III SIMGETEC: SIMPÓSIO DE GESTÃO E TECNOLOGIA

Centro Estadual de Educação Tecnológica Paula Souza
Faculdade de Tecnologia de Carapicuíba
16 a 18 de Novembro de 2017
ISSN: 2526-6241

ANÁLISE E OTIMIZAÇÃO DE UM *PIPELINE* DE IMPLANTAÇÃO ATRAVÉS DE PRÁTICAS DEVOPS

Christian Gabriel Bernini Alves da Silva – Fatec Carapicuíba

Prof. Me. Mario Marques – Fatec Carapicuíba (Orientador)

RESUMO

O termo DevOps, cunhado em 2009, é composto por práticas de gestão de ciclo de vida de *software*, como o Ágil e a Programação Extrema, além de competências humanas, como a colaboração e a afinidade. Através de um estudo de caso em uma empresa do ramo bancário, com o levantamento do fluxo de atuação da equipe responsável por colaborar com a transformação digital da instituição, o processo de entrega de *software* atual é levantado e compreendido, e um fluxo alternativo é proposto, buscando alinhar-se com práticas de mercado e inspirando-se na literatura de referência sobre o assunto. A adoção de uma ferramenta de gestão de configuração é analisada, juntos aos prós e contras da automatização no dia a dia da equipe. A lei de Conway é observada durante a realização do estudo de caso, de forma que a proposta de valor apresentada vise diminuir a complexidade, mais transparência e aproximação as áreas envolvidas no processo de entrega de *software*, principalmente desenvolvedores, analistas de infraestrutura e equipe de operações.

Palavras-chave: Gestão de configuração, DevOps, Entrega Contínua, Colaboração, Automatização

ABSTRACT

The term DevOps, coined in 2009, composed of both software life-cycle practices like Agile and eXtreme Programming, but also human competencies, such as collaboration and affinity. Through a case study at banking company, first understanting the workflow of a team responsible for enabling the company's digital transformation, the software delivery process is then mapped and understood, and an alternative flow is then suggested, aimed at being aligned with industry's best practices and inspired by the reference literature about it. The adoption of a configuration management tool is analyzed, as well as the benefits and downsides of automation and its impact on the team. Conway's law is considered during the case study, just so the value proposal targets reduced complexity, increased transparency, and the aproximation of the teams involved in the software delivery process, mainly developers, infrastructure analysts and operations.

Keywords: *Configuration Management, DevOps, Continuous Delivery, Collaboration, Automation*

ISSN: 2526-6241

1 INTRODUÇÃO

O Manifesto Ágil em seu primeiro princípio, diz: "Nossa maior prioridade é satisfazer o cliente por meio da entrega adiantada e contínua de *software* de valor". (BECK; et al, 2001). Entrega essa que, para ser contínua, necessita ser consistente, controlada e reprodutível. (HUMBLE; FARLEY, 2011)

Tal prática busca harmonizar os ganhos, alcançados com o desenvolvimento, através de modelos Ágeis às necessidades da infraestrutura e operações subjacentes, utilizando-se de gestão de configuração, automatização, entre outras ferramentas e técnicas para uma entrega eficiente e de baixo risco.

Através de uma compreensão das necessidades em interações realizadas com os clientes, aqui compreendidos como integrantes de *squads* ágeis, iniciou-se a construção de ferramentas que tornassem transparente o provisionamento de infraestrutura e seu acesso aos mesmos.

Entende-se que ferramentas são aceleradores de cultura e não a cultura em si, de forma que a utilização e a adaptação das mesmas são contínuas, e visam atender as necessidades dos reais envolvidos com o processo de entrega de *software* desde o ambiente de desenvolvimento até o de produção: os colaboradores.

Todos os modelos aqui discutidos partem da premissa de que a Integração Contínua (CI) no projeto em questão já existe, porém não avaliaremos seu grau de maturidade. Durante o projeto foi percebida uma carência em como tais artefatos, gerados pelo processo de CI, são promovidos nos ambientes e customizados de acordo com necessidades específicas da aplicação.

A infraestrutura que sustenta o objeto de estudo consiste de uma arquitetura de micro serviços de tecnologias diversas, implementada em uma nuvem privada através da solução OpenShift, do fornecedor RedHat. (REDHAT, 2017)

Os serviços foram desenhados e implementados originalmente para terem seu acesso direto via API aos desenvolvedores após a fase de CI, ou por interface gráfica através da ferramenta Rundeck.

Este artigo busca retratar, em um cenário real, como a utilização de práticas ágeis, alinhadas à cultura DevOps, fornecem a velocidade desejada na entrega de *software* em produção, demonstrar a evolução das ferramentas e apresentar um fluxo alternativo ao atual. Quais são as dificuldades observadas no processo de entrega de *software* e como as mesmas podem ser minimizadas?

ISSN: 2526-6241

2 FUNDAMENTAÇÃO TEÓRICA

Os conceitos aqui definidos, quando utilizados, serão referenciados por seu acrônimo, à maneira como se faz no meio profissional.

2.1 DEVOPS

Cunhado em 2009, o termo DevOps descreve uma abordagem onde as equipes de desenvolvimento e operações têm metas compartilhadas e utilizam-se das práticas de CI de forma a facilitar a implementação do *software* (KIM; et al, 2016). Hoje, em um modelo mais maduro, compreende-se por DevOps um conjunto de práticas e cultura sobre quatro pilares: colaboração, afinidade, ferramentas (automatização), tudo isso em escala. (DAVIS; DANIELS, 2016)

A cultura DevOps busca delegar a rotinas automatizadas todo o trabalho repetitivo durante o ciclo de vida de uma aplicação, buscando prover mais autonomia e agilidade aos membros de uma equipe, também conhecida por *squad*, composta por não mais que dez integrantes, de maneira a evitar o aumento de complexidade de um *software* como previsto pela Lei de Conway, segundo Kim; et al. (2016 *apud* CONWAY, 1968), onde é exposto que uma organização, ao desenhar um sistema, irá produzir na estrutura de tal desenho uma cópia da estrutura de comunicação de tal organização.

2.2 INTEGRAÇÃO CONTÍNUA (CI)

É o processo de integrar o código na *branch* principal frequentemente durante o dia. Uma série de testes automatizados é disparada contra a nova versão assim que a mesma passa a fazer parte da *branch*. Através de entregas pequenas, porém constantes, torna-se muito mais fácil de encontrar, resolver e minimizar regressões causadas por alguma mudança no código. (DAVIS; DANIELS, 2016)

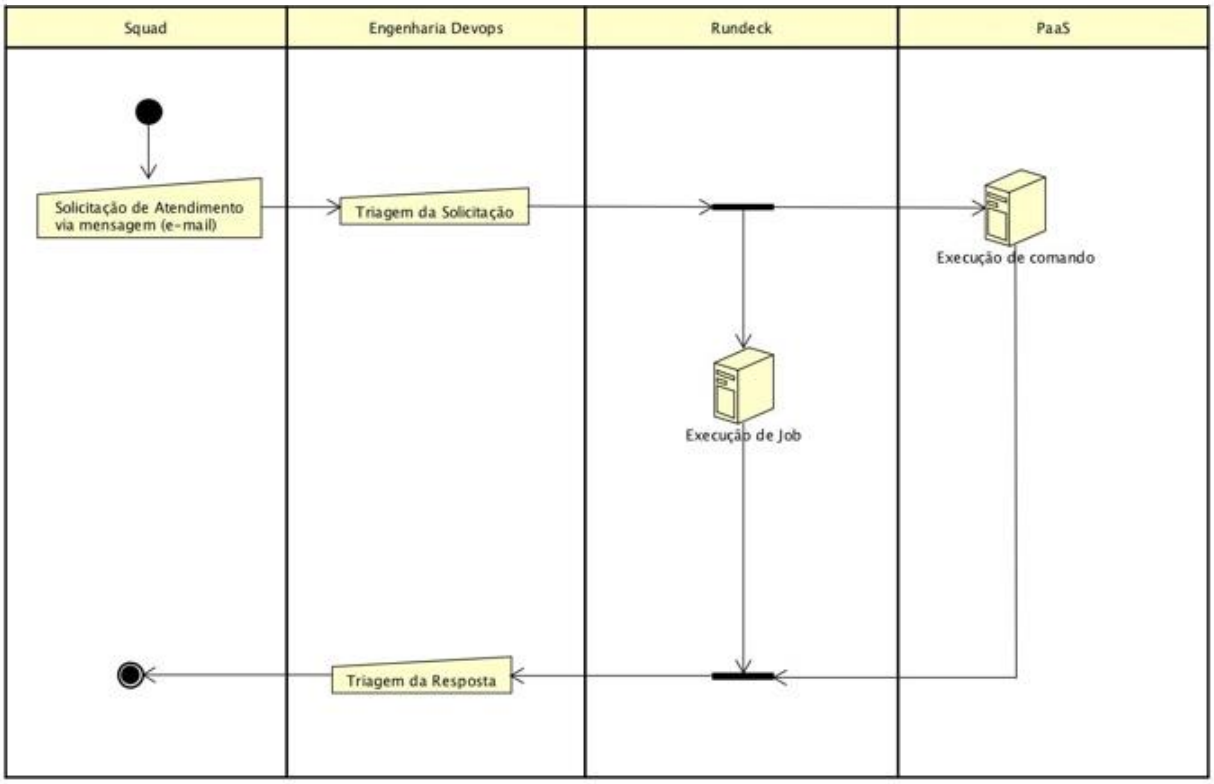
Caso um build quebre após algum código entrar na *branch*, o *feedback* é imediato e a solução tende a ser mais rápida. Uma ferramenta amplamente utilizada para CI e utilizada no projeto, disponibilizada como *software* livre, é o Jenkins.

ISSN: 2526-6241

2.3 ENTREGA CONTÍNUA (CD)

Também conhecida por CD, ou *Continuous Delivery*, consiste na utilização de *pipelines* (uma abstração direta dos pipelines observados na arquitetura de microprocessadores, como mecanismos de otimização de processamento) de implantação, com alto grau de automatização tanto de testes como de entrega e uso adequado de gerência de configuração, possibilitando entregas apertando apenas um botão ou dependente de apenas um estímulo, quando necessário, em qualquer ambiente, seja ele teste, desenvolvimento ou produção). (HUMBLE; FARLEY, 2011)

Figura 1 - Fluxo de atendimento Squads x Engenharia DevOps



Fonte: Próprio Autor

A CD é vista como uma evolução natural do CI, de forma a abranger não só o processo de desenvolvimento e garantir uma visão holística de todo o ciclo de vida do *software*. A ferramenta escolhida para centralizar as automatizações é o Rundeck, também disponível como *software* livre.

ISSN: 2526-6241

3 PROCEDIMENTOS METODOLÓGICOS

A modelagem de processo de negócio foi realizada inicialmente para entender como as interações entre a equipe de Engenharia DevOps aconteciam. Uma vez compreendido o fluxo e quais processos poderiam ser otimizados, foram levantados os requisitos funcionais e não funcionais para a automatização do processo. Foram utilizados diagramas UML para visualização destas etapas, como o fluxo de atendimento observado na figura 1.

A proposta de fluxo alternativo descrita no artigo foi apresentada como melhoria a ser oferecida ao cliente, cabendo à liderança a autorização da mesma. A mesma busca seguir a boa prática, como descrito por Humble e Farley (2011), de manter as configurações de cada ambiente separadas, em um controle de versão, com o arquivo correto sendo recuperado através de uma variável de ambiente passada ao script de implementação, indicando qual o entorno deve ser utilizado (desenvolvimento, homologação e produção).

4 DESENVOLVIMENTO

A área de Engenharia DevOps, consiste em uma equipe responsável por otimizar, viabilizar e propor melhores práticas de entrega contínua, além de disseminar a cultura DevOps no cliente, uma multinacional do ramo financeiro.

Além disso a equipe se tornou responsável, após o início do projeto, pela configuração e provisionamento dos pipelines de implementação, criação dos ambientes na solução de plataforma como serviço (PaaS), o OpenShift, em interface direta com os *squads* de desenvolvimento e sua liderança imediata.

O estudo de caso também buscou explorar o aspecto humano do que é hoje chamado DevOps e, erroneamente, interpretado como a aplicação de, somente, automatização e novas tecnologias. Davis e Daniels (2016) entendem que nem tudo pode ou sequer deve ser automatizado e, frequentemente, quanto mais complexas se tornam as automatizações, maior a necessidade de intervenção humana para manter ou analisar problemas percebidos nas mesmas. Deve-se levar em conta, também, a necessidade de evitar o *deskilling*, processo que se dá quando há um número tão grande de automatizações em vigor que as pessoas podem esquecer como devem realizar tal procedimento e sua experiência em tal área irá diminuir com o tempo. (DAVIS; DANIELS, 2016)

ISSN: 2526-6241

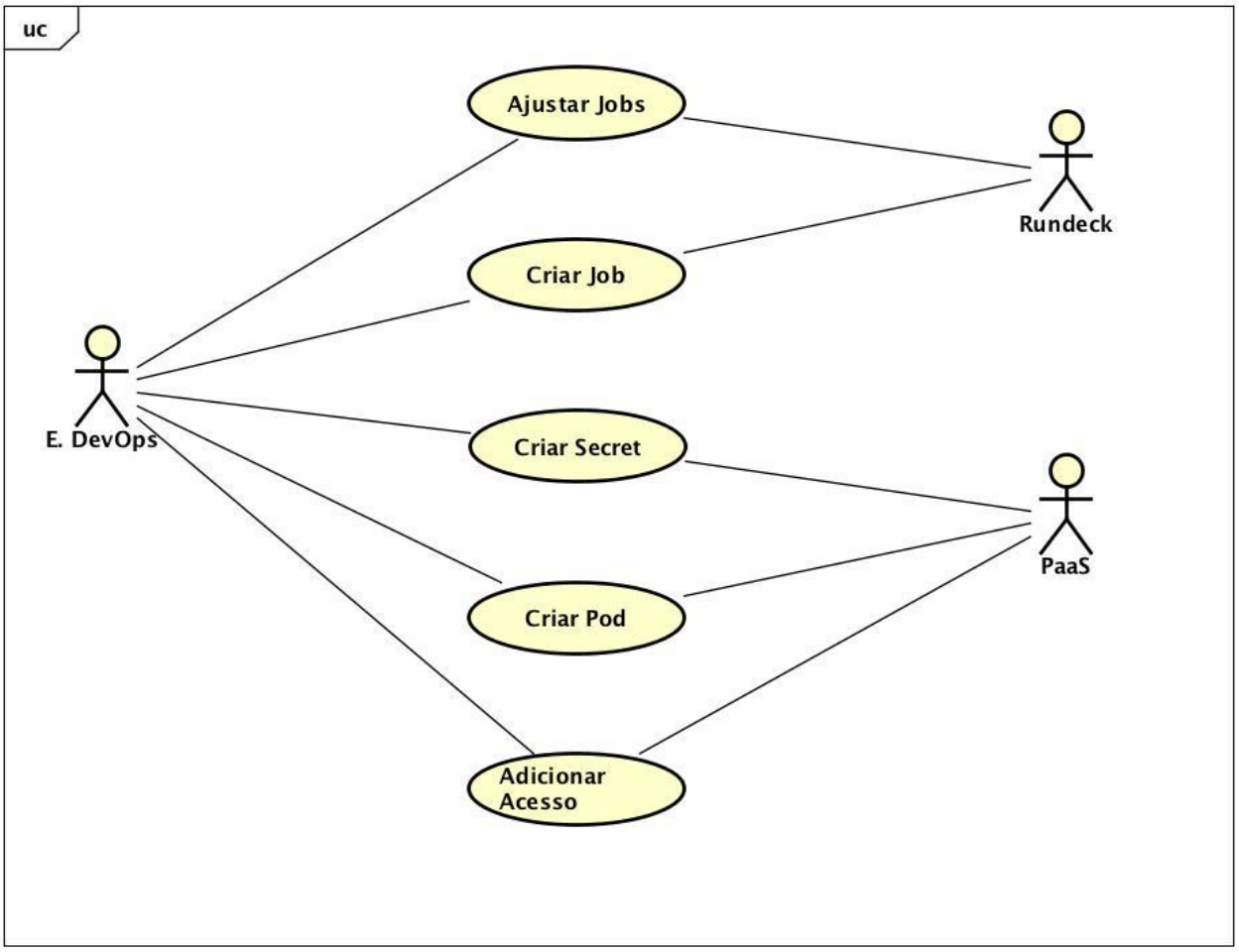
Através da utilização do Rundeck, a equipe de Engenharia DevOps pôde disponibilizar, via API e pela própria ferramenta em uma interface *web*, um *pipeline* de implementação que garantisse aos desenvolvedores autonomia nos ambientes de Desenvolvimento e Homologação enquanto a equipe de operações pudesse gerir o ambiente de produção.

Originalmente, a equipe disponibilizou a possibilidade de integrar a ferramenta de CI Jenkins com o Rundeck de forma a garantir a promoção dos artefatos para os ambientes de Desenvolvimento e Homologação via API, além de garantir Jobs de implementação e contingência para a equipe de operação.

As automatizações implementadas via Rundeck são (figura 2):

1. Criação de *secrets*, para configuração de usuário/senha no Openshift;
2. Criação automática dos *jobs* de implementação e contingência de aplicações;
3. Liberação de acesso no Openshift.

Figura 2 - Processos automatizados via Rundeck



Fonte: Próprio Autor

ISSN: 2526-6241

Durante a interação com os *squads* de desenvolvimento, foi percebido que a criação e customização de aplicações nos projetos do OpenShift utilizados eram solicitações constantes à equipe de Engenharia DevOps e para isso foram entregues duas novas automatizações:

4. Criador de aplicações no OpenShift, já também criando os *jobs* de implementação e contingência;
5. Atualizador de *jobs* no Rundeck.

A entrega mais recente, o Atualizador de *jobs*, surgiu da necessidade de adequar os *jobs* já existentes às constantes evoluções do ambiente, visto que a ferramenta em si não fornece uma opção de edição em lote com customização suficiente. É válido pontuar que todas as automatizações dependem de algum input, seja ele manual ou oriundo da plataforma de CI.

5 RESULTADOS E DISCUSSÃO

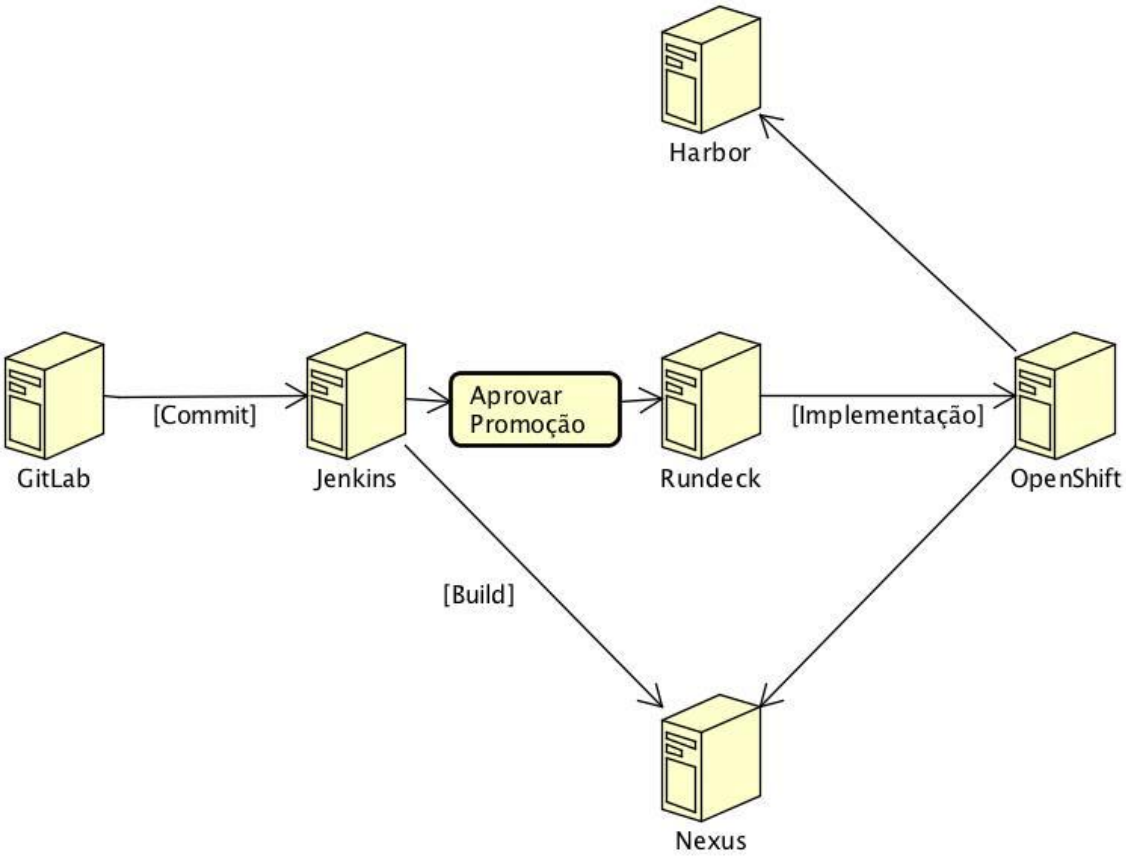
Em tempo de projeto, percebeu-se através de estudo da literatura referenciada sobre as boas práticas de mercado descritas na mesma, que a utilização mais rigorosa da disciplina de gerência de configuração diminuiria os acionamentos à equipe de Engenharia DevOps, pelo fato de garantir que toda informação relacionada à área de CD estivesse sob um controle de versão e mudanças nos processos fossem automaticamente documentadas e validadas. Tal deficiência passou a gerar impacto negativo para a equipe devido à quantidade de acionamentos feitos a cada nova implementação, quando era percebido que, uma vez que a configuração do ambiente não estava persistida em controle de versão como o código, não havia maneira fácil e ou transparente de validar que o ambiente funcionaria até que fosse realizada a referida implementação.

Somente no mês de agosto de 2017, os Jobs implementados na ferramenta Rundeck foram executados mais de 22 mil vezes, permitindo dezenas de *squads* entregar *software* em produção sem necessidade de intervenção manual ou acionamento de alguma equipe externa ao processo, exceto a validação humana para a promoção em produção. Toda essa infraestrutura é

ISSN: 2526-6241

hoje administrada e provisionada por uma equipe de 5 (cinco) engenheiros de sistemas e seu estado atual pode ser observado na figura 3.

Figura 3 - Fluxo Atual do Sistema



Fonte: Próprio Autor

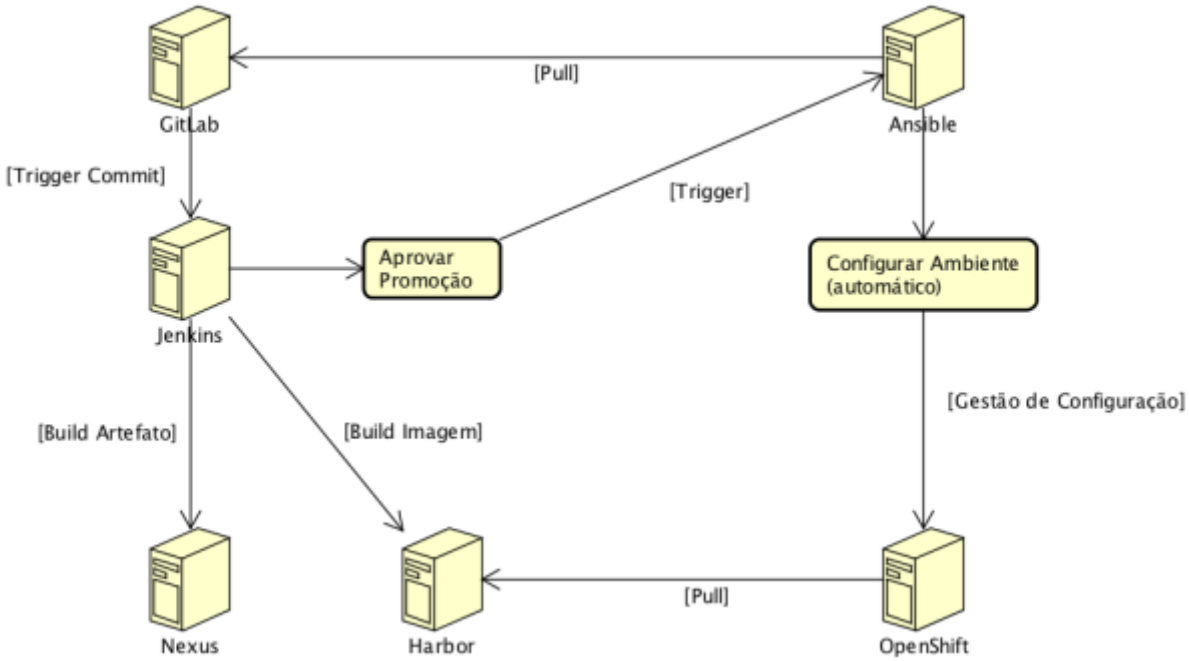
Um cenário alternativo foi proposto, de maneira a aumentar o nível de automatização, diminuir a necessidade de acionamentos por incidentes e reduzir o tempo de entrega da infraestrutura. A figura 4 demonstra um fluxo onde toda a gestão de configuração passa a ser automática, através da customização na ferramenta Ansible e o controle de promoção passa a ser diretamente pela plataforma de CI, Jenkins. A possibilidade de configuração automática através de um ponto central dá a garantia e a visibilidade de que as necessidades de segurança estejam presentes e sejam facilmente auditáveis, além de minimizar o erro humano durante a configuração e ou provisionamento do ambiente.

O modelo busca respeitar o conceito de infraestrutura como código (FOWLER, 2016), garantindo que toda a configuração do ambiente é definida e persistida, em código, num sistema

ISSN: 2526-6241

de controle de versão, tratada com o mesmo cuidado que o código da aplicação, e de infraestrutura imutável (MORRIS, 2013), a qual consiste em provisionar servidores que, uma vez instanciados, não sofrem nenhuma alteração visto que são atômicos a partir da soma de uma imagem base, configuração automaticamente gerenciada e dados, amarrados antes da subida do mesmo, de maneira a garantir maior confiabilidade no processo de entrega e diminuir o atrito entre as áreas envolvidas uma vez que qualquer engenheiro do *squad* passa a ter mais autonomia sobre a infraestrutura do seu projeto e o acionamento direto de recursos da equipe de Engenharia DevOps deixa de acontecer, visto que uma plataforma de autosserviço passa a ser a principal interface de interação com os clientes internos.

Figura 4 – Fluxo Proposto



Fonte: Próprio Autor

6 CONSIDERAÇÕES FINAIS

Implementar *pipelines* para requisitos não funcionais se mostrou essencial para garantir o cumprimento do nível de serviço, SLA, com o cliente além de permitir que a equipe de Engenharia DevOps pudesse desenhar, propor e implementar uma solução mais robusta de entrega contínua e colaborar na transformação digital desejada pelo cliente, alinhado à cultura DevOps.

ISSN: 2526-6241

Percebe-se que, apesar da utilização de ferramentas comuns a outras empresas, as soluções de CD são exclusivas e não um modelo exato de como tudo deve ser feito. As práticas que sustentam essa abordagem são agnósticas à tecnologia, de forma que a escolha das mesmas se alinha aos objetivos de negócio, familiaridade com o uso e modelo de suporte com o fornecedor.

Os objetivos com o fluxo alternativo proposto são o de centralizar a gestão de configuração e automatizá-la de maneira a garantir cobertura de testes sobre a mesma e eliminar a ocorrência de incidentes nos ambientes produtivos em razão da ausência de configuração ou de falta de validação da mesma.

Ainda a respeito da Lei de Conway, a opção por tornar a configuração e o processo de implantação algo transparente através de *pipelines* disponibilizados via Jenkins, visa diminuir uma barreira a mais de complexidade durante o processo de entrega de *software*. Utilizar uma ferramenta já existente e de conhecimento comum busca, ao mesmo tempo, eliminar a resistência em relação ao fluxo alternativo proposto e consolidar o conhecimento coletivo, de forma a unir as equipes que dependem do mesmo durante o ciclo de vida de suas atividades.

ISSN: 2526-6241

REFERÊNCIAS

DAVIS, J.; DANIEL, K. ***Efffective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale***. 1.ed. California: O'REILLY, 2016. 410p.

HUMBLE, J.; FARLEY, D. **Entrega Contínua: como entregar *software* de forma rápida e confiável**. 1.ed. Rio Grande do Sul: BOOKMAN, 2014. 464p.

KIM, G. et al. ***The DevOps Handbook: How to create world-class agility, reliability, and security in technology organizations***. 1.ed. Oregon: IT REVOLUTION PRESS, 2016. 480p.

BECK, K. et al. ***Manifesto for Agile software development***. Disponível em: <https://agilemanifesto.org> Acesso em: 20 set. 2017.

MORRIS, K. ***ImmutableServer***. Disponível em: < https://martinfowler.com/bliki/ImmutableServer.html> Acesso em: 07 nov. 2017.

FOWLER, M. ***InfrastructureAsCode***. Disponível em: < https://martinfowler.com/bliki/InfrastructureAsCode.html> Acesso em: 07 nov. 2017

CONWAY, M. ***How Do Committees Invent?***. Disponível em: <http://www.melconway.com/Home/pdf/committees.pdf> Acesso em: 07 nov. 2017

RED HAT, OpenShift Blog. ***How to Avoid Cloud Vendor Lock-In when Evaluatin PaaS***. Disponível em: <https://blog.openshift.com/how-to-avoid-cloud-vendor-lock-in-when-evaluating-a-paas/>Acesso em: 26 out. 2017.