БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет прикладной математики и информатики
Кафедра технологий программирования

Лабораторная работа №3
Мониторинг в распределенных системах

По дисциплине «Непрерывное интегрирование и сборка программного обеспечения»

Методические указания по выполнению лабораторной работы

Подготовила:
Давидовская М. И.,
Ст. преподаватель кафедры ТП

Минск, 2024 г.

Содержание

Цель работы

Изучение средств мониторинга для сбора и обработки телеметрии приложений и настройка системы, в которой данные метрик, журналов и трассировок собираются для микросервисного приложения на языке Python.

Для демонстрации этапов интеграции **OpenTelemetry** в Python мы будем использовать простое приложение, реализованное с помощью программного средства разработки Flask. Первая версия данного приложения основывается на официальной документации.

Задачи работы

1. подготовить приложение на Python к анализу трассировки с помощью Jaeger;
2. упаковать приложение в образ Docker и поднимать на его основе контейнер через Docker compose;
3. обеспечить приложение механизмами экспортирования телеметрии в Jaeger, Prometheus и Grafana.
4. Самостоятельно выполнить сборку метрик, трассировки и журналов по заданию.

Краткие теоретические сведения

Метрики, журналы и трассировки

Метрики, журналы и трассировки (metics, logs, traces — MLT) — это три компонента наблюдаемости, которые могут обеспечить полную видимость программной системы и ее мониторинг.

Что представляют собой метрики, журналы и трассировки (MLT)?

Метрики

Метрики показывают то, как используется системный ресурс. Обычно они числовые. Например:

- Сколько ресурсов процессора было использовано за последний час?
- Сколько дискового пространства потребляется?
- Какая пропускная способность была использована?

Журналы

Журналы (логи) — это события, которые записывает работающее программное обеспечение. Например:

- Регистрируют трассировку стека ошибки времени выполнения.
- Регистрируют, когда пользователь обращается к системе.
- Регистрируют критическую ошибку.

Трассировки

Трассировки показывают путь выполнения программы. В микросервисной/распределенной системе запрос от клиента может обрабатываться несколькими службами. Важно знать, по какому пути был обработан запрос и сколько времени потребовалось на обработку в каждом узле, чтобы можно было выявить ошибки и узкие места.

Подробнее о MLT

Необязательно собирать все три компонента MLT. **Журналы** являются наиболее важным элементом для сбора, поскольку они сообщают, какие ошибки произошли и как выполняется программное обеспечение.

Метрики — следующий по важности элемент для сбора данных. С помощью метрик можем узнать, насколько хорошо работает программное обеспечение и нужно ли оптимизировать функции или масштабировать серверы.

Наконец, с помощью трассировки можем отслеживать, в какой последовательности был обработан запрос. Ранее в этом не было необходимости, поскольку большинство систем были монолитными и не требовали трассировки. Но поскольку микросервисы и распределенные системы становятся все более популярными, становится все более важным отслеживать, как запросы проходят через каждую службу в системе.

OpenTelemetry

OpenTelemetry (OTel) — это платформа наблюдения с открытым исходным кодом, которая позволяет собирать данные телеметрии из облачных приложений. Она предлагает инструменты, API и SDK для сбора и генерации метрик, журналов и трассировок. OpenTelemetry

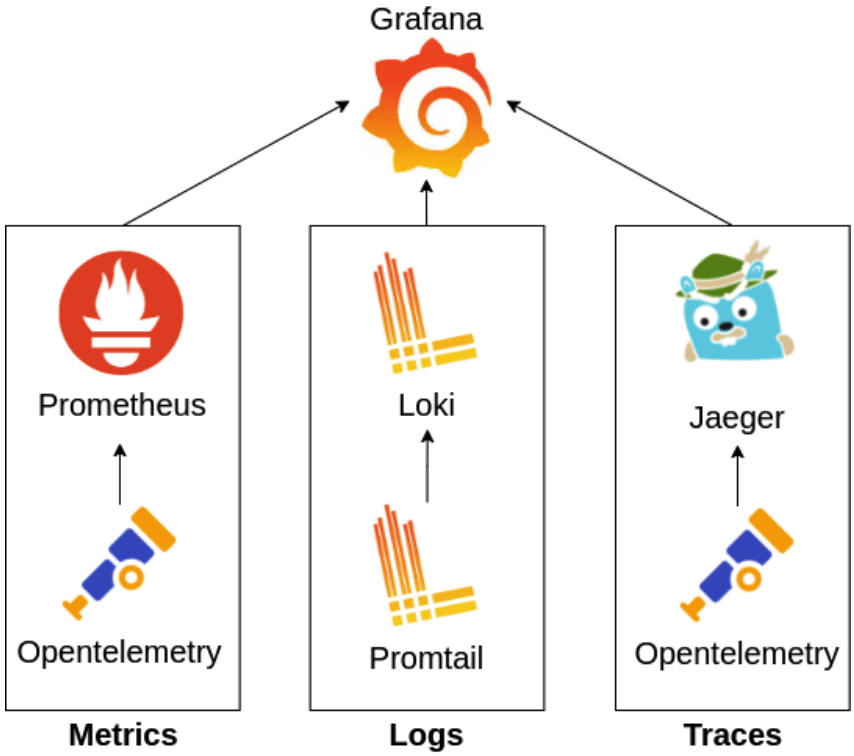
поддерживает несколько языков, включая Java, Python, Go, Ruby, C++ и Javascript.

Как мы можем собирать MLT?

Для сбора данных MLT можно использовать следующие инструменты, приведенные ниже:

- **Панель мониторинга** *Grafana*: это панель мониторинга, где мы будем наблюдать за данными MLT, собранными остальными сервисами.
- **Мониторинг** *Prometheus*: собирает и хранит показатели из приложений.
- **Ведение журнала** *Promtail*: собирает данные журнала из приложений и отправляет их в *Loki*.
- **Ведение журнала** *Loki*: агрегирует и хранит все журналы, отправленные Promtail.
- **Трассировка** *Opentelemetry*: настраивает приложение для сбора трассировок и отправляет их Jaeger.
- **Трассировка** *Jaeger*: собирает и хранит информацию о трассировке. Также помогает в визуализации трассировки.

На следующем рисунке приведены зависимости инструментов:



Задания

Методические указания

Все результаты лабораторной работы должны быть опубликованы в git-репозитории, ссылка на который доступна в курсе «Непрерывное интегрирование и сборка программного обеспечения».

Для демонстрации этапов интеграции OpenTelemetry в Python мы будем использовать простое приложение, реализованное с помощью программного средства разработки Flask. Первая версия данного приложения основывается на официальной документации.

Критерии оценивания

Для групп 11-13 выполнить все 3 задания, для группы 14 — задание 1.

Содержание отчета

1. Цель работы.
2. Вариант задания.
3. Код приложений, конфигурационных файлов.
4. Ответы на контрольные вопросы.

Отчет должен быть опубликован в git-репозитории на github. Все результаты лабораторной работы должны быть опубликованы в git-репозитории, ссылка на который доступна в курсе «Непрерывное интегрирование и сборка программного обеспечения».

В файле Readme проекта на github должна быть ссылка на отчёт. Отчет опубликовать во внешнем хранилище или в репозитории в каталоге /docs. Если в лабораторной работе необходимо написать программу/ы, то отчёт должен результаты тестов по каждой программе и ответы на контрольные вопросы.

Пример оформления файла Readme может быть таким:

```
# Overview

Report on LabRabota1.

# Usage

// Заменить <<link>> и <<folder>> на соответствующие ссылки/названия
To check, please, preview report by <<link>> and source files
in <<folder>>.

# Author

Your name and group number.

# Additional Notes

// СКОПИРОВАТЬ И ВСТАВИТЬ ССЫЛКУ НА СВОЙ РЕПОЗИТОРИЙ,НАПРИМЕР
https://github.com/maryiad/lab3-task1-gr16-david
```

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом:

1. Студент выполняет разработку программ.
2. В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведёнными в приложении.
3. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Задание 1. Подключение инструментов и библиотек OpenTelemetry и настройка трассировки приложения

В качестве операционной системы рекомендуется использовать ОС Ubuntu, установленную в виртуальной машине или же как вторая система.

1. Изучить материал «Сбор и обработка телеметрии приложений»
2. Изучить «Главу 12. Shipping Logs and Monitoring Containers» из книги The_Ultimate_Docker_Container_Book_3rd_Edition_(2023)

3. Выполнить шаги:
    - 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
4. Опубликовать в репозиторий лабораторной работы в каталог task1 конфигурационные файлы, необходимые для сборки веб-приложения и средства трассировки Jaeger.

Задание 2. Настройка сбора метрик приложения

1. Изучить материал «Сбор и обработка телеметрии приложений»
2. Изучить «Главу 12. Shipping Logs and Monitoring Containers» из книги The_Ultimate_Docker_Container_Book_3rd_Edition_(2023)
3. Выполнить шаги:
    - Шаг 6. Переход на экспортер на конечную точку OTLP
    - Шаг 7. Сбор метрик с помощью Prometheus
    - Шаг 8. Рефакторинг приложения
4. Опубликовать в репозиторий лабораторной работы в каталог task2 конфигурационные файлы, необходимые для сборки веб-приложения и сбора метрик.

Задание 3. Анализ журналов, визуализация и исследование телеметрии

1. Изучить материал «Сбор и обработка телеметрии приложений»
2. Изучить «Главу 12. Shipping Logs and Monitoring Containers» из книги The_Ultimate_Docker_Container_Book_3rd_Edition_(2023)
3. Выполнить шаги:
    - Шаг 9. Развертывание Grafana
    - Шаг 10. Работа с журналом
    - Шаг 11. Loki и Promtail
    - Шаг 12. Исследование телеметрии в Grafana
4. Опубликовать в репозиторий лабораторной работы в каталог task3 конфигурационные файлы, необходимые для сборки веб-приложения и средств анализа журналов и визуализации данных телеметрии.

Контрольные вопросы

1. Что такое OpenTelemetry? Атрибуты, события, контекст, журналы, трассировки, показатели?
2. Что такое наблюдаемость? Надежность и показатели?
3. Что такое трассировка? Что такое распределенная трассировка? Как собирать распределенную трассировку и какие проблемы имеются в распределенной архитектуре?
4. Сигналы OpenTelemetry: трассировки, метрики, логи?
5. Инструменты OpenTelemetry: автоматические, ручные, библиотеки?
6. Компоненты OpenTelemetry: спецификация, сборщики, библиотеки инструментальных средств, экспортеры, автоматические измерительные инструменты?
7. Ресурс OpenTelemetry (телеметрия)?
8. Экспортеры OpenTelemetry?
9. Что такое Prometheus? Как работает Prometheus?
10. Концепции Prometheus?
11. Типы метрик?
12. Запрос и преобразование данных Grafana?
13. Источники данных Grafana?
14. Панели мониторинга Grafana?
15. Процесс обработки журналов Grafana Loki?
16. Варианты сборки журналов для Grafana Loki?