всему списку критериев. И чтобы быть максимально конкурентно способной, именно проектируемая в этой работе система мониторинга и реагирования, должна соответствовать вышеописанным свойствам.

В качестве финального этапа главы были определены функциональные требования, также были определены и описаны нефункциональные требования к системе мониторинга и реагирования.

3. Проектирование системы мониторинга и реагирования

3.1. Построение модели сценариев использования

В качестве основных действующих лиц (Actors) в сценариях работы системы мониторинга и реагирования выделены следующие:

1. Исследуемая система требующая оркестрации – система, данные о работе которой, собирает и анализирует АСМР;
2. Оператор системы – лицо, участвующее в процессе оповещений и реагировании на критические изменения в показателях исследуемой системы;
3. Администратор АСМР – лицо конфигурирующее систему АСМР и задающее сценарии реагирования.

В таблице 3 приведен профиль действующих лиц системы мониторинга и реагирования.

Таблица 3 – Профили действующих лиц.

Название	Профиль
Администратор АСМР	Специально подготовленный, квалифицированный персонал, обладающий навыками работы с контейнеризированными и оркестрованными системами. А также понимающий основные принципы мониторинга за сложноорганизованными системами [11].
Оператор АСМР	Лицо чьи задачи решает исследуемая система, и в чьи интересы входит организация бесперебойной и безотказной работы исследуемой системы посредством АСМР.
Исследуемая система	Набор контейнеризованных частей системы объединенных в кластер для решения определенных бизнес-задач [12].

Модель сценариев использования системы мониторинга и реагирования за сервисами требующих оркестрации изображена на рисунке 6.

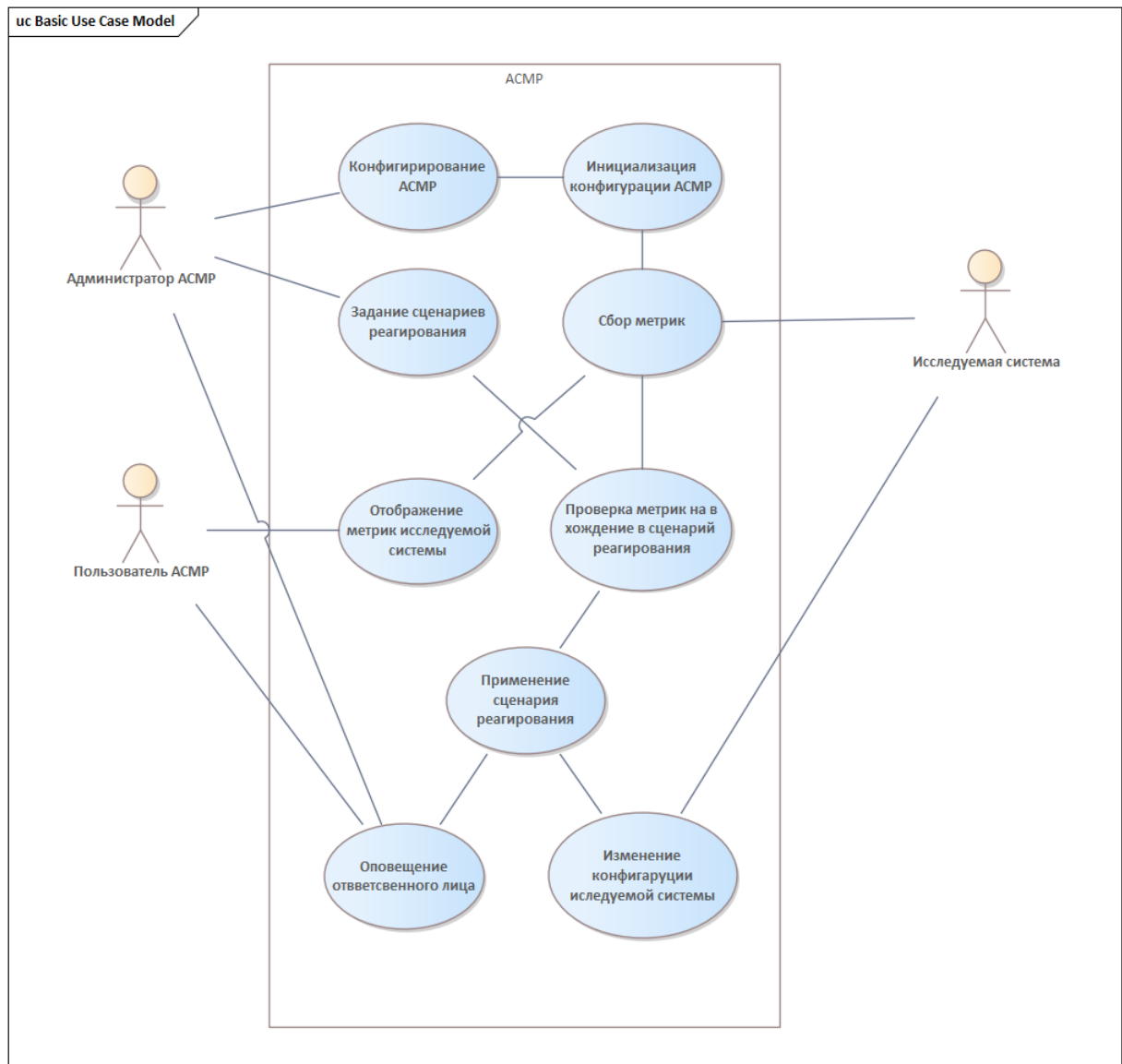


Рисунок 6 – Use case модель системы мониторинга и реагирования

В системе мониторинга и реагирования участвуют следующие сценарии использования:

1. Конфигурирование АСМР.
2. Инициализация конфигурации АСМР.
3. Задание сценариев реагирования.
4. Сбор метрик.
5. Отображение метрик исследуемой системы.
6. Проверка метрик на вхождение в сценарий реагирования.

7. Оповещение ответственного лица.
8. Изменение конфигурации исследуемой системы.

3.2. Основные состояния системы мониторинга и реагирования

В АСМР предусмотрены состояния работы, в рамках каждого из которых система мониторинга и реагирования решает определенные задачи [13]:

1. Отключена – объекты мониторинга АСМР и сценарии реагирования не заданы или статус отслеживания показателей исследуемой системы – «не активен». При этом АСМР успешно установлена и не имеет препятствий для задания конфигурации.

2. Инициализация новой конфигурации мониторинга – АСМР считывает новые объекты мониторинга, система проверяет подключение к кластерам и наличие исследуемых метрик.

3. Инициализация новой конфигурации сценариев реагирования – АСМР считывает новые сценарии реагирования, система проверяет подключение к кластерам и наличие исследуемых метрик.

4. Мониторинг – штатный режим работы, в ходе которого АСМР накапливает метрики исследуемой системы.

5. Применение сценария – режим работы, в ходе которого АСМР изменяет конфигурацию исследуемой системы или оповещает ответственное лицо.

3.3. Подробное описание основных сценариев использования

3.3.1. Конфигурирование АСМР

Участвующие лица: Администратор АСМР.

Начальные условия: не требуются.

Область действия: АСМР.

Цель пользователя: изменить объекты мониторинга.

Подробно сценарий конфигурирования АСМР описан в таблице 4.

Таблица 4 – Конфигурирование АСМР

Шаг	Состояние	Событие	Действие системы
1	Любое из описанных в разделе 2.2.	Администратор АСМР посредством конфигурационного YAML файла задает объекты мониторинга [14].	Бездействие
2	Любое из описанных в разделе 2.2.	Администратор АСМР сохраняет конфигурацию системы в отличном от предыдущего файле.	Бездействие

3.3.2. Инициализация конфигурации АСМР

Участвующие лица: Администратор АСМР.

Начальные условия: не требуются.

Область действия: АСМР.

Цель пользователя: применить новую конфигурацию мониторинга АСМР.

Подробно сценарий инициализации новой конфигурации мониторинга описан в таблице 5.

Таблица 5 – Инициализация новой конфигурации

Шаг	Состояние	Событие	Действие системы
1	Любое из описанных в разделе 2.2.	Администратор АСМР запускает команду инициализации нового конфигурационного файла	Система проверяет конфигурационных файл на наличие синтаксических ошибок.
1.1	Любое из описанных в разделе 2.2.	АСМР обнаружила ошибку в заданном конфигурационном файле	АСМР выводит сообщение о том, что при инициализации нового файла произошла ошибка. Сценарий прерывается.

Продолжение таблицы 5 – Инициализация новой конфигурации

Шаг	Состояние	Событие	Действие системы
2	Любое из описанных в разделе 2.2.	АСМР не обнаружила ошибок в новом конфигурационном файле	АСМР выводит сообщение о подтверждении новой конфигурации мониторинга
2.1	Любое из описанных в разделе 2.2.	Администратор отклоняет применение новой конфигурации	АСМР прекращает процесс инициализации новой конфигурации. Сценарий прерывается.
3	Любое из описанных в разделе 2.2.	Администратор подтверждает применение новой конфигурации	Система переходит в режим инициализации новой конфигурации мониторинга. Проверяет подключение к объектам мониторинга и проверяет наличие исследуемых метрик системы.
3.1	Инициализация новой конфигурации мониторинга	АСМР не обнаружила объекты мониторинга или файл метрик кластера	Система выводит пользователю сообщение о том объекты мониторинга или файлы метрик кластера не обнаружены. Предлагает прервать процесс инициализации или продолжить в текущем виде.
3.1.2	Инициализация новой конфигурации мониторинга	Администратор подтверждает прерывание процесса инициализации новой конфигурации	АСМР прекращает процесс инициализации новой конфигурации, и возвращает систему в предыдущее состояние. Сценарий прерывается.

Шаг	Состояние	Событие	Действие системы
3.2	Инициализация новой конфигурации мониторинга	Пользователь соглашается продолжить инициализацию в текущем виде (Подразумевая, что объекты мониторинга или исследуемые метрики появятся позже)	Система сохраняет новую конфигурацию, и переходит в режим «Мониторинг». Создает в БД АСМР соответствующую таблицу. Сценарий прерывается.
4	Инициализация новой конфигурации мониторинга	Система обнаружила объекты мониторинга и исследуемые показатели системы [15].	Система сохраняет новую конфигурацию, и переходит в режим «Мониторинг». Создает в БД АСМР соответствующую таблицу. Сценарий прерывается.

3.3.3. Задание сценариев реагирования

Участвующие лица: Администратор АСМР.

Начальные условия: АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».

Область действия: АСМР.

Цель пользователя: задать сценарий реагирования нештатную ситуацию исследуемой системы.

Подробно сценарий описан в таблице 6.

Таблица 6 – Задание сценария реагирования

Шаг	Состояние	Событие	Действие системы
1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	Администратор АСМР посредством конфигурационного YAML файла задает сценарии реагирования в формате, описанном в главе 3.	Бездействие
2	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	Администратор АСМР сохраняет конфигурацию системы в отличном от предыдущего файле.	Бездействие
3	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	Администратор АСМР запускает процесс инициализации новых сценариев реагирования	Система проверяет конфигурационных файл на наличие синтаксических ошибок.
3.1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	АСМР обнаружила ошибку в заданном конфигурационном файле	АСМР выводит сообщение о том, что при инициализации нового файла произошла ошибка. Сценарий прерывается.

Шаг	Состояние	Событие	Действие системы
4	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	АСМР не обнаружила ошибок в новом конфигурационном файле	АСМР выводит сообщение о подтверждении новой конфигурации сценариев реагирования
4.1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	Администратор отклоняет применение новой конфигурации	АСМР прекращает процесс инициализации новой конфигурации. Сценарий прерывается.
5	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»	Администратор подтверждает применение новой конфигурации сценариев реагирования	Система переходит в режим инициализации новой конфигурации сценариев реагирования. Проверяет подключение к объектам мониторинга и проверяет наличие исследуемых метрик системы, используемых в сценарии [16].
5.1	Инициализация новых сценариев реагирования	АСМР обнаружила объекты мониторинга и метрики используемые в сценарии реагирования	Система сохраняет конфигурацию конкретного сценария прошедшего проверку.

Шаг	Состояние	Событие	Действие системы
5.2	Инициализация новых сценариев реагирования	АСМР не обнаружила объекты мониторинга или исследуемые метрики системы	Система выводит сообщение о том, что не обнаружены объекты мониторинга или исследуемые метрики, участвующие в сценарии. Система не инициализирует сценарий с ошибкой к подключению и переходит инициализации следующего сценария.
6	Инициализация новых сценариев реагирования	АСМР проверила подключение и наличие всех объектов мониторинга и метрик, используемых в сценариях, описанных в конфигурационном файле	Система выводит сообщение со списком сценариев и статусом их инициализации в АСМР. Сценарий прерывается.

3.3.4. Сбор метрик

Участвующие лица: АСМР, исследуемая система

Начальные условия: Система находится в режиме «Мониторинг»

Область действия: АСМР

Цель: собрать показатели исследуемой системы

Подробно сценарий сбора метрик описан в таблице 7.

Таблица 7 – Сбор метрик

Шаг	Состояние	Событие	Действие системы
1	Мониторинг	Число временных тактов системы с предыдущего сбора сравнялось с конфигурационным интервалом сбора метрик.	Согласно конфигурации, система подключается к объекту мониторинга.
1.1	Мониторинг	АСМР не удалось подключиться к объекту мониторинга.	Система записывает сообщение о неудачном подключении к объекту мониторинга. Возможен переход к сценарию реагирования. АСМР переходит к следующему объекту мониторинга.
2	Мониторинг	АСМР успешно подключилась к объекту мониторинга.	Система, согласно конфигурации, запрашивает у объекта мониторинга его показатели.
2.1	Мониторинг	Объект мониторинга не предоставил метрики описанные в конфигурационном файле АСМР.	Система записывает сообщение о том, что объект мониторинга не предоставил метрики, описанные в конфигурационном файле. АСМР переходит к следующему исследуемому показателю.
3	Мониторинг	Объект мониторинга предоставил исследуемые метрики .	АСМР сохраняет в БД собранные показатели объекта мониторинга. Сценарий прерывается.

3.3.5. Отображение метрик исследуемой системы

Участвующие лица: Администратор АСМР, пользователь АСМР

Начальные условия: АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации»

Область действия: АСМР

Цель пользователя: посмотреть исследуемые метрики объектов мониторинга

Подробно сценарий описан в таблице 8.

Таблица 8 – Отображение метрик исследуемой системы

Шаг	Состояние	Событие	Действие системы
1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	Пользователь запрашивает доступный список объектов мониторинга.	АСМР посылает запрос на получение списка доступных объектов мониторинга в БД.
1.1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	АСМР не получила ни одного объекта мониторинга в БД.	Система выводит пользователю сообщение о том, в АСМР не содержится доступных объектов мониторинга.

Продолжение таблицы 8 – Отображение метрик исследуемой системы

Шаг	Состояние	Событие	Действие системы
2	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	АСМР успешно получила в БД не нулевой список объектов мониторинга.	Система выводит пользователю список доступных объектов мониторинга.
3	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	Пользователь запрашивает доступные метрики определенного объекта мониторинга.	Система запрашивает в БД доступные метрики заданного объекта мониторинга [17].
3.1	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	АСМР не обнаружила ни одной доступной метрики заданного объекта мониторинга.	Система выводит пользователю сообщение о том, что для данного объекта мониторинга нет доступных метрик.

Продолжение таблицы 8 – Отображение метрик исследуемой системы

Шаг	Состояние	Событие	Действие системы
4	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	АСМР успешно запросила в БД список доступных метрик объекта мониторинга.	Система выводит список доступных метрик определенного объекта мониторинга.
5	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	Пользователь запрашивает значения определенной метрики объекта мониторинга, указывая период выборки, и формат вывода результатов метрики.	Система запрашивает в БД значения определенной метрики конкретного объекта мониторинга.
6	АСМР не находится в режиме «Отключена» или в режиме «Инициализация новой конфигурации».	АСМР получила значение определенной метрики конкретного объекта мониторинга.	Система формирует формат вывода, выбранный пользователем, и предоставляет ему доступ. Сценарий завершается.

3.3.6. Проверка метрик на вхождение в сценарий реагирования

Участвующие лица: АСМР

Начальные условия: Система находится в режиме «Мониторинг»

Область действия: АСМР

Цель пользователя: Проверка на необходимость реагирования

Подробно сценарий описан в таблице 9.

Таблица 9 – Проверка метрик на вхождение в сценарий реагирования

Шаг	Состояние	Событие	Действие системы
1	Мониторинг	Число тактов системы с предыдущей проверки объекта мониторинга сравнялось с конфигурационным интервалом проверки на вхождение показателей в сценарий	АСМР запрашивает в БД последние значения метрик объекта исследования, описанных в конфигурации
2	Мониторинг	АСМР получила последние показатели метрики объекта мониторинга в БД	АСМР сравнивает показатели метрики со условием (условие может быть комбинированное и учитывающее тренд) вхождения в сценарий реагирования.
2.1	Мониторинг	Условия вхождения метрик системы в сценарий соблюдены	Система переходит в режим «Применение сценария». Запускается сценарий «Применение сценария реагирования»
3	Мониторинг	Условия вхождения метрик в сценарий реагирования не соблюдены	Система записывает результаты текущей проверки в БД. Сценарий прерывается.

3.3.7. Применение сценария реагирования

Участвующие лица: АСМР, исследуемая система

Начальные условия: Применение сценария реагирования

Область действия: АСМР, исследуемая система

Цель: применить сценарий реагирования к исследуемой системе

Подробно сценарий описан в таблице 10.

Таблица 10 – Применение сценария реагирования

Шаг	Состояние	Событие	Действие системы
1	Мониторинг	Система перешла в режим «Применение», в связи с тем, что было удовлетворено условие вхождения в сценарий реагирования одной или несколькими метриками объекта исследования	АСМР запрашивает в БД сценарий реагирования на изменение метрики объекта мониторинга
1.1	Применение сценария	АСМР получила сценарий оповещения ответственного лица как реакцию на текущее условие	Система запускает сценарий «Оповещение ответственного лица»
1.2	Применение сценария	АСМР получила сценарий изменения конфигурации исследуемой системы как реакцию на текущее условие	АСМР запускает сценарий «Изменение конфигурации исследуемой системы»
2	Применение сценария	Система запрашивает статус исполнения сценария реагирования	АСМР запрашивает статус исполнения сценария
2.1	Применение сценария	АСМР получила статус реагирования «В работе»	АСМР в соответствии с конфигурацией выставляет интервал запроса статуса исполнения и возвращается к шагу 2.

Продолжение таблицы 10 – Применение сценария реагирования

Шаг	Состояние	Событие	Действие системы
2.1.1	Применение сценария	Суммарное значение интервалов запроса превысило значение ожидания статуса	Система записывает инцидент, запускает сценарий «Оповещение ответственного лица» если таковое имеется, помечает сценарий как «Ошибка» и переходит к шагу 2.2.
2.2	Применение сценария	АСМР получила статус реагирования «Ошибка»	АСМР в соответствии с конфигурацией выставляет интервал запроса статуса исполнения и возвращается к шагу 2.
2.2.1	Применение сценария	АСМР получила статус исполнения «Ошибка» более 3 раз.	Система записывает инцидент, запускает сценарий «Оповещение ответственного лица» если таковое имеется, помечает сценарий как «Ошибка». Завершает сценарий и выставляет интервал следующей проверки сценариев данного объекта мониторинга 1 день [18]
2.3	Применение сценария	АСМР получила статус реагирования «Завершен»	Система записывает инцидент в БД АСМР. Сбрасывает интервалы повторных проверок объекта мониторинга до конфигурационных. Сценарий прерывается

3.3.8. Изменение конфигурации исследуемой системы

Участвующие лица: АСМР, исследуемая система

Начальные условия: АСМР находится в режиме «Применение сценария»

Область действия: Исследуемая система

Цель: изменить конфигурацию исследуемой системы

Подробно сценарий описан в таблице 11.

Таблица 11 – Изменение конфигурации исследуемой системы

Шаг	Состояние	Событие	Действие системы
1	Применение сценария	АСМР получила команду изменения конфигурации исследуемой системы согласно сценарию реагирования.	АСМР устанавливает соединение с объектом исследования и применяет команду согласно сценарию реагирования. Устанавливает статус реагирования «В работе».
2	Применение сценария	АСМР запрашивает у исследуемой системы статус исполнения команды реагирования	АСМР получает статус исполнения команды на исследуемой системе.
2.1	Применение сценария	В ходе получения статуса исполнения команды на исследуемой системе, АСМР получила ошибку	АСМР устанавливает статус реагирования «Ошибка». Сценарий завершается.
2.2	Применение сценария	В ходе получения статуса исполнения команды на исследуемой системе, АСМР получила статус «Выполнена»	АСМР устанавливает статус реагирования «Завершен». Сценарий завершается.

Шаг	Состояние	Событие	Действие системы
2.3	Применение сценария	В ходе получения статуса исполнения команды на исследуемой системе, АСМР не получила статус «Выполнена», но ошибок не обнаружено	Статус реагирования не изменяется система переходит к шагу 2.

3.4. Построение модели данных

Концептуальная схема данных системы мониторинга и реагирования в формате концептуальной модели представлена на рисунке 7. В модели изображены основные сущности АСМР, и взаимосвязи основных сущностей. Также модель включает в себя основные атрибуты АСМР. Атрибуты сущностей могут иметь более широкий характер, на диаграмме изображены базовые атрибуты, без которых работа АСМР в текущей реализации будет невозможна.

ER модель состоит из следующих сущностей:

1. Объекты мониторинга – сущность, в которой содержится описание основных объектов мониторинга, а также данные необходимые для подключения, включая тип оркестрационного и ПО.
2. Ответственные лица – сущность, в которой описаны ответственные лица системы мониторинга и реагирования.
3. Контакты – сущность, в которой содержатся возможные контакты ответственных лиц АСМР.
4. Коллекция метрик – сущность, в которой описаны основные показатели объекта мониторинга.
5. Типы метрик – сущность, в которой содержатся типы метрик используемые в АСМР.
6. Сценарии реагирования – сущность, содержащая в себе сценарии реагирования на нештатные ситуации АСМР.

7. Реакции – сущность, содержащая историю реакций АСМР на нештатные ситуации.

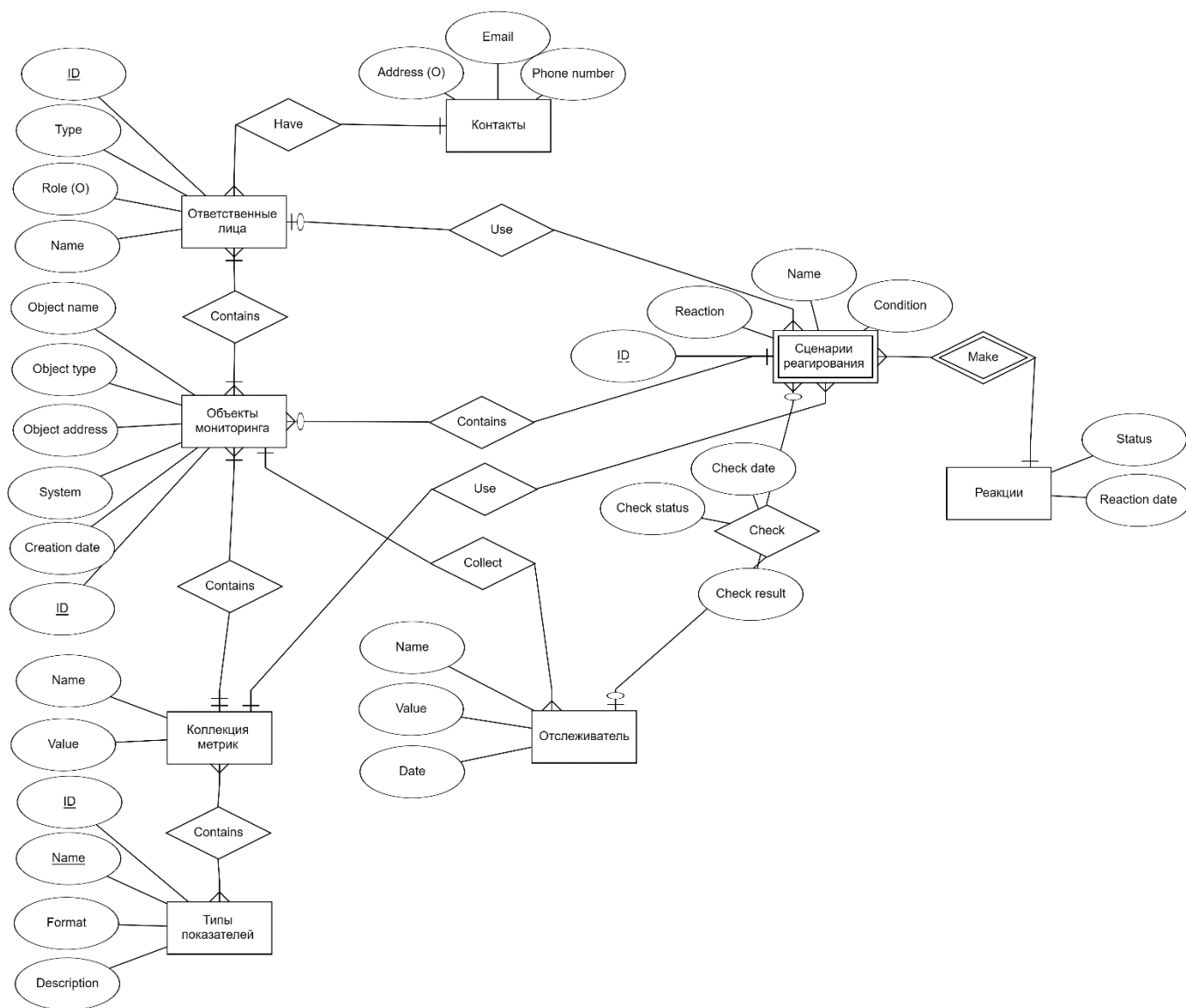


Рисунок 7 – Концептуальная модель данных АСМР

3.5. Построение компонентной модели

Компонентная модель системы мониторинга и реагирования представлена на рисунке 8.

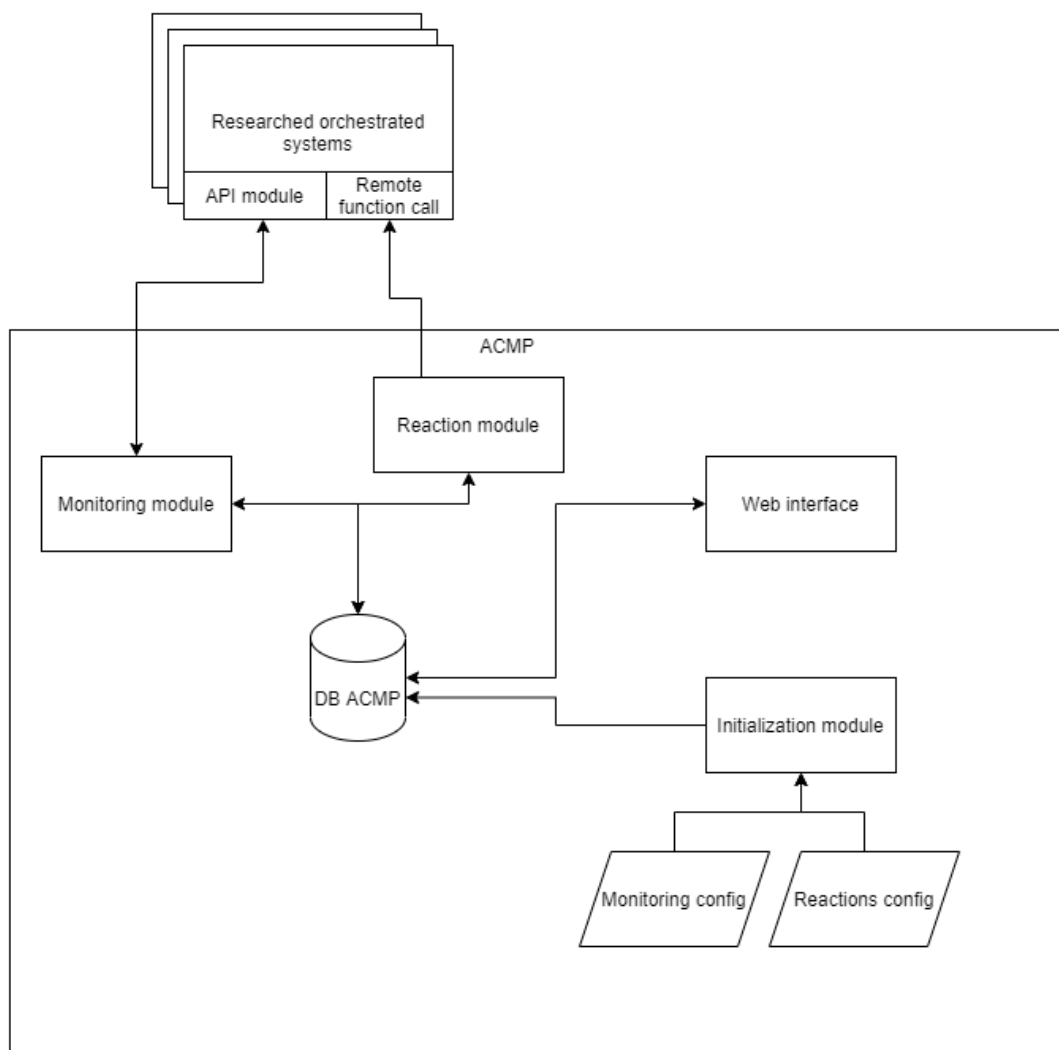


Рисунок 8 – Компонентная модель системы мониторинга и реагирования

Компонентная модель АСМР имеет модульную структуру и состоит из следующих обособленных модулей:

1. Monitoring module (модуль мониторинга).
2. Reaction module (модуль реакций).
3. Initialization module (модуль инициализации).

Конфигурационных YAML файлов:

1. Monitoring config (конфигурация мониторинга).
2. Reaction config (конфигурация реакций).

И веб-интерфейса выполняющего функции отображения: объектов мониторинга, показателей исследуемой системы, а также сценариев реагирования на нештатные ситуации.

3.6. Построение модели потоков данных

Модель потоков данных изображена на рисунке 9.

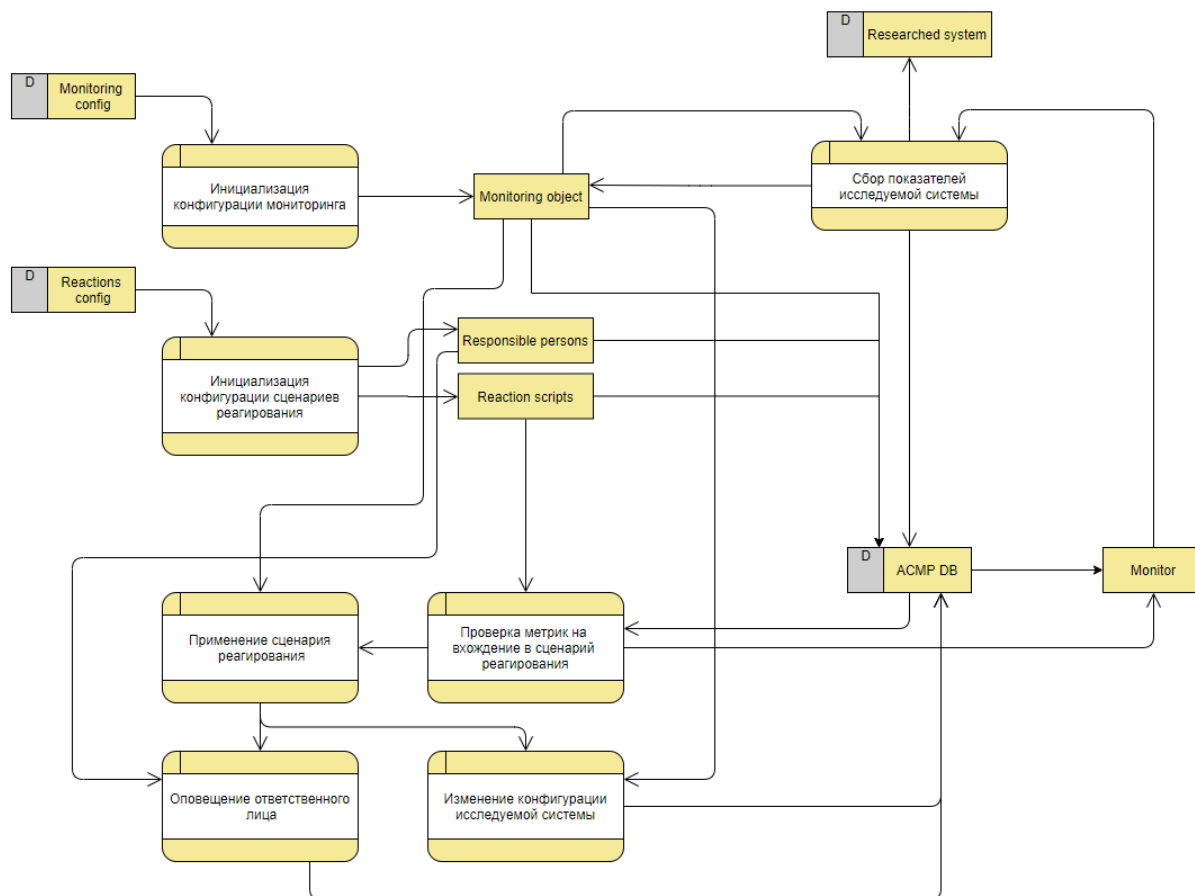


Рисунок 9 – Модель потоков данных системы мониторинга и реагирования

В модели потоков данных описаны основные процессы, сущности, а также пути, в рамках которых АСМР работает с данными [19].

3.7. Построение диаграмм последовательности

Основной целью диаграммы последовательности является отразить запросы и переходами между системами или обособленными компонентами одной системы, исходя из чего на рисунках 10 и 11 изображены диаграммы последовательности для сценария сбора метрик и сценария реагирования на нештатную ситуацию [20].

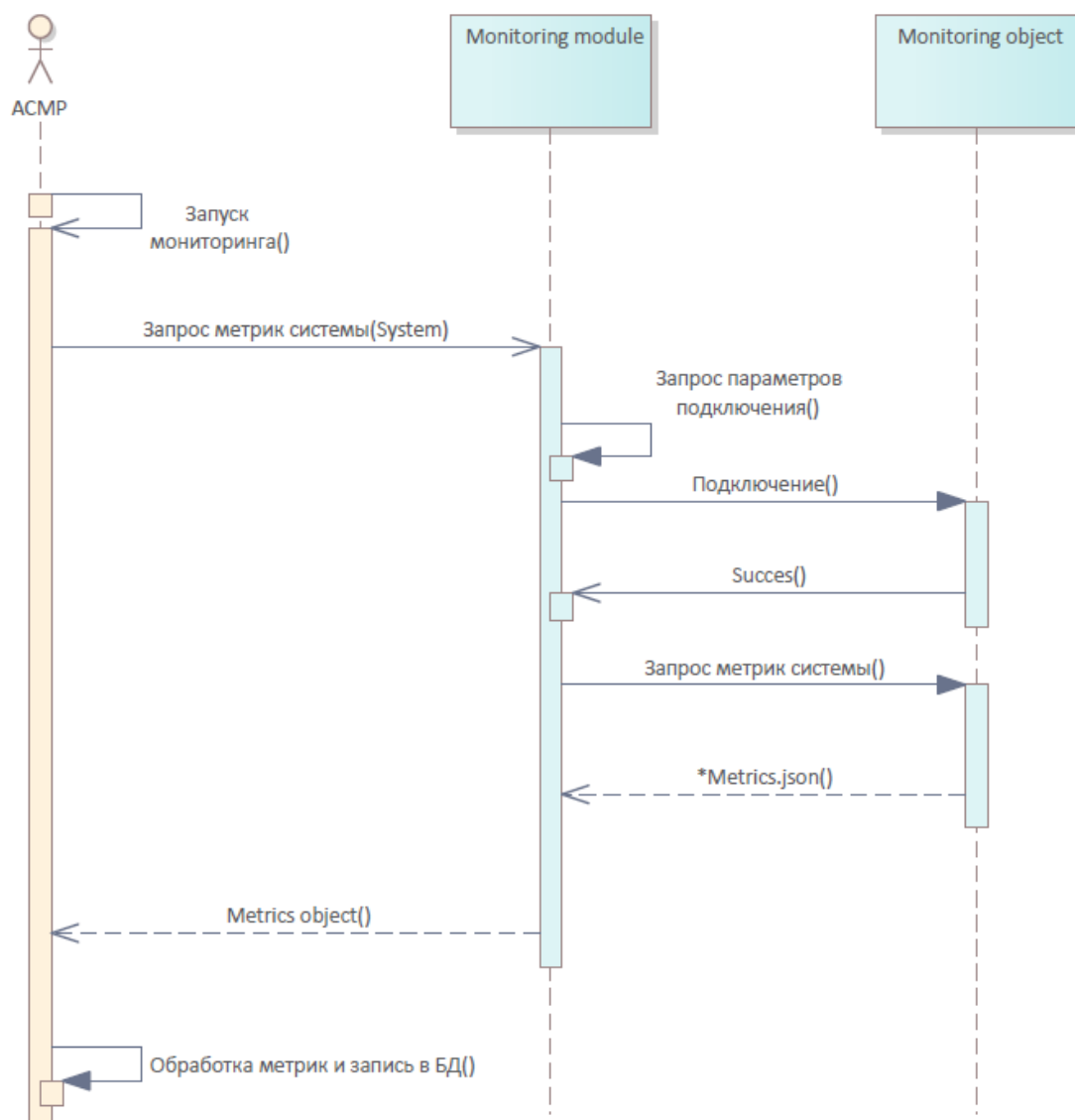


Рисунок 10 – Диаграмма последовательности сценария мониторинга

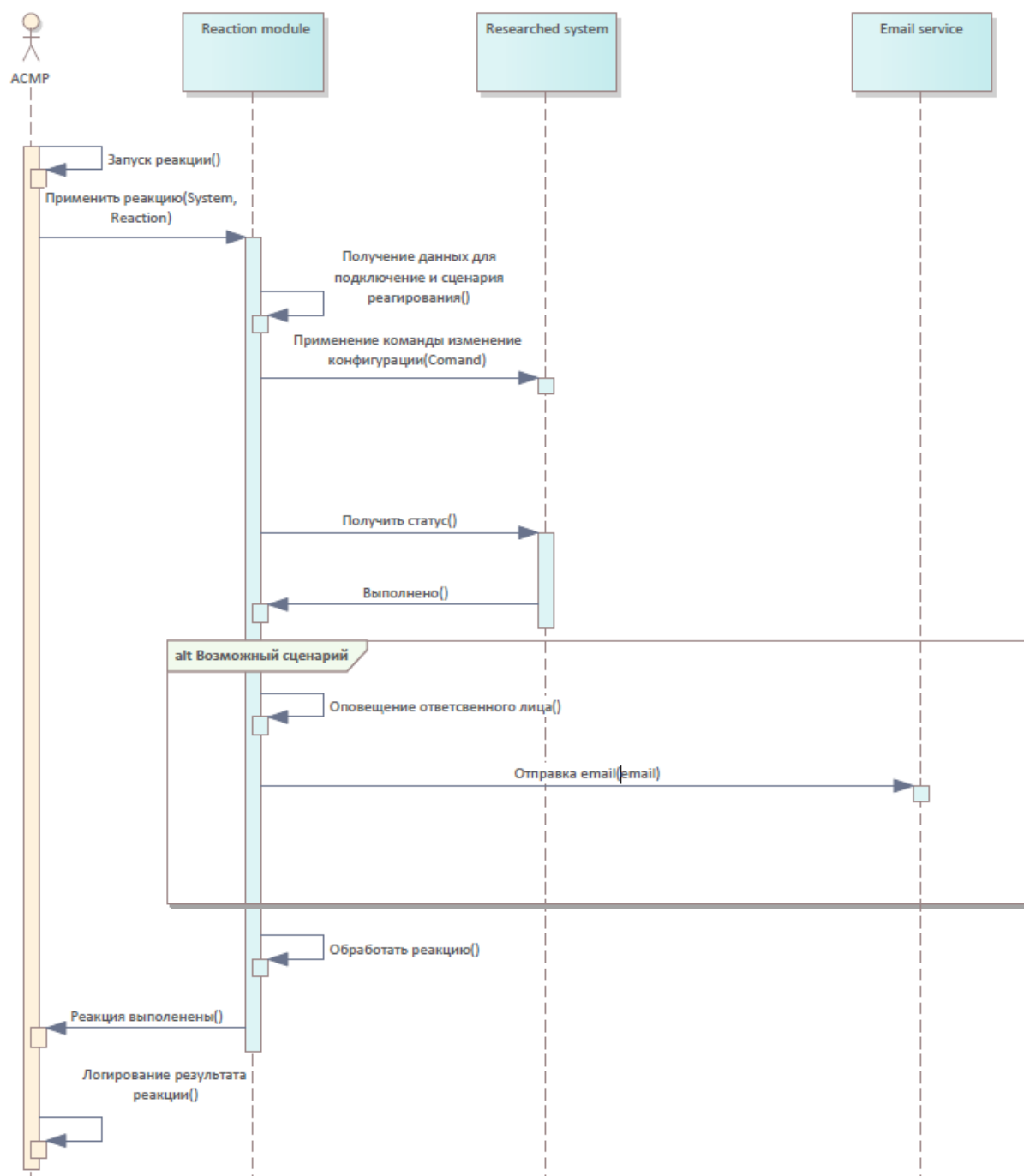


Рисунок 11 – Диаграмма последовательности сценария реагирования

3.8. Построение модели классов

Модель классов системы мониторинга и реагирования, как графическое представление статической структуры декларативных элементов представлена на рисунке 12.

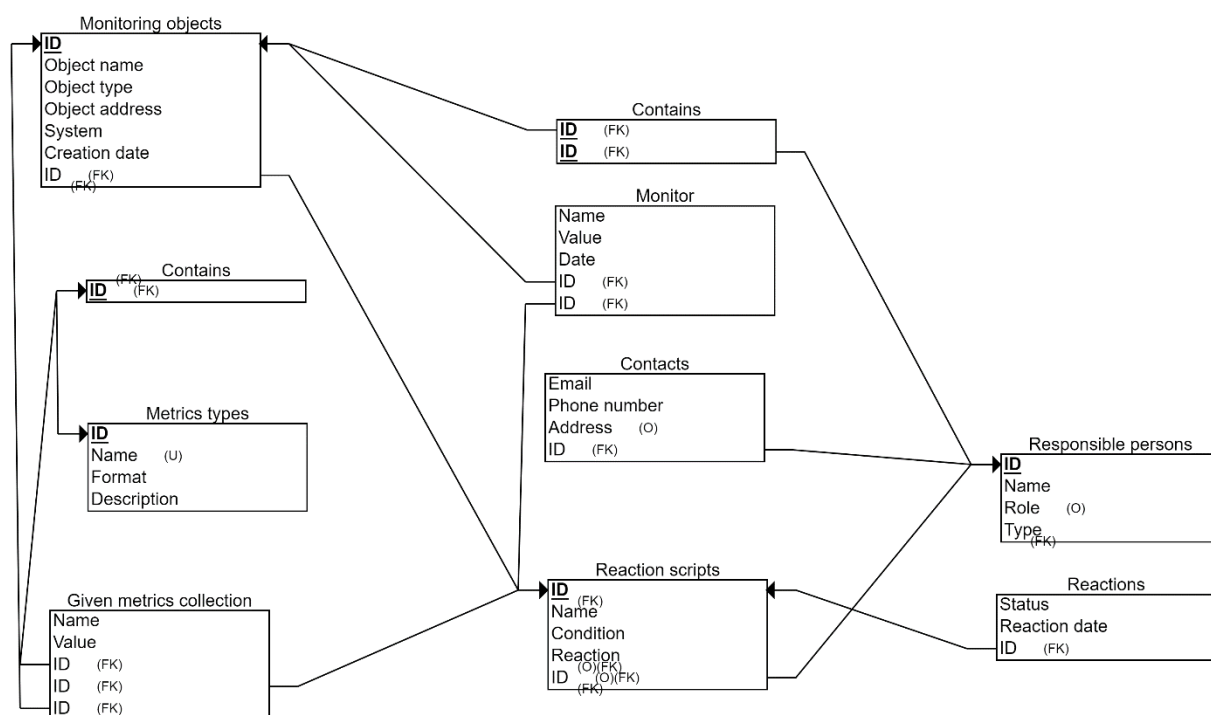


Рисунок 12 – Модель классов системы мониторинга и реагирования

3.9. Вывод к главе

В этой главе были определены и построены основные модели сценариев использования системы мониторинга и реагирования, также были описаны состояния системы вместе с подробным описанием сценариев использования, как раз опираясь на эти состояния. После чего была определена и построена модель данных АСМР вместе с моделью потоков данных. Далее определив и построив компонентную модель, были спроектированы диаграммы последовательности. В качестве завершающего этапа проектирования была определена модель классов системы мониторинга и реагирования.

Можно отметить, что именно этот набор моделей является минимальным, для открытия следующего этапа разработки системы мониторинга и реагирования – реализации [23].

4. Разработка механизма реагирования на критические состояния

Разработка механизма реагирования и формата описания сценариев одна из ключевых задач данной работы. На механизм реагирования ложатся следующие задачи:

1. Проверка показателей системы условию вхождения в сценарий реагирования.
2. Вызов процедур реагирования.
3. Проверка исполнения процедур реагирования.

4.1. Концепция и формат описания сценариев реагирования

В качестве инструмента описания сценариев реагирования в ASCMP считаем конфигурационный файл сценариев YAML. В котором последовательно задаются исследуемые метрики и процедуры реагирования участвующие в сценарии, а также условия срабатывания сценария, которое может быть комбинированным и зависящем от исторических показателей объекта мониторинга [24].

4.1.1. Используемые выражения механизма задания сценариев реагирования

Механизм задания сценариев реагирования базируется на синтаксисе YAML, где основным элементом является «последовательность», этот синтаксический элемент имеет следующую структуру:

--- Название последовательности “:”

“-“ Элемент последовательности

“-“ Элемент последовательности

“-“ Элемент последовательности

Количество элементов последовательности неограниченно. Так же элементом последовательности может быть другая последовательность. Название последовательности может любое кроме следующих зарезервированных значений:

1. ip_addr – строка объявления адреса к метрикам исследуемой системы.
2. target_clusters – строка объявления последовательности названий исследуемых кластеров.
3. target_parameters – последовательность объявления исследуемых параметров.
4. changes – последовательность объявления условий активации сценария.
 - a. metric – название исследуемого параметра (строка поддерживает арифметические операторы: “+”, “-”, “*”, “/”, “>”, “<”, “>=”, “<=”, “(”, “)”).
 - i. metric_date_start – дата старта выборки в формате UTC.
 - ii. metric_date_end – дата конца выборки в формате UTC (По умолчанию TODAY).
 - b. value – изменение исследуемого параметра (строка поддерживает арифметические операторы: “+”, “-”, “*”, “/”, “>”, “<”, “>=”, “<=”, “(”, “)”).

- c. MAX – получение максимального значения выборки.
 - d. MIN – получение минимального значения выборки.
 - e. VAR – расчет дисперсии.
 - f. EXP – расчет мат. Ожидания.
 - g. AVG – расчет среднего арифметического выборки.
 - h. LAST – получение последнего значение выборки.
 - i. MODE – получение моды выборки.
 - j. MEDIAN – получение медианы выборки.
5. reaction – последовательность объявления реакций системы. Реакции могут быть только поддерживаемые системой ASCMP.
- a. NOTIFICATION – оповещение ответственного лица.
 - b. REBOOT – перезагрузка инстанса.
 - c. ROLLOUT – откат к предыдущей версии.
 - d. DELETE – удалить кластер.
 - e. RUN – развернуть новый кластер.
 - f. PATCH – обновить кластер.
 - g. APPLY – создать ресурс кластер.
 - h. CORDON – пометить узел как не назначенный.
 - i. UNCORDON – снять отметку о не назначении.
 - j. DRAIN – вытеснить узел.
 - k. SCALE – масштабировать кластер.
6. reaction_parameter – параметр обработки процедуры реакции.

4.1.2. Синтаксис механизма задания сценариев реагирования

Синтаксис механизма реагирования основывается на синтаксисе YAML.

Из особых моментов стоит отметить следующее:

- a. Названия сценариев реагирования являются названиями словарей первого уровня в YAML файле.

- б. Условиями объявления сценариев реагирования являются строки включающие математические и логические операторы, а также статические операторы, агрегирующие выборку значений метрик.
- с. Описанием реакции сценарии является словарь, состоящий из функций реакции и параметров этих функций [25].

4.1.2.1. Примеры сценариев описанных в формате ASCMP

Увеличить счетчик репликации на 1 при росте последних дневных средних значений загрузки CPU более 90% и оповестить ответственное лицо, пример такого сценария представлен на рисунке 13.

```
#Увеличить счетчик репликации на 1 при росте последних дневных средних значений загрузки CPU более 90% и оповестить ответственное лицо
#Название сценария
CPU_90:
#Путь к метрикам инстанса
ip_addr: http://localhost:9323/metrics
#Названия исследуемых кластеров инстанса
targets_clusters:
- Njinx
- Red hat
- MySQL
#Названия исследуемых метрик системы
target_parameters:
- CPU
#Текущее число реплик
- REP_NUMBER
changes:
#Названия изменений
change1:
#Дата в формате UTC старта выборки
metric_start_date: 1620647984
#Если не указана metric_end_date, используется значение по умолчанию - TODAY
#Метрика
metric: AVG(CPU)
#Значение метрики
value: "> 0.9"
#Тип реакции, масштабирование инстанса
reaction: SCALE
#Параметр масштабирования
reaction_parameter: LAST(REP_NUMBER) + 1
Change2:
metric_start_date: 1620647984
metric: AVG CPU
value: "> 0.9"
reaction: NOTIFICATION
reaction_parameter:
```

Рисунок 13 – Пример конфигурации увеличение счетчика репликации

Ниже описано пример конфигурации позволяющий откатить инстанс на предыдущую версию при достижении 80% загрузки RAM, представлена на рисунке 14.

```
#Откатить инстанс на предыдущую версию при достижении 80% загрузки RAM

#Название сценария
MEM_80:
  #Путь к метрикам инстанса
  ip_addr: http://localhost:9323/metrics
  #Названия исследуемых кластеров инстанса
  targets_clusters:
    - Njinx
    - Red hat
    - MySql
  #Названия исследуемых метрик системы
  target_parameters:
    #Значение загрузки памяти
    - MEM
  changes:
    #Названия изменений
    change1:
      #Дата в формате UTC старта выборки
      metric_start_date: 1620647984
      #Если не указана metric_end_date, используется значение по умолчанию - TODAY
      #Метрика
      metric: LAST(MEM)
      #Значение метрики
      value: "> 0.8"
      #Тип реакции, откат
      reaction: ROLLOUT
      #Параметр отката
      reaction_parameter: UNDO
```

Рисунок 14 – Пример конфигурации отката на предыдущую версию

Ниже на рисунке 15 описано правило позволяющее снять отметку о назначении зависимого инстанса при превышении суммарного значения количества имеющихся реплик на основном инстансе более 10:

```
#Снять отметку о назначении зависимого инстанса при превышении суммарного значения количества имеющихся реплик на основном инстансе более 10

#Название сценария
REP_10:
  #Путь к метрикам инстанса
  ip_addr: http://localhost:9323/metrics
  #Названия исследуемых кластеров инстанса
  targets_clusters:
    - Njinx
    - Red hat
    - MySql
  #Названия исследуемых метрик системы
  target_parameters:
    #Текущее число реплик
    - REP_NUMBER
  changes:
    #Названия изменений
    change1:
      #Дата в формате UTC старта выборки
      metric_start_date: 1620647984
      #Если не указана metric_end_date, используется значение по умолчанию - TODAY
      #Метрика
      metric: LAST(REP_NUMBER)
      #Значение метрики
      value: 10
      #Тип реакции, откат
      reaction: CORDON
      reaction_parameter:
```

Рисунок 15 – Пример конфигурации – снятия отметки о назначении

4.2. Обработка сценариев системой мониторинга и реагирования

Основным инструментом интерпретации сценариев реагирования является скрипт Python 3, с использованием библиотек yaml, requests,

paramiko. Обработывая yaml файл скрипт реагирования на основе заранее заданного словаря метрик (рисунок 14) запрашивает метрики системы по URL указанном в ip_addr с диапазоном указанным в metric_start_date и metric_end_date. Далее согласно словарю команд, получает численное значение metric и value. После чего с использование интерпретационного инструмента python – eval (выбор интерпретационного инструмента, является одной из проблем данной работы, и может быть одним из моментов для дальнейшего развития) получает логический результат активации сценария (True/False).

Следующим этапом при получении положительного логического результата активации сценария, устанавливает защищенное подключение с инстансом с помощью библиотеки docker-py или аналогичной для оркестратора, и согласно словарю реакций запускает команду реакции на инстансе.

4.3. Вывод к главе

Результатом данной главы является описание концепции и механизма формулирования сценария реагирования для системы мониторинга и реагирования. Стоит отметить, что формулирование сценариев реагирования, является одной из ключевых функций АСМР и именно от него в большей мере зависит обеспечение бесперебойной работы и повышение надежности систем требующих оркестрации.

5. Протипирование системы мониторинга и реагирования

Основные цели прототипа системы мониторинга и реагирования для систем требующих оркестрации следующие:

- Продемонстрировать работу части функционала системы.
- Сформировать условия для возможности проверить целесообразность реализации системы мониторинга и реагирования для пользователя.
- Сформировать условия для возможности проверить целесообразность дальнейшей полномасштабной разработки.
- Сформировать условия для возможности определить пользовательское виденье работы с системой мониторинга и реагирования.

5.1. Описание инструментов реализации прототипа системы

В качестве основных инструментом реализации системы мониторинга и реагирования выбраны следующие:

- Python 3 – основной инструмент реализации функций мониторинга и реагирования.
- Flask 2 – фреймворк для обеспечения обмена запросами в архитектуре REST.
- Angular 12 – фронтенд фреймворк выступающий в качестве интерфейса управления системой мониторинга и реагирования.
- Tarantool 2.7 – высоко производительная система управления базой данных с неблокирующим сервером приложений.
- Kafka 2.8 – распределенный программный брокер. Представляет из себя распределённую, горизонтально масштабируемую систему, обеспечивающую наращивание пропускной способности, как при росте числа и нагрузки со стороны источников, так и количества систем-подписчиков. Подписчики могут быть объединены в группы. Поддерживается возможность временного хранения данных для последующей пакетной обработки.

На рисунке 16 отображена организация компонентов прототипа системы мониторинга и реагирования.

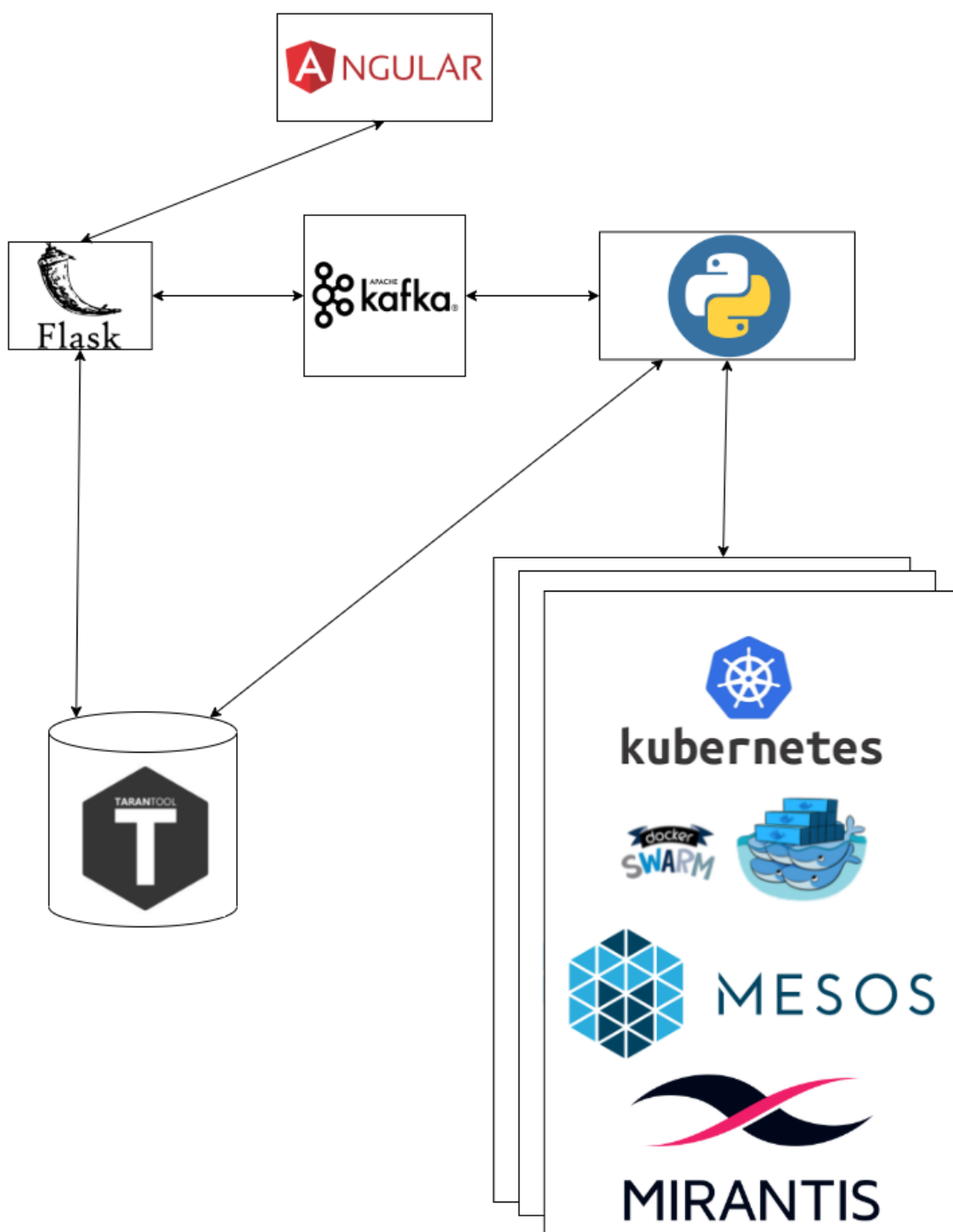


Рисунок 16 - Организация компонентов прототипа системы мониторинга и реагирования

5.2. Конфигурация прототипа

5.2.1. Система управления базой данных Taratool

Структура СУБД tarantool для прототипа системы мониторинга и реагирования состоит из 9 сущностей представленных на рисунке 17. Подробное описание конфигурации tarantool представлено в приложении 5.

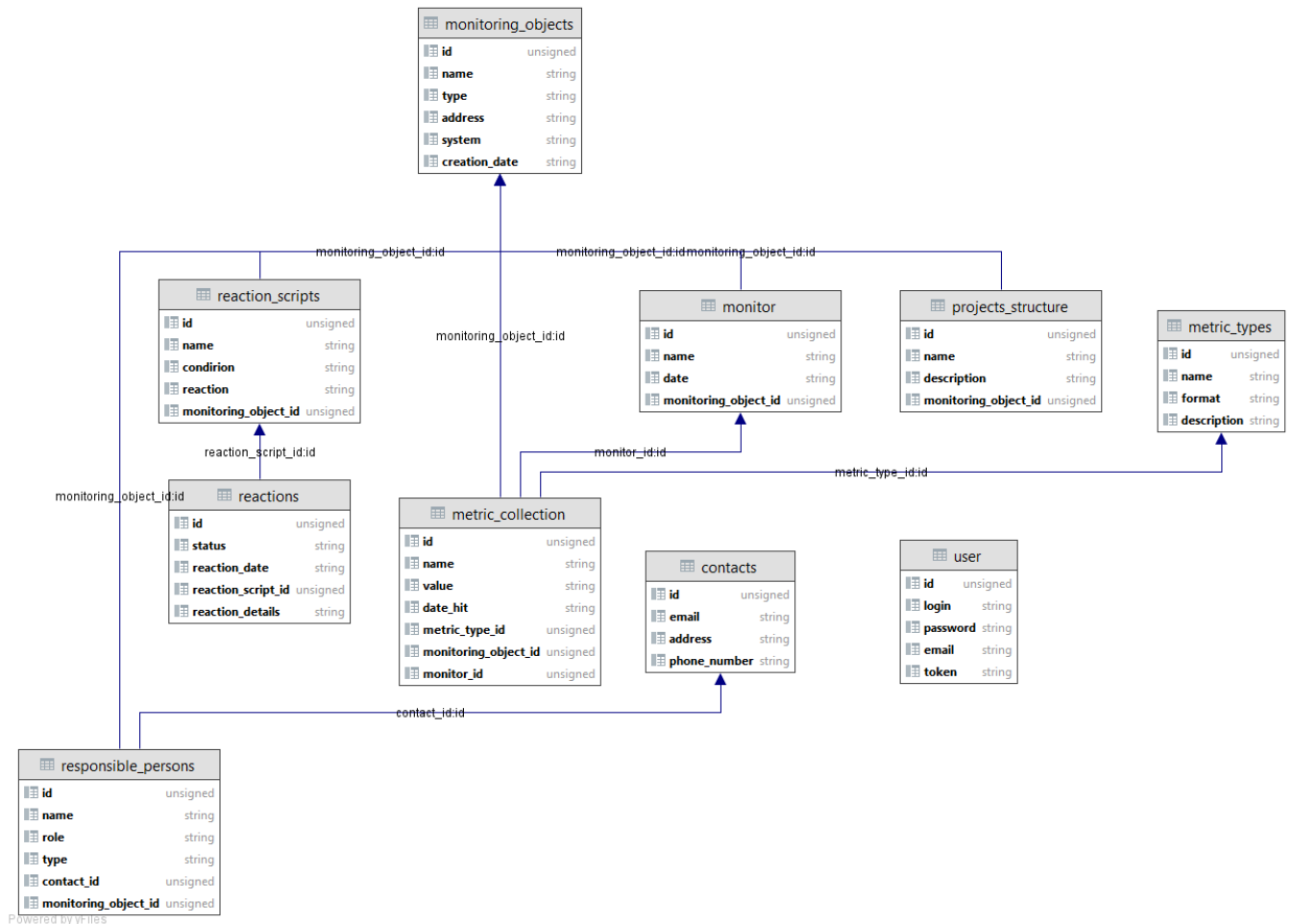


Рисунок 17 – Сущности tarantool

5.2.2. Flask инструмент организации API

В данном прототипе Flask представляет из себя API прослойку между Tarantool и Angular. В нем содержатся следующие методы, обеспечивающие веб интерфейсу доступ к данным Tarantool, полный код содержится в приложении 1:

- `add_user` – метод flask осуществляющий добавление нового пользователя к веб интерфейсу АСМР;
- `login` – метод flask осуществляющий проверку наличия доступа у пользователя к интерфейсу АСМР;
- `add_contact` – метод flask осуществляющий добавление новых контактов для пользователя;
- `get_contact` – метод, осуществляющий получение контактов определенного пользователя;
- `add_rperson` – метод, осуществляющий добавления нового ответственного лица с привязкой его к существующему объекту мониторинга;
- `get_rperson` - метод осуществляющий получение ответственного лица определенного объекта мониторинга;
- `add_m_types` – метод, осуществляющий добавление нового типа метрики;
- `get_m_types` – метод, позволяющий получить имеющиеся типы метрик.
- `add_m_collection` – метод осуществляющий добавление новой коллекции метрик;
- `get_m_collection` – метод осуществляющий получение коллекции метрик определенного монитора;
- `add_monitor` – метод, осуществляющий добавление нового монитора;
- `get_monitor` – метод позволяющий получить доступные мониторы объекта мониторинга;
- `add_monitoring_o` – метод осуществляющий добавление нового объекта мониторинга;
- `get_monitoring_o` – метод осуществляющий получение доступных объектов мониторинга;
- `add_r_script` – метод осуществляющий добавление нового сценария реагирования;

- `get_r_script` – метод позволяющий получить доступные сценарии реагирования;
- `add_reaction` – метод позволяющий добавить новую реакцию;
- `get_reaction` – метод позволяющий получить доступные реакции сценария реагирования;
- `add_project` – метод позволяющий добавить новый проект;
- `get_project` – метод позволяющий получить доступные проекты;

5.2.3. Инструмент организации веб-интерфейса Angular

Конфигурация веб интерфейса реализованного на Angular создана с использованием набора элементов Nebular и состоит из следующих модулей и компонентов изображенных на рисунке 18.

Модули:

- `NbSidebarModule` – Модуль организации боковых панелей интерфейса.
- `NbButtonModule` – Модуль организации кнопок.
- `NbListModule` – Модуль организации списков.
- `NbCardModule` – Модуль организации графических карточек.
- `NbUserModule` – Модуль организации профиля пользователя.
- `NbThemeService` – Модуль организации тем.
- `NbInputModule` – Модуль организации полей ввода.
- `NbActionsModule` – Модуль организации событий.
- `NbCheckboxModule` – Модуль организации особых вводных полей.
- `NbDatepickerModule` – Модуль организации интерфейса вывода дат.
- `NbIconModule` – Модуль поддержки иконок.
- `NbRadioModule` – Модуль организации особых полей ввода.
- `NbSelectModule` – Модуль организации вводных групп.
- `NbContextMenuModule` Модуль организации контекстного меню.

- NbMenuModule – Модуль организации меню.
- NbAccordionModule – модуль организации раскрывающихся списков.
- NbTreeGridModule – Модуль организации интерфейсных деревьев.
- NbAlertModule – Модуль организации оповещений.

Компоненты:

- AppComponent – главный компонент интерфейса.
- LoginComponent – компонент входа в систему.
- MainComponent – компонент главного экрана.
- DashboardComponent – компонент отображения метрик.
- ComponentsComponent – компонент состав полей в интерфейсе.
- TestApiComponent – компонент тестирования API.
- ClusterDetailsComponent – Компонент просмотра объектов мониторинга и их подробностей.

Подробно код основных Angular компонентов веб интерфейса системы мониторинга и реагирования описан в приложении 2.

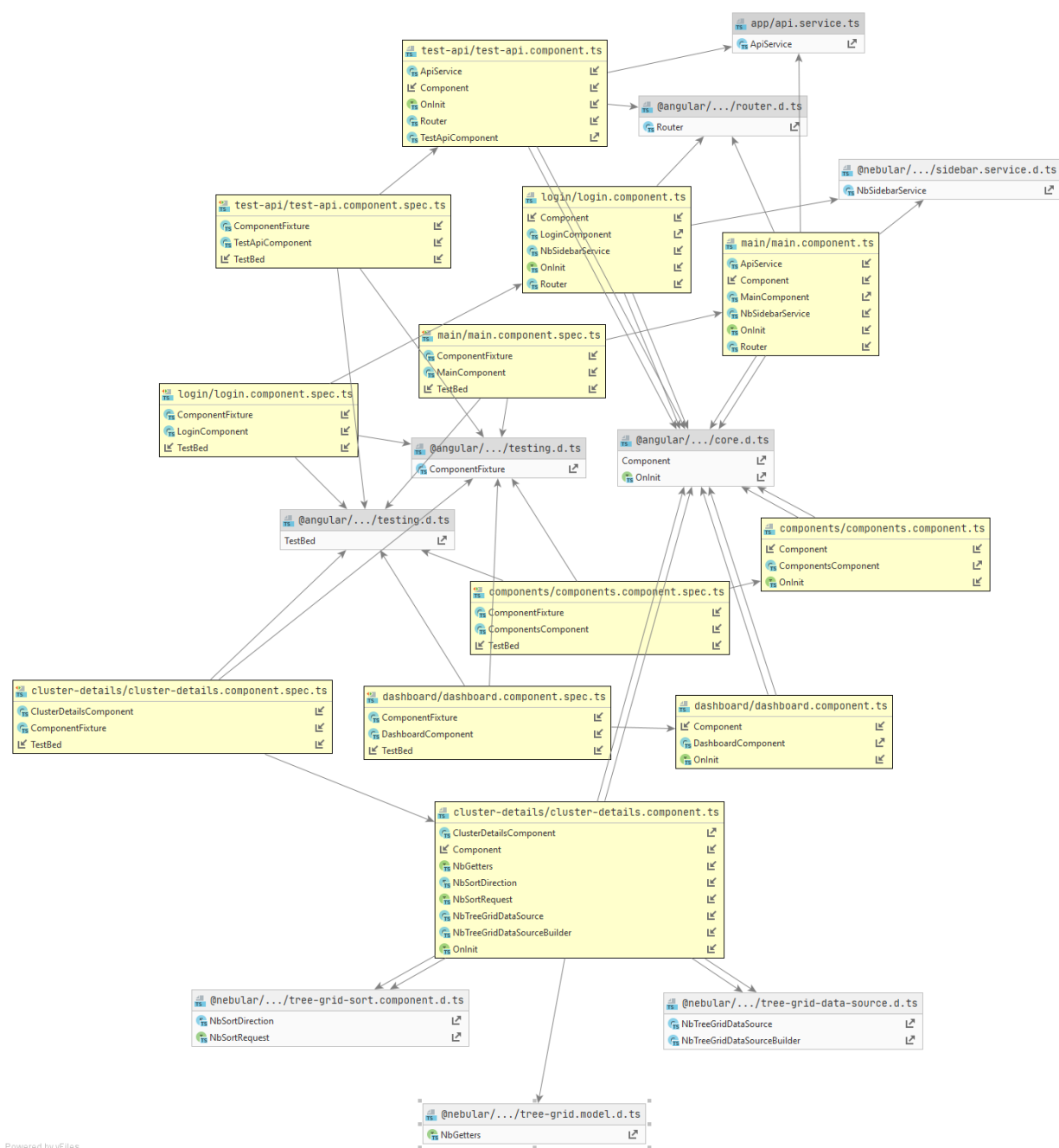


Рисунок 18 – конфигурация компонентов Angular

5.2.4. Kafka – программный брокер

Конфигурация системы очередей Kafka для прототипа системы мониторинга и реагирования описана при помощи docker compose файла, описанного в приложении 3.

5.3. Интеграционное описание

В качестве процесса интеграционного описания системы мониторинга и реагирования в среду кластера можно выделить следующее:

Получение метрик исследуемой системы определяется следующим образом. При использовании стандартного функционала Docker или с установкой сервиса Cadvisor, запрос на порт 9323 и подкаталог “/metrics” создает сетевой маршрут к файлу метрик в формате Prometheus. Именно такой формат будет включать в себя максимальный набор метрик кластера. Пример такого файла описан в приложении 4.

Использование Python 3 библиотеки docker-py позволяет установить соединение с кластером или отдельным контейнером системы, и выполнить необходимую команду, соответствующую сценарию реагирования.

5.4. Демонстрация работы системы

На рисунке 19 представлен интерфейс входа в систему мониторинга и реагирования.

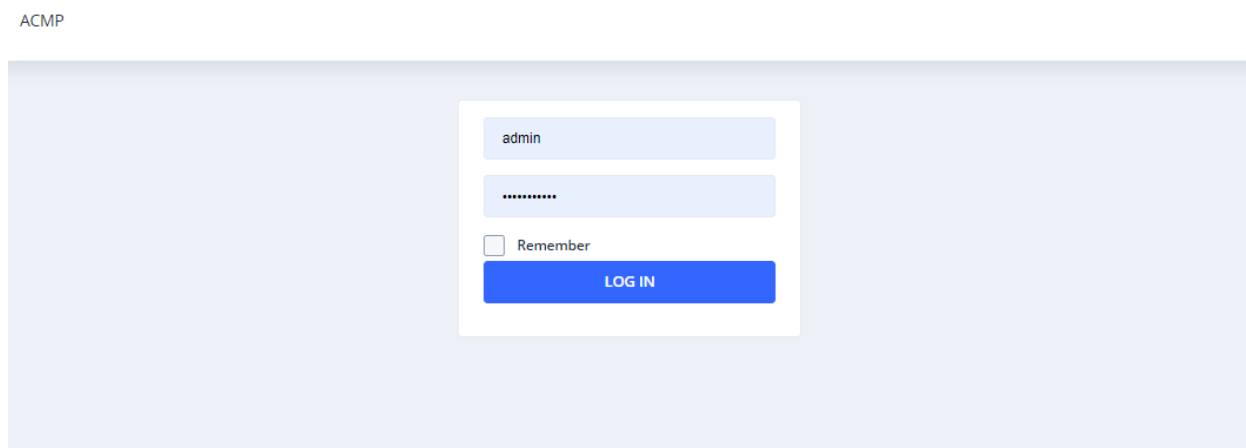


Рисунок 19 – Интерфейс входа в АСМР

После успешной авторизации в системе АСМР пользователя перенаправляет на главную страницу веб интерфейса системы мониторинга и реагирования, изображенной на рисунке 20. На которой пользователю демонстрируются доступные проекты, кластеры, а также графики с изображенными на них исследуемыми показателями.

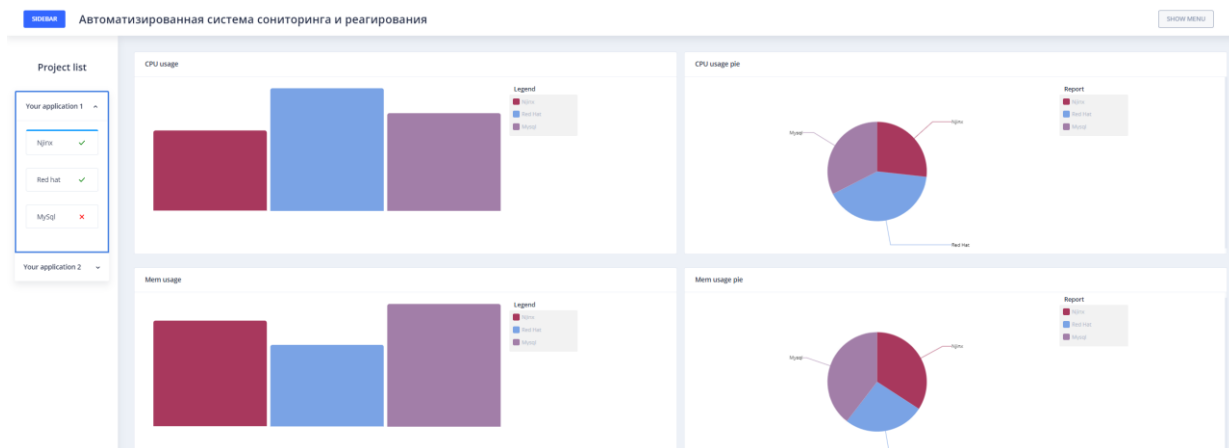


Рисунок 20 – Главная страница АСМР

При переходе на страницу реакций, изображенной на рисунке 21, пользователю отображается таблица с существующими сценариями реакции к его приложениям, с их описанием. А также список уже выполненных реакций с их конкретным статусом.

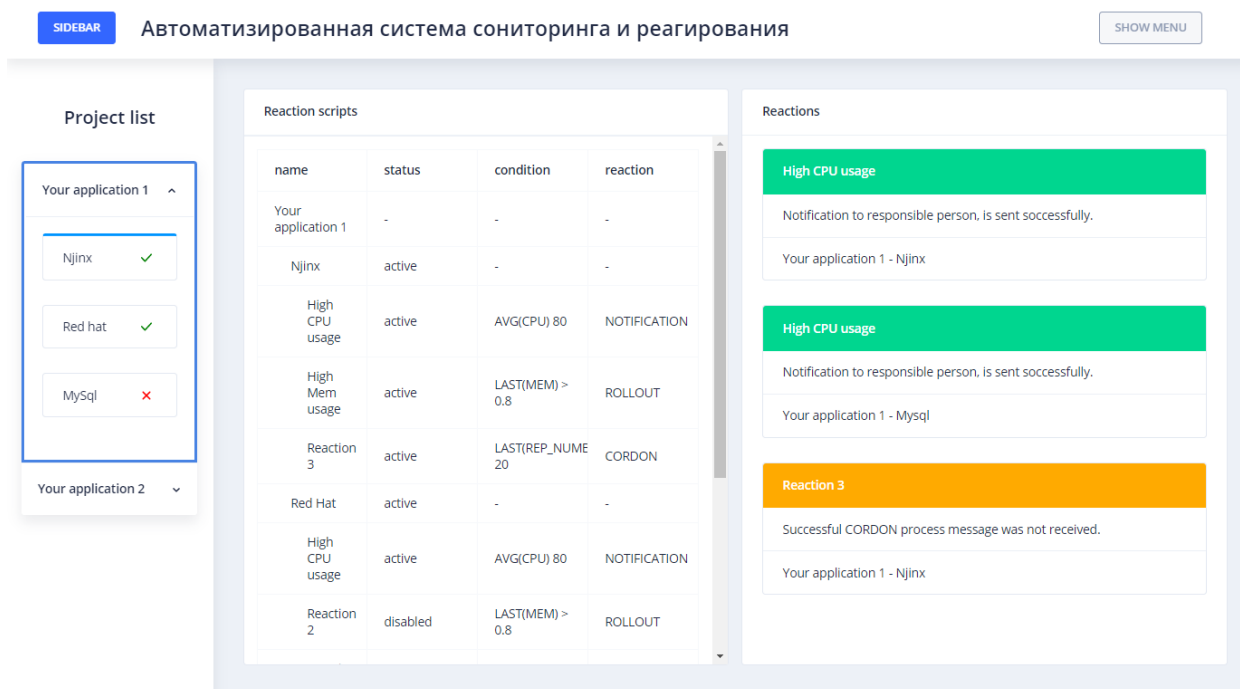


Рисунок 21 – Страница просмотра сценариев реагирования АСМР

5.5. Вывод к главе

В данной главе были описаны основные компоненты прототипа системы мониторинга и реагирования, вместе с процессом интеграции прототипа АСМР в среду кластеров исследуемой системы. А также приведены необходимые схемы и диаграммы компонентов.

Основным результатом главы является демонстрация работы части системы мониторинга и реагирования, а именно: взаимодействие веб-интерфейса Angular АСМР с системой хранения данных Tarantool посредством Flask и Kafka.

Из задачи прототипа следует, что он лишь демонстрирует работу части автоматизированной системы мониторинга и реагирования. Для того, чтобы в дальнейшем перейти от прототипа к полномасштабной разработке следует выполнить следующее:

- Реализовать полноценный транслятор задания сценариев реагирования.
- Реализовать полноценную логику сбора метрик с множества объектов мониторинга, описанную в главе 3.3.3.
- Реализовать полноценную логику внесения изменений в конфигурацию объекта мониторинга, описанную в главе 3.3.8.

Именно вышеописанный список изменений отличает прототип от готовой автоматизированной системы мониторинга и реагирования.

6. Оценка полученных результатов

При реализации прототипа, с учетом ограниченных временных ресурсов, удалось выполнить следующие задачи:

- В среде Angular реализован веб-интерфейс с поддержкой графической визуализации данных, способный успешно обмениваться данными с СУБД Tarantool и системой очередей Kafka посредством API интерфейса.
- Была развернута СУБД tarantool, создано 10 сущностей и 13 индексов.
- Был реализован API интерфейс, состоящий из 21 Flask метода, осуществляющий подключение к tarantool и Kafka.
- Был развернут программный брокер Kafka для обмена сообщениями между компонентами системы мониторинга и реагирования.

В целом можно сказать, что условия, при которых возможен дальнейший анализ использования и формирования пользовательского опыта с использованием систем мониторинга и реагирования, проект которых описан в разделе 3 - сформированы. При использовании текущего прототипа можно вывести следующее:

- Наличие возможности прозрачно задавать сценарии реагирования, упрощает процесс увеличения надежности системы, и значительно повышает предсказуемость её поведения.
- Развитие данной системы мониторинга и реагирования, вероятно может перерасти в конкурентноспособный продукт, помогающий пользователям решать комплексные задачи, связанные с мониторингом и реагированием.

Заключение

Исходя из всего выше описано можно сделать вывод, что в работе большинства современных систем требующих оркестрации необходим инструмент мониторинга и реагирования, так как на него возложены важные функции обеспечения бесперебойной работы системы.

В ходе данной работы были проанализированы существующие системы мониторинга и реагирования, а также проведено сравнение их показателей. А также сформировано техническое задание для автоматизированной системы мониторинга и реагирования. Далее был полностью сформирован и описан проект системы мониторинга и реагирования, включая ее описание с помощью средств UML.

После чего был разработан механизм реагирования на нештатные ситуации, включающий в себя описание синтаксиса конфигурационных файлов, а также концепцию применения сценариев реагирования к объектам мониторинга.

В качестве финальной задачи данной работы, с помощью актуальных инструментов разработки был реализован прототип системы мониторинга и реагирования. Основной задачей которого являлось - продемонстрировать работу части функционала АСМР, а также создание условий для проверки целесообразности использования такой системы мониторинга и реагирования.

Список используемых источников

1. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения: ГОСТ 34.003-90 – Издание официальное. -М. : ИПК Издательство стандартов, 2020. - 24 с
2. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем : ГОСТ 34.201-89 – Издание официальное. -М. : ИПК Издательство стандартов, 2020. - 24 с
3. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания : ГОСТ 34.601-90 – Издание официальное. -М. : ИПК Издательство стандартов, 2020. - 24 с
4. Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования : ГОСТ Р 50739-95 : ИПК Издательство стандартов, 2020. - 24 с
5. Доктрина информационной безопасности Российской Федерации (утв.5 декабря 2016 года Президентом Российской Федерации В.В. Путиным).
6. Сайман, Г. Mastering Kubernetes: Master the art of container management by using the power of Kubernetes, 2nd Edition / Г. Сайман.
7. OpenStack Kubernetes on OpenStack at eBay [Конференция] // Kubernetes on OpenStack at eBay. - Seattle : [б.н.], 2015.
8. И. С. Лебедев, В. В. Семенов, М. Е. Сухопаров «Идентификация состояния отдельных элементов киберфизических систем на основе внешних поведенческих характеристик» 2018.
9. Lukša, M. Kubernetes in Action / M. Lukša. — 2017 : Manning Publications, 2017.

10. Turnbull, J. The Docker Book: Containerization is the new virtualization Kindle Edition / J. Turnbull. — 2014 : Turnbull Press, 2014. — 344 c.
11. Poulton, N. The Kubernetes Book: The fastest way to get your head around Kubernetes / N. Poulton. — 2019 : O'Reilly Media, Inc., 2019. — 228 c. — ISBN 9781491935675.
12. Keith Adams, Ole Agesen A Comparison of Software and Hardware Techniques for x86 Virtualization;
13. VMWare. Virtualize Your IT Infrastructure. 2011;
14. Nathan Liefing and Brian van Baekel. Zabbix 5 IT Infrastructure Monitoring Cookbook. 2021;
15. Joel Bastos and Pedro Araujo. Hands-On Infrastructure Monitoring with Prometheus/ 2019;
16. Navin Sabharwal and Piyush Pandey. Monitoring Microservices and Containerized Applications: Deployment, Configuration, and Best Practices for Prometheus and Alert Manager. 2020;
17. Brian Brazil. Prometheus: Up & Running. 2018";
18. Jason Dixon. Monitoring with Graphite. 2017;
19. James Turnbull. The Art of Monitoring. 2016;
20. Rob Ewaschuk and Betsy Beyer. Monitoring Distributed Systems. 2016;
21. Patrik Uytterhoeven and Rihards Olups. Zabbix 4 Network Monitoring - Third Edition. 2019;
22. Rihards Olups, Andrea Vacche and Patrik Uytterhoeven. Zabbix: Enterprise Network Monitoring Made Easy. 2017;
23. Rihards Olups. Zabbix Network Monitoring - Second Edition
24. N. Rin. Virtual Machines Detection Enhanced. 2013. // URL: <https://artemonsecurity.com/vmde.pdf> (дата обращения 10.04.2021).
25. Virt-what - detect if we are running in a virtual machine // redhat.com. URL: <https://people.redhat.com/~rjones/virt-what/virt-what.txt> (дата обращения 11.04.2021).

Приложение 1 – Реализация API интерфейса прототипа АСМР

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# vim:fileencoding=utf-8
from flask import Flask
from flask import Flask, jsonify
from flask import Flask, request # import main Flask class and request object
# from flask_restplus import Api, Resource, fields
import tarantool
import time
import hashlib, uuid
from datetime import date, datetime
from flask_cors import CORS
from flask import Response

app = Flask(__name__)
CORS(app)
app.config['JSON_AS_ASCII'] = False
# apps = Api(app = app)
# name_space = apps.namespace('main', description='Main APIs')
server = tarantool.connect("localhost", 3301)
users = server.space('user')
contacts = server.space('contacts')
met_collection = server.space('metric_collection')
met_types = server.space('metric_types')
monitor = server.space('monitor')
m_objects = server.space('monitoring_objects')
r_scripts = server.space('reaction_scripts')
reactions = server.space('reactions')
projects = server.space('projects_structure')
r_persons = server.space('responsible_persons')
main_token = 'test_token'
codes = ['10001', '10002', '10003', '10004', '10005', '10006']

def isuser_noexist(username):
    print(str(users.select(username, index='secondary'))))

    try:
        if str(users.select(username, index='secondary')) == "":
            return True
        else:
            return False
    except:
        return 'DB_ERROR'

def has_str(line):
    return hashlib.sha512(line.encode('utf-8')).hexdigest()

@app.route('/')
def main():
    return 'ACMP'

@app.route('/user/add', methods=['GET'])
def add_user():
    username = request.args.get('username')
    password = request.args.get('password')
```

```

email = request.args.get('email')
if isuser_noexist(username) == True:
    users.insert((hash(time.time()), username, has_str(password), email, str(hash(username))))
    response = 'Пользователь добавлен'
    print(response)
    return jsonify({'response': response, 'token': main_token}), 200
else:
    response = 'Такой логин есть в системе'
    print(response)
    return jsonify({'response': response}), 200

@app.route('/login', methods=['GET'])
def login():
    username = request.args.get('username')
    password = request.args.get('password')
    if isuser_noexist(username) == True:
        response = 'No user'
        print(response)
        return jsonify({'response': response}), 200
    else:
        if str(users.select(username, index='secondary')[0][2]) == has_str(password):
            response = 'Success login'

            else:
                response = 'Access denied'
                print(response)
        return jsonify(response), 200

@app.route('/contact/add', methods=['get'])
def add_contact():
    email = request.args.get('email')
    address = request.args.get('address')
    phone_number = request.args.get('phone_number')
    token = request.args.get('token')

    if token == main_token:
        contacts.insert((hash(time.time()), email, address, phone_number))
        response = 'Contact add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/contact/get', methods=['get'])
def get_contact():
    token = request.args.get('token')
    if token == main_token:
        response = list(contacts.select())
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/r_person/add', methods=['get'])
def add_rperson():
    name = request.args.get('name')

```

```

role = request.args.get('role')
type = request.args.get('type')
monitoring_object_id = request.args.get('monitoring_object_id')
contact_id = request.args.get('contact_id')
token = request.args.get('token')

if token == main_token:
    print(hash(time.time()), name, role, type, contact_id, token)
    r_persons.insert((hash(time.time()), name, role, type, int(contact_id), int(monitoring_object_id)))
    response = 'Person add'
    print(response)
    return jsonify({'response': response}), 200
else:
    response = 'Access denied'
    return jsonify(response), 200

@app.route('/r_person/get', methods=['get'])
def get_rperson():
    token = request.args.get('token')

    if token == main_token:
        response = list(r_persons.select())
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/m_types/add', methods=['get'])
def add_m_types():
    name = request.args.get('name')
    format = request.args.get('format')
    description = request.args.get('description')
    token = request.args.get('token')

    if token == main_token:
        print(hash(time.time()), name, format, description)
        met_types.insert((hash(time.time()), name, format, description))
        response = 'Metric type add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/m_types/get', methods=['get'])
def get_m_types():
    token = request.args.get('token')

    if token == main_token:
        response = list(met_types.select())
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/m_collection/add', methods=['get'])

```

```

def add_m_collection():
    name = request.args.get('name')
    value = request.args.get('value')
    metric_type_id = request.args.get('metric_type_id')
    monitoring_object_id = request.args.get('monitoring_object_id')
    monitor_id = request.args.get('monitor_id')
    token = request.args.get('token')

    if token == main_token:
        met_collection.insert(
            (hash(time.time()), name, value, str(datetime.now()), int(metric_type_id), int(monitoring_object_id),
             int(monitor_id)))
        response = 'Metric collection add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/m_collection/get', methods=['get'])
def get_m_collection():
    monitor_id = request.args.get('monitor_id')
    token = request.args.get('token')

    if token == main_token:
        response = list(met_collection.select(int(monitor_id), index='secondary'))
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/monitor/add', methods=['get'])
def add_monitor():
    name = request.args.get('name')
    monitoring_object_id = request.args.get('monitoring_object_id')
    token = request.args.get('token')

    if token == main_token:
        monitor.insert((hash(time.time()), name, str(datetime.now()), int(monitoring_object_id)))
        response = 'Monitor add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/monitor/get', methods=['get'])
def get_monitor():
    monitoring_object_id = request.args.get('monitoring_object_id')
    token = request.args.get('token')

    if token == main_token:
        response = list(monitor.select(int(monitoring_object_id), index='secondary'))
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

```

```

@app.route('/monitoring_o/add', methods=['get'])
def add_monitoring_o():
    name = request.args.get('name')
    type = request.args.get('type')
    address = request.args.get('address')
    system = request.args.get('system')
    status = request.args.get('status')
    token = request.args.get('token')

    if token == main_token:
        m_objects.insert((hash(time.time()), name, type, address, system, str(date.today()),status))
        response = 'Monitoring object add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/monitoring_o/get', methods=['get'])
def get_monitoring_o():
    token = request.args.get('token')

    if token == main_token:
        response = list(m_objects.select())
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/r_script/add', methods=['get'])
def add_r_script():
    name = request.args.get('name')
    condition = request.args.get('condition')
    reaction = request.args.get('reaction')
    monitoring_object_id = request.args.get('monitoring_object_id')
    token = request.args.get('token')

    if token == main_token:
        r_scripts.insert((hash(time.time()), name, condition, reaction, int(monitoring_object_id)))
        response = 'Reaction script add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/r_script/get', methods=['get'])
def get_r_script():
    monitoring_object_id = request.args.get('monitoring_object_id')
    token = request.args.get('token')

    if token == main_token:
        response = list(r_scripts.select(int(monitoring_object_id), index='secondary'))
        print(response)
        return jsonify(response), 200
    else:

```



```

    response = 'Access denied'
    return jsonify(response), 200

@app.route('/reaction/add', methods=['get'])
def add_reaction():
    status = request.args.get('status')
    reaction_details = request.args.get('reaction_details')
    reaction_script_id = request.args.get('reaction_script_id')
    token = request.args.get('token')

    if token == main_token:
        r_scripts.insert(
            (hash(time.time()), status, str(datetime.now()), str(reaction_details), int(reaction_script_id)))
        response = 'Monitoring object add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/reaction/get', methods=['get'])
def get_reaction():
    reaction_script_id = request.args.get('reaction_script_id')
    token = request.args.get('token')

    if token == main_token:
        response = list(r_scripts.select(int(reaction_script_id), index='secondary'))
        print(response)
        return jsonify(response), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/project/add', methods=['get'])
def add_project():
    name = request.args.get('name')
    description = request.args.get('description')
    monitoring_object_id = request.args.get('monitoring_object_id')
    token = request.args.get('token')

    if token == main_token:
        projects.insert((hash(time.time()), name, description, int(monitoring_object_id) ))

        response = 'Project add'
        print(response)
        return jsonify({'response': response}), 200
    else:
        response = 'Access denied'
        return jsonify(response), 200

@app.route('/project/get', methods=['get'])
def get_project():
    token = request.args.get('token')

    if token == main_token:
        response = list(projects.select())
        print(response)
        return jsonify(response), 200

```

```
    else:
        response = 'Access denied'
    return jsonify(response), 200

if __name__ == '__main__':
    app.run()
```

Приложение 2 – Реализация веб-интерфейса прототипа АСМР

App.module.ts:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { NbThemeModule, NbLayoutModule } from '@nebular/theme';
import { NbEvaIconsModule } from '@nebular/eva-icons';
import { AppRoutingModuleModule } from './app-routing.module';
import {
  NbSidebarModule,
  NbButtonModule,
  NbSidebarService,
  NbListModule,
  NbCardModule,
  NbUserModule,
  NbThemeService,
  NbInputModule,
  NbActionsModule,
  NbCheckboxModule,
  NbDatepickerModule, NbIconModule,
  NbRadioModule,
  NbSelectModule,
  NbContextMenuModule,
  NbMenuModule,
  NbAccordionModule,
  NbTreeGridModule,
  NbAlertModule,
} from '@nebular/theme';

import { HttpClientModule } from '@angular/common/http';
import { FormsModule as ngFormsModule } from '@angular/forms';
import { MainComponent } from './main/main.component';
import { DashboardComponent } from './dashboard/dashboard.component';

import { NgxChartsModule } from '@swimlane/ngx-charts';
import { ComponentsComponent } from './components/components.component';
import { TestApiComponent } from './test-api/test-api.component';
import { ClusterDetailsComponent } from './cluster-details/cluster-details.component';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    MainComponent,
    DashboardComponent,
    ComponentsComponent,
    TestApiComponent,
    ClusterDetailsComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    NbThemeModule.forRoot({ name: 'default' }),
    NbLayoutModule,
    NbEvaIconsModule,
```

```

    AppRoutingModule,
    NbLayoutModule,
    NbSidebarModule.forRoot(), // NbSidebarModule.forRoot(), //if this is your app.module
    NbButtonModule,
    NbListModule,
    NbCardModule,
    NbUserModule,
    ngFormsModule,
    NbInputModule,
    NbActionsModule,
    NbCheckboxModule,
    NbRadioModule,
    NbDatepickerModule,
    NbSelectModule,
    NbIconModule,
    NbContextMenuModule,
    NbMenuModule.forRoot(),
    NbAccordionModule,
    NgxChartsModule,
    NbTreeGridModule,
    HttpClientModule,
    NbAlertModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

Api.service.ts:

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError, retry } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class ApiService {

  constructor(private httpClient: HttpClient) {
  }
  url='http://127.0.0.1:5000';
  token = 'test_token';
  public adduser(username: string, password: string, email: string) {

    return this.httpClient.get(this.url + '/user/add?username=' + username + '&password=' + password + '&email=' + email);
  }

  public login(username: string, password: string) {

    return this.httpClient.get(this.url + '/login?username=' + username + '&password=' + password);
  }

  public add_contact(email: string, address: string, phone_number: string) {

    return this.httpClient.get(this.url + '/contact/add?email=' + email + '&address=' + address + '&phone_number=' + phone_number + '&token=' + this.token);
  }

```

```

    }

    public get_contact() {

        return this.httpClient.get(this.url + '/contact/get?token=' + this.token);
    }
    public add_rperson(name: string, role: string, type: string, contactid: number, monitoring_object_id: number) {

        return this.httpClient.get(this.url +
'/r_person/add?name='+name+'&role='+role+'&type='+type+'&contact_id='+contactid+'&monitoring_object_id='+
monitoring_object_id+'&token=' + this.token);
    }

    public get_rperson() {

        return this.httpClient.get(this.url + '/r_person/get?token=' + this.token);
    }

    public add_m_types(name: string, format: string, description: string) {

        return this.httpClient.get(this.url +
'/m_types/add?name='+name+'&format='+format+'&description='+description+'&token=' + this.token);
    }

    public get_m_types() {

        return this.httpClient.get(this.url + '/m_types/get?token=' + this.token);
    }

    public add_m_collection(name: string, value: string, metric_type_id: number, monitoring_object_id: number,
monitor_id: number) {

        return this.httpClient.get(this.url + '/m_collection/add?name=' + name + '&value=' + value + '&metric_type_id='
+
        metric_type_id + '&monitoring_object_id=' + monitoring_object_id + '&monitor_id=' + monitor_id + '&token='
+ this.token);
    }

    public get_m_collection(monitor_id: number) {

        return this.httpClient.get(this.url + '/m_collection/get?monitor_id=' + monitor_id + '&token=' + this.token);
    }

    public add_monitor( name: string, monitoring_object_id: number) {

        return this.httpClient.get(this.url + '/monitor/add?name=' + name + '&monitoring_object_id='
+ monitoring_object_id + '&token=' + this.token);
    }

    public get_monitor( monitoring_object_id: number) {

        return this.httpClient.get(this.url + '/monitor/get?monitoring_object_id=' + monitoring_object_id + '&token=' +
this.token);
    }

    public add_monitoring_o( name: string, type: string, address: string, system: string, status: string) {

        return this.httpClient.get(this.url + '/monitoring_o/add?name=' + name + '&type=' + type + '&address=' + address
+ '&system=' + system + '&status=' + status + '&token=' + this.token);
    }
}

```

```

public get_monitoring_o() {

    return this.httpClient.get(this.url + '/monitoring_o/get?token=' + this.token);

}

public add_r_script( name: string, condition: string, reaction: string, monitoring_object_id: number) {

    return this.httpClient.get(this.url + '/r_script/add?name=' + name + '&condition=' + condition + '&reaction=' +
    reaction + '&monitoring_object_id=' + monitoring_object_id + '&token=' + this.token);

}

public get_r_script(monitoring_object_id: number) {

    return this.httpClient.get(this.url + '/r_script/get?monitoring_object_id='+monitoring_object_id+'&token=' +
    this.token);

}

public add_reaction( status: string, reaction_details: string, reaction_script_id: number) {

    return this.httpClient.get(this.url + '/reaction/add?status=' + status + '&reaction_details=' + reaction_details +
    '&reaction_script_id=' + reaction_script_id + '&token=' + this.token);

}

public get_reaction(reaction_script_id: number) {

    return this.httpClient.get(this.url + '/reaction/get?reaction_script_id='+reaction_script_id+'&token=' + this.token);

}

public add_project(name: string,description: string,monitoring_object_id:number) {
    return this.httpClient.get(this.url +
'/project/add?name='+name+'&description='+description+'&monitoring_object_id='+monitoring_object_id+'&token
=' + this.token);
}
public get_project() {
    return this.httpClient.get(this.url + '/project/get?token=' + this.token);
}
}

```

Приложение 3 – Конфигурация KAFKA для прототипа АСМР

```
---
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:6.1.1
    hostname: zookeeper
    container_name: zookeeper
    ports:
      - "2181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
  broker:
    image: confluentinc/cp-server:6.1.1
    hostname: broker
    container_name: broker
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
      - "9101:9101"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS:
PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_METRIC_REPORTERS:
io.confluent.metrics.reporter.ConfluentMetricsReporter
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_CONFLUENT_LICENSE_TOPIC_REPLICATION_FACTOR: 1
```

KAFKA_CONFLUENT_BALANCER_TOPIC_REPLICATION_FACTOR: 1
KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
KAFKA_JMX_PORT: 9101
KAFKA_JMX_HOSTNAME: localhost
KAFKA_CONFLUENT_SCHEMA_REGISTRY_URL: http://schema-registry:8081
CONFLUENT_METRICS_REPORTER_BOOTSTRAP_SERVERS: broker:29092
CONFLUENT_METRICS_REPORTER_TOPIC_REPLICAS: 1
CONFLUENT_METRICS_ENABLE: 'true'
CONFLUENT_SUPPORT_CUSTOMER_ID: 'anonymous'

schema-registry:

image: confluentinc/cp-schema-registry:6.1.1
hostname: schema-registry
container_name: schema-registry
depends_on:
- broker
ports:
- "8081:8081"
environment:
SCHEMA_REGISTRY_HOST_NAME: schema-registry
SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS: 'broker:29092'
SCHEMA_REGISTRY_LISTENERS: http://0.0.0.0:8081

connect:

image: cnfldemos/cp-server-connect-datagen:0.4.0-6.1.0
hostname: connect
container_name: connect
depends_on:
- broker
- schema-registry
ports:
- "8083:8083"
environment:
CONNECT_BOOTSTRAP_SERVERS: 'broker:29092'
CONNECT_REST_ADVERTISED_HOST_NAME: connect
CONNECT_REST_PORT: 8083


```

CONNECT_GROUP_ID: compose-connect-group
CONNECT_CONFIG_STORAGE_TOPIC: docker-connect-configs
CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: 1
CONNECT_OFFSET_FLUSH_INTERVAL_MS: 10000
CONNECT_OFFSET_STORAGE_TOPIC: docker-connect-offsets
CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: 1
CONNECT_STATUS_STORAGE_TOPIC: docker-connect-status
CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: 1
CONNECT_KEY_CONVERTER: org.apache.kafka.connect.storage.StringConverter
CONNECT_VALUE_CONVERTER: io.confluent.connect.avro.AvroConverter
CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL: http://schema-
registry:8081

# CLASSPATH required due to CC-2422
CLASSPATH:      /usr/share/java/monitoring-interceptors/monitoring-interceptors-
6.1.1.jar

CONNECT_PRODUCER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringProducerInterceptor"
CONNECT_CONSUMER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor"
CONNECT_PLUGIN_PATH: "/usr/share/java,/usr/share/confluent-hub-components"
CONNECT_LOG4J_LOGGERS:
org.apache.zookeeper=ERROR,org.I0Itec.zkclient=ERROR,org.reflections=ERROR

control-center:
  image: confluentinc/cp-enterprise-control-center:6.1.1
  hostname: control-center
  container_name: control-center
  depends_on:
    - broker
    - schema-registry
    - connect
    - ksqldb-server
  ports:
    - "9021:9021"
  environment:
    CONTROL_CENTER_BOOTSTRAP_SERVERS: 'broker:29092'

```

```

CONTROL_CENTER_CONNECT_CLUSTER: 'connect:8083'
CONTROL_CENTER_KSQL_KSQLDB1_URL: "http://ksqldb-server:8088"
CONTROL_CENTER_KSQL_KSQLDB1_ADVERTISED_URL:
"http://localhost:8088"
CONTROL_CENTER_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
CONTROL_CENTER_REPLICATION_FACTOR: 1
CONTROL_CENTER_INTERNAL_TOPICS_PARTITIONS: 1
CONTROL_CENTER_MONITORING_INTERCEPTOR_TOPIC_PARTITIONS: 1
CONFLUENT_METRICS_TOPIC_REPLICATION: 1
PORT: 9021
ksqldb-server:
  image: confluentinc/cp-ksqldb-server:6.1.1
  hostname: ksqldb-server
  container_name: ksqldb-server
  depends_on:
    - broker
    - connect
  ports:
    - "8088:8088"
  environment:
    KSQL_CONFIG_DIR: "/etc/ksql"
    KSQL_BOOTSTRAP_SERVERS: "broker:29092"
    KSQL_HOST_NAME: ksqldb-server
    KSQL_LISTENERS: "http://0.0.0.0:8088"
    KSQL_CACHE_MAX_BYTES_BUFFERING: 0
    KSQL_KSQL_SCHEMA_REGISTRY_URL: "http://schema-registry:8081"
    KSQL_PRODUCER_INTERCEPTOR_CLASSES:
      "io.confluent.monitoring.clients.interceptor.MonitoringProducerInterceptor"
    KSQL_CONSUMER_INTERCEPTOR_CLASSES:
      "io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor"
    KSQL_KSQL_CONNECT_URL: "http://connect:8083"
    KSQL_KSQL_LOGGING_PROCESSING_TOPIC_REPLICATION_FACTOR: 1
    KSQL_KSQL_LOGGING_PROCESSING_TOPIC_AUTO_CREATE: 'true'
    KSQL_KSQL_LOGGING_PROCESSING_STREAM_AUTO_CREATE: 'true'

```

ksqldb-cli:

image: confluentinc/cp-ksqldb-cli:6.1.1

container_name: ksqldb-cli

depends_on:

- broker
- connect
- ksqldb-server

entrypoint: /bin/sh

tty: true

ksql-datagen:

image: confluentinc/ksqldb-examples:6.1.1

hostname: ksql-datagen

container_name: ksql-datagen

depends_on:

- ksqldb-server
- broker
- schema-registry
- connect

command: "bash -c 'echo Waiting for Kafka to be ready... && \\
cub kafka-ready -b broker:29092 1 40 && \\
echo Waiting for Confluent Schema Registry to be ready... && \\
cub sr-ready schema-registry 8081 40 && \\
echo Waiting a few seconds for topic creation to finish... && \\
sleep 11 && \\
tail -f /dev/null'"

environment:

KSQL_CONFIG_DIR: "/etc/ksql"

STREAMS_BOOTSTRAP_SERVERS: broker:29092

STREAMS_SCHEMA_REGISTRY_HOST: schema-registry

STREAMS_SCHEMA_REGISTRY_PORT: 8081

rest-proxy:

image: confluentinc/cp-kafka-rest:6.1.1

depends_on:

- broker
- schema-registry

ports:

- 8082:8082

hostname: rest-proxy

container_name: rest-proxy

environment:

KAFKA_REST_HOST_NAME: rest-proxy

KAFKA_REST_BOOTSTRAP_SERVERS: 'broker:29092'

KAFKA_REST_LISTENERS: "http://0.0.0.0:8082"

KAFKA_REST_SCHEMA_REGISTRY_URL: 'http://schema-registry:8081'

Приложение 4 – Пример файла метрик, образованного кластером

```
# HELP builder_builds_failed_total Number of failed image builds
# TYPE builder_builds_failed_total counter
builder_builds_failed_total{reason="build_canceled"} 0
builder_builds_failed_total{reason="build_target_not_reachable_error"} 0
builder_builds_failed_total{reason="command_not_supported_error"} 0
builder_builds_failed_total{reason="dockerfile_empty_error"} 0
builder_builds_failed_total{reason="dockerfile_syntax_error"} 0
builder_builds_failed_total{reason="error_processing_commands_error"} 0
builder_builds_failed_total{reason="missing_onbuild_arguments_error"} 0
builder_builds_failed_total{reason="unknown_instruction_error"} 0
# HELP builder_builds_triggered_total Number of triggered image builds
# TYPE builder_builds_triggered_total counter
builder_builds_triggered_total 0
# HELP engine_daemon_container_actions_seconds The number of seconds it takes to process each container
action
# TYPE engine_daemon_container_actions_seconds histogram
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="+Inf"} 1
engine_daemon_container_actions_seconds_sum{action="changes"} 0
engine_daemon_container_actions_seconds_count{action="changes"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="+Inf"} 1
engine_daemon_container_actions_seconds_sum{action="commit"} 0
engine_daemon_container_actions_seconds_count{action="commit"} 1
engine_daemon_container_actions_seconds_bucket{action="create",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="create",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="create",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="create",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="create",le="0.1"} 2
engine_daemon_container_actions_seconds_bucket{action="create",le="0.25"} 9
engine_daemon_container_actions_seconds_bucket{action="create",le="0.5"} 9
engine_daemon_container_actions_seconds_bucket{action="create",le="1"} 9
engine_daemon_container_actions_seconds_bucket{action="create",le="2.5"} 10
engine_daemon_container_actions_seconds_bucket{action="create",le="5"} 10
engine_daemon_container_actions_seconds_bucket{action="create",le="10"} 10
engine_daemon_container_actions_seconds_bucket{action="create",le="+Inf"} 11
engine_daemon_container_actions_seconds_sum{action="create"} 26.696194000000002
engine_daemon_container_actions_seconds_count{action="create"} 11
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.005"} 1
```

```

engine_daemon_container_actions_seconds_bucket{action="delete",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="delete",le="1"} 3
engine_daemon_container_actions_seconds_bucket{action="delete",le="2.5"} 3
engine_daemon_container_actions_seconds_bucket{action="delete",le="5"} 3
engine_daemon_container_actions_seconds_bucket{action="delete",le="10"} 4
engine_daemon_container_actions_seconds_bucket{action="delete",le="+Inf"} 9
engine_daemon_container_actions_seconds_sum{action="delete"} 74.6072289
engine_daemon_container_actions_seconds_count{action="delete"} 9
engine_daemon_container_actions_seconds_bucket{action="start",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="start",le="0.5"} 2
engine_daemon_container_actions_seconds_bucket{action="start",le="1"} 7
engine_daemon_container_actions_seconds_bucket{action="start",le="2.5"} 18
engine_daemon_container_actions_seconds_bucket{action="start",le="5"} 19
engine_daemon_container_actions_seconds_bucket{action="start",le="10"} 19
engine_daemon_container_actions_seconds_bucket{action="start",le="+Inf"} 19
engine_daemon_container_actions_seconds_sum{action="start"} 24.5670439
engine_daemon_container_actions_seconds_count{action="start"} 19
# HELP engine_daemon_container_states_containers The count of containers in various states
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 11
engine_daemon_container_states_containers{state="stopped"} 0
# HELP engine_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
# TYPE engine_daemon_engine_cpus_cpus gauge
engine_daemon_engine_cpus_cpus 4
# HELP engine_daemon_engine_info The information related to the engine and the OS it is running on
# TYPE engine_daemon_engine_info gauge
engine_daemon_engine_info{architecture="x86_64",commit="363e9a8",daemon_id="VXZX:PE23:KKP7:WA5P:C6K7:ZBOB:GSQF:EDJV:7S7I:TLLW:3VO3:DCMD",graphdriver="overlay2",kernel="5.4.72-microsoft-standard-WSL2",os="Docker Desktop",os_type="linux",os_version="",version="20.10.5"} 1
# HELP engine_daemon_engine_memory_bytes The number of bytes of memory that the host system of the engine has
# TYPE engine_daemon_engine_memory_bytes gauge
engine_daemon_engine_memory_bytes 1.335228416e+10
# HELP engine_daemon_events_subscribers_total The number of current subscribers to events
# TYPE engine_daemon_events_subscribers_total gauge
engine_daemon_events_subscribers_total 3
# HELP engine_daemon_events_total The number of events logged
# TYPE engine_daemon_events_total counter
engine_daemon_events_total 132487
# HELP engine_daemon_health_checks_failed_total The total number of failed health checks
# TYPE engine_daemon_health_checks_failed_total counter
engine_daemon_health_checks_failed_total 11
# HELP engine_daemon_health_checks_total The total number of health checks
# TYPE engine_daemon_health_checks_total counter
engine_daemon_health_checks_total 44116
# HELP engine_daemon_host_info_functions_seconds The number of seconds it takes to call functions gathering info about the host
# TYPE engine_daemon_host_info_functions_seconds histogram
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.005"} 5
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.01"} 5
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.025"} 5

```

```

engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.05"} 5
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.1"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.25"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="0.5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="1"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="2.5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="operating_system",le="10"} 6
engine_daemon_host_info_functions_seconds_sum{function="operating_system",le="+Inf"} 6
engine_daemon_host_info_functions_seconds_sum{function="operating_system"} 0.09396180000000001
engine_daemon_host_info_functions_seconds_count{function="operating_system"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.005"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.01"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.025"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.05"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.1"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.25"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="0.5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="1"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="2.5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="10"} 6
engine_daemon_host_info_functions_seconds_bucket{function="os_version",le="+Inf"} 6
engine_daemon_host_info_functions_seconds_sum{function="os_version"} 0.0001546
engine_daemon_host_info_functions_seconds_count{function="os_version"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.005"} 0
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.01"} 1
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.025"} 1
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.05"} 2
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.1"} 5
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.25"} 5
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="0.5"} 5
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="1"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="2.5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="5"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="10"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_info",le="+Inf"} 6
engine_daemon_host_info_functions_seconds_sum{function="system_info"} 0.9334072
engine_daemon_host_info_functions_seconds_count{function="system_info"} 6
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.005"} 0
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.01"} 3
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.025"} 3
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.05"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.1"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.25"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="0.5"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="1"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="2.5"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="5"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="10"} 4
engine_daemon_host_info_functions_seconds_bucket{function="system_version",le="+Inf"} 4
engine_daemon_host_info_functions_seconds_sum{function="system_version"} 0.051587299999999996
engine_daemon_host_info_functions_seconds_count{function="system_version"} 4
# HELP engine_daemon_image_actions_seconds The number of seconds it takes to process each image action
# TYPE engine_daemon_image_actions_seconds histogram
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.005"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.01"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.025"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.05"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.1"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.25"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="0.5"} 0

```

```

engine_daemon_image_actions_seconds_bucket{action="pull",le="1"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="2.5"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="5"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="10"} 0
engine_daemon_image_actions_seconds_bucket{action="pull",le="+Inf"} 10
engine_daemon_image_actions_seconds_sum{action="pull"} 1270.2042975999998
engine_daemon_image_actions_seconds_count{action="pull"} 10
# HELP engine_daemon_network_actions_seconds The number of seconds it takes to process each network action
# TYPE engine_daemon_network_actions_seconds histogram
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.005"} 0
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.01"} 0
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.025"} 0
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.05"} 0
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.1"} 3
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.25"} 7
engine_daemon_network_actions_seconds_bucket{action="allocate",le="0.5"} 11
engine_daemon_network_actions_seconds_bucket{action="allocate",le="1"} 15
engine_daemon_network_actions_seconds_bucket{action="allocate",le="2.5"} 18
engine_daemon_network_actions_seconds_bucket{action="allocate",le="5"} 18
engine_daemon_network_actions_seconds_bucket{action="allocate",le="10"} 18
engine_daemon_network_actions_seconds_bucket{action="allocate",le="+Inf"} 18
engine_daemon_network_actions_seconds_sum{action="allocate"} 8.8137919
engine_daemon_network_actions_seconds_count{action="allocate"} 18
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.005"} 0
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.01"} 0
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.025"} 0
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.05"} 1
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.1"} 4
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.25"} 7
engine_daemon_network_actions_seconds_bucket{action="connect",le="0.5"} 12
engine_daemon_network_actions_seconds_bucket{action="connect",le="1"} 15
engine_daemon_network_actions_seconds_bucket{action="connect",le="2.5"} 18
engine_daemon_network_actions_seconds_bucket{action="connect",le="5"} 18
engine_daemon_network_actions_seconds_bucket{action="connect",le="10"} 18
engine_daemon_network_actions_seconds_bucket{action="connect",le="+Inf"} 18
engine_daemon_network_actions_seconds_sum{action="connect"} 8.3260763
engine_daemon_network_actions_seconds_count{action="connect"} 18
engine_daemon_network_actions_seconds_bucket{action="release",le="0.005"} 0
engine_daemon_network_actions_seconds_bucket{action="release",le="0.01"} 0
engine_daemon_network_actions_seconds_bucket{action="release",le="0.025"} 0
engine_daemon_network_actions_seconds_bucket{action="release",le="0.05"} 0
engine_daemon_network_actions_seconds_bucket{action="release",le="0.1"} 0
engine_daemon_network_actions_seconds_bucket{action="release",le="0.25"} 1
engine_daemon_network_actions_seconds_bucket{action="release",le="0.5"} 1
engine_daemon_network_actions_seconds_bucket{action="release",le="1"} 1
engine_daemon_network_actions_seconds_bucket{action="release",le="2.5"} 6
engine_daemon_network_actions_seconds_bucket{action="release",le="5"} 7
engine_daemon_network_actions_seconds_bucket{action="release",le="10"} 7
engine_daemon_network_actions_seconds_bucket{action="release",le="+Inf"} 7
engine_daemon_network_actions_seconds_sum{action="release"} 11.1318685
engine_daemon_network_actions_seconds_count{action="release"} 7
# HELP etcd_debugging_snap_save_marshallling_duration_seconds The marshallling cost distributions of save
called by snapshot.
# TYPE etcd_debugging_snap_save_marshallling_duration_seconds histogram
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.001"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.002"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.004"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.008"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.016"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.032"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.064"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.128"} 0

```



```

etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.256"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="0.512"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="1.024"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="2.048"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="4.096"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="8.192"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_bucket{le="+Inf"} 0
etcd_debugging_snap_save_marshallling_duration_seconds_sum 0
etcd_debugging_snap_save_marshallling_duration_seconds_count 0
# HELP etcd_debugging_snap_save_total_duration_seconds The total latency distributions of save called by
snapshot.
# TYPE etcd_debugging_snap_save_total_duration_seconds histogram
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.001"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.002"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.004"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.008"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.016"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.032"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.064"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.128"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.256"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="0.512"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="1.024"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="2.048"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="4.096"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="8.192"} 0
etcd_debugging_snap_save_total_duration_seconds_bucket{le="+Inf"} 0
etcd_debugging_snap_save_total_duration_seconds_sum 0
etcd_debugging_snap_save_total_duration_seconds_count 0
# HELP etcd_disk_wal_fsync_duration_seconds The latency distributions of fsync called by wal.
# TYPE etcd_disk_wal_fsync_duration_seconds histogram
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.001"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.002"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.004"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.008"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.016"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.032"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.064"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.128"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.256"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="0.512"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="1.024"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="2.048"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="4.096"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="8.192"} 0
etcd_disk_wal_fsync_duration_seconds_bucket{le="+Inf"} 0
etcd_disk_wal_fsync_duration_seconds_sum 0
etcd_disk_wal_fsync_duration_seconds_count 0
# HELP etcd_snap_db_fsync_duration_seconds The latency distributions of fsyncing .snap.db file
# TYPE etcd_snap_db_fsync_duration_seconds histogram
etcd_snap_db_fsync_duration_seconds_bucket{le="0.001"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.002"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.004"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.008"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.016"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.032"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.064"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.128"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.256"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="0.512"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="1.024"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="2.048"} 0

```

```

etcd_snap_db_fsync_duration_seconds_bucket{le="4.096"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="8.192"} 0
etcd_snap_db_fsync_duration_seconds_bucket{le="+Inf"} 0
etcd_snap_db_fsync_duration_seconds_sum 0
etcd_snap_db_fsync_duration_seconds_count 0
# HELP etcd_snap_db_save_total_duration_seconds The total latency distributions of v3 snapshot save
# TYPE etcd_snap_db_save_total_duration_seconds histogram
etcd_snap_db_save_total_duration_seconds_bucket{le="0.1"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="0.2"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="0.4"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="0.8"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="1.6"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="3.2"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="6.4"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="12.8"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="25.6"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="51.2"} 0
etcd_snap_db_save_total_duration_seconds_bucket{le="+Inf"} 0
etcd_snap_db_save_total_duration_seconds_sum 0
etcd_snap_db_save_total_duration_seconds_count 0
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 8e-06
go_gc_duration_seconds{quantile="0.25"} 1.08e-05
go_gc_duration_seconds{quantile="0.5"} 1.28e-05
go_gc_duration_seconds{quantile="0.75"} 3.38e-05
go_gc_duration_seconds{quantile="1"} 0.0053299
go_gc_duration_seconds_sum 32.5016842
go_gc_duration_seconds_count 195833
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 130
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.13.15"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.6742992e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 2.687859756144e+12
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 3.133875e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 6.728477567e+09
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the
program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0.003659979972674019
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.074944e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.6742992e+07
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.00048896e+08
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge

```

```

go_memstats_heap_inuse_bytes 3.239936e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 224116
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 9.8861056e+07
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 1.32448256e+08
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 1.6222426667013512e+09
# HELP go_memstats_lookups_total Total number of pointer lookups.
# TYPE go_memstats_lookups_total counter
go_memstats_lookups_total 0
# HELP go_memstats_mallocs_total Total number of mallocs.
# TYPE go_memstats_mallocs_total counter
go_memstats_mallocs_total 6.728701683e+09
# HELP go_memstats_mcache_inuse_bytes Number of bytes in use by mcache structures.
# TYPE go_memstats_mcache_inuse_bytes gauge
go_memstats_mcache_inuse_bytes 6944
# HELP go_memstats_mcache_sys_bytes Number of bytes used for mcache structures obtained from system.
# TYPE go_memstats_mcache_sys_bytes gauge
go_memstats_mcache_sys_bytes 16384
# HELP go_memstats_mspan_inuse_bytes Number of bytes in use by mspan structures.
# TYPE go_memstats_mspan_inuse_bytes gauge
go_memstats_mspan_inuse_bytes 453424
# HELP go_memstats_mspan_sys_bytes Number of bytes used for mspan structures obtained from system.
# TYPE go_memstats_mspan_sys_bytes gauge
go_memstats_mspan_sys_bytes 1.163264e+06
# HELP go_memstats_next_gc_bytes Number of heap bytes when next garbage collection will take place.
# TYPE go_memstats_next_gc_bytes gauge
go_memstats_next_gc_bytes 2.8596208e+07
# HELP go_memstats_other_sys_bytes Number of bytes used for other system allocations.
# TYPE go_memstats_other_sys_bytes gauge
go_memstats_other_sys_bytes 1.263429e+06
# HELP go_memstats_stack_inuse_bytes Number of bytes in use by the stack allocator.
# TYPE go_memstats_stack_inuse_bytes gauge
go_memstats_stack_inuse_bytes 1.769472e+06
# HELP go_memstats_stack_sys_bytes Number of bytes obtained from system for stack allocator.
# TYPE go_memstats_stack_sys_bytes gauge
go_memstats_stack_sys_bytes 1.769472e+06
# HELP go_memstats_sys_bytes Number of bytes obtained from system.
# TYPE go_memstats_sys_bytes gauge
go_memstats_sys_bytes 1.44869624e+08
# HELP go_threads Number of OS threads created.
# TYPE go_threads gauge
go_threads 26
# HELP logger_log_entries_size_greater_than_buffer_total Number of log entries which are larger than the log
buffer
# TYPE logger_log_entries_size_greater_than_buffer_total counter
logger_log_entries_size_greater_than_buffer_total 2
# HELP logger_log_read_operations_failed_total Number of log reads from container stdio that failed
# TYPE logger_log_read_operations_failed_total counter
logger_log_read_operations_failed_total 0
# HELP logger_log_write_operations_failed_total Number of log write operations that failed
# TYPE logger_log_write_operations_failed_total counter
logger_log_write_operations_failed_total 0
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 14680.84

```

```

# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 1.048576e+06
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 145
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 1.3899776e+08
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.62215164799e+09
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 2.101641216e+09
# HELP process_virtual_memory_max_bytes Maximum amount of virtual memory available in bytes.
# TYPE process_virtual_memory_max_bytes gauge
process_virtual_memory_max_bytes -1
# HELP promhttp_metric_handler_requests_in_flight Current number of scrapes being served.
# TYPE promhttp_metric_handler_requests_in_flight gauge
promhttp_metric_handler_requests_in_flight 1
# HELP promhttp_metric_handler_requests_total Total number of scrapes by HTTP status code.
# TYPE promhttp_metric_handler_requests_total counter
promhttp_metric_handler_requests_total{code="200"} 0
promhttp_metric_handler_requests_total{code="500"} 0
promhttp_metric_handler_requests_total{code="503"} 0
# HELP swarm_dispatcher_scheduling_delay_seconds Scheduling delay is the time a task takes to go from NEW to
RUNNING state.
# TYPE swarm_dispatcher_scheduling_delay_seconds histogram
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.005"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.01"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.025"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.05"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.1"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.25"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="0.5"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="1"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="2.5"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="5"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="10"} 0
swarm_dispatcher_scheduling_delay_seconds_bucket{le="+Inf"} 0
swarm_dispatcher_scheduling_delay_seconds_sum 0
swarm_dispatcher_scheduling_delay_seconds_count 0
# HELP swarm_manager_configs_total The number of configs in the cluster object store
# TYPE swarm_manager_configs_total gauge
swarm_manager_configs_total 0
# HELP swarm_manager_leader Indicates if this manager node is a leader
# TYPE swarm_manager_leader gauge
swarm_manager_leader 0
# HELP swarm_manager_networks_total The number of networks in the cluster object store
# TYPE swarm_manager_networks_total gauge
swarm_manager_networks_total 0
# HELP swarm_manager_nodes The number of nodes
# TYPE swarm_manager_nodes gauge
swarm_manager_nodes{state="disconnected"} 0
swarm_manager_nodes{state="down"} 0
swarm_manager_nodes{state="ready"} 0
swarm_manager_nodes{state="unknown"} 0
# HELP swarm_manager_secrets_total The number of secrets in the cluster object store
# TYPE swarm_manager_secrets_total gauge
swarm_manager_secrets_total 0
# HELP swarm_manager_services_total The number of services in the cluster object store

```

```

# TYPE swarm_manager_services_total gauge
swarm_manager_services_total 0
# HELP swarm_manager_tasks_total The number of tasks in the cluster object store
# TYPE swarm_manager_tasks_total gauge
swarm_manager_tasks_total{state="accepted"} 0
swarm_manager_tasks_total{state="assigned"} 0
swarm_manager_tasks_total{state="complete"} 0
swarm_manager_tasks_total{state="failed"} 0
swarm_manager_tasks_total{state="new"} 0
swarm_manager_tasks_total{state="orphaned"} 0
swarm_manager_tasks_total{state="pending"} 0
swarm_manager_tasks_total{state="preparing"} 0
swarm_manager_tasks_total{state="ready"} 0
swarm_manager_tasks_total{state="rejected"} 0
swarm_manager_tasks_total{state="remove"} 0
swarm_manager_tasks_total{state="running"} 0
swarm_manager_tasks_total{state="shutdown"} 0
swarm_manager_tasks_total{state="starting"} 0
# HELP swarm_node_manager Whether this node is a manager or not
# TYPE swarm_node_manager gauge
swarm_node_manager 0
# HELP swarm_raft_snapshot_latency_seconds Raft snapshot create latency.
# TYPE swarm_raft_snapshot_latency_seconds histogram
swarm_raft_snapshot_latency_seconds_bucket{le="0.005"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.01"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.025"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.05"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.1"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.25"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="0.5"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="1"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="2.5"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="5"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="10"} 0
swarm_raft_snapshot_latency_seconds_bucket{le="+Inf"} 0
swarm_raft_snapshot_latency_seconds_sum 0
swarm_raft_snapshot_latency_seconds_count 0
# HELP swarm_raft_transaction_latency_seconds Raft transaction latency.
# TYPE swarm_raft_transaction_latency_seconds histogram
swarm_raft_transaction_latency_seconds_bucket{le="0.005"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.01"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.025"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.05"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.1"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.25"} 0
swarm_raft_transaction_latency_seconds_bucket{le="0.5"} 0
swarm_raft_transaction_latency_seconds_bucket{le="1"} 0
swarm_raft_transaction_latency_seconds_bucket{le="2.5"} 0
swarm_raft_transaction_latency_seconds_bucket{le="5"} 0
swarm_raft_transaction_latency_seconds_bucket{le="10"} 0
swarm_raft_transaction_latency_seconds_bucket{le="+Inf"} 0
swarm_raft_transaction_latency_seconds_sum 0
swarm_raft_transaction_latency_seconds_count 0
# HELP swarm_store_batch_latency_seconds Raft store batch latency.
# TYPE swarm_store_batch_latency_seconds histogram
swarm_store_batch_latency_seconds_bucket{le="0.005"} 0
swarm_store_batch_latency_seconds_bucket{le="0.01"} 0
swarm_store_batch_latency_seconds_bucket{le="0.025"} 0
swarm_store_batch_latency_seconds_bucket{le="0.05"} 0
swarm_store_batch_latency_seconds_bucket{le="0.1"} 0
swarm_store_batch_latency_seconds_bucket{le="0.25"} 0
swarm_store_batch_latency_seconds_bucket{le="0.5"} 0

```

```

swarm_store_batch_latency_seconds_bucket{le="1"} 0
swarm_store_batch_latency_seconds_bucket{le="2.5"} 0
swarm_store_batch_latency_seconds_bucket{le="5"} 0
swarm_store_batch_latency_seconds_bucket{le="10"} 0
swarm_store_batch_latency_seconds_bucket{le="+Inf"} 0
swarm_store_batch_latency_seconds_sum 0
swarm_store_batch_latency_seconds_count 0
# HELP swarm_store_lookup_latency_seconds Raft store read latency.
# TYPE swarm_store_lookup_latency_seconds histogram
swarm_store_lookup_latency_seconds_bucket{le="0.005"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.01"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.025"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.05"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.1"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.25"} 0
swarm_store_lookup_latency_seconds_bucket{le="0.5"} 0
swarm_store_lookup_latency_seconds_bucket{le="1"} 0
swarm_store_lookup_latency_seconds_bucket{le="2.5"} 0
swarm_store_lookup_latency_seconds_bucket{le="5"} 0
swarm_store_lookup_latency_seconds_bucket{le="10"} 0
swarm_store_lookup_latency_seconds_bucket{le="+Inf"} 0
swarm_store_lookup_latency_seconds_sum 0
swarm_store_lookup_latency_seconds_count 0
# HELP swarm_store_memory_store_lock_duration_seconds Duration for which the raft memory store lock was held.
# TYPE swarm_store_memory_store_lock_duration_seconds histogram
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.005"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.01"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.025"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.05"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.1"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.25"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="0.5"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="1"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="2.5"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="5"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="10"} 0
swarm_store_memory_store_lock_duration_seconds_bucket{le="+Inf"} 0
swarm_store_memory_store_lock_duration_seconds_sum 0
swarm_store_memory_store_lock_duration_seconds_count 0
# HELP swarm_store_read_tx_latency_seconds Raft store read tx latency.
# TYPE swarm_store_read_tx_latency_seconds histogram
swarm_store_read_tx_latency_seconds_bucket{le="0.005"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.01"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.025"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.05"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.1"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.25"} 0
swarm_store_read_tx_latency_seconds_bucket{le="0.5"} 0
swarm_store_read_tx_latency_seconds_bucket{le="1"} 0
swarm_store_read_tx_latency_seconds_bucket{le="2.5"} 0
swarm_store_read_tx_latency_seconds_bucket{le="5"} 0
swarm_store_read_tx_latency_seconds_bucket{le="10"} 0
swarm_store_read_tx_latency_seconds_bucket{le="+Inf"} 0
swarm_store_read_tx_latency_seconds_sum 0
swarm_store_read_tx_latency_seconds_count 0
# HELP swarm_store_write_tx_latency_seconds Raft store write tx latency.
# TYPE swarm_store_write_tx_latency_seconds histogram
swarm_store_write_tx_latency_seconds_bucket{le="0.005"} 0
swarm_store_write_tx_latency_seconds_bucket{le="0.01"} 0
swarm_store_write_tx_latency_seconds_bucket{le="0.025"} 0
swarm_store_write_tx_latency_seconds_bucket{le="0.05"} 0

```

```
swarm_store_write_tx_latency_seconds_bucket{le="0.1"} 0
swarm_store_write_tx_latency_seconds_bucket{le="0.25"} 0
swarm_store_write_tx_latency_seconds_bucket{le="0.5"} 0
swarm_store_write_tx_latency_seconds_bucket{le="1"} 0
swarm_store_write_tx_latency_seconds_bucket{le="2.5"} 0
swarm_store_write_tx_latency_seconds_bucket{le="5"} 0
swarm_store_write_tx_latency_seconds_bucket{le="10"} 0
swarm_store_write_tx_latency_seconds_bucket{le="+Inf"} 0
swarm_store_write_tx_latency_seconds_sum 0
swarm_store_write_tx_latency_seconds_count 0
```

Приложение 5 – Конфигурация СУБД Tarantool

```
u = box.schema.space.create('user')
u:format({
  {name = 'id', type = 'unsigned'},
  {name = 'login', type = 'string'},
  {name = 'password', type = 'string'},
  {name = 'email', type = 'string'},
  {name = 'token', type = 'string'}
})
u:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
u:create_index('secondary', {
  type = 'tree',
  unique = false,
  parts = {'login'}
})
u:insert{1, 'admin', 'admin', 'admin@admin.com', '12345' }
m = box.schema.space.create('monitoring_objects')
m:format({
  {name = 'id', type = 'unsigned'},
  {name = 'name', type = 'string'},
  {name = 'type', type = 'string'},
  {name = 'address', type = 'string'},
  {name = 'system', type = 'string'},
  {name = 'creation_date', type = 'string'},
  {name = 'status', type = 'string'},
})
m:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
c = box.schema.space.create('contacts')
c:format({
  {name = 'id', type = 'unsigned'},
  {name = 'email', type = 'string'},
  {name = 'address', type = 'string'},
  {name = 'phone_number', type = 'string'}
})
```



```

c:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
r = box.schema.space.create('responsible_persons')
r:format({
  {name = 'id', type = 'unsigned'},
  {name = 'name', type = 'string'},
  {name = 'role', type = 'string'},
  {name = 'type', type = 'string'},
  {name = 'contact_id', type = 'unsigned'},
  {name = 'monitoring_object_id', type = 'unsigned'},
})
r:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
r:create_index('secondary', {
  type = 'tree',
  unique = false,
  parts = {'monitoring_object_id'}
})
rs = box.schema.space.create('reaction_scripts')
rs:format({
  {name = 'id', type = 'unsigned'},
  {name = 'name', type = 'string'},
  {name = 'condirion', type = 'string'},
  {name = 'reaction', type = 'string'},
  {name = 'monitoring_object_id', type = 'unsigned'}
})
rs:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
rs:create_index('secondary', {
  type = 'tree',
  unique = false,
  parts = {'monitoring_object_id'}
})

```

```

rcts = box.schema.space.create('reactions')
rcts:format({
  {name = 'id', type = 'unsigned'},
  {name = 'status', type = 'string'},
  {name = 'reaction_date', type = 'string'},
  {name = 'reaction_script_id', type = 'unsigned'},
  {name = 'reaction_details', type = 'string'}
})
rcts:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
rcts:create_index('secondary', {
  type = 'tree',
  unique = false,
  parts = {'reaction_script_id'}
})
mon = box.schema.space.create('monitor')
mon:format({
  {name = 'id', type = 'unsigned'},
  {name = 'name', type = 'string'},
  {name = 'date', type = 'string'},
  {name = 'monitoring_object_id', type = 'unsigned'}
})
mon:create_index('primary', {
  type = 'hash',
  parts = {'id'}
})
mon:create_index('secondary', {
  type = 'tree',
  unique = false,
  parts = {'monitoring_object_id'}
})
mtrc = box.schema.space.create('metric_collection')
mtrc:format({
  {name = 'id', type = 'unsigned'},
  {name = 'name', type = 'string'},
  {name = 'value', type = 'string'},
  {name = 'date_hit', type = 'string'},
  {name = 'metric_type_id', type = 'unsigned'},
  {name = 'monitoring_object_id', type = 'unsigned'},

```

```

    {name = 'monitor_id', type = 'unsigned'}
  })
  mtrc:create_index('primary', {
    type = 'hash',
    parts = {'id'}
  })
  mtrc:create_index('secondary', {
    type = 'tree',
    unique = false,
    parts = {'monitor_id'}
  })
  mt = box.schema.space.create('metric_types')
  mt:format({
    {name = 'id', type = 'unsigned'},
    {name = 'name', type = 'string'},
    {name = 'format', type = 'string'},
    {name = 'description', type = 'string'}
  })
  mt:create_index('primary', {
    type = 'hash',
    parts = {'id'}
  })
  prs = box.schema.space.create('projects_structure')
  prs:format({
    {name = 'id', type = 'unsigned'},
    {name = 'name', type = 'string'},
    {name = 'description', type = 'string'},
    {name = 'monitoring_object_id', type = 'unsigned'}
  })
  prs:create_index('primary', {
    type = 'hash',
    parts = {'id'}
  })
  prs:create_index('secondary', {
    type = 'tree',
    unique = false,
    parts = {'monitoring_object_id'}})

```