DEVOPS

Marcos Vinicius Bião Cerqueira

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Introdução ao DevOps

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Descrever os principais aspectos da cultura de DevOps.
- Definir os conceitos básicos relativos à infraestrutura ágil.
- Demonstrar os efeitos da aplicação de integração e de entrega contínuas.

Introdução

A ideia central do DevOps está na organização, simplificação e automação de processos, abrangendo desde as etapas de desenvolvimento de *software*, passando pelo ciclo de subida do código ao servidor até o estágio de configuração do servidor. Essa é uma prática que ainda é pouco utilizada pelas empresas, porém ela possui a capacidade de elevar a qualidade do sistema produzido e alcançar um maior controle do código.

Neste capítulo, você conhecerá o DevOps, bem como os diversos conceitos que o cercam e as metodologias que contribuem para a sua continuidade.

1 A cultura DevOps

Antigamente, grandes empresas possuíam equipes especializadas para cada uma das etapas do desenvolvimento de um *software*. A equipe de desenvolvedores era responsável pelo planejamento arquitetural, o cronograma das atividades e a codificação do projeto. O grupo de testes, por sua vez, era incumbido de analisar o *software* em busca de erros. Por fim, o setor de infraestrutura ficava encarregado dos servidores, tanto dos *updates* das atualizações quanto da configuração dos servidores em si.

Cada uma dessas equipes conseguia trabalhar de forma fluida, e os seus membros contribuíam uns com os outros, a fim de elucidar os problemas que surgiam, aprender novas tecnologias e pensar em formas inovadoras de realizar o seu trabalho. Obviamente, há organizações em que os profissionais da mesma equipe não se auxiliam, porém será considerada uma situação em que isso não ocorre. O problema maior está na comunicação entre as equipes, visto que, em parte, os colaboradores visam a facilitar e reduzir o seu próprio trabalho, o que acaba tornando o serviço do outro mais complexo.

Por exemplo, imagine que a equipe de desenvolvedores realiza entregas semanais, sendo que cada programador lida com um arquivo diferente do projeto, a fim de evitar problemas de versões. Foi gerado um documento com todos os requisitos, entretanto, não foram documentadas todas as funções que surgiram durante o andamento do projeto. Assim, o grupo de testes recebe o projeto no *pen drive*, sem saber com exatidão as alterações que foram realizadas na semana, podendo, assim, deixar trechos importantes sem serem testados. Por fim, o sistema é colocado no servidor de produção, pois não apresentava mais erros. A quantidade de problemas que podem surgir nessa situação é alarmante, os quais, muitas vezes, acabam gerando retrabalho para corrigir falhas que poderiam ter sido evitadas.

Fique atento

A periodicidade de entregas varia de acordo com a metodologia de desenvolvimento utilizada na empresa. Contudo, independentemente da frequência, os mesmos problemas podem vir a ocorrer.

O termo **DevOps** surgiu com a junção das áreas de desenvolvimento e operação (FARROCHA, 2014). Contudo, isso não significa que o conhecimento fica restrito a esses dois campos, uma vez que diversas áreas acabam sendo correlacionadas com as práticas dessa metodologia.

Há muito tempo se ouve falar sobre desenvolvimento ágil e, em 2001, houve uma elaboração de um documento por parte de diversos especialistas da área, chamado de **Manifesto Ágil**, que contém os princípios que fundamentam o desenvolvimento de *software* (MATTIOLI *et al*., 2009). O evento Agile 2008 abriu caminhos para o DevOps, com discussões e debates acerca de infraestrutura ágil. Entretanto, o termo DevOps só surgiu em 2009, com o objetivo de unir administradores de infraestrutura e programadores, promovendo automação nos processos de integração e entrega.

Fique atento

Ao contrário da metodologia ágil, o DevOps não possui um documento similar ao Manifesto Ágil. Dessa forma, mesmo sendo conhecido mundialmente, cada empresa adapta o conceito de acordo com as suas necessidades.

A ideia central por trás do DevOps é ter processos bem definidos, automatizar ações e possibilitar cooperação mútua. Dessa forma, os profissionais ficam livres para focar as suas atividades em soluções inteligentes (MUELLER, 2016). Com a liderança e o uso dos *software* corretos, é possível viabilizar um ambiente onde desenvolvedores e administradores de infraestrutura trabalhem em harmonia. A eficiência da cultura DevOps pode ser percebida quando a organização compreende o seu funcionamento.

A Figura 1, a seguir, apresenta as áreas que se relacionam, formando o DevOps. Logicamente, cada uma dessas áreas sofre influências de outras disciplinas, o que enriquece ainda mais as práticas utilizadas.

Figura 1. Áreas correlacionadas que formam o DevOps.
Fonte: Adaptada de Traversin (2017).

Cada uma dessas áreas possui suas próprias metodologias, problemas e processos, de modo que, independentemente do setor, os profissionais devem estar aptos a resolver as problemáticas que surgem no cotidiano. Vale lembrar que a intenção do DevOps é criar formas de interação entre as equipes, para que a passagem de atividades sejam simplificadas.

2 Conceitos básicos

O DevOps é um movimento profissional que objetiva o apoio, a comunicação e a integração entre as pessoas envolvidas no processo de desenvolvimento de um projeto (TRAVERSIN, 2017). Nesse sentido, a padronização e a automação das ações são imprescindíveis para a manutenção do trabalho.

Conceitualmente, o DevOps é uma metodologia de desenvolvimento de *software* que interage diretamente com profissionais de infraestrutura. Diversas ideias giram em torno dessa cultura, propiciando o bom trabalho dos profissionais, sendo as principais ideias apresentadas a seguir.

Automação

Muitas empresas possuem um ritmo de entrega acelerado e, a cada atualização, é necessário realizar diferentes ações para garantir a qualidade do *software* produzido. Os processos realizados manualmente dificultam a entrega mais rápida e baixam a produtividade dos colaboradores por estarem presos àquela atividade (TRAVERSIN, 2017), de modo que a automação dos processos é indispensável para o DevOps.

Quando os processos são automatizados, eles acabam ficando repetitivos, o que permite os entender mais facilmente, descomplica os processos de auditorias e possibilita melhorá-los sem grandes problemas (TRAVERSIN, 2017).

Qualidade

De modo geral, por meio do *software*, as empresas buscam propiciar informações, valores e experiências para os usuários. Muitas vezes, o *software* é o ponto-chave de uma empresa, gerando resultados e benefícios para os clientes. Portanto, a qualidade do sistema deve ser primordial na sua construção, o que, frequentemente, é difícil de ser alcançado, pois existem muitas barreiras a serem superadas (FORRESTER, 2016).

Em algumas situações, qualidade e agilidade são fatores inversamente proporcionais, entretanto, o DevOps faz essas duas características andarem lado a lado. A ideia em torno desse aspecto está em reduzir o tamanho das entregas, para que elas sejam realizadas mais rapidamente, com maior controle e menor risco associado, criando, assim, um bom nível de qualidade em toda a cadeia de entrega do *software* (MICRO FOCUS, 2018).

No entanto, existe um impacto negativo na qualidade quando a expectativa do usuário não é alcançada. De acordo com Micro Focus (2018), os requisitos do usuário devem estar sincronizados com os testes, para garantir que todas as exigências dos usuários sejam atendidas.

Produtividade

Em alguns casos, é possível observar alguns pontos que, inicialmente, parecerão falar do mesmo conteúdo, mas, na verdade, tratam de aspectos diferentes, ou até mesmo de causa e efeito. Por exemplo, ao automatizar processos, tem-se uma menor carga de trabalho e mais tempo para realizar outras atividades. Como consequência, é possível aumentar a produtividade.

Com a eliminação de trabalhos cansativos, repetitivos e enfadonhos, os colaboradores conseguem otimizar o período de trabalho. Dessa forma, é possível utilizar esse tempo para dar continuidade ao desenvolvimento do projeto, reduzir o prazo de entrega, realizar intervenções proativas ou estudar uma nova tecnologia.

Segurança

Um dos pontos principais de qualquer tecnologia é a segurança, visto que, sem ela, a retenção dos usuários torna-se uma tarefa árdua. Desse modo, faz-se necessário manter ações que analisem possíveis falhas no código, bem como retroceder para momentos estáveis do *software*, se houver erros graves.

A possibilidade de monitoramento, registro e análise em tempo real das funções do *software* contribui para o diagnóstico de erros. Isso aumenta a velocidade da resolução de problemas e possibilita entrega contínua, permitindo que a equipe responsável realize mais testes.

As técnicas utilizadas no DevOps permitem definir ações para serem executadas ao fim do processo de produção. Após a criação do *software*, testes são realizados, garantindo, assim, a segurança operacional e a satisfação do cliente.

Colaboração

Quando uma organização opta por trabalhar com metodologia DevOps, o fluxo de trabalho torna-se mais constante. Isso ocorre em virtude de os processos serem bem definidos e porque cada equipe conhece as responsabilidades das outras. Desse modo, na definição dos processos, tudo é pensado de forma a facilitar o trabalho do próximo.

Fique atento

É importante que todas as pessoas envolvidas na organização tenham conhecimento das etapas de criação, envio e execução da aplicação. Contudo, isso não significa que os desenvolvedores sejam bons operadores, e vice-versa, mas sim que eles devem entender as atividades do outro e colaborar para concluir cada novo ciclo do *software*.

3 Integração e entrega contínuas

A integração contínua (CI, *continuous integration*) e a entrega contínua (CD, *continuous delivery*) são práticas da engenharia de *software* que visam a testar e disponibilizar o *software* de forma automatizada, contínua e consistente (DESTRO; FRANÇA, 2019). Essas ações são aplicadas a projetos de desenvolvimento de *software*, com o intuito de manter o nível de qualidade elevado, a estabilidade da aplicação e propiciar uma menor quantidade de erros.

Essas práticas são fixadas em um conjunto, intitulado *pipeline*, e organizadas de forma lógica e sequencial, conforme a Figura 2.

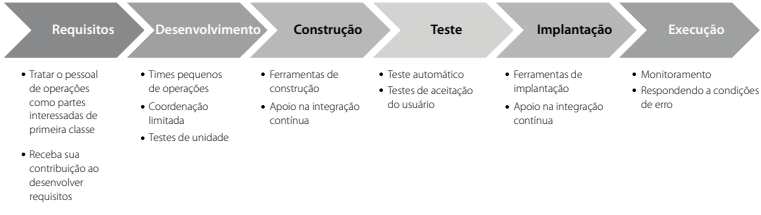

Figura 2. Abstração do *pipeline* DevOps.

Fonte: Adaptada de Souza Neto (2018).

O *pipeline* pode variar de acordo com o projeto, considerando-se que cada sistema suporta tecnologias distintas e possui ferramentas específicas para realizar diferentes ações (DESTRO; FRANÇA, 2019). A metodologia ágil recomenda que o processo de desenvolvimento de *software* seja executado com maior frequência de entregas. Como resultado, o trabalho necessário para coletar, integrar e testar todo o código desenvolvido pela equipe no repositório central se tornou mais frequente.

Assim, a CI foi introduzida para limitar o trabalho manual que era necessário para o processo de integração do sistema. Além disso, acrescentou-se o processo de testes automatizados, garantindo o funcionamento do sistema após cada atualização. Para adotar essa prática, é essencial utilizar ferramentas de controle de versão, como GitHub e GitLab, a fim de obter garantia do controle entre as versões dos códigos, possibilidade de retorno para versões anteriores e viabilidade de diferentes programadores trabalharem no mesmo código através de *branches* distintas.

Antes de falar sobre CD, é preciso entender os tipos de servidores. O primeiro tipo é o **servidor de teste**, que possui recursos limitados e é utilizado para realizar os testes de forma mais controlada. O segundo é o **servidor de homologação**, que é basicamente uma cópia do servidor de produção, possuindo integração com todas as APIs (*application programming interface*; ou interface de programação de aplicações, em português) utilizadas. Com isso, os testes são realizados de forma mais fidedigna. Por fim, tem-se o **servidor de produção**, que se refere ao ambiente final que é acessado pelos usuários.

A CD é uma evolução da CI. Com ela, além das ações apresentadas anteriormente, é possível garantir que o código esteja pronto para ir ao servidor de produção. Para tanto, existe um *deploy* automatizado que sobe o código para o servidor de homologação, realiza os testes pré-programados e, caso não ocorra nenhum erro, encaminha o *software* para o servidor de produção. A Figura 3, a seguir, apresenta um detalhamento da divisão de tarefas e responsabilidades das práticas DevOps.

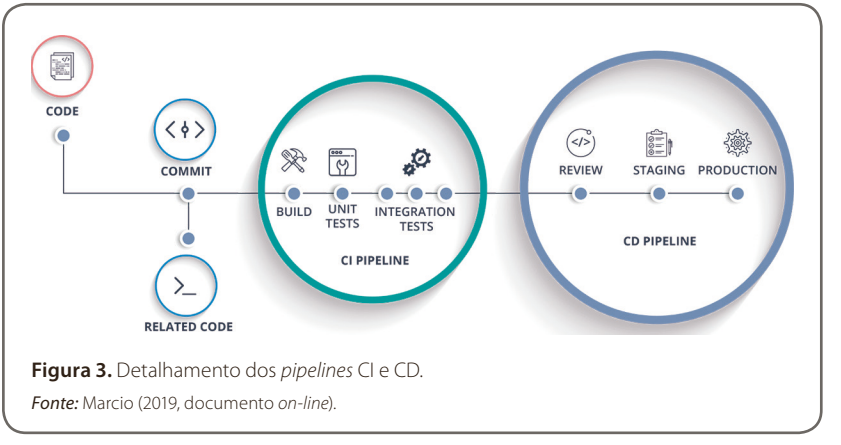

Figura 3. Detalhamento dos *pipelines* CI e CD.
Fonte: Marcio (2019, documento *on-line*).

Para os profissionais, trabalhar com CI e CD implica diminuição do trabalho, redução de erros e entregas rápidas. Já para a organização, o tempo necessário para lançar uma atualização é encurtado. Além disso, os usuário recebem novos *updates* com maior frequência.

Referências

DESTRO, G. A.; FRANÇA, B. B. N. *Avaliação do nível da aplicação de práticas de entrega e integração contínua em repositórios de código aberto*. 2019. Projeto Final (Graduação) – Universidade Estadual de Campinas, Campinas, 2019. Disponível em: https://www.ic.unicamp.br/~reltech/PFG/2019/PFG-19-31.pdf. Acesso em: 31 jul. 2020.

FORRESTER, A. *Accelerate your DevOps to awesome*: learn how to take your DevOps to the next level. London: Forrester, 2016.

MARCIO. *CI/CD*: continuous integration and continuous delivery. [*S. l.*]: Medium, 2019. Disponível em: https://medium.com/tecnologia-e-afins/ci-cd-continuous-integration--and-continuous-delivery-fb5d0aed4bf5. Acesso em: 31 jul. 2020.

MATTIOLI, F. E. R. *et al*. Uma proposta para o desenvolvimento ágil de ambientes virtuais. *In*: WORKSHOP DE REALIDADE VIRTUALE AUMENTADA, 6., 2009, *Santos*. *Anais* [...]. Santos: SBC, 2009.

MICRO FOCUS. *Measuring DevOps success*: how do you know DevOps is working? Watch these KPIs. Berkshire: Micro Focus, 2018. Disponível em: http://files.asset.microfocus.com/4aa6-3036/en/4aa6-3036.pdf. Acesso em: 9 jul. 2020.

SOUZA NETO, P. J. *Integração e entrega contínua para aplicações móveis desenvolvidas em React Native*. 2018. Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Pernambuco, Recife, 2018. Disponível em: https://www.cin.ufpe.br/~tg/2018-2/TG_SI/pjsn.pdf. Acesso em: 31 jul. 2020.

TRAVERSIN, G. *DevOps*: habilidades e capacidades necessárias para sua utilização. 2017. Monografia (Especialização) – Universidade de Taubaté, Taubaté, 2017. Disponível em: http://repositorio.unitau.br:8080/jspui/bitstream/20.500.11874/1246/1/GUSTAVO%20TRAVERSIN.pdf. Acesso em: 31 jul. 2020.

Leituras recomendadas

BOAGLIO, F. *Jenkins*: automatize tudo sem complicações. 2. ed. São Paulo: Casa do Código, 2019.

KIM, G. *et al*. *Manual de DevOps*: como obter agilidade, confiabilidade e segurança em organizações tecnológicas. Rio de Janeiro: Alta Books, 2018.

MORAES, G. *Caixa de ferramentas DevOps*: um guia para construção, administração e arquitetura de sistemas modernos. São Paulo: Casa do Código, 2015.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS