Trabalho final de Sistemas Distribuídos

Orquestração de serviços com Docker Swarm

Antônio José Brogni, Lucas Bauchspiess, Mateus Albano Santos

Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

ajbrogni@furb.br, lbauchspiess@furb.br, mateusalbano@furb.br

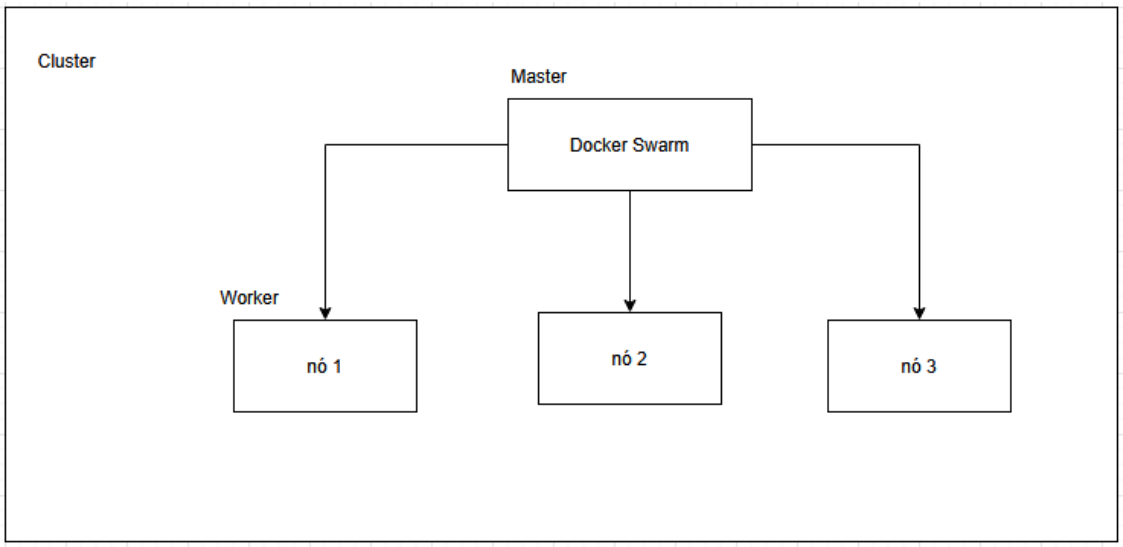
Abstract. *This paper presents the development of a distributed alert system, implemented in Python and based on container orchestration with Docker Swarm. The proposed solution is composed of sensors that periodically send data to a central service, demonstrating fundamental concepts such as scalability, high availability and resilience. The use of Docker Swarm allows load balancing and automatic restart of containers in case of failures, ensuring service continuity. The project reinforces the viability of simple, effective and accessible distributed architectures, being a viable alternative to more complex solutions.*

Resumo. *Este trabalho apresenta o desenvolvimento de um sistema distribuído de alertas, implementado em Python e baseado na orquestração de containers com Docker Swarm. A solução proposta é composta por sensores que enviam dados periodicamente a um serviço central, demonstrando conceitos fundamentais como escalabilidade, alta disponibilidade e resiliência. A utilização do Docker Swarm permite o balanceamento de carga e a reinicialização automática de containers em caso de falhas, assegurando a continuidade do serviço. O projeto reforça a viabilidade de arquiteturas distribuídas simples, eficazes e acessíveis, sendo uma alternativa viável a soluções mais complexas.*

1. Introdução

Este trabalho apresenta um sistema distribuído de alertas implementado com Docker Swarm, demonstrando na prática os conceitos fundamentais de orquestração de containers. Desenvolvemos uma solução escalável que utiliza múltiplas réplicas de serviços e balanceamento de carga utilizando Nginx. A proposta do trabalho é utilizar o Docker Swarm para garantir a redundância de um sistema de alta prioridade, onde se a instância do processo cair, o Docker Swarm garante que outra será inicializada em seu lugar garantindo a resiliência do serviço. Outro papel importante que o Docker Swarm desempenhará é o de garantir o balanceamento de carga entre os containers.

2. Desenvolvimento



Para o desenvolvimento deste sistema distribuído de alertas, optou-se pela linguagem de programação Python. Essa escolha foi fundamentada em sua versatilidade, facilidade de uso e ampla compatibilidade com bibliotecas voltadas à comunicação em rede, envio de alertas e monitoramento. Além disso, Python apresenta excelente integração com containers, recurso essencial para a arquitetura proposta neste projeto.

A gestão e orquestração dos containers foram realizadas com o Docker Swarm. Essa ferramenta possibilita a transformação de um conjunto de máquinas físicas ou virtuais em um cluster coordenado de nós (nodes), facilitando a execução e o gerenciamento de aplicações distribuídas. Como orquestrador nativo do Docker, o Swarm gerencia múltiplos containers distribuídos em diferentes máquinas como se fossem um único sistema integrado, o que simplifica significativamente a configuração e administração do ambiente.

A escolha do Docker Swarm é dada por sua capacidade de alta disponibilidade, resiliência e escalabilidade. O Swarm monitora constantemente o estado dos serviços e, em caso de falha em uma instância de container, realiza automaticamente sua reinicialização em outro nó disponível. Esse comportamento garante a continuidade do sistema, aspecto crítico em aplicações de alerta. Outro ponto forte é a escalabilidade dinâmica. O número de réplicas pode ser ajustado com facilidade por meio de um único comando, permitindo que o sistema se adapte rapidamente a variações na carga de trabalho. O balanceamento de carga entre containers ativos também contribui para o desempenho, distribuindo as requisições de forma eficiente. Outro ponto que impactou na decisão foi quanto à aprendizagem ser mais fácil que outros sistemas, a exemplo do Kubernetes.

3. Conclusões

O desenvolvimento deste sistema distribuído de alertas com o uso do Docker Swarm demonstrou, de forma prática, como é possível construir soluções confiáveis, escaláveis e com alta disponibilidade por meio da orquestração de containers. A escolha do Python como linguagem principal contribuiu bastante para uma implementação ágil, tanto pela

sua simplicidade quanto pela vasta quantidade de bibliotecas disponíveis. Já o Docker Swarm mostrou-se eficiente no gerenciamento do ciclo de vida das aplicações, garantindo que o sistema continue operando mesmo em caso de falhas pontuais.

A arquitetura que foi desenvolvida permite que diversos sensores funcionem simultaneamente, enviando dados de forma contínua para um coletor central. Esse coletor permanece sempre disponível graças aos recursos de monitoramento e reinicialização automática que o Docker Swarm oferece. Além disso, a possibilidade de escalar os serviços dinamicamente e o balanceamento de carga embutido tornam essa solução adequada para cenários com alta demanda e necessidade de alto desempenho.

De modo geral, este trabalho mostra que é viável construir sistemas distribuídos robustos utilizando ferramentas acessíveis e de fácil aprendizado. Por isso, o Docker Swarm se apresenta como uma ótima alternativa, não só para fins acadêmicos e protótipos, mas também para aplicações reais de pequeno e médio porte.