Visualização de Logs com Grafana e Loki

Marco Leone Merini
Universidade Católica de Joinville
Glauco Vinicius Scheffel

Resumo

Este artigo descreve a implementação de um sistema de logging em Python, utilizando a biblioteca `logging`, e a subsequente análise dos logs gerados através da ferramenta de visualização Grafana em conjunto com Loki. O objetivo é apresentar uma solução eficiente para o gerenciamento e monitoramento de logs, destacando como essas tecnologias podem ser integradas para fornecer uma visão clara e detalhada do comportamento de sistemas de software.

1 Introdução

A análise e visualização de logs desempenham um papel crucial na manutenção e monitoramento de sistemas de software. Com a crescente complexidade dos ambientes de TI, a necessidade de ferramentas robustas e flexíveis para gerenciar logs tornou-se essencial. Este trabalho explora a implementação de um sistema de logging em Python, a escolha da ferramenta de visualização Grafana em conjunto com Loki, e os resultados obtidos.

2 Implementação do Sistema de Logging

2.1 Arquivo `main.py`

O arquivo `main.py` é responsável por configurar o sistema de logging e gerar logs em diferentes níveis de severidade. Abaixo está o código utilizado:

```
import logging
import logging.config
import yaml

def setup_logging(default_path='logging_config.yaml',
                  default_level=logging.DEBUG):
    with open(default_path, 'r') as f:
        config = yaml.safe_load(f.read())
        logging.config.dictConfig(config)

def main():
    setup_logging()
    logger = logging.getLogger('example_logger')
    logger.debug('This is a DEBUG message.')
    logger.info('This is an INFO message.')
    logger.warning('This is a WARNING message.')
    logger.error('This is an ERROR message.')
    logger.critical('This is a CRITICAL message.')

if __name__ == "__main__":
    main()
```

2.2 Explicação do Código

O código apresentado configura e gera logs em vários níveis de severidade (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`). A função `setup_logging` lê um arquivo YAML de configuração (`logging_config.yaml`) e aplica essas configurações ao sistema de logging.

2.3 Arquivo `logging_config.yaml`

O arquivo `logging_config.yaml` define como o sistema de logging deve ser configurado, especificando detalhes como formato das mensagens e onde os logs devem ser armazenados:

```
version: 1
formatters:
  detailed:
```

```
    format: '%(asctime)s - %(name)s - %(levelname)s - %(message)s'

handlers:
  file:
    class: logging.FileHandler
    level: DEBUG
    formatter: detailed
    filename: 'app.log'
    encoding: 'utf8'

loggers:
  example_logger:
    level: DEBUG
    handlers: [file]
    propagate: no

root:
  level: DEBUG
  handlers: [file]
```

3 Como funciona?

O Grafana e o Loki trabalham em conjunto para fornecer uma plataforma completa de visualização de logs.

Loki é um sistema de gerenciamento de logs projetado para ser altamente eficiente em termos de armazenamento e desempenho. Diferente de sistemas tradicionais que indexam o conteúdo completo dos logs, o Loki indexa apenas os rótulos (*labels*) associados aos logs, como o nome da aplicação, o ambiente, ou o nível de severidade. Isso reduz drasticamente a quantidade de dados a serem armazenados e processados, tornando o Loki uma solução leve e rápida para grandes volumes de logs.

O Grafana, por outro lado, é uma ferramenta de visualização de dados que permite criar dashboards personalizados para monitorar métricas e logs. Integrado ao Loki, o Grafana permite que os logs sejam exibidos de maneira intuitiva, em tempo real, com a possibilidade de aplicar filtros, criar gráficos, e visualizar tendências ao longo do tempo.

No contexto desta tarefa, os logs gerados pelo sistema de logging em

Python foram coletados pelo Loki. Estes logs, que incluem informações detalhadas sobre a execução do programa (mensagens de `DEBUG`, `INFO`, `WARNING`, `ERROR` e `CRITICAL`), foram enviados ao Loki através de um simples agente de coleta. Em seguida, esses logs foram visualizados no Grafana, onde dashboards foram criados para monitorar os diferentes níveis de logs, possibilitando uma análise detalhada do comportamento da aplicação.

4 Aplicação e Resultados

A implementação do sistema de logging em Python foi realizada com sucesso, gerando logs de diferentes níveis de severidade, que foram armazenados no arquivo `app.log`. Após a configuração do Loki e do Grafana, os logs foram integrados ao Grafana para visualização.

Durante a análise dos logs no Grafana, foi possível observar o fluxo de execução do programa e identificar pontos críticos onde erros ou avisos ocorreram. Por exemplo, ao visualizar os logs em nível `ERROR`, foi possível identificar situações específicas onde o programa encontrou problemas que impediram a execução de determinadas funções. Da mesma forma, os logs em nível `WARNING` ajudaram a antecipar possíveis problemas que poderiam surgir, permitindo uma intervenção preventiva.

O uso de filtros no Grafana permitiu focar em logs de interesse específico, como aqueles gerados durante um período de tempo específico ou relacionados a uma parte particular da aplicação. Além disso, o Grafana ofereceu a capacidade de correlacionar os logs com outras métricas do sistema, como uso de CPU ou memória, proporcionando uma visão abrangente da saúde e desempenho da aplicação.

O resultado final foi a criação de um dashboard interativo e informativo que facilitou a análise contínua e o monitoramento da aplicação. Este dashboard pode ser utilizado por equipes de desenvolvimento e operações para melhorar a resposta a incidentes e otimizar o desempenho do sistema.

5 Conclusão

A integração do sistema de logging em Python com Grafana e Loki demonstrou ser uma solução eficaz para o monitoramento e análise de logs em ambientes de produção. O Loki se destacou por sua eficiência no gerenciamento

de grandes volumes de logs, enquanto o Grafana proporcionou uma interface poderosa e flexível para a visualização desses logs.

Este trabalho mostrou que, ao adotar uma solução como Grafana e Loki, as equipes de desenvolvimento e operações podem obter uma visão clara e em tempo real do comportamento das suas aplicações. Isso não só melhora a capacidade de resposta a problemas, como também facilita a manutenção preventiva e a otimização contínua dos sistemas.

No futuro, a integração de Grafana e Loki pode ser expandida para incluir monitoramento mais avançado, como a correlação de logs com eventos de rede ou a análise preditiva de falhas, o que potencialmente transformaria a abordagem reativa em proativa. Assim, Grafana e Loki se configuram como ferramentas indispensáveis para qualquer infraestrutura de TI moderna que busca eficiência e confiabilidade.