|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №8** | |
| **по дисциплине** |  |
| **«Технологии виртуализации клиент-серверных приложений»** | |
| Выполнил студент группы ИКБО-01-22 | Прокопчук Р.О. |
| Принял преподаватель кафедры ИиППО | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2025 г. |  |
| «Зачтено» | « » 2025 г. |  |

Москва 2025

**Теоретическое введение**

В данной практической работе необходимо будет работать со множеством сервисов. В теоретическом введение будет дана краткая информация о них.

Kraken - это расширяемый, декларативный, высокопроизводительный APIшлюз с открытым исходным кодом.

Его основная функциональность заключается в создании API, который действует как агрегатор многих микросервисов в конечные точки, автоматически выполняя тяжелую работу: агрегирование, преобразование, фильтрацию, декодирование, дросселирование, аутентификацию и многое другое.

Конфигурацию для него можно писать самостоятельно в формате JSON или же использовать онлайн конфигуратор KrakenD Designer.

Kafka Apache — распределенная система обмена сообщениями между серверными приложениями в режиме реального времени. Благодаря высокой пропускной способности, масштабируемости и надежности применяется в компаниях, работающих с большими объемами данных. Написана на языках Java и Scala. Kafka — связующее звено между отдельными функциональными модулями большой системы. Например, с ее помощью можно подписать микросервис на другие компоненты для регулярного получения обновлений.

Управление приложениями — сложный аспект Kubernetes. Helm значительно упрощает его, предоставляя единый метод упаковки программного обеспечения, поддерживающий контроль версий. Helm устанавливает пакеты (называются Чартами в Helm) для Kubernetes и управляет ими, как это делают yum и apt. Чарты содержат все определения ресурсов, необходимые для запуска приложения, инструмента или службы внутри кластера Kubernetes.

Jaeger – это программное обеспечение, которое вы можете использовать для отслеживания и устранения проблем во взаимосвязанных программных компонентах, называемых микросервисами. Несколько микросервисов взаимодействуют между собой с целью выполнения единой программной функции. Разработчики используют Jaeger для визуализации цепочки событий при взаимодействии микросервисов, чтобы изолировать проблему, когда что-то идет не так, как требуется. Jaeger также называется трассировкой Jaeger, потому что отслеживает, или трассирует, путь запроса через серию взаимодействий между микросервисами.

**Постановка задачи**

Разработать Backend или часть Backend системы в соответствии с выбранной для ВКР темой с использованием Spring/Spring Boot. Темы в группах не должны повторяться.

Система должна иметь сервис авторизации с хранением токенов в RedisDB и минимум 2 сервиса, которые должны общаться между собой с помощью шины сообщений с использованием Kafka. Хранение данных должно производиться с помощью СУБД PostgreSQL.

В качестве API-шлюза к сервисам должен выступать KrakenD.

Необходимо логировать все обращения к сервисам с помощью Graylog. В логах должны указываться HTTP-метод, URL, IP-адрес отправителя.

Также необходимо собирать метрики с баз данных с помощью связки Prometheus + Grafana, а все транзакции подвергать трассировке с помощью Jaeger.

В случае невозможности реализации системы с использованием предложенного стека технологий необходимо обосновать это и предложить решение на собственном стеке.

Для каждого сервиса необходимы:

⎯ Установка в Minikube при помощи Helm-чартов;

⎯ Настройка лимитов;

⎯ Настройка Liveness, Readiness and Startup Probes;

⎯ Настройка перезапуска при падении;

⎯ Настройка использования ресурсов пода на 60% и использование Autoscaling.

Для СУБД:

⎯ Использование persistent volume;

⎯ Конфигурация макс количества соединений;

⎯ Создание пользователя для считывания данных мониторингом;

⎯ Настройка мониторинга пода с БД при помощи Prometheus и Grafana;

⎯ Должен соблюдаться паттерн “1 БД на 1 сервис”.

В отчете вам необходимо отобразить все этапы конфигурации системы, показать конфигурационные файлы Minikibe, Helm Charts, KrakenD, отобразить работу системы с учетом авторизации, показать процесс в сервисах Kafka, Jaeger, Prometheus, Grafana, Graylog, отобразить сохранение данных в persistent volume при отключении пода с БД.

**Ход работы**

Структура проекта представлена на рисукнке 1.

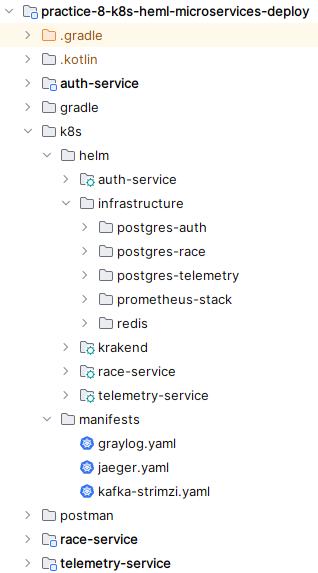
****

Рисунок 1 – Структура проекта

Gradle-файлы всех сервисов представлены на рисунках 2-4.

****

Рисунок 2 – Gradle файл сервиса авторизации

****

Рисунок 3 – Gradle файл сервиса гонок

****

Рисунок 4 – Gradle файл сервиса телеметрии

Docker-файлы сервисов представлены на рисунках 5-7.

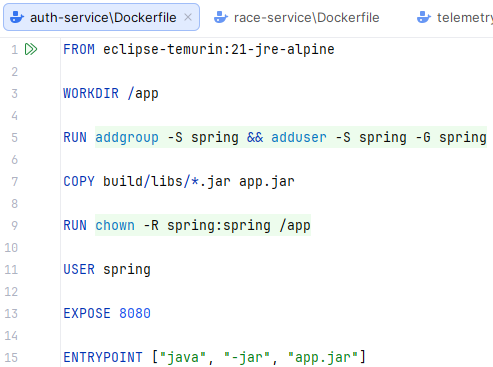
****

Рисунок 5 –Dockerfile сервиса авторизации

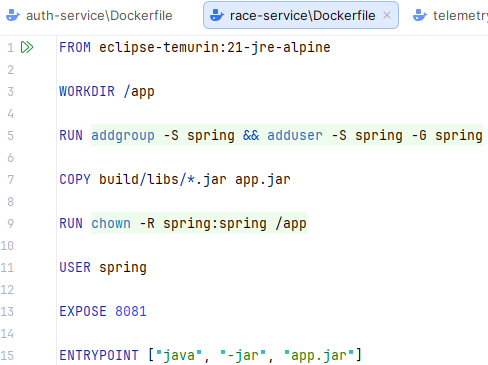
****

Рисунок 6 –Dockerfile сервиса авторизации

****

Рисунок 7 –Dockerfile сервиса авторизации

Values-файлы инфраструктурных сервисов, которые будут разворачиваться через Bitnami предаставлены на рисунках 8-13.

****

Рисунок 8 – Values-файл БД сервиса авторизации

****

Рисунок 9 – Values-файл Redis сервиса авторизации

****

Рисунок 10 – Values-файл БД сервиса гонок

****

Рисунок 11 – Values-файл БД сервиса телеметрии

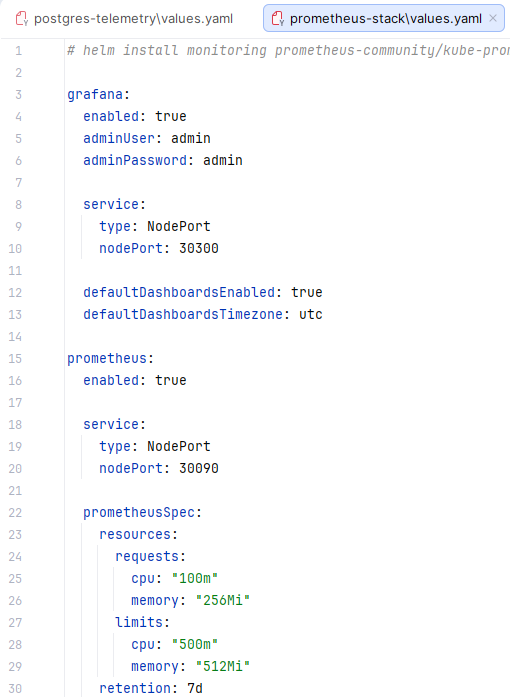
****

Рисунок 12 – Values-файл сервисов мониторинга (1/2)

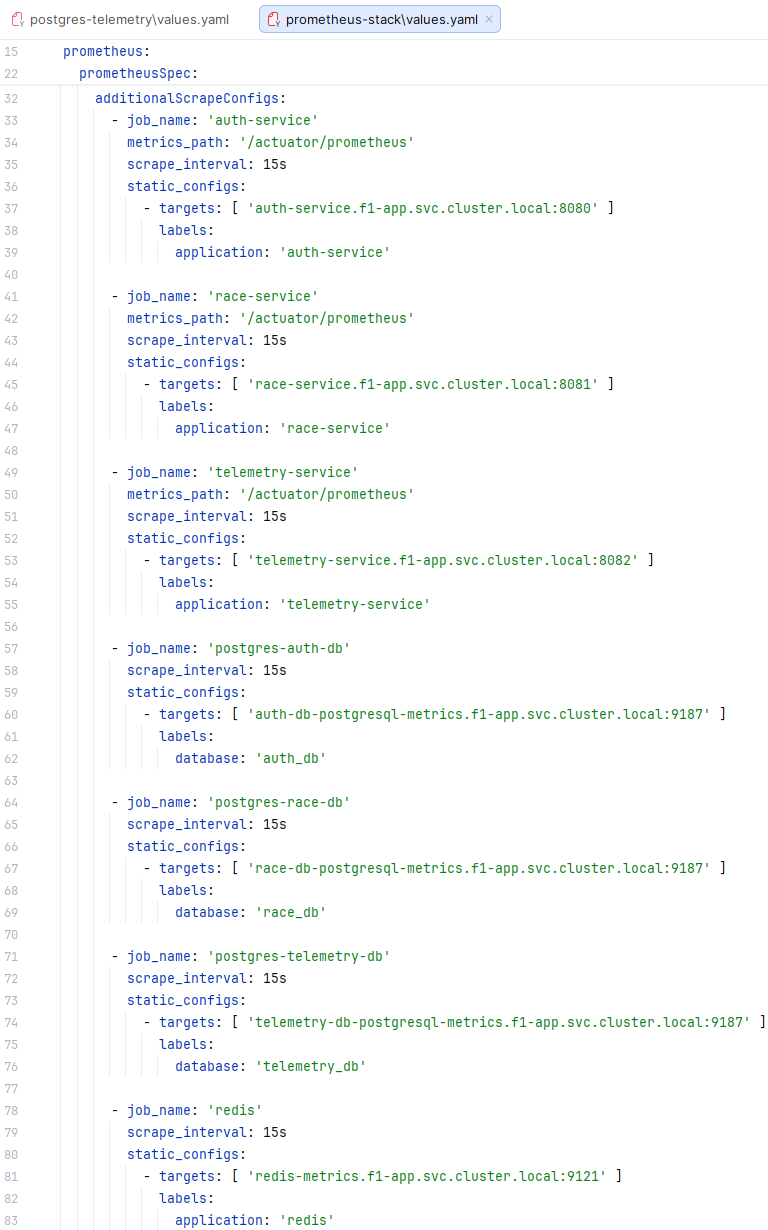
****

Рисунок 13 – Values-файл сервисов мониторинга (2/2)

Файлы helm-чарта сервиса авторизации представлены на рисунках 14-21.

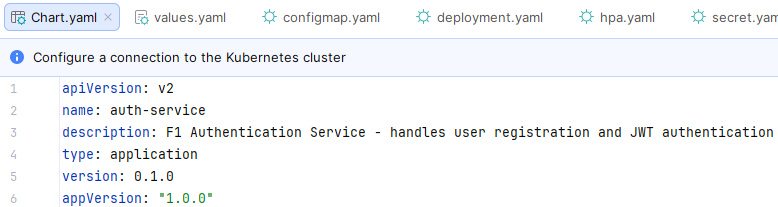
****

Рисунок 14 – Chart.yaml сервиса авторзизации

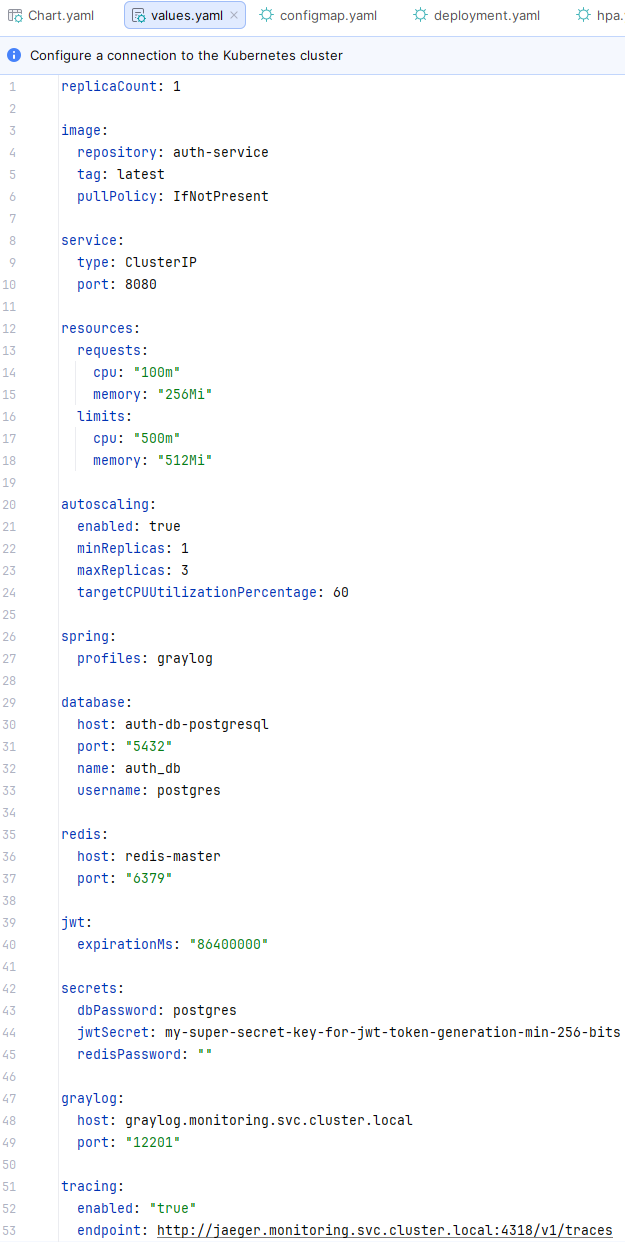
****

Рисунок 15 – Values.yaml сервиса авторизации (1/2)

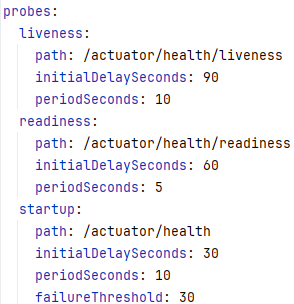
****

Рисунок 16 – Values.yaml сервиса авторизации (2/2)

****

Рисунок 17 – Configmap.yaml сервиса авторизации

****

Рисунок 18 – Deployment.yaml сервиса авторизации

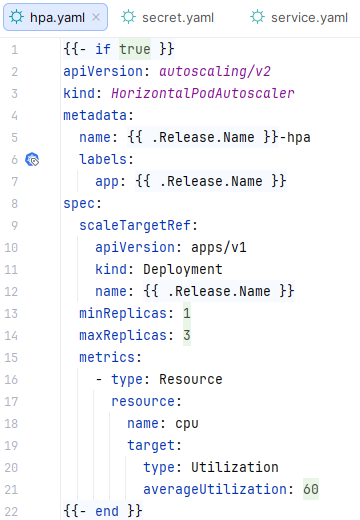
****

Рисунок 19 – Hpa.yaml сервиса авторизации

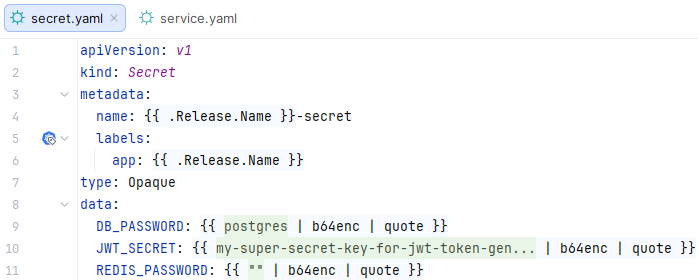
****

Рисунок 20 – Secret.yaml сервиса авторизации

****

Рисунок 21 – Service.yaml сервиса авторизации

Файлы helm-чарта сервиса гонок представлены на рисунках 22-29.

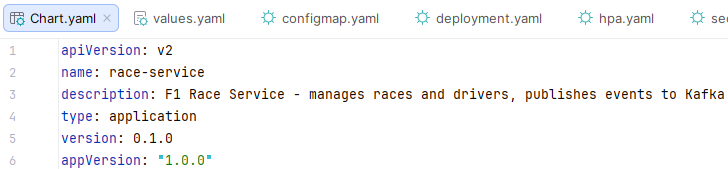
****

Рисунок 22 – Chart.yaml сервиса гонок

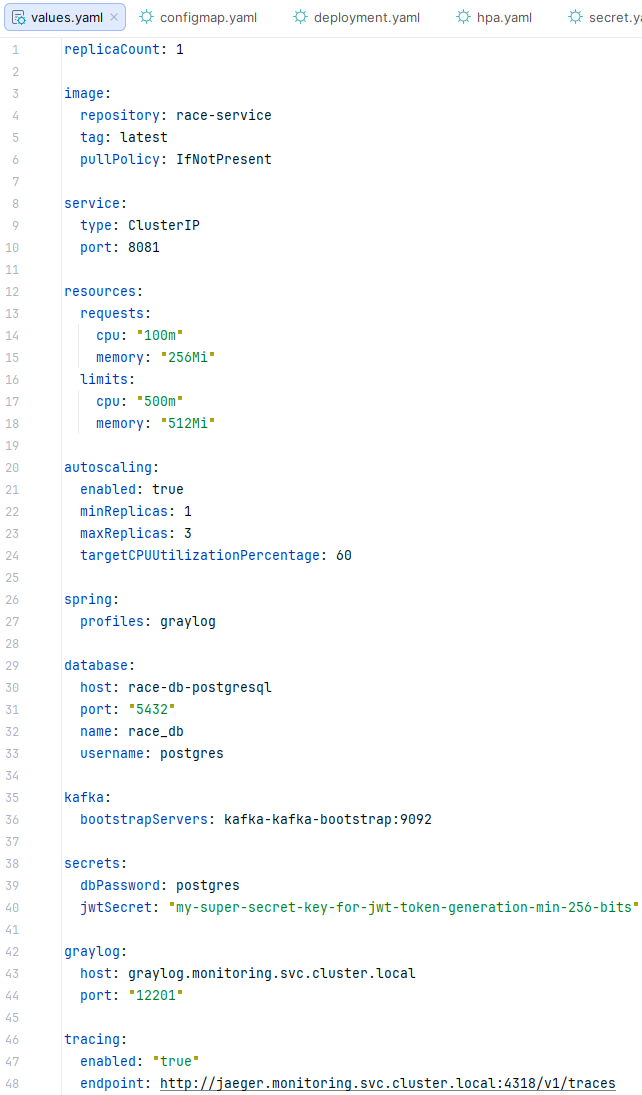
****

Рисунок 23 – Values.yaml сервиса гонок (1/2)

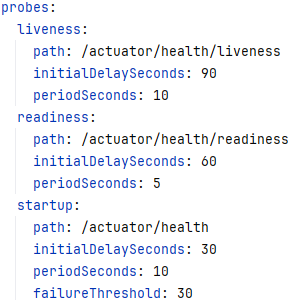
****

Рисунок 24 – Values.yaml сервиса гонок (2/2)

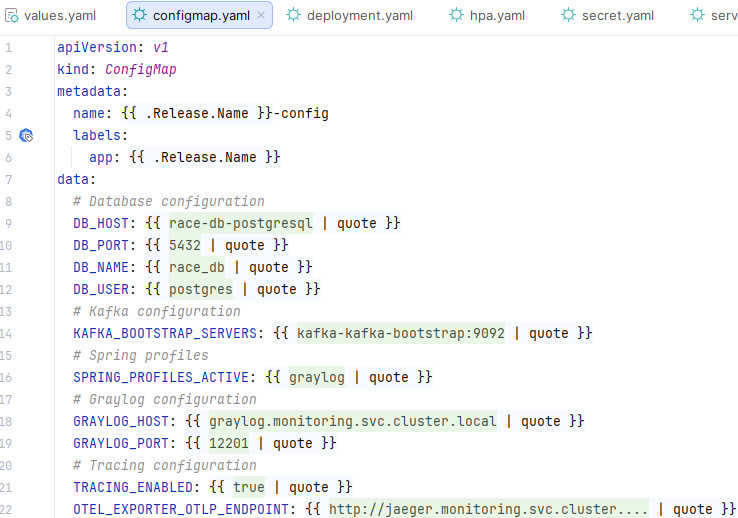
****

Рисунок 25 – Configmap.yaml сервиса гонок

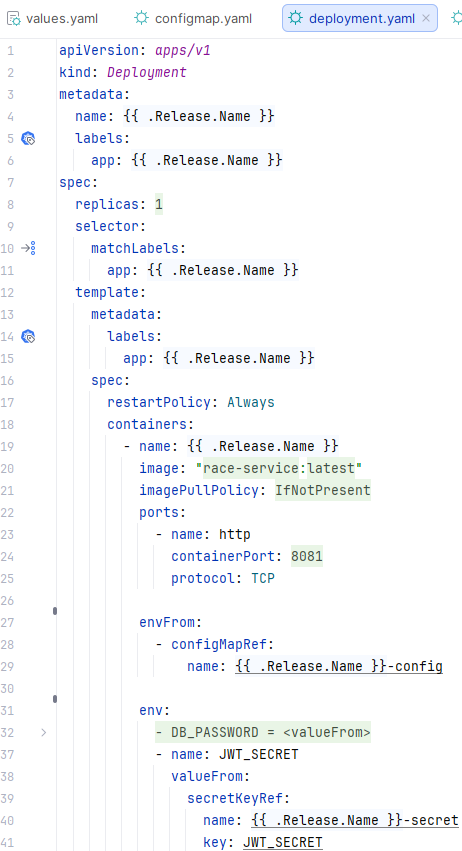
****

Рисунок 26 – Deployment.yaml сервиса гонок (1/2)

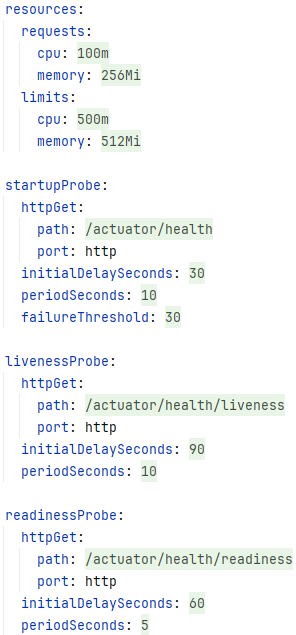
****

Рисунок 27 – Deployment.yaml сервиса гонок (2/2)

****

Рисунок 28 – Hpa.yaml сервиса гонок

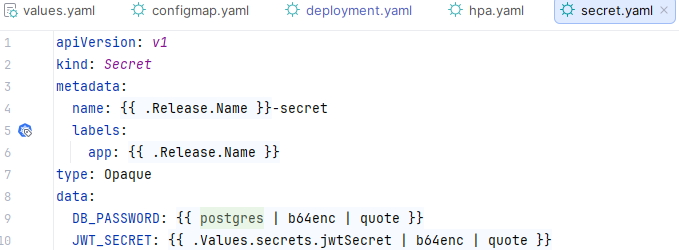
****

Рисунок 29 – Secret.yaml сервиса гонок

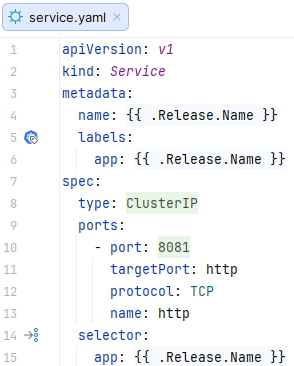
****

Рисунок 30 – Service.yaml сервиса гонок

Файлы helm-чарта сервиса телеметрии представлены на рисунках 31-37.

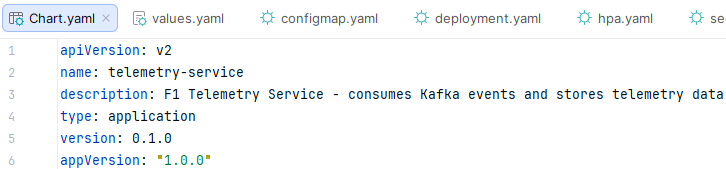
****

Рисунок 31 – Chart.yaml сервиса телеметрии

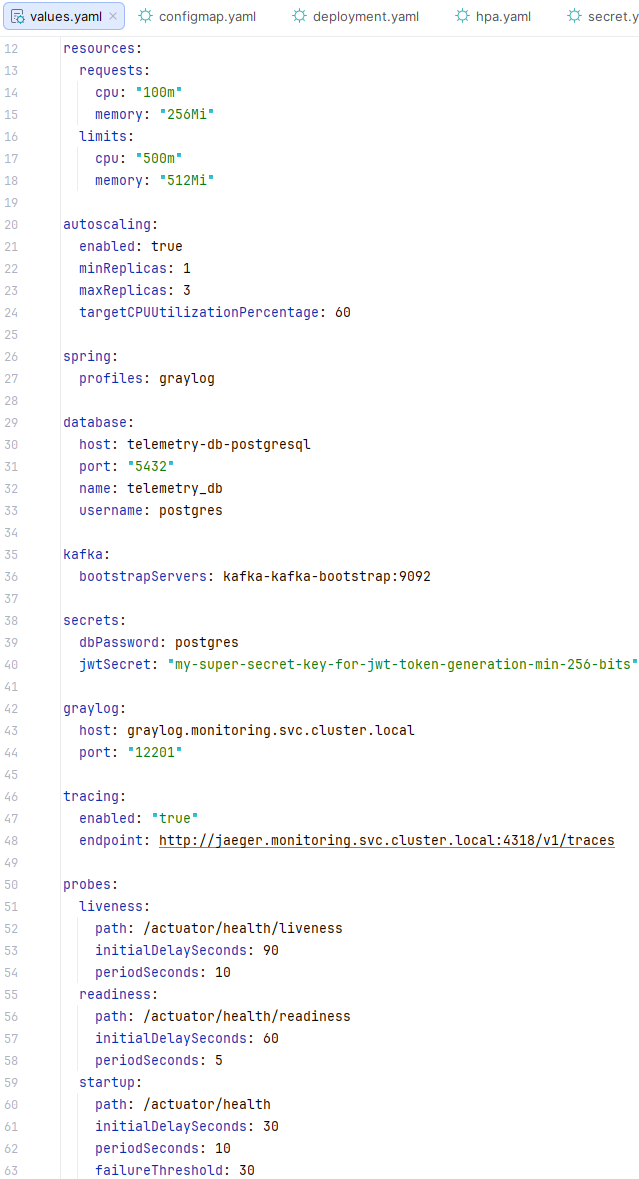
****

Рисунок 32 – Values.yaml сервиса телеметрии

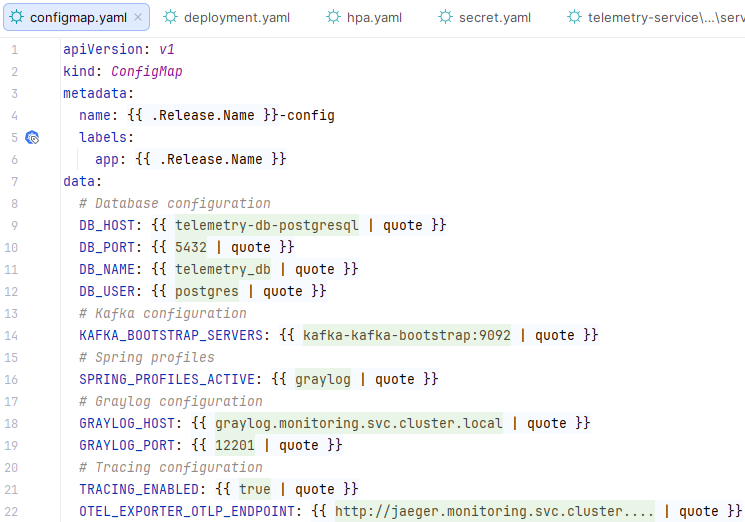
****

Рисунок 33 – Configmap.yaml сервиса телеметрии

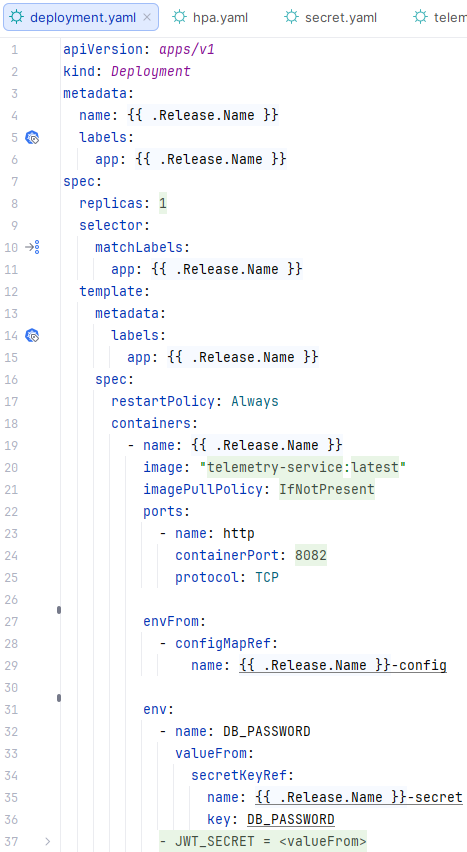
****

Рисунок 34 – Deployment.yaml сервиса телеметрии(1/2)

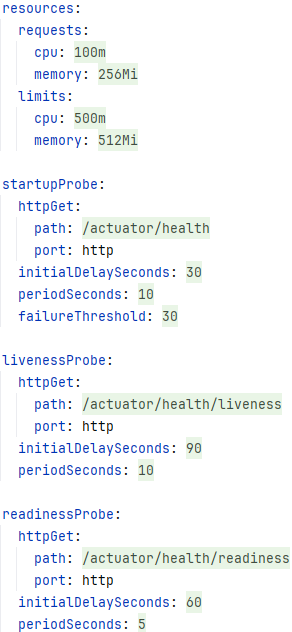
****

Рисунок 35 – Deployment.yaml сервиса телеметрии (2/2)

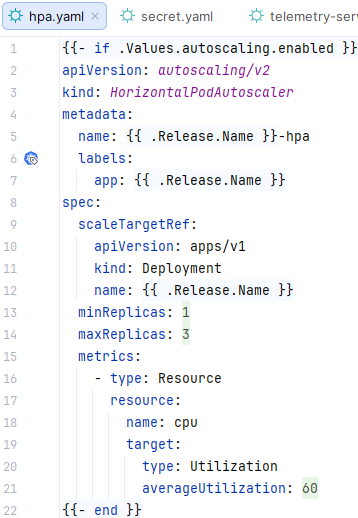
****

Рисунок 36 – Hpa.yaml сервиса телеметрии

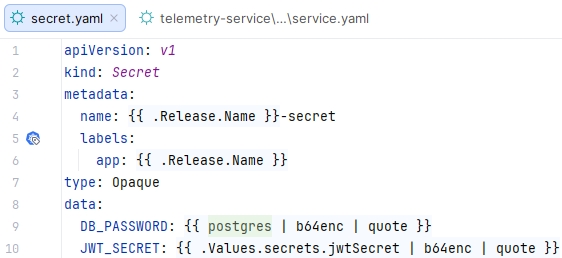
****

Рисунок 37 – Secret.yaml сервиса телеметрии

****

Рисунок 38 – Service.yaml сервиса телеметрии

Файлы helm-чарта KrakenD представлены на рисунках 39-44.

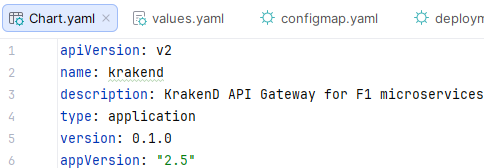
****

Рисунок 39 – Chart.yaml KrakenD

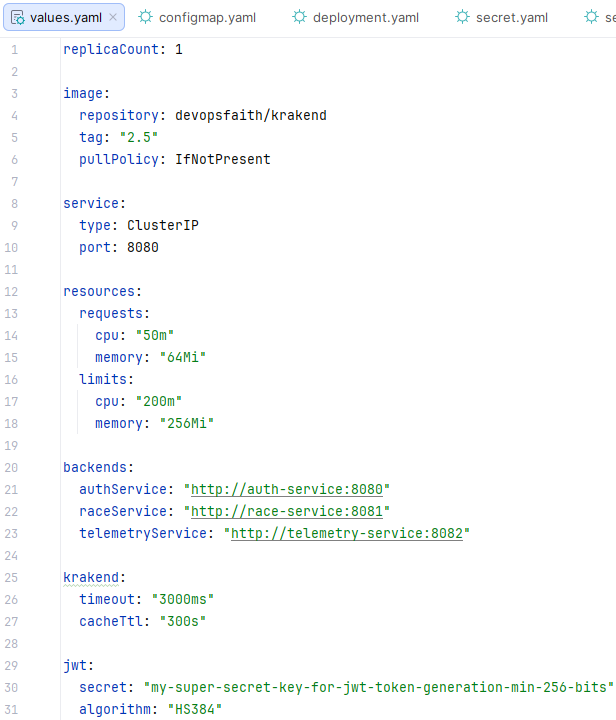
****

Рисунок 40 – Values.yaml KrakenD

****

Рисунок 41 – Configmap.yaml KrakenD

****

Рисунок 42 – Deployment.yaml KrakenD

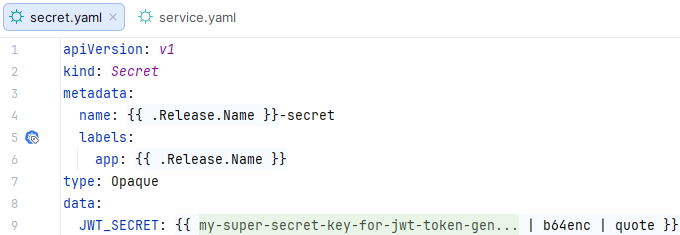
****

Рисунок 43 – Secret.yaml KrakenD

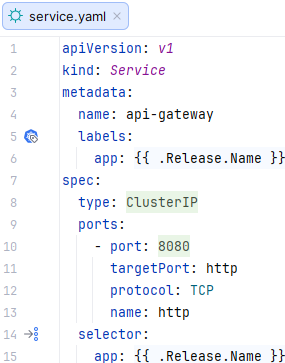
****

Рисунок 44 – Service.yaml KrakenD

Файлы для запуска GrayLog и его окружения представлены на рисунках 45-49.

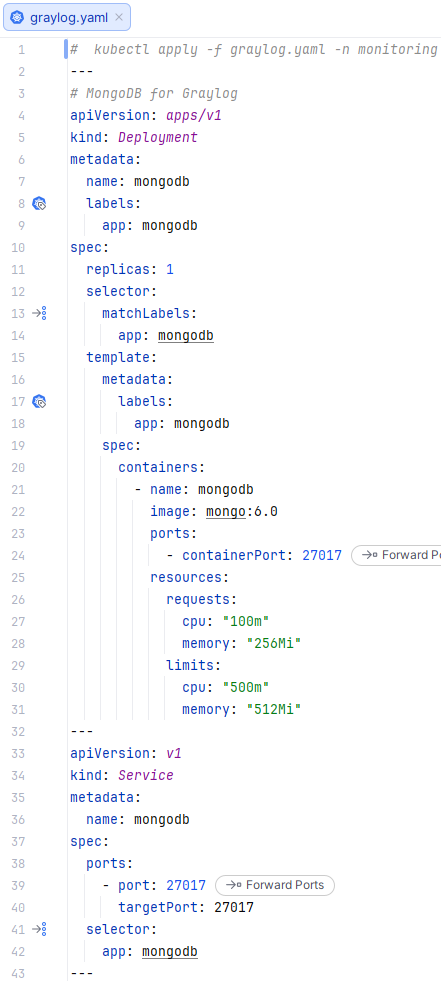
****

Рисунок 45 – Deployment и Service MongoBD для Graylog

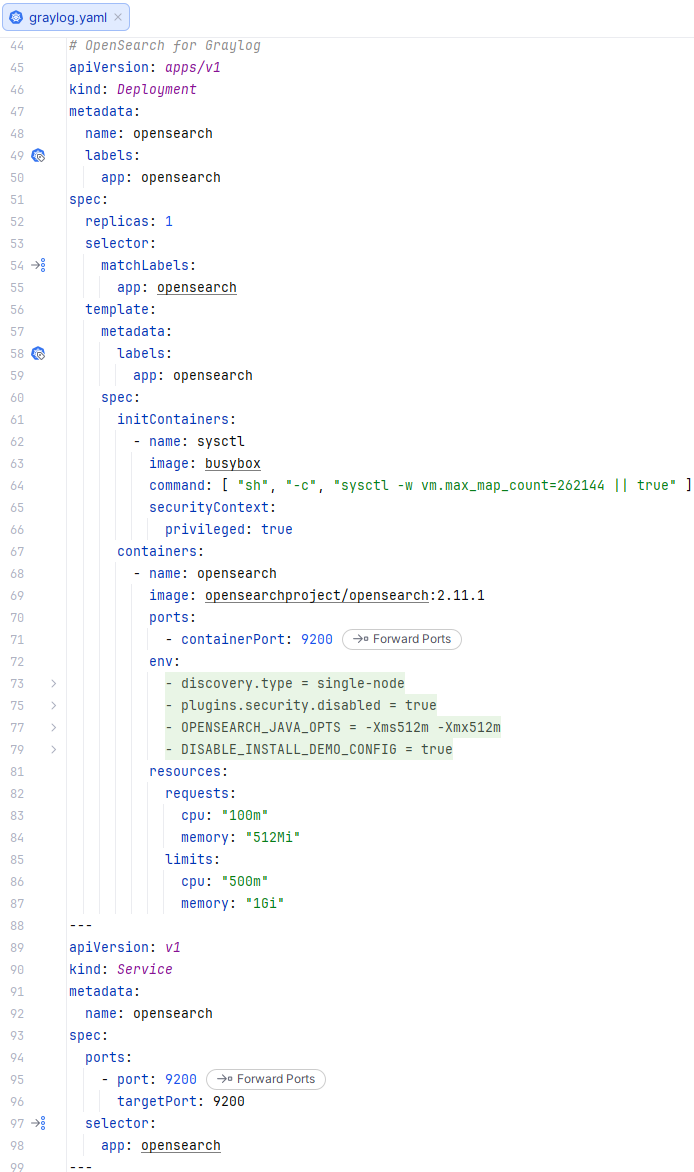
****

Рисунок 46 – Deployment и Service opensearch для Graylog

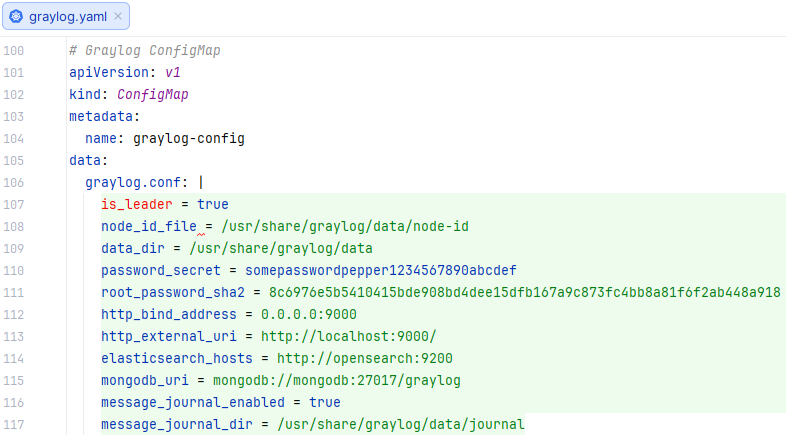
****

Рисунок 47 – ConfigMap для Graylog

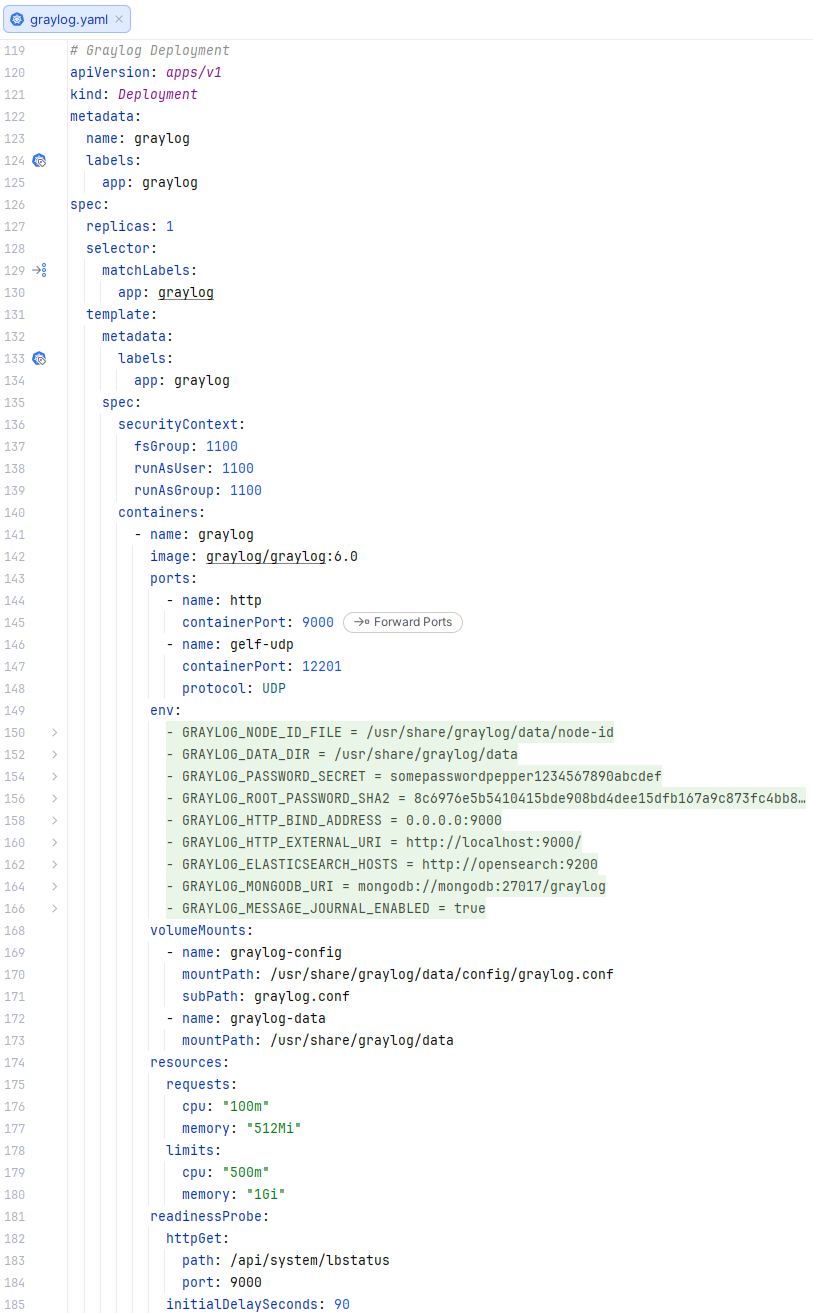
****

Рисунок 48 – Deployment для Graylog

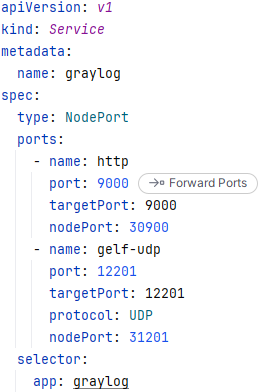
****

Рисунок 49 – Service для Graylog

Файлы для запуска Jaeger представлены на рисунках 50-51.

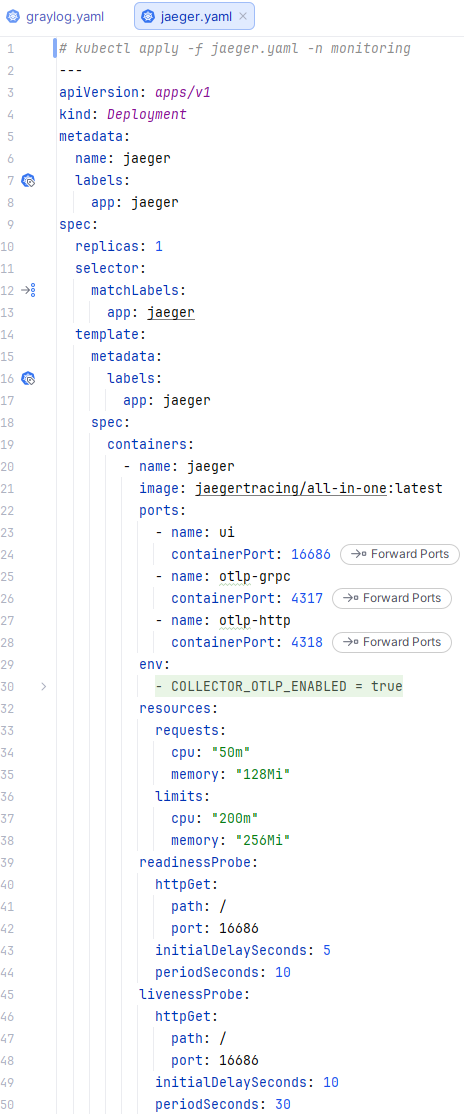
****

Рисунок 50 – Deployment для Jaeger

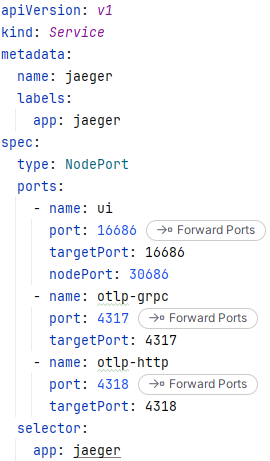
****

Рисунок 51 –Service для Jaeger

Файлы для запуска Kafka представлены на рисунке 52.

****

Рисунок 52 – KafkaNodePool и Kafka

Процесс развертывания кластема minikube со всеми сервисами представлен на рисунках 53-69.

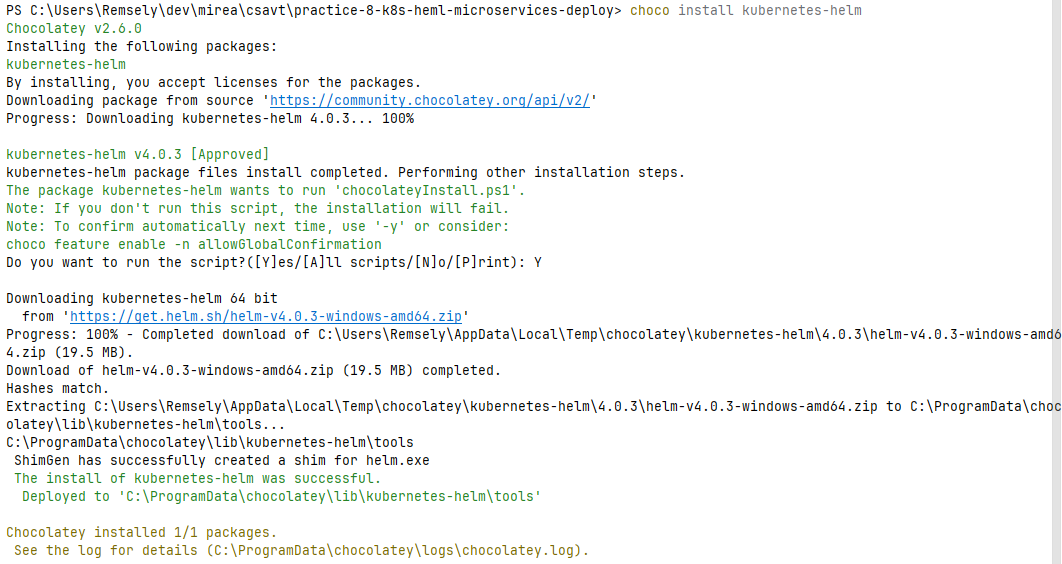
****

Рисунок 53 – Установка Helm

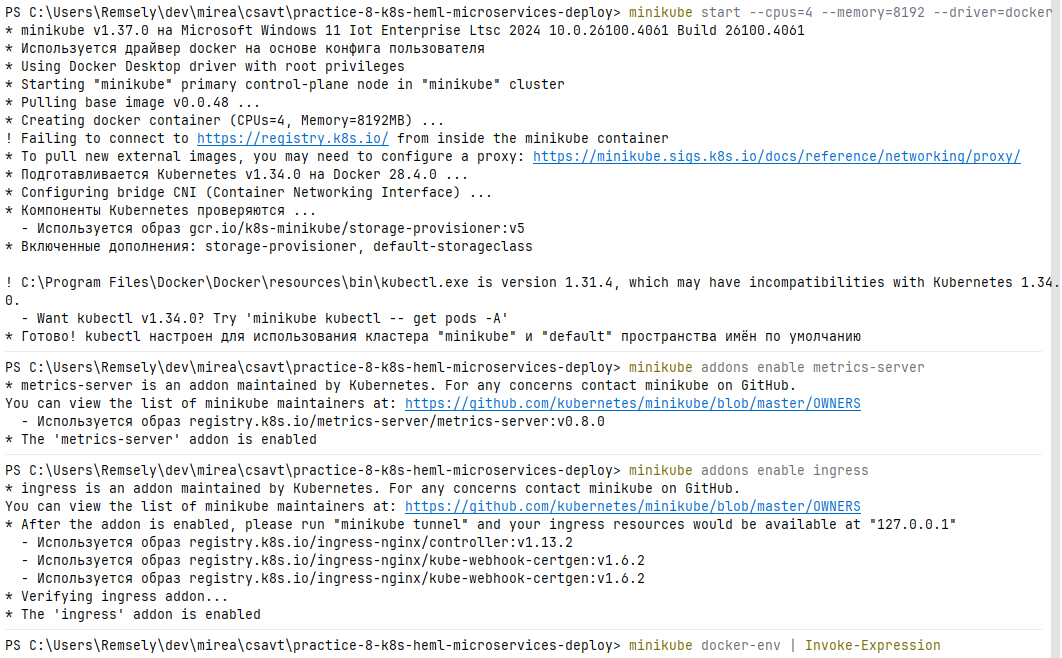
****

Рисунок 54 – Запуск minikube и включение аддонов

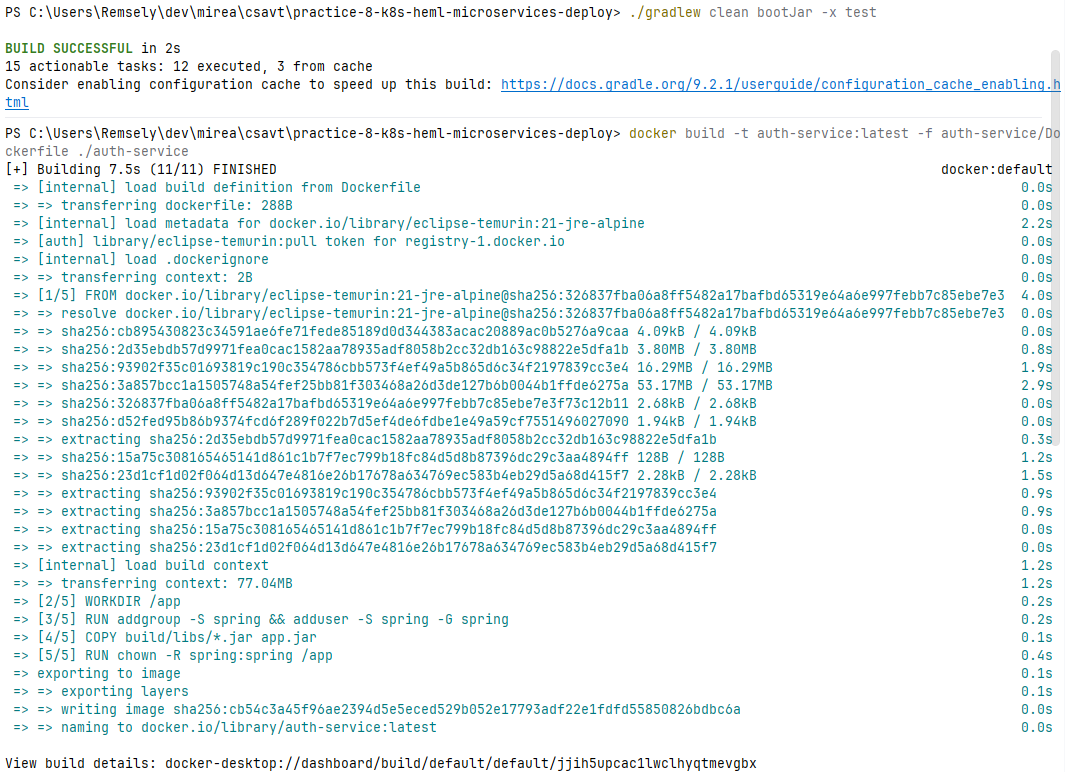
****

Рисунок 55 – Сборка образов сервисов (1/2)

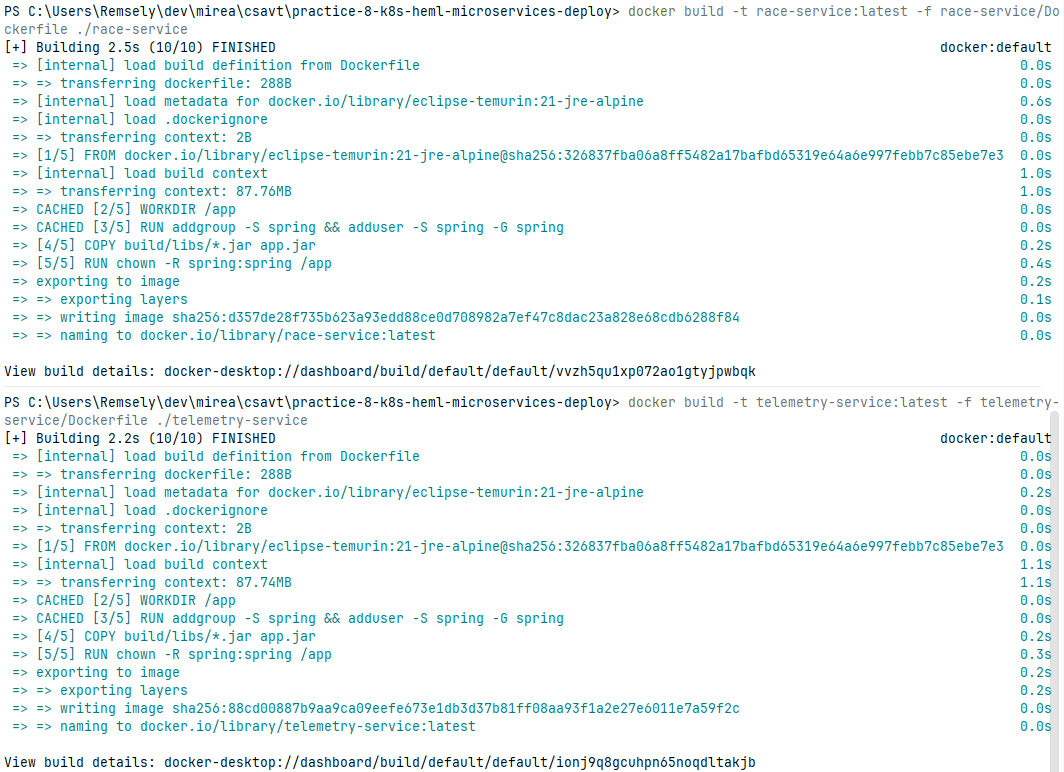
****

Рисунок 56 – Сборка образов сервисов (2/2)

****

Рисунок 57 – Добавление Helm репозиториев

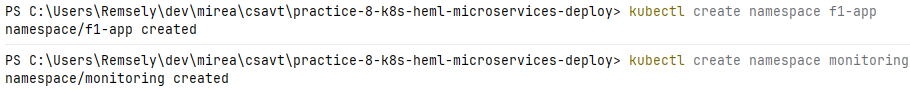
****

Рисунок 58 – Создание неймспейсов

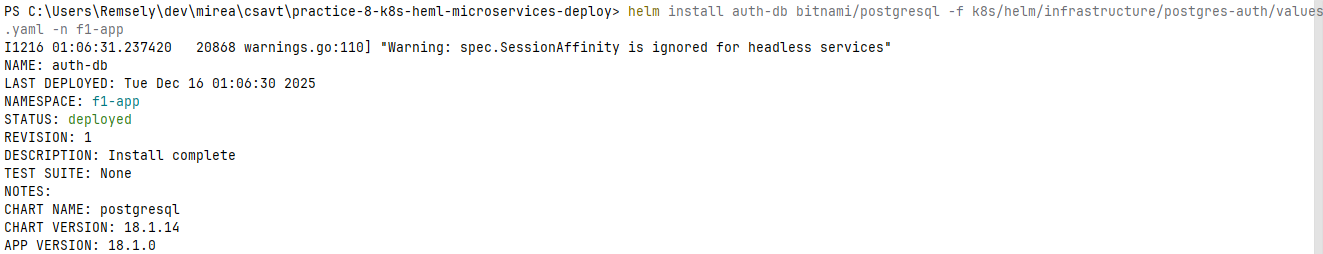
****

Рисунок 59 – Установка БД сервиса авторизации

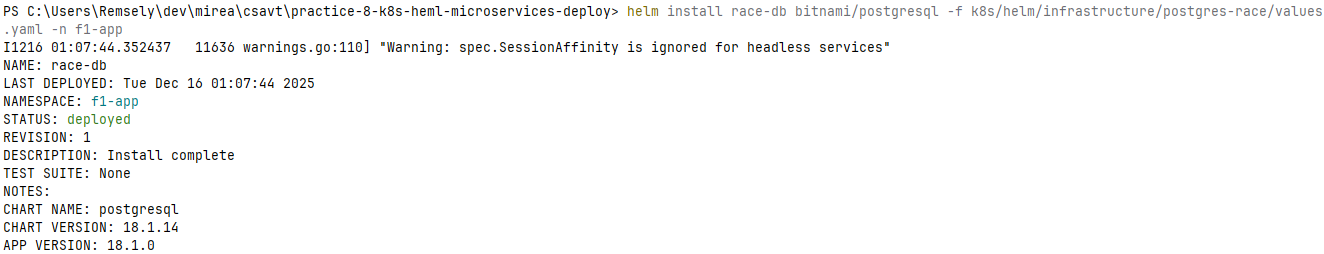
****

Рисунок 60 – Установка БД сервиса гонок

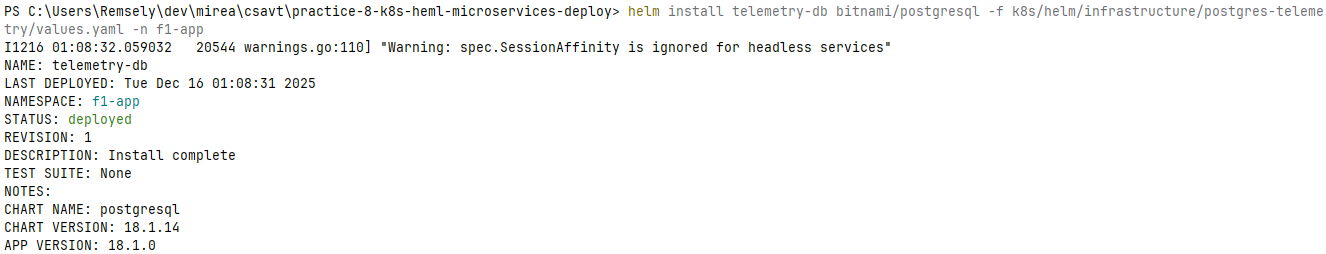
****

Рисунок 61 – Установка БД сервиса телеметрии

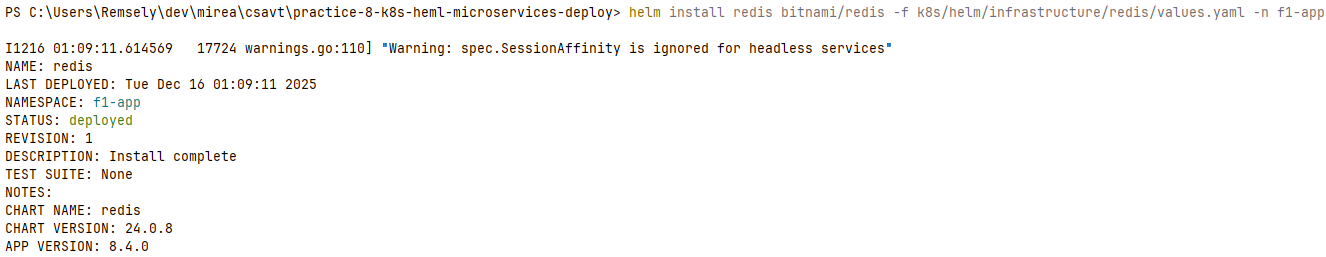
****

Рисунок 62 – Установка Redis для хранения токенов

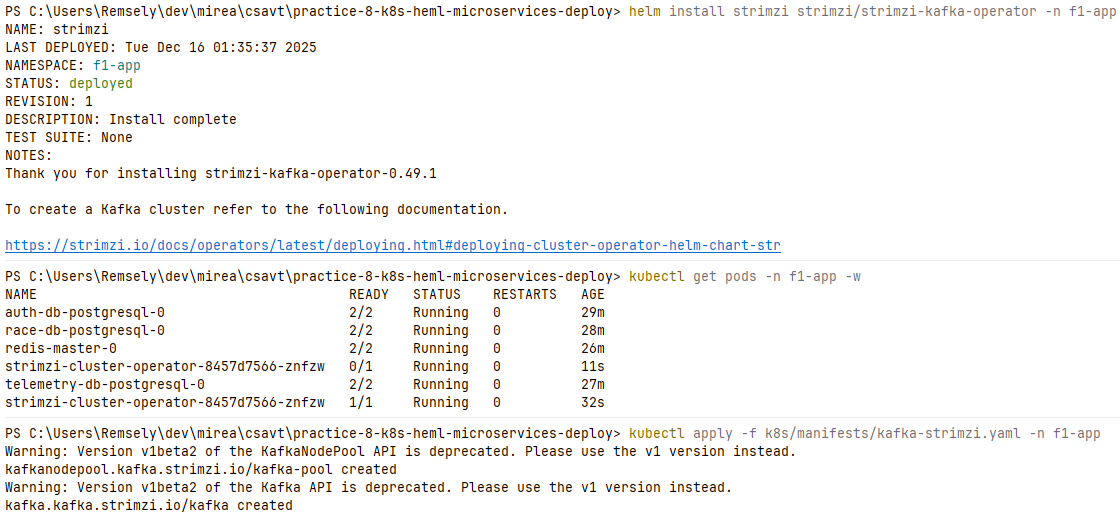
****

Рисунок 63 – Установка Kafka и ее окружения

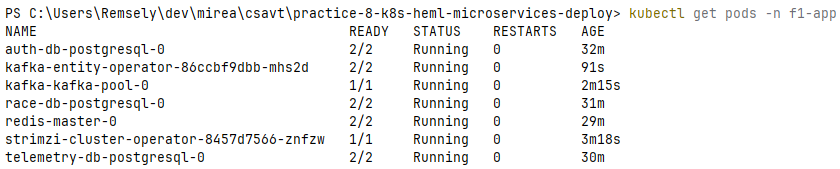
****

Рисунок 64 – Проверка статусов инфраструктурных подов

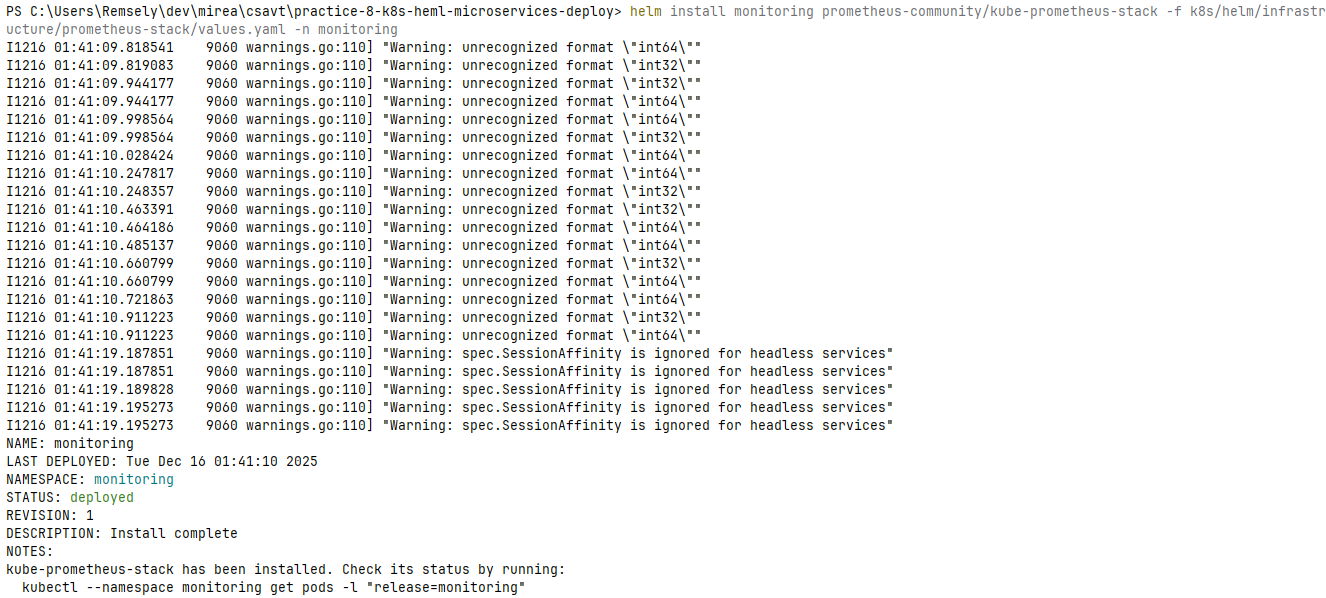
****

Рисунок 65 – Установка Prometheus и его окружения

****

Рисунок 66 – Применение манифестов Jaeger и Graylog

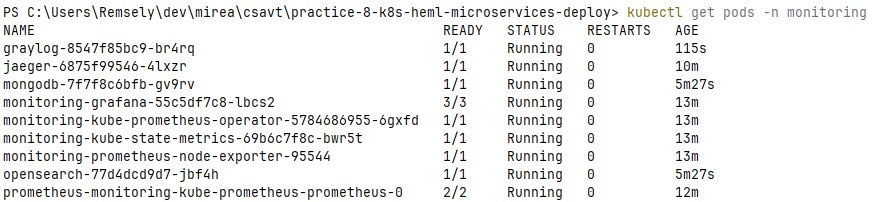
****

Рисунок 67 – Проверка статусов всех под в неймспейсе monitoring

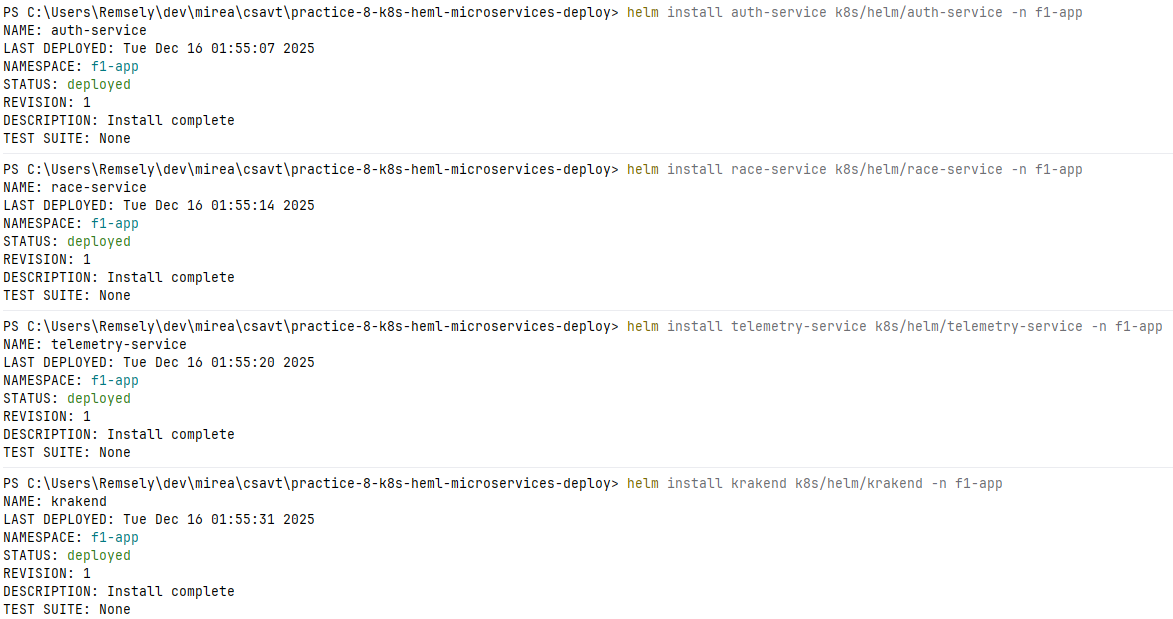
****

Рисунок 68 – Установка основных сервисов и API Gateway

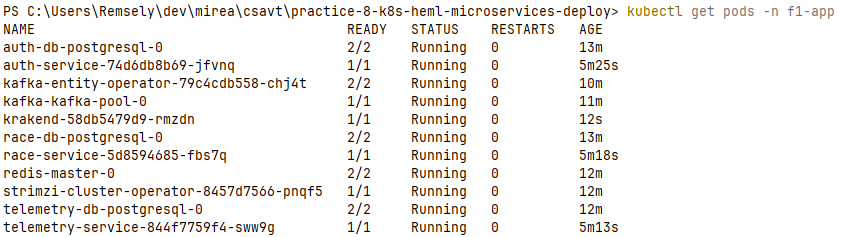
****

Рисунок 69 – Проверка статусов всех под в неймспейсе f1-app

Проверка работоспособности авторизации представлена на рисунках 70-73.

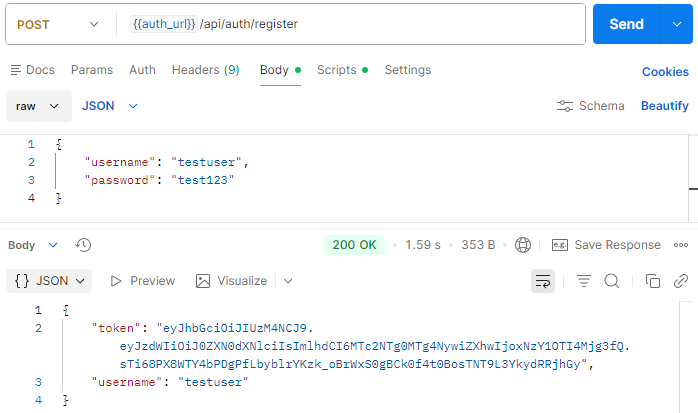
****

Рисунок 70 – Регистрация пользователя

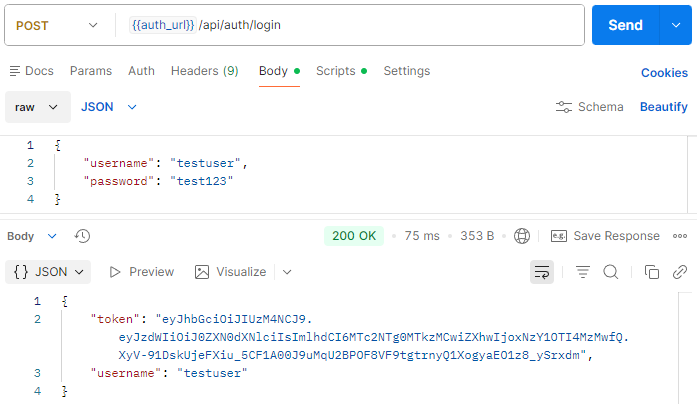
****

Рисунок 71 – Авторизация пользователя

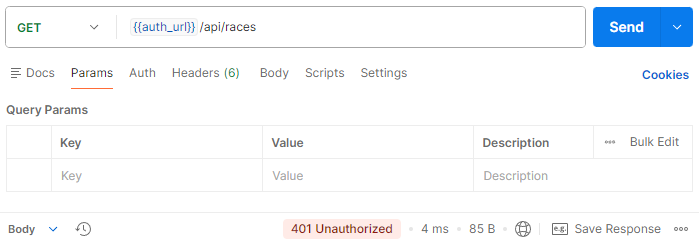
****

Рисунок 72 – Запрос к сервису гонок без токена авторизации

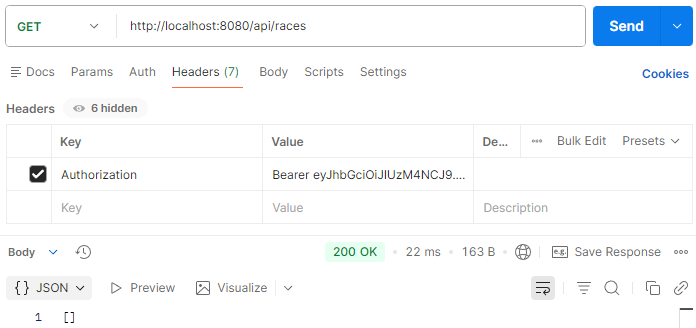
****

Рисунок 73 – Запрос к сервису гонок с токеном авторизации

Проверка работоспособности окружения для мониторинга представлена на рисунках 74-78.

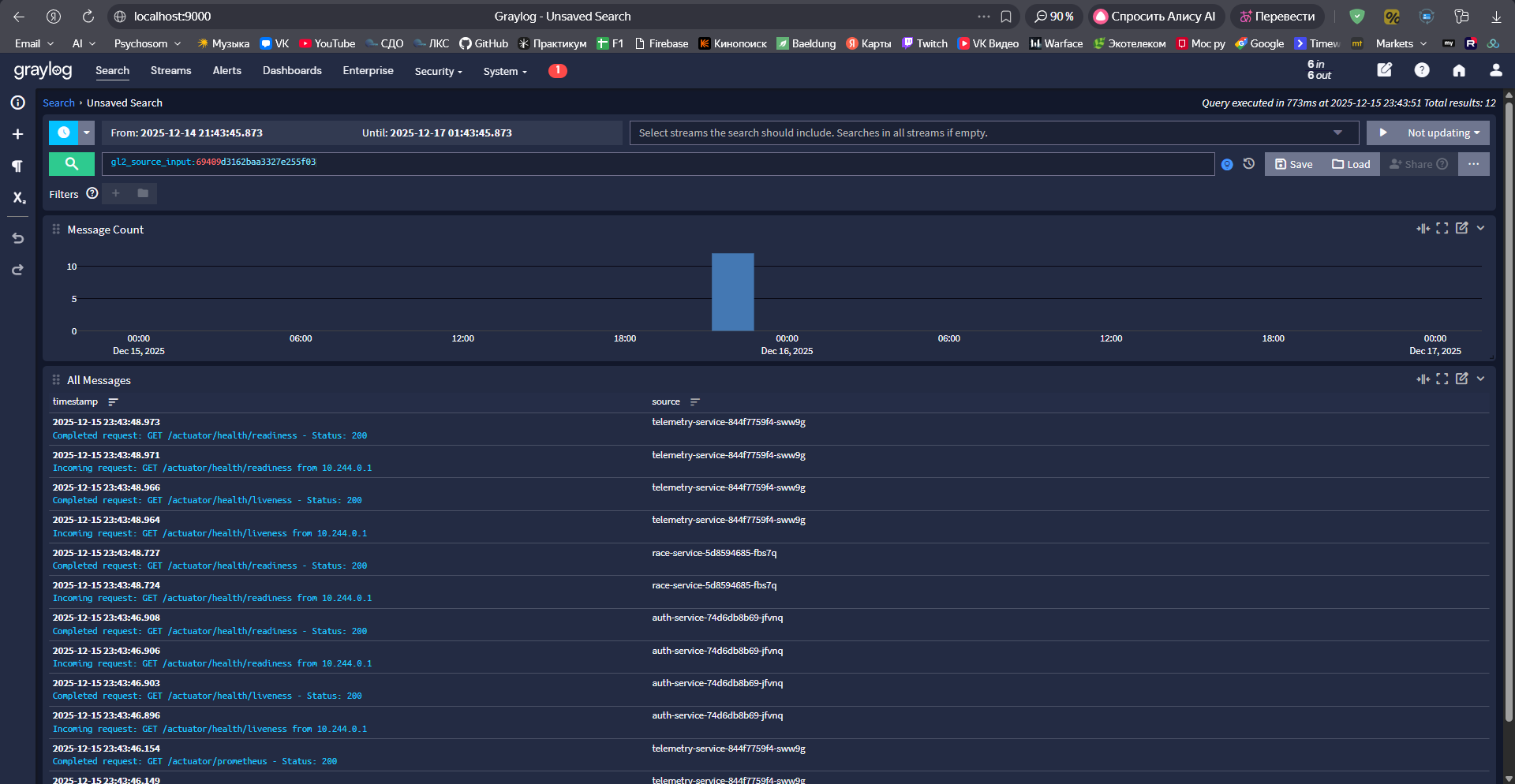
****

Рисунок 74 – Логи в интерфейсе Graylog

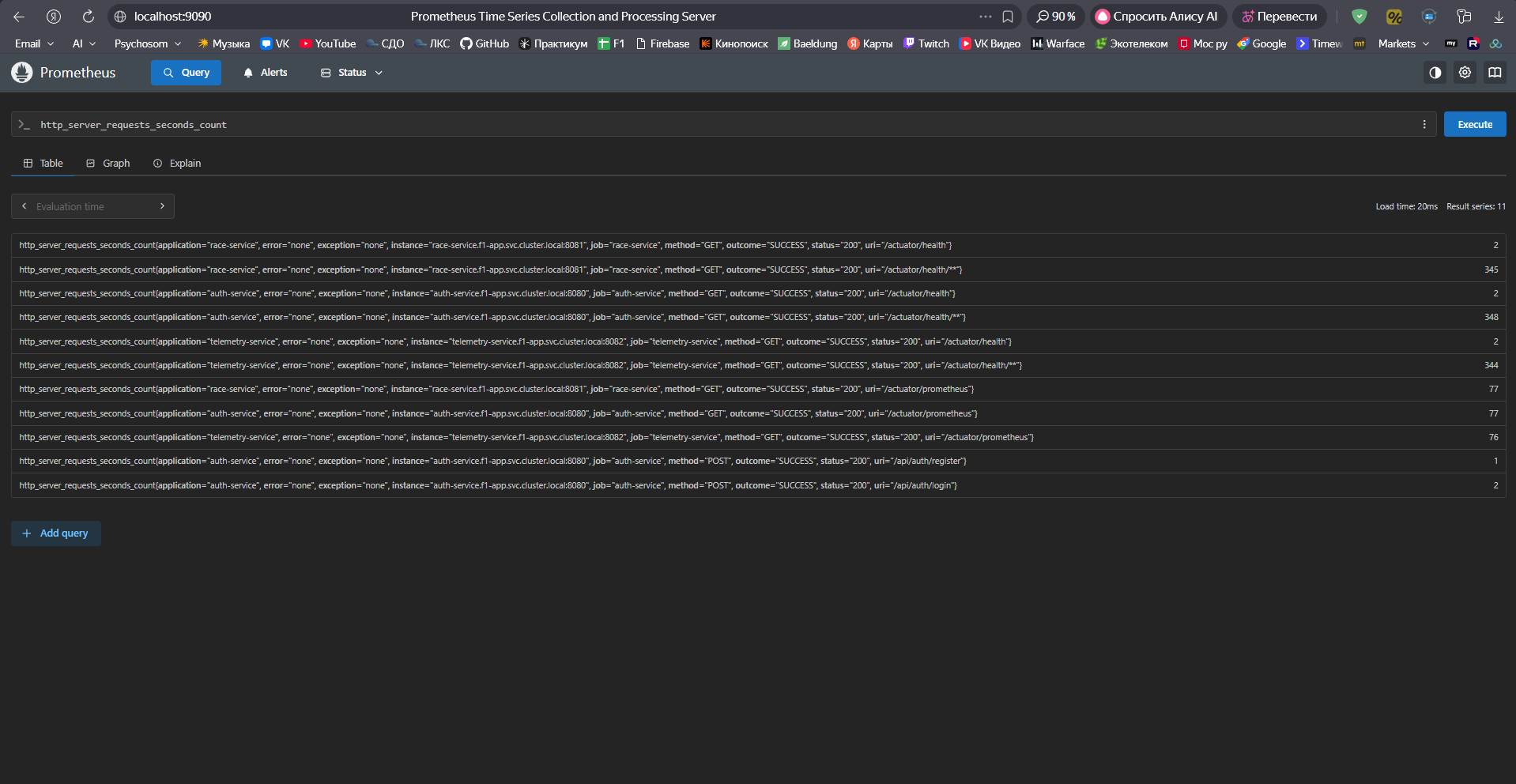
****

Рисунок 75 – Сообщения метрик в интерфейсе Prometheus

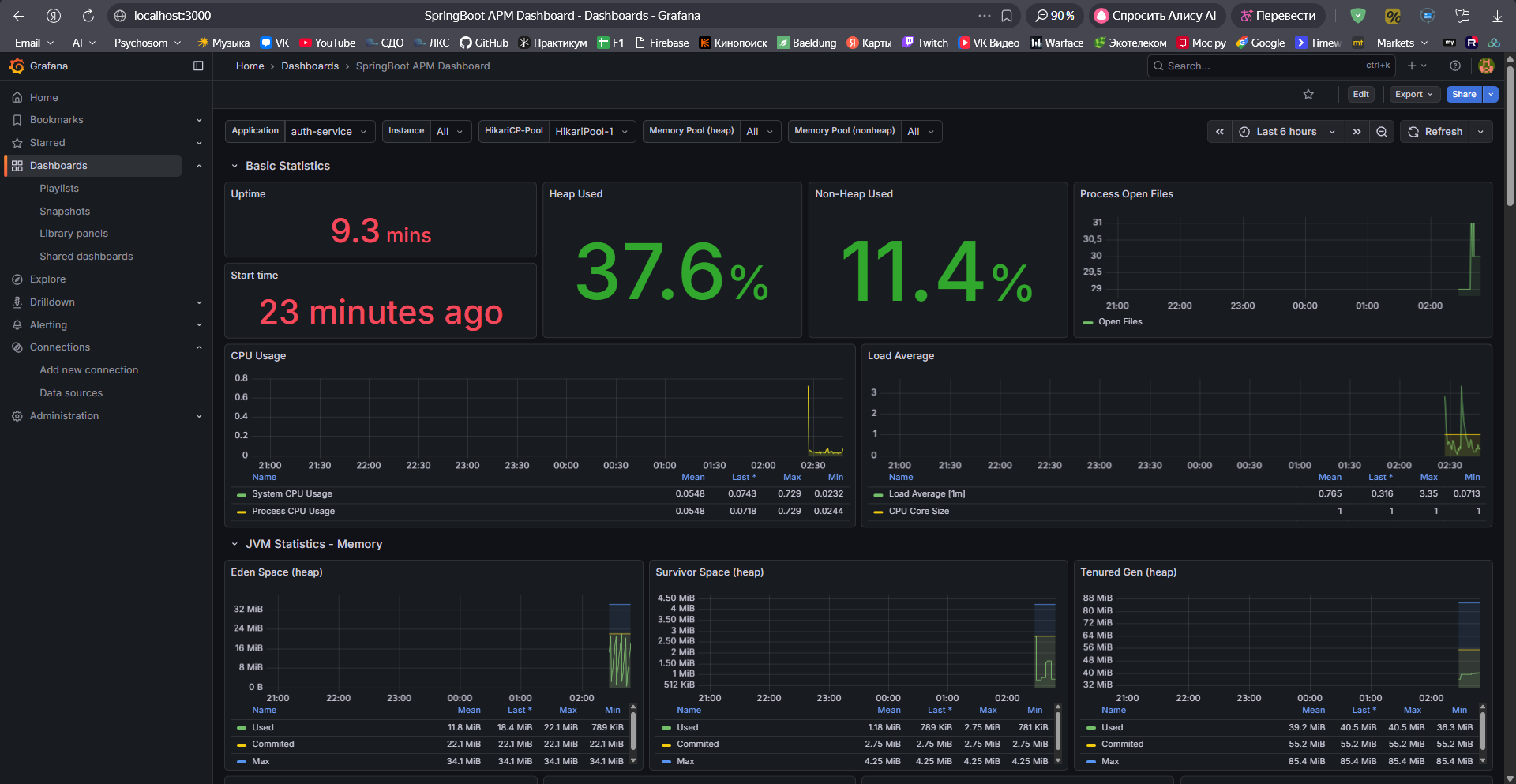
****

Рисунок 76 – Spring Boot дашборд в интерфейсе Grafana

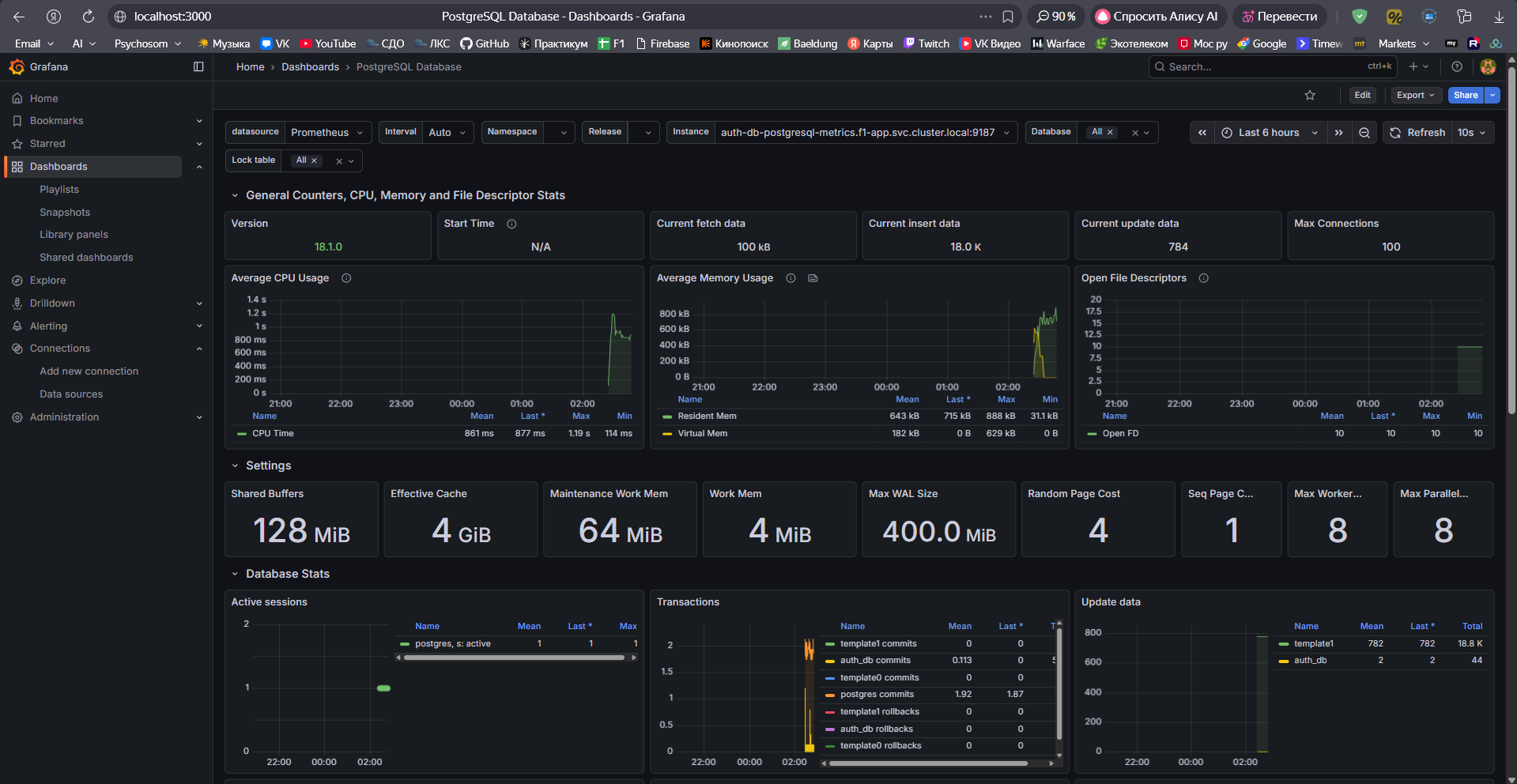
****

Рисунок 77 – PostgreSQL дашборд в интерфейсе Grafana

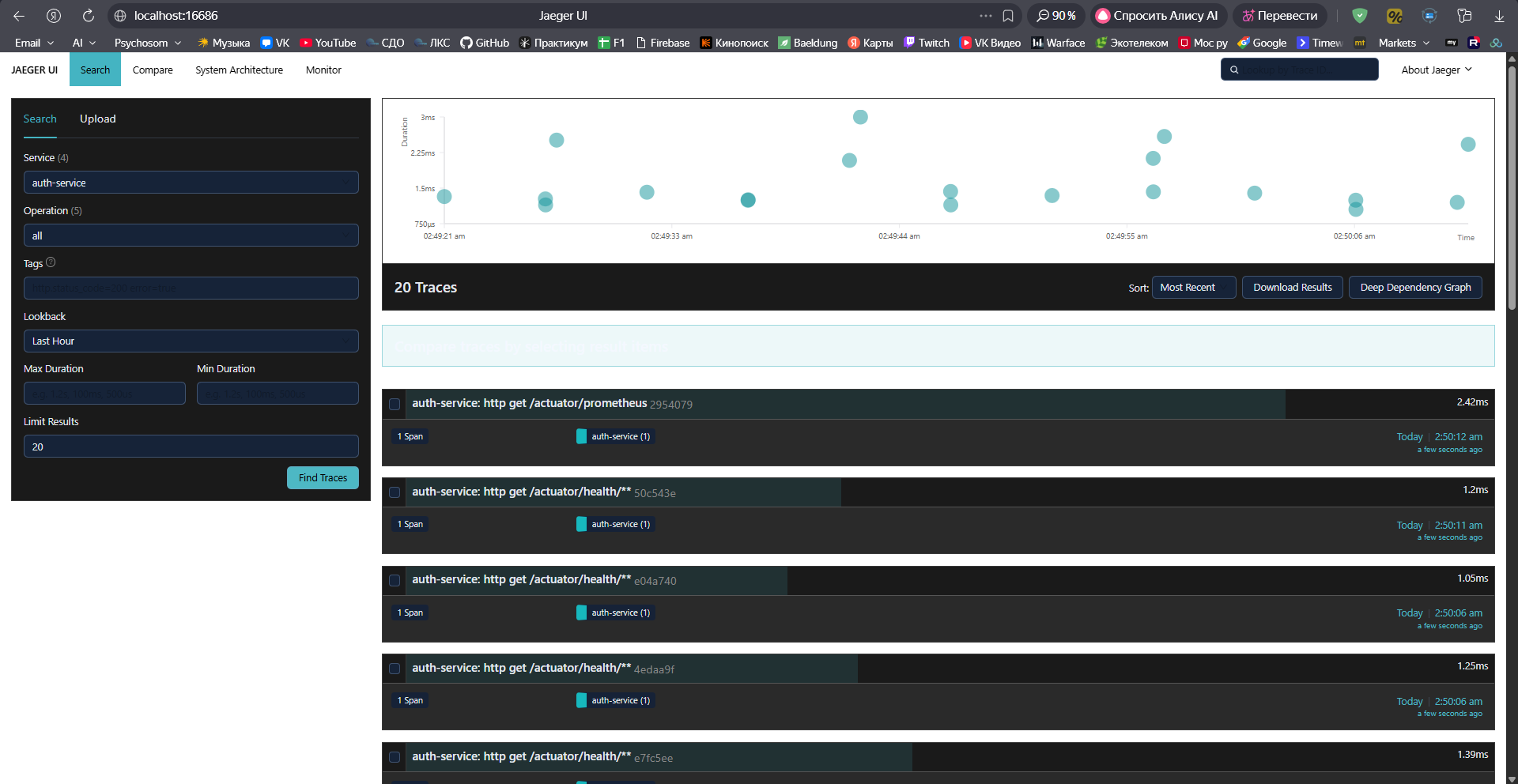
****

Рисунок 78 – Jaeger UI

Демонстрация работы кластера представлена на рисунках 79-85.

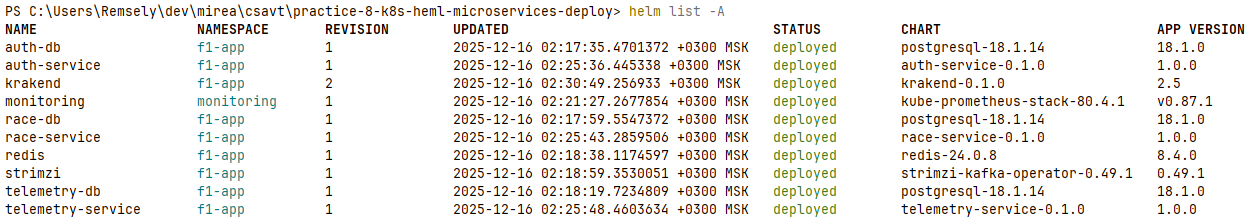
****

Рисунок 79 – Все сущности Helm

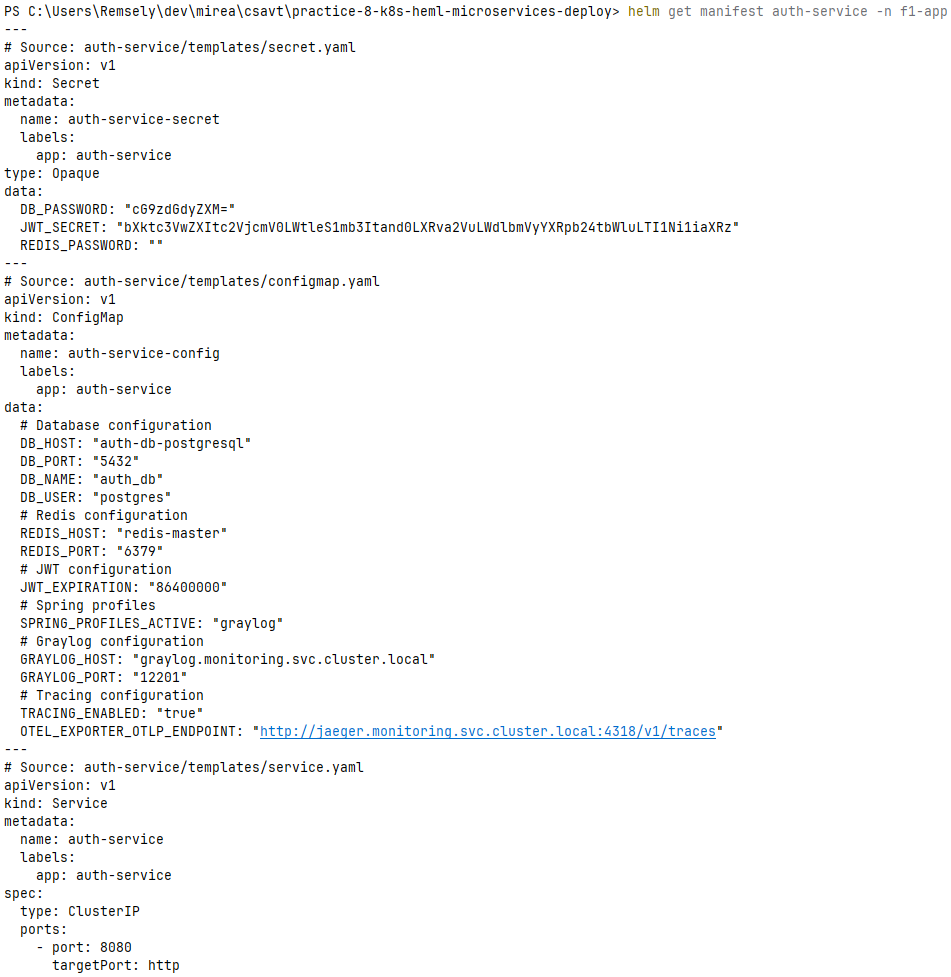
****

Рисунок 80 – Манифест сервиса авторизации

****

Рисунок 81 – Настройки проб в сервисе авторизации

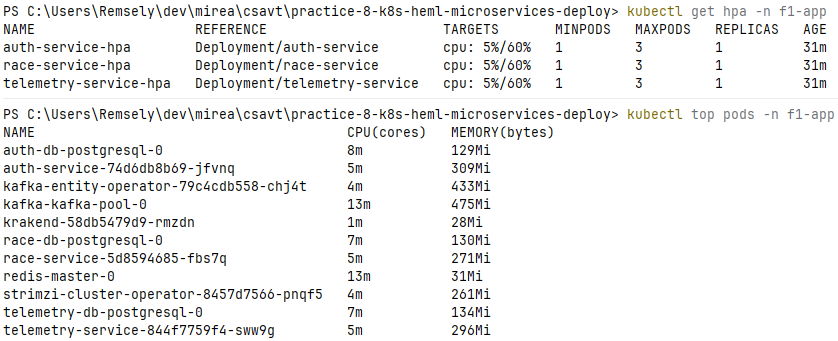
****

Рисунок 82 – Конфигурации управления ресурсами и нагрузкой

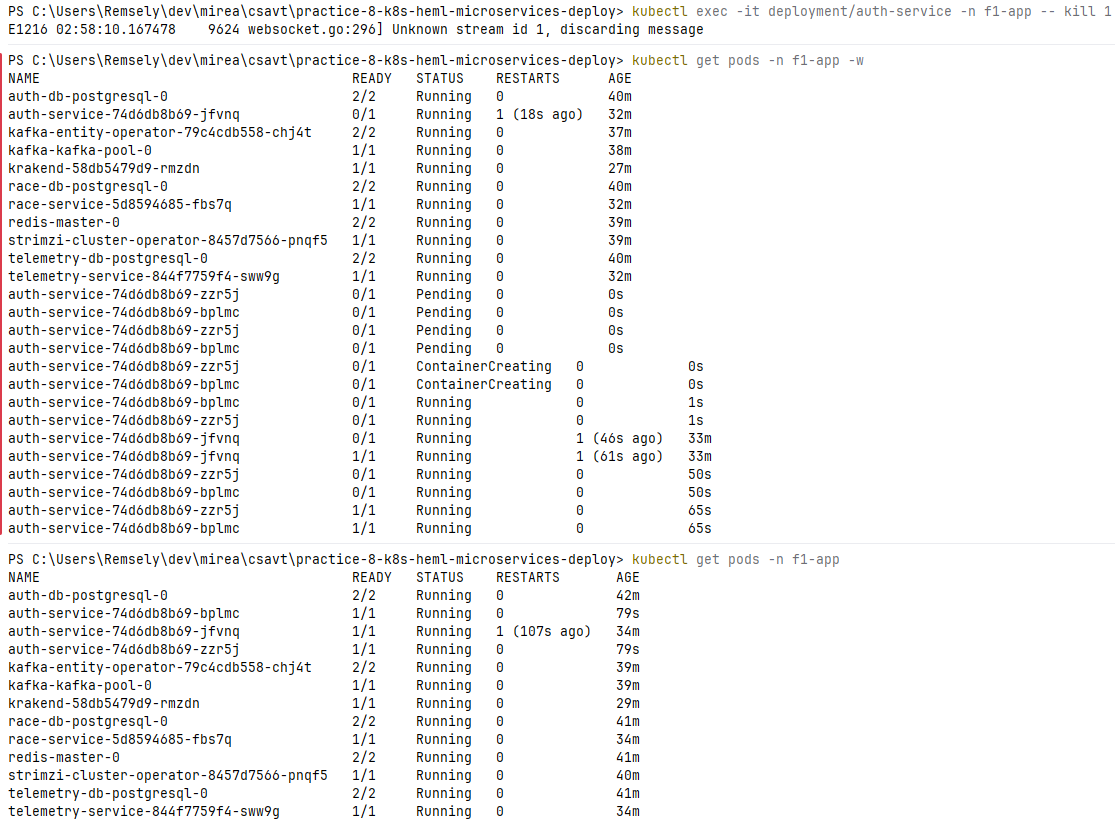
****

Рисунок 83 – Проверка автоматического перезапуска сервиса авторизации

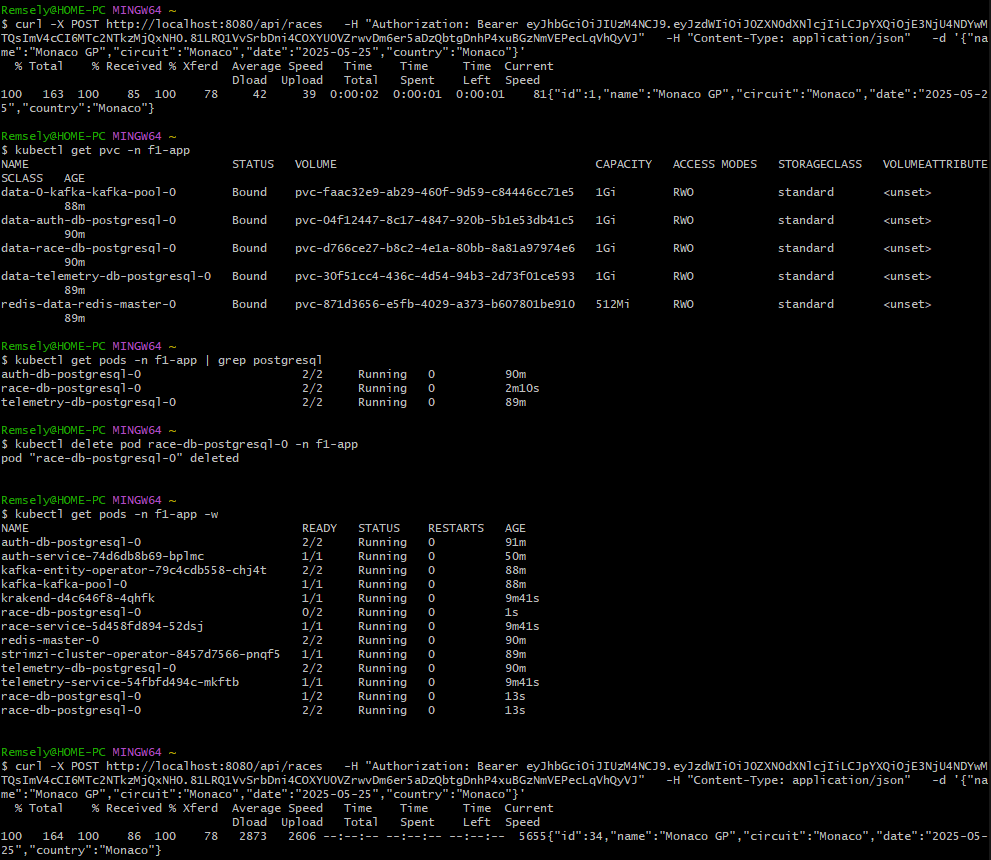
****

Рисунок 84 – Проверка сохранения данных в Persistent Volume

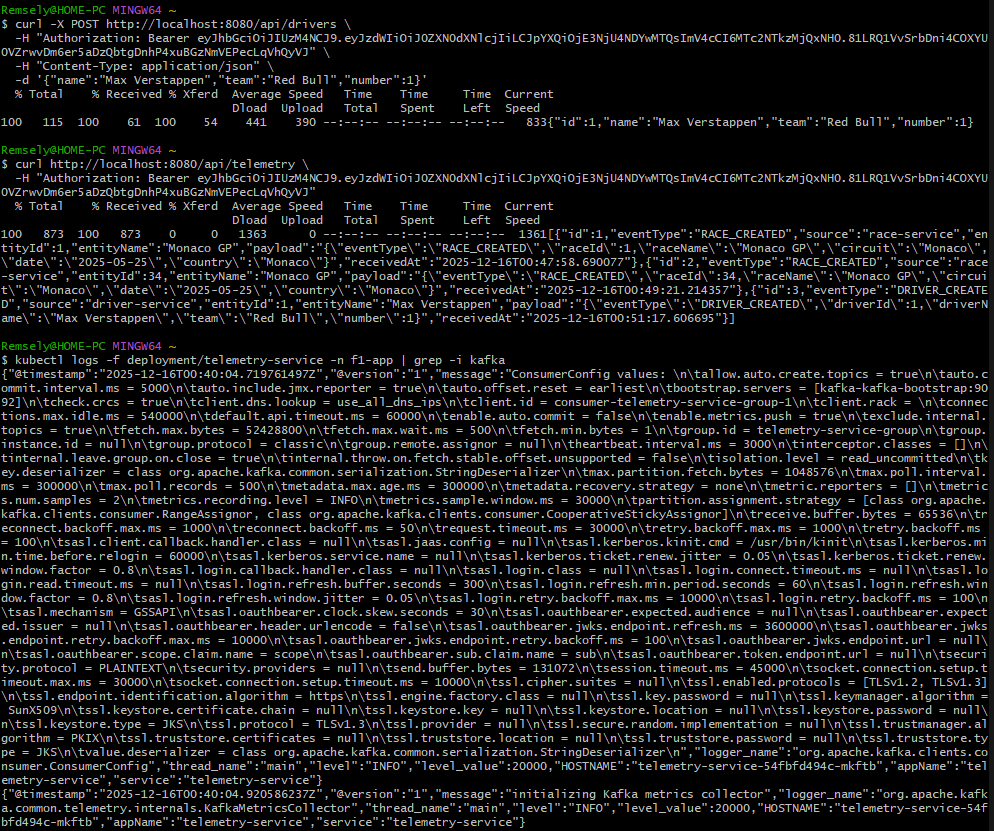
****

Рисунок 85 – Демонстрация работы Kafka

**Вывод**

В результате выполнения данной практической было выполнено развертывание полноценной микросервисной системы с окружением для мониторинга при помощи пакетного менеджера Helm в Kubernetes.

**Ответы на вопросы к практической работе**

**1. Назовите плюсы и минусы использования helm. Какие форматы конфигурационных файлов он поддерживает? Какие существуют аналоги?**

Плюсы: упрощение деплоя, версионирование релизов, переиспользование чартов. Минусы: сложность для простых задач, трудная отладка шаблонов. Форматы: YAML с Go-templates. Аналоги: Kustomize, Jsonnet, Kapitan.

**2. Почему для микросервисного взаимодействия используется шина сообщений? Какие аналоги для микросервисного взаимодействия существуют? Каковы их преимущества и недостатки?**

Обеспечивает асинхронность, слабую связанность и надежность доставки. Аналоги: REST (простой, но синхронный), gRPC (быстрый, но сложнее), GraphQL (гибкий, но избыточный). Плюсы шины: устойчивость к сбоям. Минусы: сложная отладка, eventual consistency.

**3. Для чего используется persistent volume в БД? Какие преимущества и недостатки у паттерна "1 БД на 1 сервис"?**

Сохраняет данные при перезапуске подов. Плюсы паттерна: изоляция данных, независимое масштабирование. Минусы: сложность транзакций между сервисами, дублирование данных, больше операционных затрат.

**4. Почему для чтения данных из БД системами мониторинга создается отдельный пользователь? К чему может привести использование аккаунта с лишними правами?**

По принципу минимальных привилегий. Риски: случайное удаление данных, компрометация дает полный контроль над БД, усложнение аудита.

**5. Какой функционал предоставляет KrakenD? Какие существуют аналоги?**

API Gateway: маршрутизация, агрегация ответов, аутентификация, rate limiting, кеширование. Аналоги: Kong, Nginx, Traefik, Envoy, Istio.

**Список источников информации**

* 1. K8S для начинающих. Первая часть — Текст: электронный [сайт]. — URL: https://habr.com/ru/post/589415/
  2. KrakenD Designer | KrakenD — Текст: электронный [сайт]. — URL: <https://designer.krakend.io/>
  3. Operator for Kubernetes | Jaeger — Текст: электронный [сайт]. — URL: <https://www.jaegertracing.io/docs/1.24/operator/>
  4. Kafka 3.3 Documentation | Kafka — Текст: электронный [сайт]. — URL: <https://kafka.apache.org/documentation/>
  5. Helm Docs | Helm — Текст: электронный [сайт]. — URL: https://helm.sh/docs/ KrakenD Documentation |
  6. KrakenD — Текст: электронный [сайт]. — URL: <https://www.krakend.io/docs/overview/>