Разработка ИИ-приложений в Kubernetes

Студент группы М24-535 Агаев Р.Э.

Цель проекта

Целью данного проекта является разработка и локальное развертывание контейнерного Java-приложения с элементами ИИ для анализа тональности текста в среде Kubernetes с использованием Minikube.

В рамках проекта были решены следующие задачи:

- разработка REST API для анализа тональности текста;
- контейнеризация приложения с помощью Docker;
- развертывание приложения в Kubernetes с балансировкой нагрузки и автоскейлингом;
- настройка мониторинга с использованием Prometheus и Grafana;
- анализ современных тенденций в области ИИ и контейнеризации на основе научных статей arXiv.

Программные и инфраструктурные компоненты

Программные и инфраструктурные компоненты

- Язык программирования: Java SE
- Контейнеризация: Docker
- Оркестрация: Kubernetes (Minikube)
- CI/запуск образов: локально
- Мониторинг: Prometheus, Grafana
- Масштабирование: Horizontal Pod Autoscaler
- Анализ научных трендов: arXiv.org

Среда выполнения

Локальная инфраструктура

ОС: MacOS

Minikube (2 ноды):

- CPU: 4
- RAM: 8 GB
- minikube version: v1.37.0

Kubernetes версии:

- Client Version: v1.34.2
- Kustomize Version: v5.7.1
- Server Version: v1.34.0

Архитектура решения

Развёрнутое решение представляет собой Kubernetes-кластер, внутри которого функционируют следующие компоненты:

- Ingress, выполняющий маршрутизацию HTTP-запросов к приложению;
- Service, обеспечивающий доступ к приложению внутри кластера;
- Deployment с несколькими репликами Java-приложения;
- Horizontal Pod Autoscaler, отвечающий за автоматическое масштабирование подов;
- Prometheus, собирающий метрики приложения и ресурсов;
- Grafana, предоставляющая интерфейс визуализации и анализа метрик.

Реализация приложения

REST API

Эндпоинт:

GET /api/v1/sentiment?text=...

Особенности Java-приложения

- REST API без использования тяжёлых фреймворков
- Упрощённая логика анализа тональности
- Экспорт JVM и application-метрик
- Минимальная архитектура (один основной класс)

Развертывание приложения

Реализация и Контейнеризация:

Многоэтапный Dockerfile (сборка + runtime), оптимизированный размер образа, быстрое развёртывание в Kubernetes

Развертывание в Kubernetes:

- Deployment с несколькими репликами
- Ограничения по ресурсам CPU и памяти
- Readiness и liveness проверки

Сетевая конфигурация

- Service для доступа внутри кластера
- Ingress для маршрутизации внешних запросов

Масштабирование и мониторинг

- HPA по метрикам загрузки ресурсов
- Prometheus — сбор метрик
- Grafana — визуализация и анализ

Контейнеризация и Kubernetes

Docker-образ

- Многоэтапная сборка
- Отдельный этап сборки и runtime
- Минимальный базовый образ

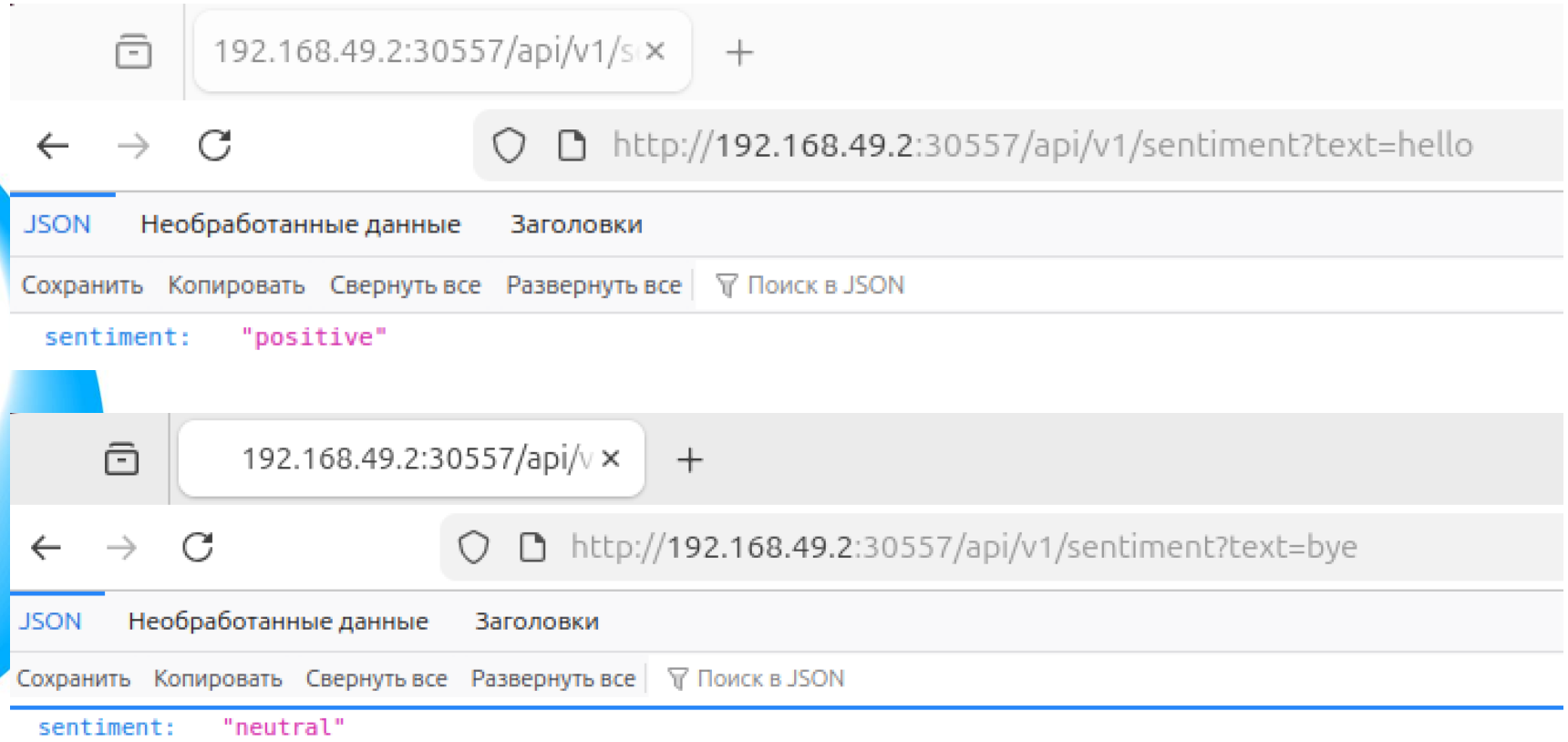
Kubernetes Deployment

- Несколько реплик приложения
- Ограничения ресурсов
- Проверки готовности и работоспособности

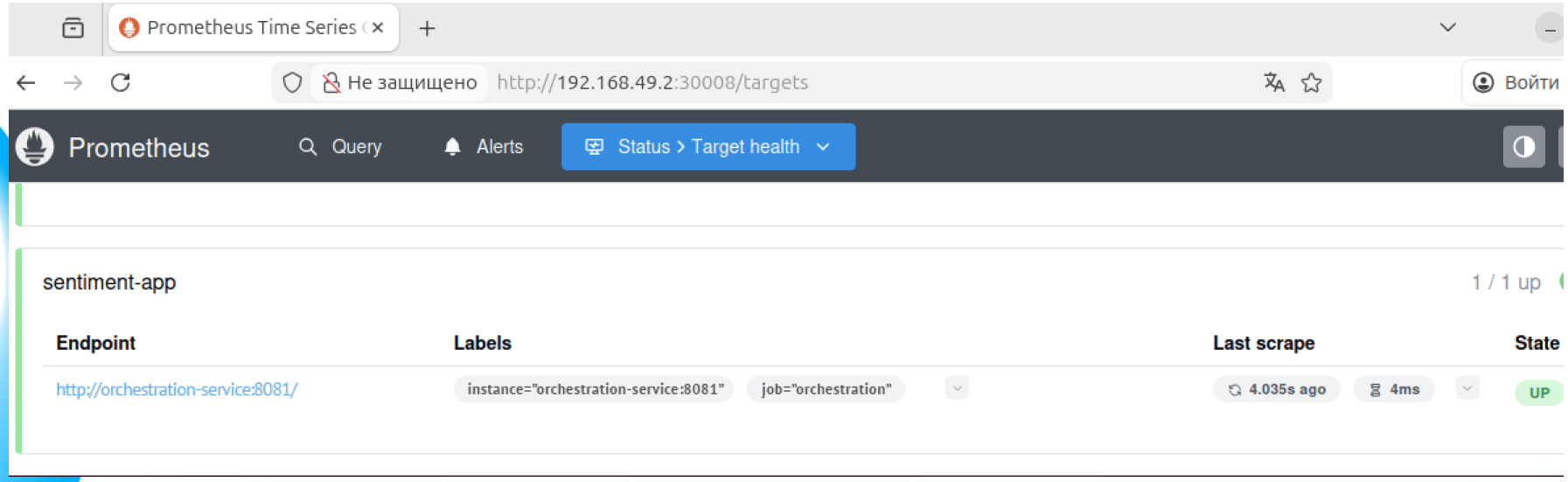
Service и Ingress

- Service — доступ внутри кластера
- Ingress — обработка внешних HTTP-запросов

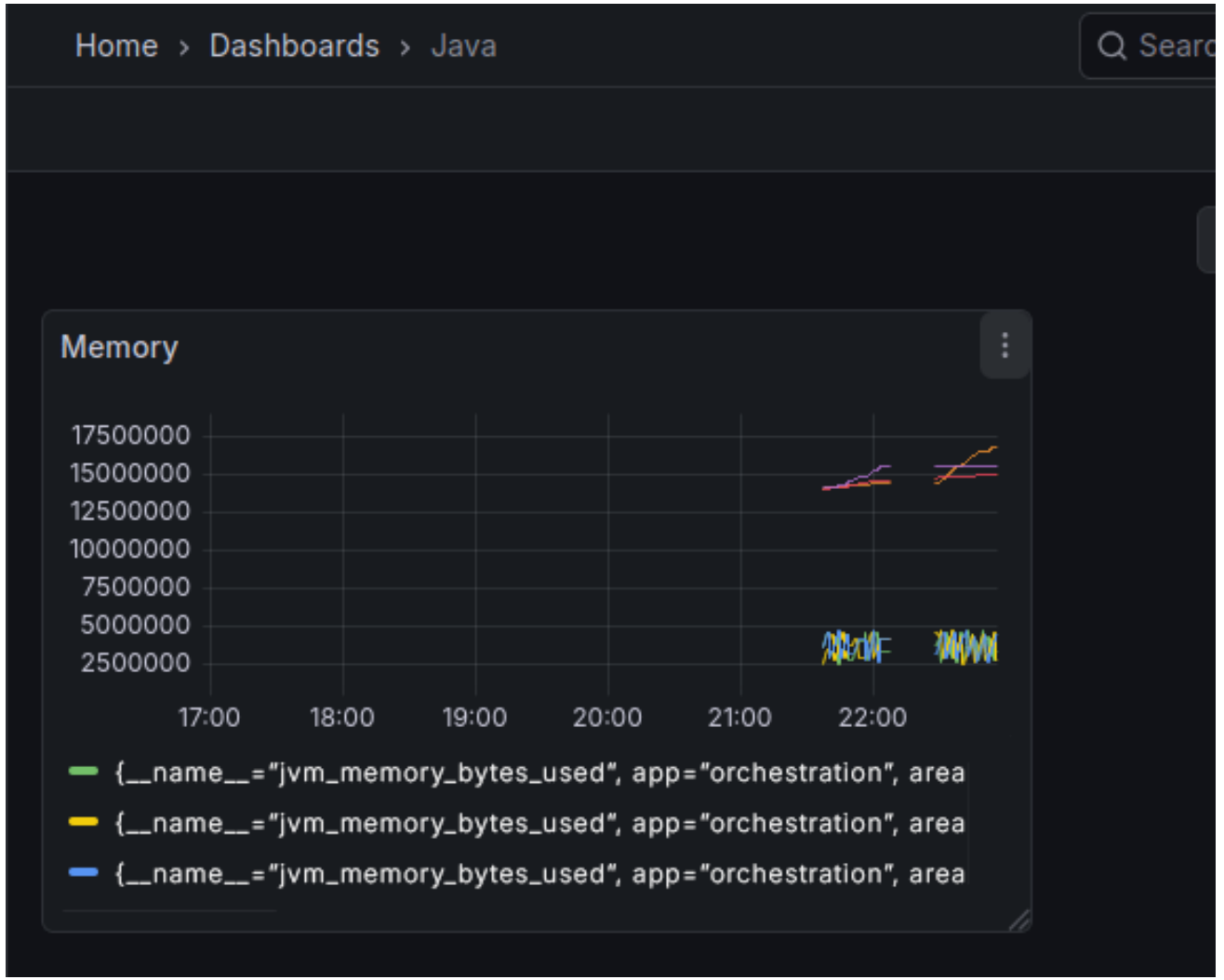
Проверки


192.168.49.2:30557/api/v1/s
http://192.168.49.2:30557/api/v1/sentiment?text=hello
JSON
Необработанные данные
Заголовки
Сохранить
Копировать
Свернуть все
Развернуть все
Поиск в JSON
sentiment: "positive"
192.168.49.2:30557/api/v
http://192.168.49.2:30557/api/v1/sentiment?text=bye
JSON
Необработанные данные
Заголовки
Сохранить
Копировать
Свернуть все
Развернуть все
Поиск в JSON
sentiment: "neutral"

Мониторинг


Prometheus Time Series
Не защищено
http://192.168.49.2:30008/targets
Войти
Prometheus
Query
Alerts
Status > Target health
sentiment-app
1 / 1 up
Endpoint
Labels
Last scrape
State
http://orchestration-service:8081/
instance="orchestration-service:8081"
job="orchestration"
4.035s ago
4ms
UP

Дашборд используемой памяти



Достигнутые результаты

В рамках данной работы были достигнуты следующие результаты:

- разработано Java-приложение с REST API для анализа тональности текста с использованием mock-логики;
- выполнена контейнеризация приложения с использованием Docker, при этом размер итогового образа был оптимизирован;
- приложение развернуто в Kubernetes-кластере Minikube с использованием Deployment, Service, Ingress и Horizontal Pod Autoscaler;
- настроен сбор и визуализация метрик с помощью Prometheus и Grafana;

Исследовательская часть: цель и контекст

Цель исследования

- Изучить современные подходы к использованию Kubernetes для ИИ-приложений
- Рассмотреть архитектурные и инфраструктурные решения для LLM
- Проанализировать тенденции развития агентных ИИ-систем

Актуальность

- Современные ИИ-сервисы требуют масштабируемой инфраструктуры
- Kubernetes всё чаще используется как базовая платформа для ИИ

Выбранные исследования (arXiv)

Рассмотренные работы

- **Containerization in Multi-Cloud Environment (2024)**
  Контейнеризация и оркестрация в мультиоблачных средах
- **An Efficient KV Cache Layer for LLM Inference (2025)**
  Оптимизация инференса LLM за счёт кеширования
- **Evolution of AI in Education: Agentic Workflows (2024)**
  Архитектуры многоагентных ИИ-систем в образовании

Тренд 1: LLM и Kubernetes

Основные наблюдения

- Kubernetes используется как платформа для инференса LLM
- Производительность повышается за счёт инфраструктурных решений
- Кеширование снижает задержки и нагрузку на ресурсы

Вывод

- Эффективность LLM-сервисов всё чаще зависит не от модели, а от архитектуры её развертывания.

Тренд 2: Модульные и агентные архитектуры

Основные наблюдения

- Переход от монолитных моделей к набору компонентов
- Использование специализированных агентов
- Повышение устойчивости и гибкости систем

Вывод

- Многоагентный подход упрощает развитие и масштабирование ИИ-систем.

Тренд 3: Контейнеризация и управление инфраструктурой

Основные наблюдения

- Рост требований к автоматизации и мониторингу
- Стандартизация через Kubernetes и CI/CD
- Контейнеризация как основа управляемости

Вывод

- Инфраструктура становится ключевым элементом ИИ-систем.

Связь исследования с практическим проектом

Что применимо в проекте

- Архитектура сервиса соответствует современным подходам
- Возможность добавления кеширования
- Потенциал перехода к модульной архитектуре

Перспективы

- Подключение реальной LLM
- Расширение сервиса новыми компонентами

Итоги исследовательской части

Ключевые выводы

- Kubernetes — де-факто платформа для ИИ-сервисов
- Производительность LLM определяется архитектурой
- Будущее за модульными и агентными системами