УДК 004.75

Разработка платформы Aletheia: мониторинг ресурсов и ошибок в информационных системах с гибкой настройкой сценариев срабатывания и системой автоматических мультиканальных уведомлений

Солопов А.С. (ИТМО), **Сидоров. И.О.** (ИТМО),
Шамис А.Р. (ИТМО**)**
Научный руководитель – Ткаченко Д.М.
(ИТМО)

Введение

В современных распределённых системах и микросервисной архитектуре особое значение приобретает своевременное обнаружение ошибок и мониторинг ресурсов. От качественно настроенной системы оповещений напрямую зависит скорость реакции разработчиков на критические сбои, а значит, и надёжность всего приложения.

Платформа Aletheia предоставляет гибкую и масштабируемую систему сбора логов, мониторинга метрик, а также автоматической доставки уведомлений о возникающих проблемах. В данной работе рассматривается архитектура Aletheia и механизмы настройки правил, позволяющих пользователю определять, какие именно события должны вызывать оповещение и в каких каналах связи (Telegram, Discord, Email и др.).

Основная часть

Архитектура:

Платформа Aletheia включает в себя Collector Service (приём логов), брокер сообщений Kafka, Rule Engine (анализ событий и применение правил) и Alert-сервисы для доставки уведомлений. Логи и метрики сохраняются в MongoDB и TimescaleDB, что обеспечивает удобную аналитику и масштабирование.

Rule Engine:
Центральный модуль, который сопоставляет полученные события с набором правил (условия, операторы AND/OR, повторяемость и т. д.). При совпадении условий система отправляет задачу на рассылку уведомлений (Telegram, Email, Discord).

Сценарии применения:

- *Мониторинг ошибок*: Отслеживание критических сбоев с учётом повторяемости и контекста.
- *Мониторинг ресурсов*: Контроль превышения пороговых значений памяти, количества goroutine и др.

Утилита aletheia-paas:
Командная строка, позволяющая одним действием разворачивать всю инфраструктуру (Kafka, сервисы, базы данных), упрощая процесс настройки и внедрения в различные проекты.

Выводы

Разработанная платформа Aletheia призвана повысить надёжность и эффективность микросервисных систем за счёт своевременного обнаружения сбоев и превышения ресурсных лимитов. Основными преимуществами являются гибкая логика правил, широкие возможности для интеграции (через SDK и веб-интерфейс) и удобная CLI-утилита для развёртывания.

Список использованных источников

1. Donovan, A., & Kernighan, B. W. (2015). The Go Programming Language. Addison-Wesley Professional.
2. William Kennedy, Hoanh An (2021). Ultimate Go Notebook
3. Burns B. (2018). Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media.
4. Narkhede N. (2017). Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale. O'Reilly Media.