Universidade Federal de Pernambuco

Centro de Informática

Bacharelado em Ciência da Computação

Automação de Canary Deployments em Clusters de Kubernetes usando Istio e GitHub Actions

Proposta Trabalho de Conclusão de Curso de Graduação

por

Gustavo Henrique Lopes de Souza

Orientador: Prof. Vinícius Cardoso Garcia

Recife, Junho / 2021

RESUMO

Neste trabalho, será descrita uma automação que fará *canary deployments* em ambientes Kubernetes utilizando *continuous delivery* (CD) com *GitHub Actions* e *Istio* como plataforma de *Service Mesh*. Inicialmente, serão apresentadas as motivações da recente larga adoção do Kubernetes como principal serviço de orquestração de *containers*. Em seguida, será feita uma breve introdução dos principais conceitos e funcionalidades do Kubernetes. Além disso, também serão apresentados os conceitos de *canary deployment*, *continuous delivery*, *service mesh* e as ferramentas que serão usadas neste trabalho. Após isso, serão apontados alguns problemas e situações comuns no dia a dia de uma pessoa Engenheira de Software que a automação resultante deste trabalho poderá mitigar ou ser útil de alguma forma. Por fim, serão apresentados o processo de desenvolvimento desta automação e sugestões de incrementos a serem feitos em trabalhos futuros.

Palavras-chave: Entrega contínua, Orquestração de containers, Canary Deployment, Confiabilidade, Disponibilidade.

ABSTRACT

This work will describe an automation to perform canary deployments in Kubernetes environments using *continuous delivery* (CD) with *GitHub Actions* and *Istio* as a *service mesh* platform. First, the reasons for the recent large adoption of Kubernetes as a container orchestration platform will be presented. Then, a brief introduction of Kubernetes main concepts and functionalities will be made. Furthermore, the concepts of *canary deployment*, *continuous delivery*, *service mesh*, and the tools which will be used in this work will be shown as well. After that, some common problems and situations of a Software Engineer routine which can be mitigated by the automation resulting from this work will be pointed for the sake of motivation. Lastly, the development process of this automation will be presented with suggestions of increments to be done in future works.

Keywords: Continuous delivery, Container orchestration, Canary deployment, Reliability, Availability...

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, especialmente na última década, os sistemas computacionais têm ficado cada vez mais distribuídos e granulares. Dessa maneira, é evidente que cada vez mais empresas de tecnologias rodam em ambientes de *cloud*, como mostra [1]. Sendo assim, novos tópicos e padrões sobre arquiteturas de microsserviços surgem a todo instante prometendo coisas como mais escalabilidade, disponibilidade e maior frequência de *deployments* que são algumas das principais métricas avaliadas em times de engenharia de software segundo [2].

Contudo, os aumentos da granularidade e da distribuição dos sistemas de software também trazem desafios. Como apontado em [3], até o mais simples ecossistema de microsserviços é complexo o suficiente para satisfazer uma verdade: Catástrofes e falhas poderão ocorrer com frequência e provavelmente todo cenário de falha possível irá acontecer pelo menos uma vez no ciclo de vida do sistema. Além disso, como é dito em [4], nenhum ambiente de testes consegue ser 100% idêntico ao ambiente de produção e os testes de um sistema raramente cobrem 100% dos cenários possíveis, portanto alguns erros e *bugs* irão chegar ao ambiente de produção.

A estratégia de *rollout* conhecida por *canary deployment* ou *canary releasing* surge como uma forma de mitigar todo esse cenário caótico e dar mais confiança às pessoas engenheiras que precisam colocar seus códigos no ar. [5] define essa estratégia como uma maneira de validar uma nova versão de algum software através da observação de como essa nova versão se comporta, em aspectos funcionais e não-funcionais, ao receber tráfego do ambiente de produção (tradução nossa)[1]. Ainda de acordo com [5], caso a nova versão se comporte de forma inesperada, basta reverter o que foi feito; e se a nova versão se comporta como esperado, o *rollout* pode ir em frente.

A estratégia de *canary deployment* pode ser integrada em *pipelines* de *Continuous Integration (CI)* e *Continuous Delivery (CD)* com o objetivo de oferecer, além da segurança, velocidade para as equipes que precisam frequentemente lançar novas versões de algum sistema. Inclusive, [6] pontua isso como um dos requisitos necessários para que um serviço seja considerado estável e confiável.

[1] With canary releasing, we are verifying our newly deployed software by directing amounts of production traffic against the system to see if it performs as expected. "Performing as expected" can cover a number of things, both functional and nonfunctional.

Este trabalho propõe uma maneira de realizar essa integração entre *canary deployments* e *Continuous Delivery* com ferramentas amplamente utilizadas pela indústria: Kubernetes, Istio e GitHub Actions. O capítulo 2 abordará os conceitos necessários para desenvolver esta integração. No capítulo 3, os componentes que irão compor a integração serão apresentados juntamente com os papéis que eles dentro do todo. No capítulo 4, será descrito o desenvolvimento da automação da integração. Nos capítulo 5, a *pipeline* resultante do desenvolvimento será mostrada. Nos capítulos finais, serão debatidos os trabalhos futuros e a conclusão deste trabalho.

2 OBJETIVO

O principal objetivo deste trabalho é descrever o desenvolvimento de uma *pipeline* de *Continuous Delivery* onde um de seus estágios consiste em fazer *canary deployments* em ambiente de produção. Outro objetivo é analisar o desempenho das instâncias *canary* em comparação com instâncias criadas por estratégias de *deployment* mais convencionais.

Os seguintes objetivos específicos devem ser alcançados para satisfazer o objetivo principal:

- Criar um cluster de Kubernetes que simulará um ambiente de produção;
- Configurar o Istio para ser a plataforma de *Service Mesh* do cluster;
- Codificar um software com a capacidade de fazer com que erros aconteçam de forma controlada e intencional;
- Criar GitHub Actions que coordenem o *deployment* desse software no cluster de Kubernetes e seja capaz de realizar *canary deployments*;
- Coletar métricas a fim de comparar os *canary deployments* com outras estratégias de *deployments*;

3 CRONOGRAMA

Atividades	**Maio**	**Junho**	**Julho**	**Agosto**
Elaboração da Proposta	– –X			
Revisão da Literatura	– –X	X– –		
Desenvolvimento do sistema		XXX	XX–	
Escrita da Monografia		XXX	XXX	
Construção da Apresentaão			– –X	XX–
Defesa do TCC				– –X

4 POSSÍVEIS AVALIADORES

- Nelson Souto Rosa (nsr@cin.ufpe.br)
- Carlos André Guimarães Ferraz (cagf@cin.ufpe.br)

REFERÊNCIAS

[1] JONES, E. *Cloud Market Share – a Look at the Cloud Ecosystem in 2021*. Available at: <https://kinsta.com/blog/cloud-market-share/>. Accessed in: 30-May-2021.

[2] BROWN, A. et al. *State of DevOps Report 2020 - presented by Puppet and CircleCI*. [S.l.], 2020.

[3] FOWLER, S. J. Production-Ready Microservices: Building Standardized Systems Across an Engineering Organization. *O'Reilly Media*, p. 150–157, Oct 2017.

[4] BEYER, B. et al. The Site Reliability Workbook: Practical Ways to Implement SRE. *O'Reilly Media*, v. 1, p. 338, July 2018.

[5] NEWMAN, S. Building Microservices: Designing Fine-Grained Systems. *O'Reilly Media*, v. 1, p. 262, Feb 2015.

[6] FOWLER, S. J. Production-Ready Microservices: Building Standardized Systems Across an Engineering Organization. *O'Reilly Media*, p. 68, Oct 2017.