|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**К ЛАБОРАТОРНОЙ РАБОТЕ № 5**

**По дисциплине «Автоматизация развертывания и эксплуатации программного обеспечения»**

**Сбор и обработка телеметрии приложений**

Студент ИУ5Ц-91Б  А.В. Рябкин

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель

(Подпись, дата) (И.О.Фамилия)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[План выполнения лабораторной работы: 3](#__RefHeading___Toc212_1356002713)

[Ход выполнения лабораторной работы 4](#__RefHeading___Toc214_1356002713)

[Шаг 1. Создание приложения на Python 4](#__RefHeading___Toc235_1356002713)

[Шаг 2. Установка измерительных инструментов и библиотек OpenTelemetry 4](#__RefHeading___Toc237_1356002713)

[Шаг 4. Развертывание Jaeger 4](#__RefHeading___Toc239_1356002713)

[Шаг 5. Упаковка приложения в Docker-контейнер 4](#__RefHeading___Toc241_1356002713)

[Шаг 6. Переход на экспортер на конечную точку OTLP 5](#__RefHeading___Toc243_1356002713)

[Шаг 7. Сбор метрик с помощью Prometheus 5](#__RefHeading___Toc245_1356002713)

[Шаг 8. Рефакторинг приложения 5](#__RefHeading___Toc247_1356002713)

[Шаг 9. Развертывание Grafana 5](#__RefHeading___Toc249_1356002713)

[Шаг 10. Работа с журналом 5](#__RefHeading___Toc251_1356002713)

[Шаг 11. Loki и Promtail 5](#__RefHeading___Toc253_1356002713)

[Шаг 12. Исследование телеметрии в Grafana 5](#__RefHeading___Toc255_1356002713)

[Задание для самостоятельной проработки 5](#__RefHeading___Toc257_1356002713)

[Контрольные вопросы 5](#__RefHeading___Toc259_1356002713)

## План выполнения лабораторной работы:

Метрики, журналы и трассировки

Шаг 1. Создание приложения на Python

Шаг 2. Установка измерительных инструментов и библиотек OpenTelemetry

Шаг 3. Подключение библиотек для трассировки приложения

Шаг 4. Развертывание Jaeger

Шаг 5. Упаковка приложения в Docker-контейнер

Шаг 6. Переход на экспортер на конечную точку OTLP

Шаг 7. Сбор метрик с помощью Prometheus

Шаг 8. Рефакторинг приложения

Шаг 9. Развертывание Grafana

Шаг 10. Работа с журналом

Шаг 11. Loki и Promtail

Шаг 12. Исследование телеметрии в Grafana

Задание для самостоятельной проработки

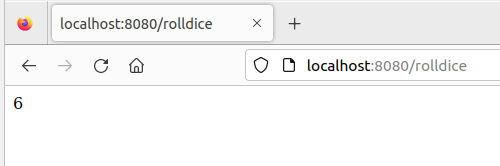
Контрольные вопросы

## Задачи лабораторной работы

## Ход выполнения лабораторной работы

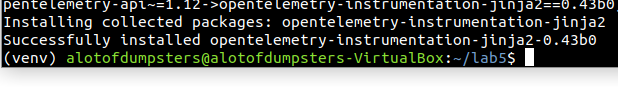
### Шаг 1. Создание приложения на Python

Настроил виртуальную среду выполнения venv, установил фласк, сделал апп, результат работы:



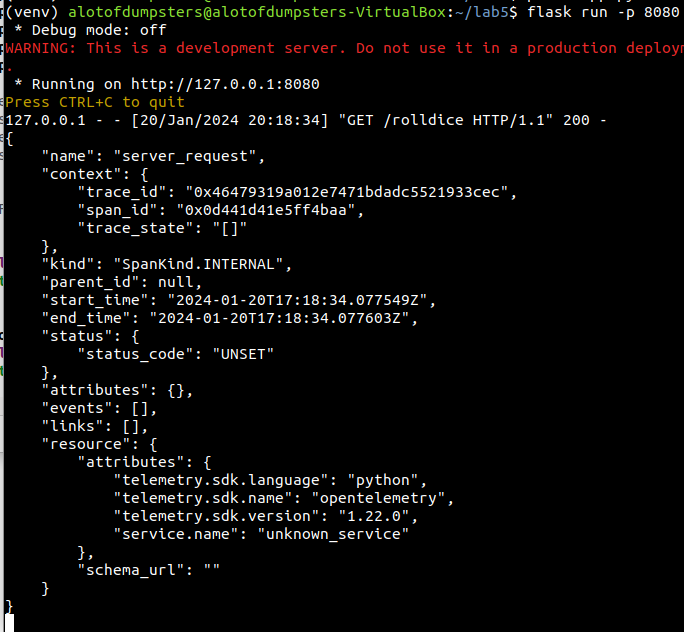
### Шаг 2. Установка инструментов и библиотек OpenTelemetry

Успешно установили opentelemetry и ее библиотеки для flask.

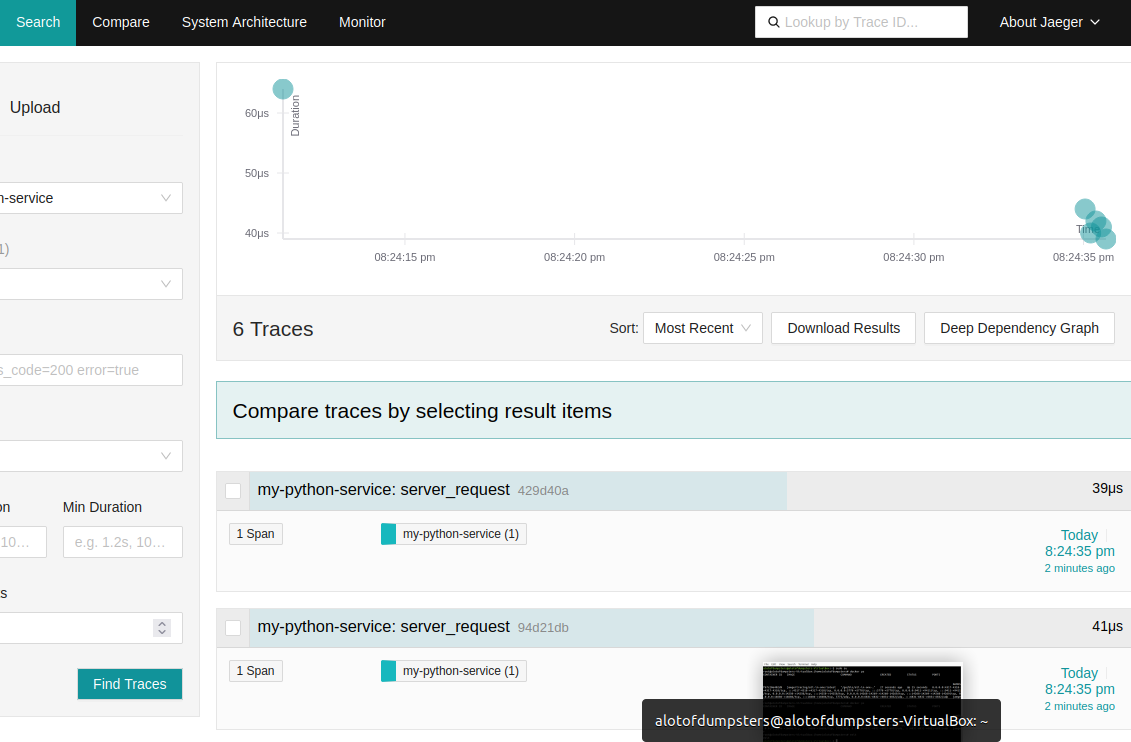


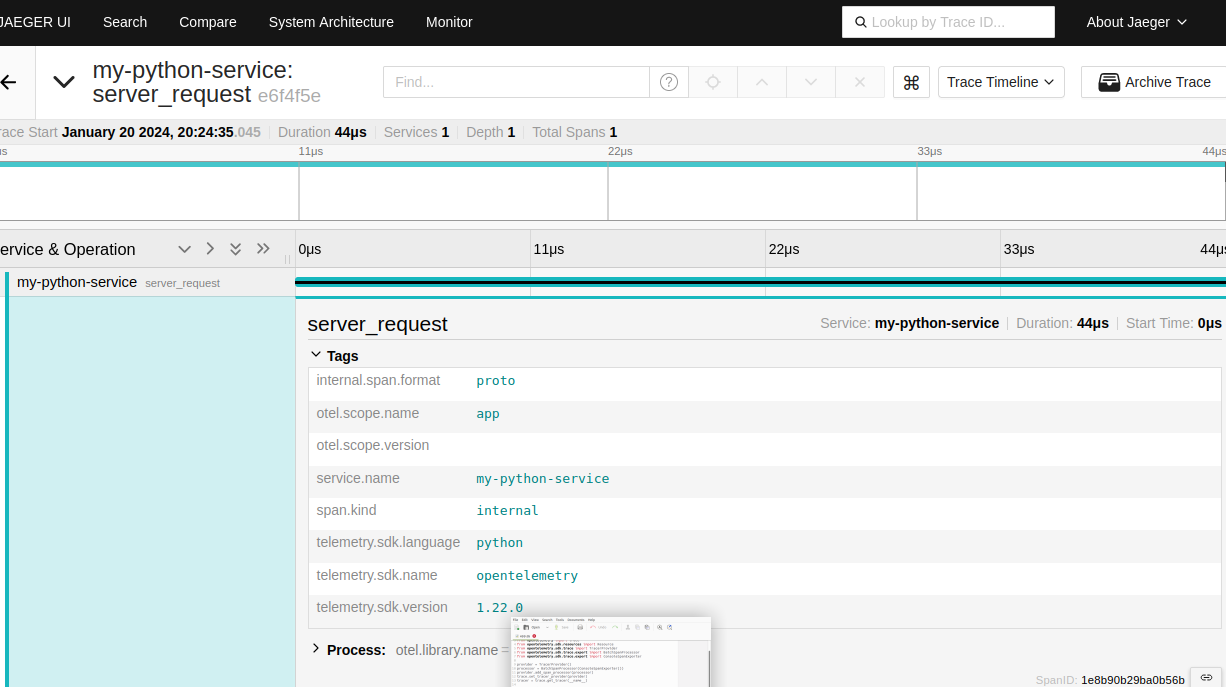
### Шаг 3. Подключение библиотек для трассировки приложения

Все сделал, результат на скриншоте



### Шаг 4. Развертывание Jaeger



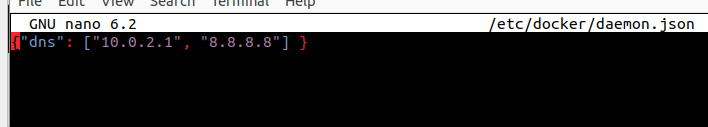


### Шаг 5. Упаковка приложения в Docker-контейнер

Тут у меня возникла та же проблема, что и в лаб работе 4, которую я тогда не доделал, но я тут

A THOUSAND OF YEARS HAS PASSED

смог ее решить. Докер не совсем правильно берет настройки ДНС из системы, нужно создать файл

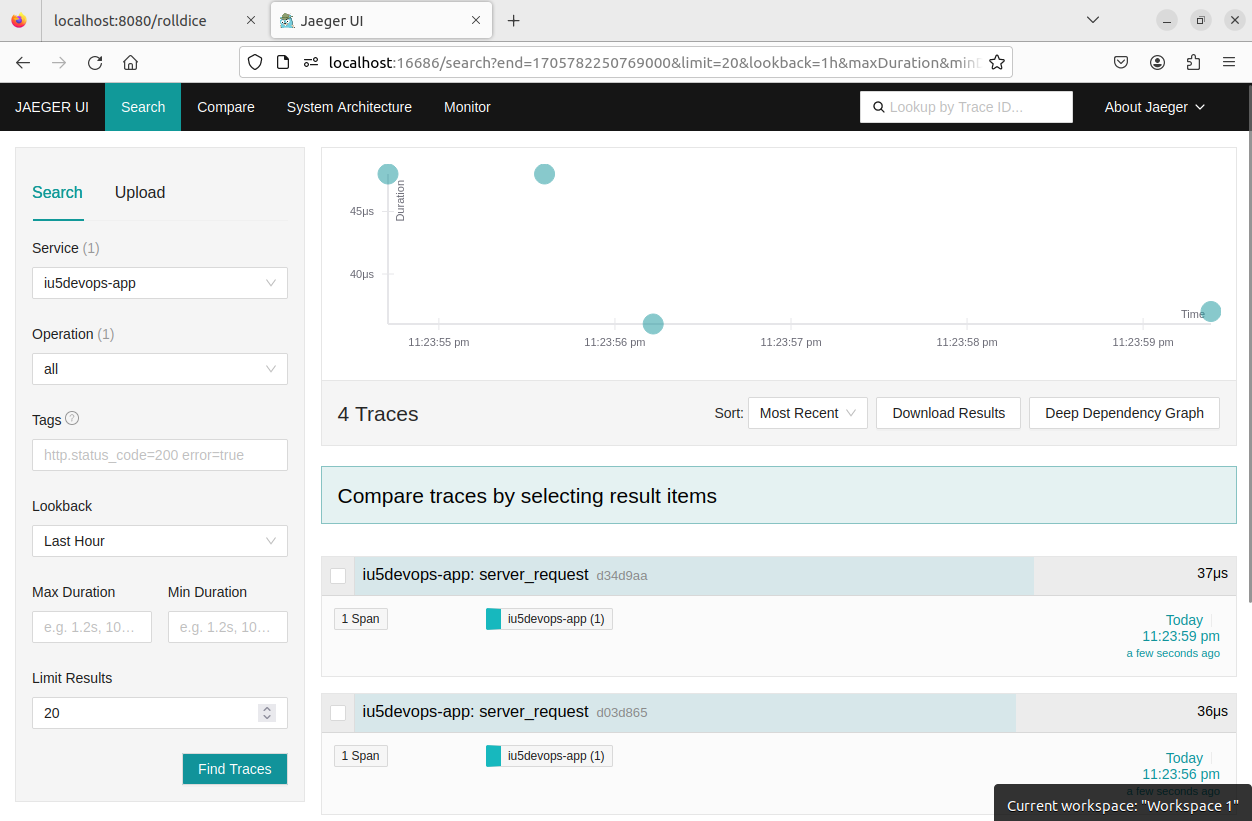


и внести в него адрес кластера, который мы создавали в 1 лаб работе.



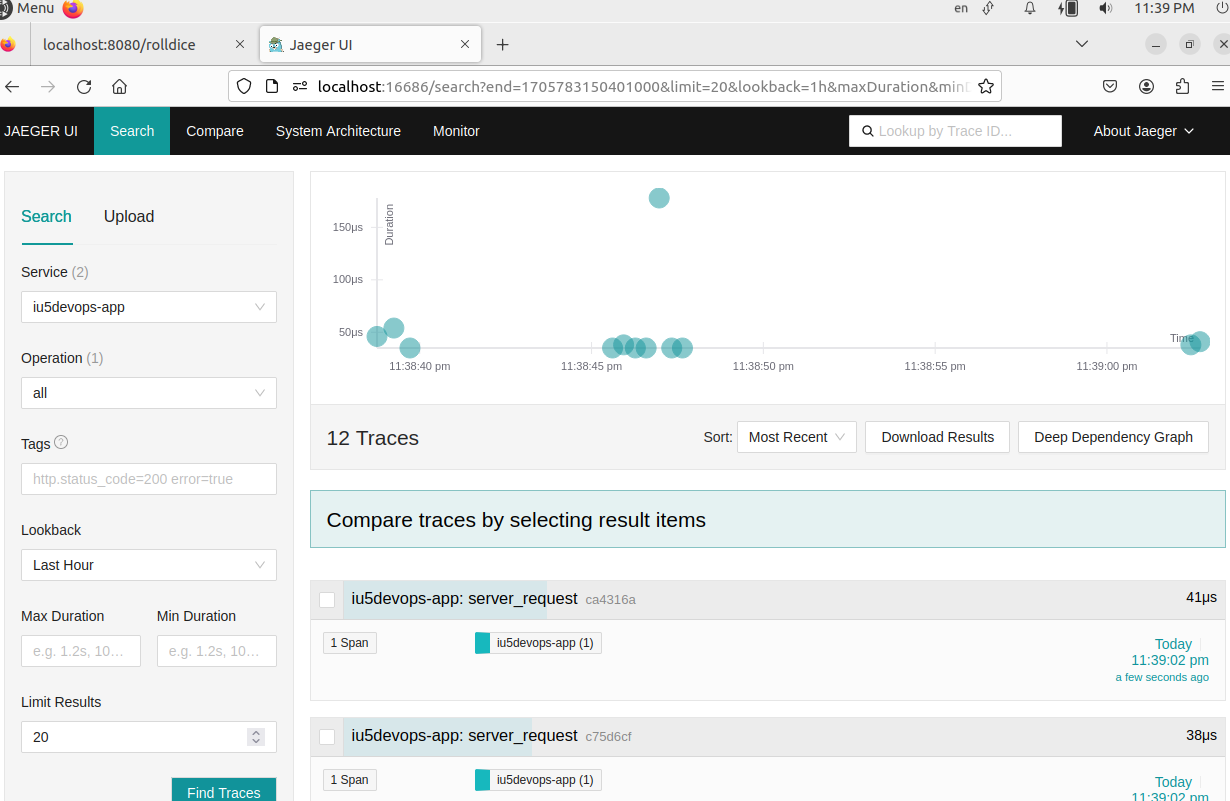
Продолжим выполнение 5 лабораторной работы, и может быть, я еще доделаю четвертую. У меня пожар был.

Все подняли, джаигир работает:

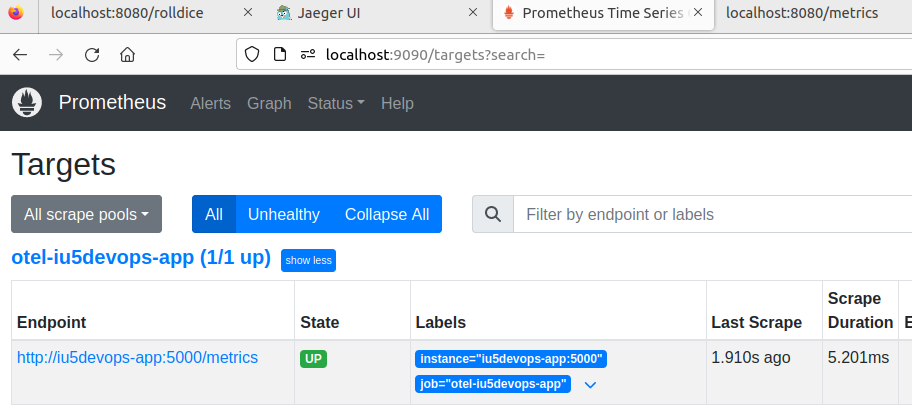


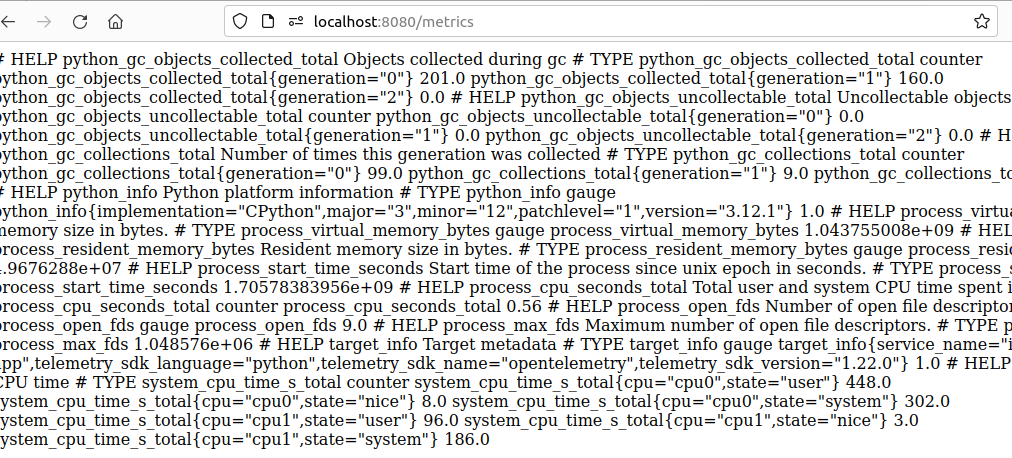
### Шаг 6. Переход на экспортер на конечную точку OTLP

Все обновили, джаигир по прежнему следы показывает

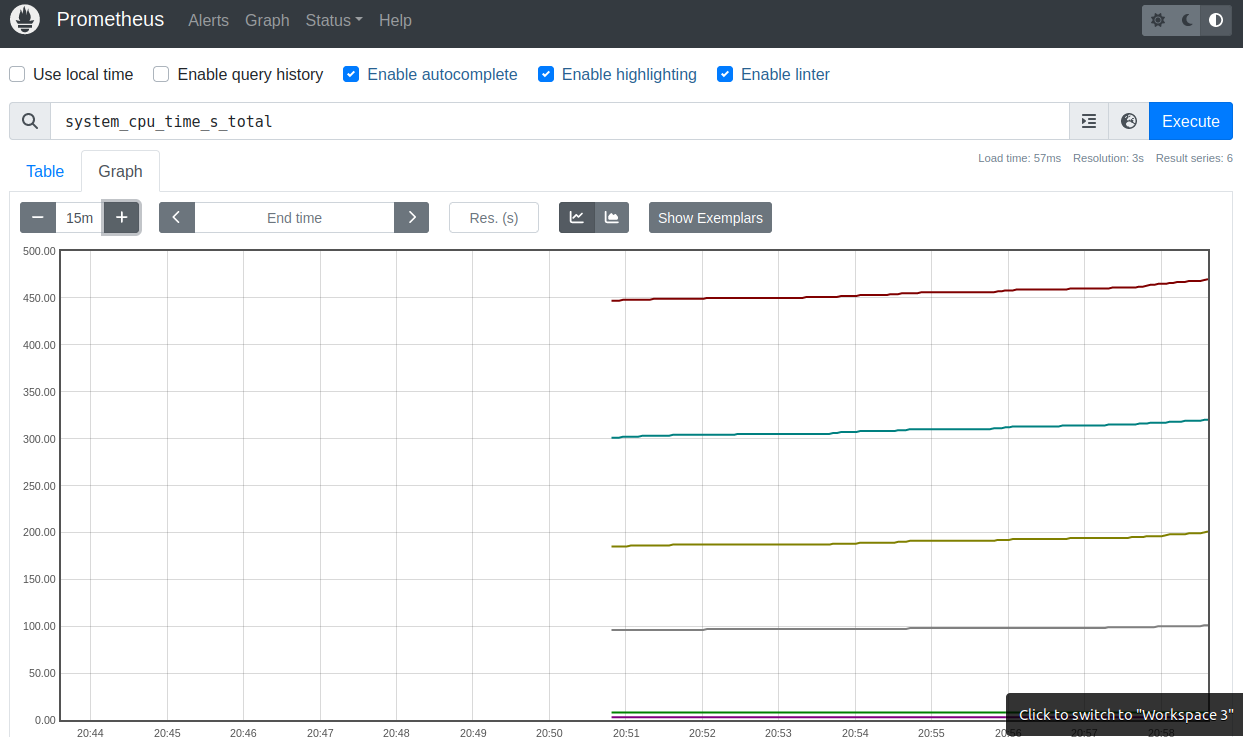


### Шаг 7. Сбор метрик с помощью Prometheus



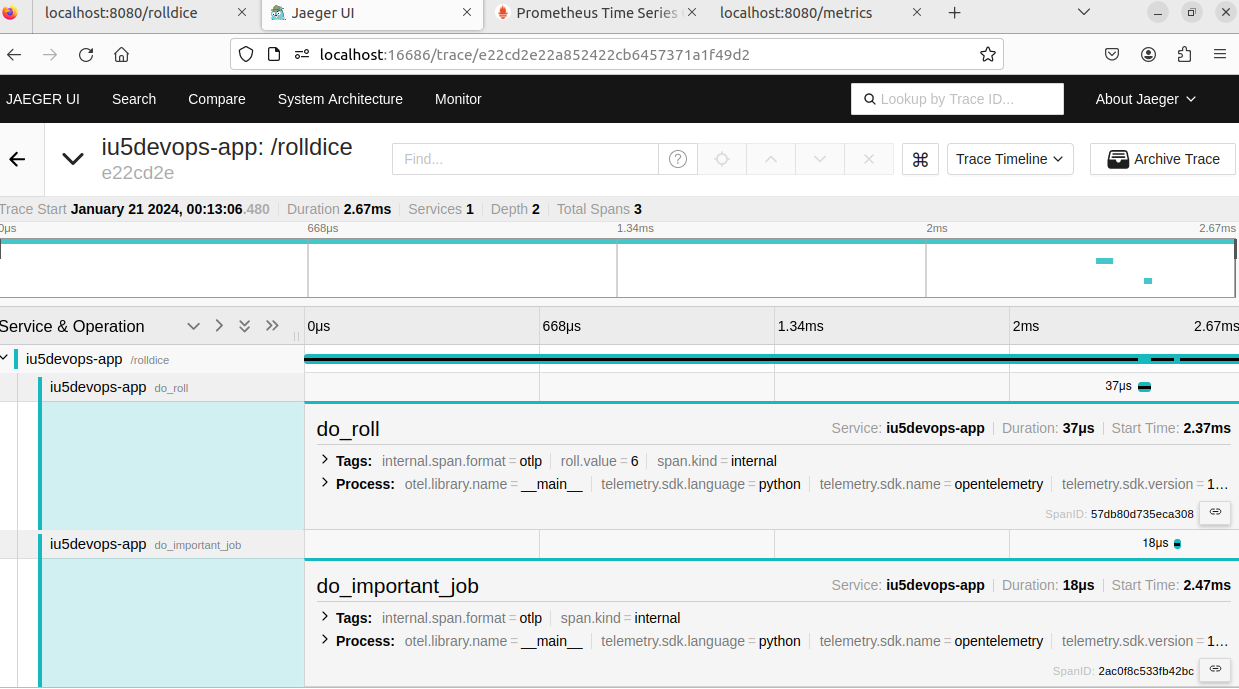


Пример работающего prometheus:

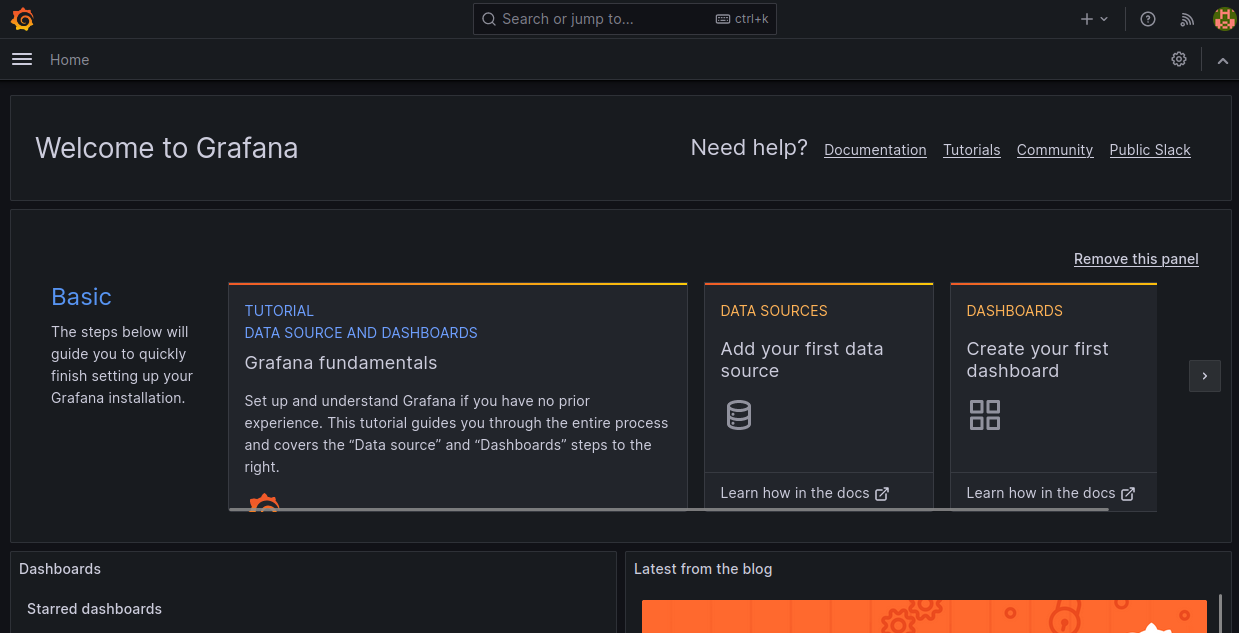


### Шаг 8. Рефакторинг приложения

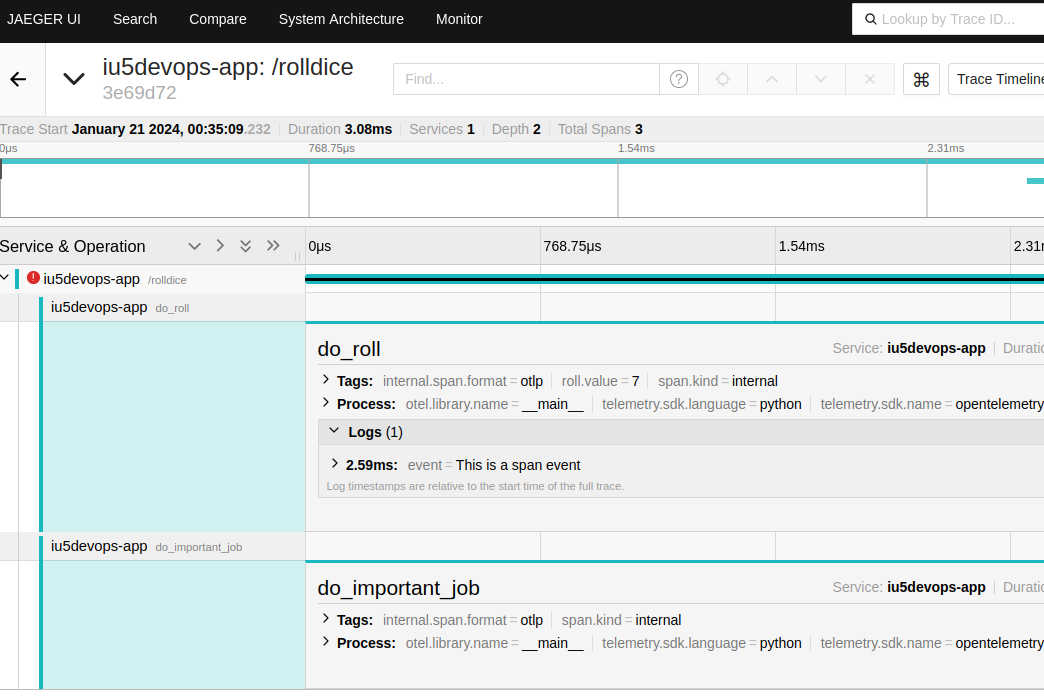
Сделаю свой скриншот, но как в приложении:



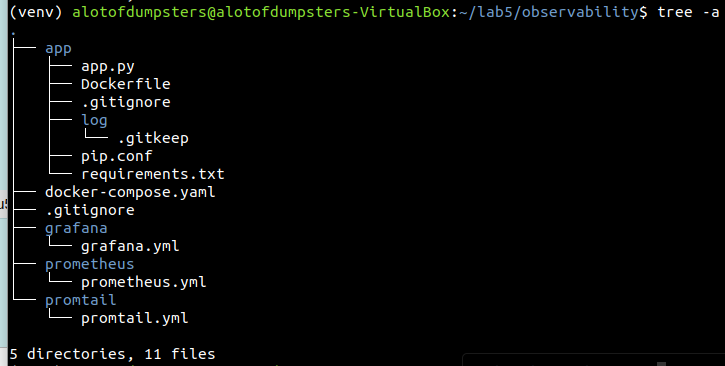
### Шаг 9. Развертывание Grafana



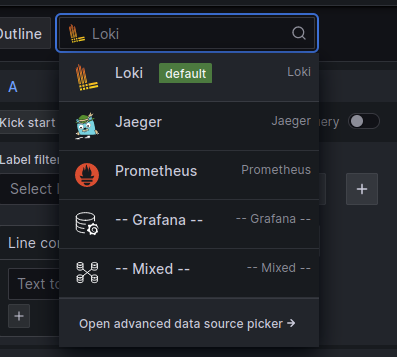
### Шаг 10. Работа с журналом



### Шаг 11. Loki и Promtail



### Шаг 12. Исследование телеметрии в Grafana



### Контрольные вопросы

1. Что такое OpenTelemetry? Атрибуты, события, контекст, журналы, трассировки, показатели?

Это (OTel) кроссплатформенный стандарт для сбора и создания данных телеметрии. OpenTelemetry включает: API для библиотек, используемых для записи данных телеметрии в виде кода.

Атрибуты — пара ключ-значение, много атрибутов представляют собой структуру данных.

Событие — это отметка о времени и набор атрибутов.

Контекст — атрибуты, общие для набора событий. Может быть статическим(определяет местонахождение событий) или динамическим(определяют активную операцию, содержащую событие, при этом значение атрибутов span изменяется при проведении операции).

Журнал — события, которые только сопровождают ресурсы, например, запуск программы.

Трассировка — это график, отображающий события, связанные с транзакцией.

Показатель - это событие, значением которого может быть количество связанных событий или некоторое вычисление значения события, например, использование системной памяти.

1. Что такое наблюдаемость? Надежность и показатели?

Наблюдаемость в контексте распределенной системы — это возможность отслеживать и анализировать данные телеметрии о состоянии каждого компонента, иметь возможность наблюдать за изменениями производительности и диагностировать причины возникновения этих изменений.

1. Что такое трассировка? Что такое распределенная трассировка? Как собирать распределенную трассировку и какие проблемы имеются в микросервисной/распределенной архитектуре?

Трассировка — это график, отображающий события, связанные с передачей данных.

Распределенная трассировка - это трассировка, пересекающая границу службы.

1. Сигналы OpenTelemetry: трассировки, метрики, логи, багаж?
2. Инструменты OpenTelemetry: автоматические, мануальные, библиотеки?
3. Компоненты OpenTelemetry: спецификация, сборщики, библиотеки инструментальных средств, экспортеры, автоматические измерительные инструменты?

Спецификация — Описывает межъязыковые требования и ожидания для всех реализаций. Определяет API, SDK и данные

Сборщик - это независимый от поставщика прокси-сервер, который может получать, обрабатывать и экспортировать данные телеметрии. Он поддерживает получение данных телеметрии в нескольких форматах (например, OTLP, Jaeger, Prometheus, а также многих коммерческих/собственных инструментов) и отправку данных на один или несколько серверов. Он также поддерживает обработку и фильтрацию данных телеметрии перед их экспортом.

Библиотеки инструментальных средств — инструменты, которые генерируют данные из телеметрии. Ожидается, что подавляющее большинство библиотек позволятиспользование без отдельных компонентов, так сказать из-коробки

Экспортеры

автоматические измерительные инструменты

1. Ресурс OpenTelemetry (телеметрия)?
2. Экспортеры OpenTelemetry?

Это компонент Otel, который отвечает, куда отправлять собранные данные. Jaeger, Grafana, Prometheus.

1. Что такое Prometheus? Как работает Prometheus?

Это коллекция метрик, агрегирование и система баз данных временных рядов. Prometheus очищает совместимые конечные точки вашего приложения (приложений) и сохраняет их внутри себя.

1. Сбор vs. отправка телеметрии? Примеры инструментария?
2. Концепции Prometheus?
3. Типы метрик?
4. Запрос и преобразование данных Grafana?
5. Источники данных Grafana?

Grafana подерживает следующие источники данных: Alertmanager, AWS CloudWatch, Azure Monitor, Elasticsearch, Google Cloud Monitoring, Graphite, InfluxDB, Jaeger, Loki, Microsoft SQL Server (MSSQL), MySQL, OpenTSDB, PostgreSQL, Prometheus, Tempo, Testdata, Zipkin и много-много других, в том числе самописные (для каких-то своих проектов) решения.