



DEVOPS E GESTÃO DO CICLO DE VIDA DE APLICAÇÕES

BRASÍLIA-DF.

Elaboração

Jorge Umberto Scatolin Marques

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

Sumário

APRESENTAÇÃO.....	5
ORGANIZAÇÃO DO CADERNO DE ESTUDOS E PESQUISA	6
INTRODUÇÃO.....	8
UNIDADE I	
INTRODUÇÃO	9
CAPÍTULO 1	
O QUE É DEVOPS.....	9
CAPÍTULO 2	
MITOS E VERDADES	16
CAPÍTULO 3	
DIFERENÇA ENTRE O CICLO TRADICIONAL E DEVOPS	21
CAPÍTULO 4	
DO LEAN AO DEVOPS	28
UNIDADE II	
PROCESSOS ÁGEIS E MÉTODOS LEAN.....	35
CAPÍTULO 1	
LEAN.....	35
CAPÍTULO 2	
DEVOPS E KANBAN	41
CAPÍTULO 3	
CORDA DE ANDON	45
CAPÍTULO 4	
MAPA DE FLUXO DE VALOR	49
UNIDADE III	
CULTURA DEVOPS.....	56
CAPÍTULO 1	
CULTURA COLABORATIVA	56
CAPÍTULO 2	
BENEFÍCIOS	63

CAPÍTULO 3	
ESTRUTURA DE EQUIPES.....	69
CAPÍTULO 4	
ETAPAS.....	76
UNIDADE IV	
ESTRUTURA E IMPLEMENTAÇÃO	81
CAPÍTULO 1	
CONCEITO DE INFRAESTRUTURA	81
CAPÍTULO 2	
PIPELINES	88
CAPÍTULO 3	
MUDANÇAS.....	93
CAPÍTULO 4	
SEGURANÇA DA INFORMAÇÃO NA INTEGRAÇÃO, ENTREGA E IMPLANTAÇÃO	97
REFERÊNCIAS	100

Apresentação

Caro aluno

A proposta editorial deste Caderno de Estudos e Pesquisa reúne elementos que se entendem necessários para o desenvolvimento do estudo com segurança e qualidade. Caracteriza-se pela atualidade, dinâmica e pertinência de seu conteúdo, bem como pela interatividade e modernidade de sua estrutura formal, adequadas à metodologia da Educação a Distância – EaD.

Pretende-se, com este material, levá-lo à reflexão e à compreensão da pluralidade dos conhecimentos a serem oferecidos, possibilitando-lhe ampliar conceitos específicos da área e atuar de forma competente e conscienciosa, como convém ao profissional que busca a formação continuada para vencer os desafios que a evolução científico-tecnológica impõe ao mundo contemporâneo.

Elaborou-se a presente publicação com a intenção de torná-la subsídio valioso, de modo a facilitar sua caminhada na trajetória a ser percorrida tanto na vida pessoal quanto na profissional. Utilize-a como instrumento para seu sucesso na carreira.

Conselho Editorial

Organização do Caderno de Estudos e Pesquisa

Para facilitar seu estudo, os conteúdos são organizados em unidades, subdivididas em capítulos, de forma didática, objetiva e coerente. Eles serão abordados por meio de textos básicos, com questões para reflexão, entre outros recursos editoriais que visam tornar sua leitura mais agradável. Ao final, serão indicadas, também, fontes de consulta para aprofundar seus estudos com leituras e pesquisas complementares.

A seguir, apresentamos uma breve descrição dos ícones utilizados na organização dos Cadernos de Estudos e Pesquisa.



Provocação

Textos que buscam instigar o aluno a refletir sobre determinado assunto antes mesmo de iniciar sua leitura ou após algum trecho pertinente para o autor conteudista.



Para refletir

Questões inseridas no decorrer do estudo a fim de que o aluno faça uma pausa e reflita sobre o conteúdo estudado ou temas que o ajudem em seu raciocínio. É importante que ele verifique seus conhecimentos, suas experiências e seus sentimentos. As reflexões são o ponto de partida para a construção de suas conclusões.



Sugestão de estudo complementar

Sugestões de leituras adicionais, filmes e sites para aprofundamento do estudo, discussões em fóruns ou encontros presenciais quando for o caso.



Atenção

Chamadas para alertar detalhes/tópicos importantes que contribuam para a síntese/conclusão do assunto abordado.

**Saiba mais**

Informações complementares para elucidar a construção das sínteses/conclusões sobre o assunto abordado.

**Sintetizando**

Trecho que busca resumir informações relevantes do conteúdo, facilitando o entendimento pelo aluno sobre trechos mais complexos.

**Para (não) finalizar**

Texto integrador, ao final do módulo, que motiva o aluno a continuar a aprendizagem ou estimula ponderações complementares sobre o módulo estudado.

Introdução

Houve um período em que os atrasos nos grandes projetos de *software* eram frequentes. Por muito tempo um desses fatores ficou obscuro, até que alguém percebeu que o problema estava no relacionamento das atividades do time de desenvolvimento e operações. De início, foi constatado que os desenvolvedores possuíam técnicas que foram evoluindo constantemente, e o mesmo não acontecia com os conceitos do lado da Operação, que é responsável pela infraestrutura. Por isso desentendimentos ocorriam a cada pedido de criação de ambientes, alteração de infra, *hardware* ou *software*.

Entretanto com os serviços de datacenters ficando mais baratos, a internet mais acessível, as grandes empresas trataram logo de disponibilizar ao pessoal de Operações uma ótima opção: atender com eficiência o time de desenvolvimento. Na ocasião, ambos os times estariam supridos de ferramentas, porém, faltava algo para que se entendessem na utilização colaborativa dessas opções. É aí que surge o conceito do DevOps.

O conceito do DevOps veio para organizar os times de desenvolvimento e operações, fazendo ambos participarem de algo mais colaborativo, focados em entregas de valor ao cliente, sem atrasos e de acordo com o projeto ofertado.

Pensar em DevOps é algo que muda completamente a cultura da empresas. Pensar sobre o conceito da cultura colaborativa do DevOps não será algo fácil de se implementar em empresas que possuem os times em posições distintas, porém o esforço valerá muito a pena, pois os clientes serão satisfeitos e terão seu produto entregue conforme o combinado. DevOps é uma cultura que dá às organizações a capacidade de responder às necessidades de mercado em constante mudança e expansão.

Objetivos

- » Compreender as melhores práticas de compor uma cultura colaborativa no desenvolvimento de *software*.
- » Estudar as principais técnicas e boas práticas de DevOps.
- » Entender técnicas para elaborar um produto de valor ao cliente.

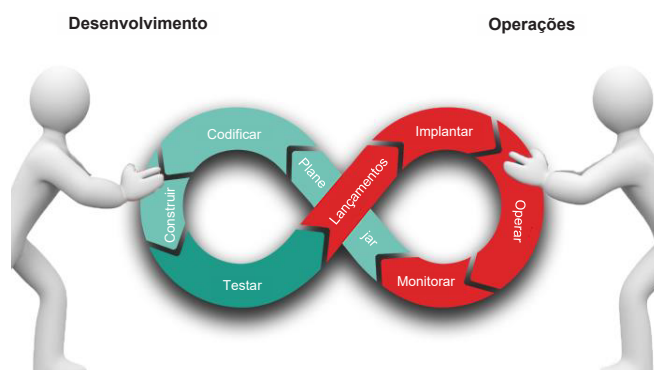
CAPÍTULO 1

O que é DevOps

Antes de começar qualquer explicação sobre DevOps, vamos traduzir literalmente essa palavra, que significa *Developers & Operation*. Isso mesmo, Desenvolvimento e Operação, entidades distintas dentro de uma mesma organização. O time de desenvolvimento é composto pelos desenvolvedores do *software*, e o time de operações é composto pelo pessoal da infraestrutura de TI. Esses dois times começaram a ter destaques bem como a percepção de que algo deveria ser feito. Tudo começou quando os desenvolvedores, apoiados pela grande evolução dos movimentos ágeis de entrega de um produto funcional, esbarraram nas opções que o time de operações tinha em disponibilizar ambientes totalmente funcionais à mesma velocidade que os desenvolvedores precisavam. Enquanto o time de operações desejava uma produção estável e segura, o time de desenvolvimento queria, a qualquer custo, implantar seus projetos.

Esse conflito estaria dando muito prejuízo às organizações, e a cobrança por algo mais produtivo foi cada vez mais necessária. É aí que entrou o conceito colaborativo do DevOps.

Figura 1. Colaboração em DevOps.

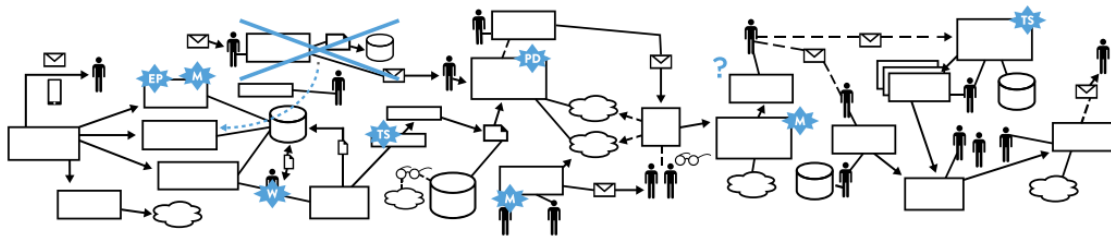


Fonte: <http://www.agilebuddha.com/wp-content/uploads/2016/12/Devops-Team.jpg>.

O DevOps não é um conjunto de ferramentas ou métodos, mas um conceito que ajudou a integrar todas as funções do desenvolvimento de *software* com o ciclo operacional. Isso, de fato, exigiu muito mais gerenciamento e coordenação entre os envolvidos: Desenvolvimento, garantia de qualidade e operações. Quando esses times se organizam por meio da compreensão das preocupações e visões de cada um, eles são capazes de construir e distribuir produtos de *software* em um ritmo acelerado.

Kim *et al.* (2018, p. 10) “encontravam em empresas situações em que o tempo de implementação exigiam meses. Isso acontecia, pois as organizações eram grandes e complexas, trabalhavam com aplicações monolíticas, fortemente acopladas, tempos longos em ambientes de testes e várias etapas para aprovações”. O fluxo de valor era muito bem representado pela figura a seguir:

Figura 2. Um fluxo de valor tecnológico com tempo de execução de implementação de três meses.

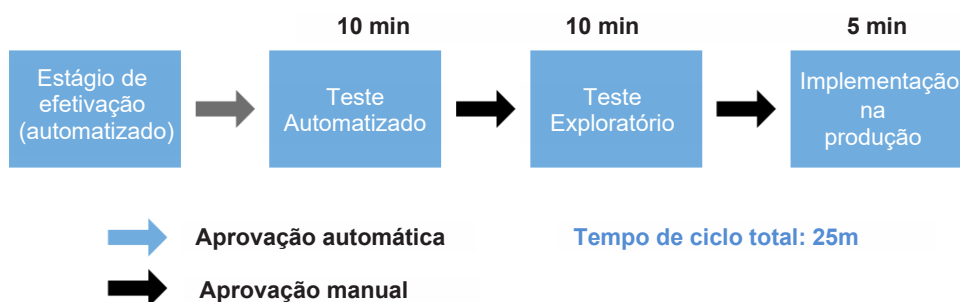


Fonte: Kim *et al.* (2018, p. 11).

Houve, portanto, a necessidade de se criar ambientes de infraestrutura mais dinâmicos para suprir a necessidade dos desenvolvedores. E esse objetivo foi prontamente atendido, porém o custo para manter essa infraestrutura dentro das organizações era muito alto.

No DevOps, o *feedback* aos desenvolvedores é muito mais rápido e constante, permitindo a integração e validação do código rápida.

Figura 3. Um fluxo de valor tecnológico com tempo de execução de minutos.



Fonte: Kim *et al.* (2018, p. 11).

Criou-se aí mais uma oportunidade que as grandes empresas não deixaram passar. Todas essas empresas possuem produtos que são utilizados na implementação da cultura DevOps. À medida que formos avançando, citaremos algumas dessas ferramentas.

Para oferecer seu portfólio de produtos e serviços, elas reforçam o conceito do DevOps, porém sempre tentam explicar o conceito por meio dos serviços prestados, como a *Amazon*, em <https://aws.amazon.com/pt/devops/what-is-devops>. Acesso em 8/11/2019:

O DevOps é a combinação de filosofias culturais, práticas e ferramentas que aceleram a capacidade de uma empresa de distribuir aplicativos e serviços em alta velocidade: otimizando e aperfeiçoando produtos em um ritmo mais rápido do que o das empresas que usam processos tradicionais de desenvolvimento de software e gerenciamento de infraestrutura.

A Atalssian, empresa que tem em seu portfólio serviços de versionamento de códigos, visualizado pelo link <https://www.atlassian.com/br/devops#!team-morale>, acessado em 08/11/2019:

DevOps é um conjunto de práticas que automatiza os processos entre as equipes de desenvolvimento de software e de TI, para que possam criar, testar e liberar software de maneira mais rápida e confiável. O conceito de DevOps é fundado na construção de uma cultura de colaboração entre equipes que historicamente funcionavam em silos relativos.

Uma das pioneiras no mundo Linux, a Redhat, atualmente adquirida pela IBM, através do link <https://www.redhat.com/pt-br/topics/devops>. Acesso em 8/11/2019, define DevOps:

A metodologia DevOps é uma abordagem de cultura, automação e design de plataforma com o objetivo de agregar mais valor aos negócios e aumentar a capacidade de resposta por meio da entrega de serviços rápida e de alta qualidade. Tudo isso é possível por meio da disponibilização de serviços de TI rápidos e iterativos. Adotar o DevOps significa conectar aplicações legadas a uma infraestrutura e aplicações modernas e nativas da *cloud*.

Umas das gigantes da tecnologia, a Microsoft, também não poderia ficar fora dessa lista, explicando sobre DevOps no link <https://azure.microsoft.com/pt-br/overview/what-is-devops/>. Acesso em 8/11/2019:

A O DevOps é a união de pessoas, processos e tecnologias a fim de proporcionar a entrega contínua do valor para os clientes. DevOps, termo composto por **dev** (de *development*, ou desenvolvimento) e **ops** (*operations*, ou operações), é uma prática de desenvolvimento de software que unifica operações de desenvolvimento e TI.

E por fim, a precursora em matéria de criação de tendências e tecnologias, IBM, podendo ser visualizado pelo link <https://www.ibm.com/br-pt/cloud/devops>. Acesso em 8/11/2019:

O DevOps é uma abordagem cada vez mais comum ao desenvolvimento ágil de software que os desenvolvedores e as equipes de operações usam para criar, testar, implantar e monitorar aplicativos com velocidade, qualidade e controle.

As três maneiras

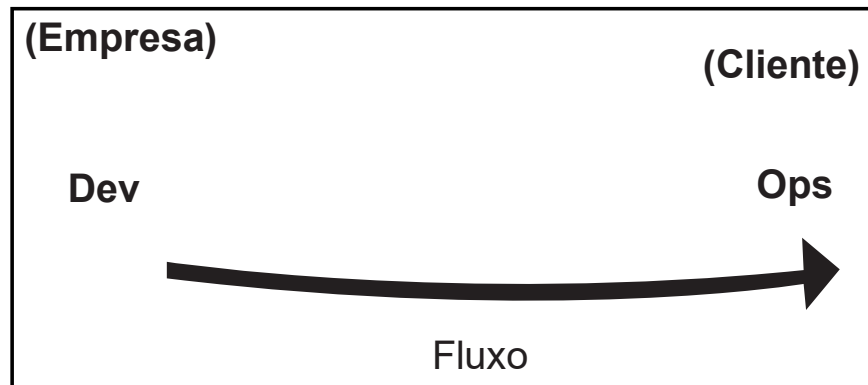
A cultura DevOps ainda conta com três princípios citados no Projeto Fênix, também conhecidos como as “Três Maneiras”:

A primeira maneira

A primeira maneira diz que o trabalho de desenvolvimento deve ser acelerado da operação para o cliente, ou, segundo a figura abaixo, da esquerda para a direita. Seu objetivo é aumentar continuamente o fluxo do desenvolvimento para as operações e minimizar qualquer fluxo de retrabalho, ou seja, o retorno. O time de operações deverá ser envolvido desde o início do desenvolvimento para que o ambiente de produção possa ser criado, alterado ou configurado sempre que necessário.

No desenvolvimento de *software*, lançar versões de *softwares* menores, e com mais frequência, reduz o desperdício e melhora seu tempo no mercado. A ideia é mover o produto mais rápido possível para a direita.

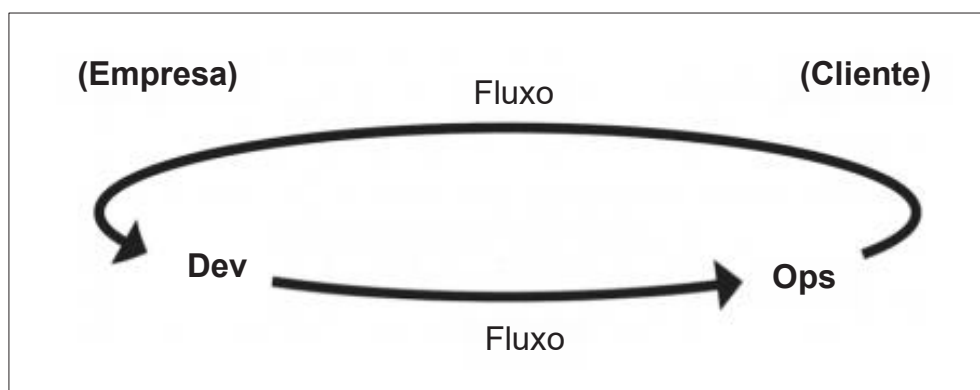
Figura 4. Fluxo de trabalho da primeira maneira.

Fonte: Kim *et al.* (2018, p.12).

A segunda maneira

A segunda maneira cria os ciclos de *feedback* da direita para a esquerda. O objetivo é encurtar e ampliar os ciclos de *feedback* para que as correções necessárias possam ser feitas continuamente. Isso exige que aumentemos o *feedback* para evitar o retrabalho e possibilitar uma detecção mais rápida. Também é possível medir o fluxo de trabalho realizado, ou seja, se a duração dos ciclos for medida constantemente, *feedback* possibilitará reduzi-los.

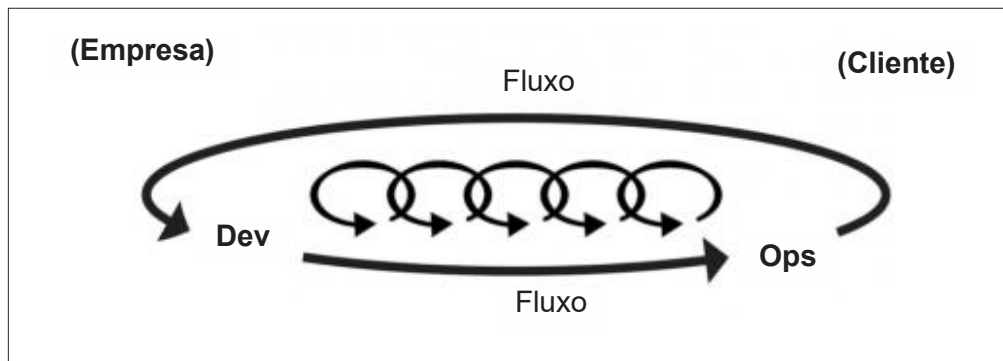
Figura 5. Fluxo de Trabalho da segunda maneira.

Fonte: Kim *et al.* (2018, p.12).

A terceira maneira

Segundo Kim *et al.* (2018, p.12), “a terceira via permite a criação de uma cultura de alta confiança, que suporta uma abordagem dinâmica, disciplinada e científica, à experimentação e tomada de riscos, facilitando a criação de aprendizado organizacional, tanto de nossos sucessos como fracassos.”

Figura 6. Fluxo de trabalho da terceira maneira.

Fonte: Kim *et al.* (2018, p.12).

Implantações frequentes ou contínuas precisam de disciplina em todos os níveis e compromisso para que isso aconteça. Você também pode ter reuniões entre os desenvolvedores e a equipe de operações antes da implantação, mesmo que as equipes de desenvolvimento e operações estiverem geograficamente dispersas. A agenda da equipe de operações tem como objetivo manter o sistema em um tempo mínimo e, nesse processo, alguns erros críticos não são resolvidos em tempo hábil ou são ignorados. O DevOps, uma vez instalado, pode eliminar todos esses fatores e tornar a jornada de implantação muito suave.

Normalmente, os *releases* planejados são implementados por meio de um conjunto de ferramentas – alguns podem ser automatizados, mas é necessário inserir comandos para acioná-los. Uma ferramenta por si só não significaria nada, mas a utilização dela e a integração de todas as ferramentas automatizadas desde o *check-in* até a implementação são oferecidas pelo DevOps.

O DevOps geralmente funciona da seguinte maneira, usando uma abordagem que enfatiza muito a automação do teste e da implantação.

- » Equipe de desenvolvimento começa a escrever um código.
- » Preparando um ambiente de QA (*Quality Assurance*).
- » Execute casos de teste.
- » Implantação no ambiente de produção.

Concluindo, DevOps é uma cultura em que todos são responsáveis pela entrega contínua, desde os ciclos de planejamento e desenvolvimento de *software* (SDLC) até o monitoramento de produção e melhoria contínua. Basicamente, é uma abordagem multifuncional em andamento, enraizada nos princípios do *Agile* e do *Lean*. Esses dois conceitos veremos em capítulos posteriores.

**DevOps and Segregation of Duties**

<https://medium.com/@jeehad.jebeile/devops-and-segregation-of-duties-9c1a1bea022e>.

My Journey To DevOps

<https://medium.com/@jvftuo/my-journey-to-devops-eb655684a814>.

Noobs Guide: Continuous Integration & Continuous Delivery

<https://medium.com/@brenn.a.hill/noobs-guide-continuous-integration-continuous-delivery-continuous-deployment-d26ac4f2beeb>.

CAPÍTULO 2

Mitos e verdades

À medida que o conceito de DevOps ganha notoriedade, muitos equívocos acontecem acerca de sua popularização. Isso ocorre pelo fato de muitos conceitos básicos, que apareceram muito antes do DevOps, já estão estarem sendo utilizados. É o caso, por exemplo, de entrega contínua, *lean*, *agile*, automação etc. Só lembrando que o DevOps é a cultura que veio amenizar o conflito entre o pessoal do Desenvolvimento e de Operações.

Hand (2015) destacou os 10 principais mitos de DevOps que surgiram após o *DevOps Enterprise Summit*, de 2015:

Utilizando ferramentas faz você DevOps

Lembraremos sempre que DevOps é uma cultura. Fazer DevOps vai muito além de implementação e utilização de ferramentas. Embora ferramentas sejam necessárias para que haja monitoramento, automação e diversas facilitações, as equipes devem entender a mudança cultural por trás do DevOps. Portanto é um mito dizer que somente adotando ferramentas estaremos implementando DevOps, pois seu conceito é mais sobre pessoas do que sobre ferramentas.

DevOps e entrega contínua são a mesma coisa

Quando uma empresa estabelece entrega contínua, indica que ela estabeleceu um dos principais conceitos de DevOps. Entretanto, apesar de aparecerem juntos dentro de um mesmo cenário, a cultura DevOps não foi implementada se cada departamento ou time estiver preocupado somente com suas entregas e não com a totalidade. Portanto é um mito dizer que faz DevOps quem faz entrega contínua e não se atentou para disseminar a cultura DevOps entre os envolvidos.

Entrega contínua é automação

A automação é uma parte importante, ou seja, faz parte do projeto da entrega contínua. Além da automação, a entrega contínua trabalha para criar processos repetíveis, criar ciclos de *feedback*, definir o escopo das mudanças e determinar o que realmente deve ser promovido ou definido como “concluído”.

DevOps significa ir a conferências

Não restam dúvidas de que se participarmos de conferências sobre DevOps, estaremos sempre por dentro das atualidades, novos conceitos, tecnologias emergentes e ferramentas inovadoras e novas tendências. Isso vai ajudá-lo a abrir a cabeça para tentar entender a cultura DevOps. Entretanto, entender a cultura não significa implantar DevOps. Para implantar, é preciso, além de ter o referencial teórico, ser um membro atuante que envolva seu time, sua empresa, seu ambiente para que todos entendam que implementar essa cultura colaborativa é a melhor solução para as empresas. Portanto participar de eventos e estar a par de tudo o que há de novo no conceito, não significa que você estará pronto para utilizar os benefícios do DevOps, pois a implementação requer muito esforço.

Ter DevOps no seu título significa que você está fazendo DevOps

Você pode ser um evangelista DevOps, porém não poderá oferecer DevOps imediatamente a uma empresa quando for solicitado. Da mesma forma, contratar um engenheiro, gerente ou consultor de DevOps não significa que você tenha adquirido o DevOps.

Aqueles que possuem DevOps em seu título, esperançosamente, estão tomando todas as medidas necessárias para começar a transformar a cultura para criar empatia em toda a organização. Essa empatia levará a um entendimento mais profundo de como as coisas são feitas e como todos fazem parte não apenas da entrega de *software*, mas também da manutenção do código e da infraestrutura em que ele reside.

Entrega contínua significa liberar o *software* a cada cinco minutos

O termo “contínuo” pode se tornar ambíguo quando mal interpretado. Entregar continuamente significa atingir um nível de confiança nos processos de entrega que lhe permite lançar novos *softwares* quando necessário. Isso não tem uma regra de ser de cinco em cinco horas, ou a cada semana, mensalmente. Nem mesmo os mais brilhantes praticantes de DevOps, como *Amazon*, *Netflix* e *Etsy*, estão entregando novas versões de seus *softwares* continuamente. Por exemplo, o lema do *Facebook* é “Navegue com frequência”. Isso significa que não tem um período de tempo específico. A qualquer momento, o *Facebook* pode implementar novos aprimoramentos ou alterações em seu serviço por diversos motivos.

A implementação de DevOps exige equipes de DevOps

Na verdade, o DevOps trabalha justamente para quebrar silos e não criar a necessidade de formar novas equipes. O DevOps faz com que as diversas equipes conversem entre si, ou seja, a equipe de negócios converse com a equipe de desenvolvimento, e que a equipe de desenvolvimento converse com a equipe de operações, etc. Todos devem conversar! Você não pode conseguir isso criando uma nova equipe que precisa falar com a equipe de desenvolvimento, a equipe de operações e todos os outros.

DevOps é apenas para engenheiros e operadores

Vendas, marketing, suporte técnico e até mesmo altos executivos de nível devem comprar a importância de criar uma cultura de automação, compartilhamento e medição. É muito importante ver uma organização inteira encontrar um senso de conhecimento em conjunto, consciência situacional e, o mais importante, empatia. Ao trazer equipes adicionais para o DevOps, melhorias ainda maiores são encontradas mais profundamente na cultura da organização.

Kim *et al.* (2018) também possui uma lista sobre o que é mito em relação à prática do DevOps:

DevOps só serve para *startup*

“Embora as práticas de DevOps tenham sido exploradas por empresas como Google, Amazon, Netflix, em algum momento, essas empresas correram o risco por possuírem as práticas tradicionais. Porém elas conseguiram transformar sua arquitetura, técnicas e cultura para criar resultados DevOps” (KIM *et al.*, 2018, p. xii).

DevOps substitui o ágil

Ao contrário do que muitos acreditam, o DevOps é uma continuação da jornada ágil. Seus princípios são compatíveis com a técnica ágil que, frequentemente, serve como facilitador eficiente para DevOps, graças ao enfoque em pequenas equipes que entregam produto de qualidade e com frequência.

DevOps é incompatível com ITIL

Apesar de muitos verem no DevOps um retrocesso ao ITIL, para suportar os tempos de execução mais curtos e as frequências de implementação mãos altas, muitas áreas de processos ITIL estão sendo totalmente automatizadas para resolver problemas dos processos de gerenciamento da configuração e *release*. Como a cultura DevOps exige uma detecção e recuperação rápidas sobre incidentes, esses processos do ITIL permanecem relevantes.

DevOps é incompatível com segurança da informação e conformidade

Por não existir controles tradicionais, muitos profissionais de segurança ficam apavorados. O que na verdade acontece é que em vez das atividades de segurança e conformidade serem executadas apenas no final do projeto, controles são feitos em cada estágio do ciclo de desenvolvimento de software. Isso oferece melhor qualidade, segurança e conformidade.

DevOps significa eliminar operações de TI

É certo que a natureza das operações de TI vai mudar. No ciclo de vida do *software*, as Operações colaboram muito antes com o desenvolvimento, que continua trabalhando com as operações de TI muito tempo depois do código já estar em produção. O que vai acontecer, de fato, é o aproveitamento da cultura colaborativa e as automatizações propostas pelo DevOps, e, como já foi dito, a quebra de paradigmas gera esse tipo de pensamento.

Fatos de DevOps

Alguns números da pesquisa para focar <https://puppet.com/resources/whitepaper/state-of-devops-report>:

- » 66% das organizações pesquisadas implementaram DevOps e entrega contínua para seus aplicativos;
- » Apenas 16% dessas organizações viram o DevOps como um mecanismo de redução de custos;

- » 49% dos participantes creditaram ao DevOps o melhor desempenho, tempo mais rápido de lançamento no mercado e melhor ROI;
- » 48% dos entrevistados achavam que o maior obstáculo para não usar o DevOps era a falta de conhecimento, enquanto 33% achavam que não tinham as ferramentas certas nem o tempo para implementá-lo. Alguns expressaram sua desconfiança na automação como uma razão para não adotar entrega contínua.
- » 84% dos entrevistados colocam os sistemas de controle de versão no topo da lista de ferramentas que suportam uma iniciativa de DevOps. Os sistemas modernos de controle de versão são os blocos básicos de todos os processos relacionados à automação e integração ou entrega contínua.

CAPÍTULO 3

Diferença entre o ciclo tradicional e DevOps

No desenvolvimento de *softwares*, o pensamento básico é a codificação, testes de integração e, em seguida, o sistema é liberado para a equipe de operações implementar. Os trabalhos podem ficar bons e harmônicos se houver uma sincronização máxima entre a equipe de desenvolvimento e a equipe de operações, mas com mais frequência há desafios. Por exemplo, o time de operações implanta a solução no ambiente de desenvolvimento, porém a resposta do time de desenvolvimento é que, no ambiente de produção, não funciona. O motivo da falha é diferença de ambientes e a falta de sincronização dos códigos.

Novos métodos de desenvolvimentos, aliados a novas ferramentas tornam a codificação mais rápida, fazendo com que a equipe de operações não consiga acompanhar as mudanças e lançamentos frequentes. Os servidores de produção podem precisar de alguns ajustes ou ajuste fino no banco de dados ou no nível do sistema operacional. A falta de habilidade ou experiência nesse cenário poderá colocar a implantação em risco.

Outra situação é que, muitas vezes, o desenvolvedor não tem acesso aos servidores para verificar o comportamento do *software* e ficam dependendo do *feedback* de usuários finais, o que, com frequência, não é recebido pelos desenvolvedores por motivos óbvios. Em alguns casos, não há instruções/detalhes claros de implantação. A equipe de operações precisa descobrir algumas coisas com base em sua experiência/habilidade – transição deficiente.

Ao comparar as operações de TI tradicionais ao DevOps, fica claro como elas diferem e por que o DevOps é cada vez mais adotado por organizações em todo o mundo. Quando um projeto de desenvolvimento de *software* é baseado em conceitos e metodologias tradicionais, o papel do time de desenvolvimento era o de converter requisitos em programas. Nessa situação, cabia ao time de operações instalar, atualizar, configurar e manter os programas necessários funcionando para a infraestrutura proposta. Nessas metodologias, cada grupo trabalhava apenas para cumprir o sucesso de sua etapa e não para o projeto como um todo. Cada equipe possuía suas formas de trabalhar e era comum ignorar os dilemas do outro time. Algumas questões geravam alguns conflitos, e opções importantes para o projeto acabavam ficando de fora. Por exemplo, era comum o gerente de projeto ter que decidir entre velocidade, recursos, acesso ou estabilidade, usabilidade e controle.

A capacidade de colaboração de equipes é comprometida, pois, além de possuir filosofias opostas, as equipes são dimensionadas erroneamente. Por isso, não são eficazes, devido à flexibilidade, quando a tecnologia evolui rapidamente.

Uma empresa DevOps compreende muito bem que a equação de valor, criada no ciclo de vida de entrega de *software*, está concentrada em dois locais. Segundo Kapadia (2015), “primeiro, quando o *software* é criado (desenvolvimento), e segundo, quando o *software* é entregue ao cliente e o *feedback* recebido.”

O desenvolvimento e a entrega de Software se concentram nos processos que agregam valor, organizam-se em torno dessas ações e fazem o melhor para minimizar o risco. A TI tradicional também entende – pelo menos conceitualmente. No entanto, eles tratam todos os estágios no SDLC com igual importância ou pior enfatizam o(s) estágio(s) errado(s). Em geral, ao comparar as organizações de DevOps à TI tradicional, o DevOps terá políticas e práticas que diferem das da TI tradicional nas oito dimensões principais a seguir (KAPADIA, 2015):

Tabela 1. Dimensão de planejamento.

	Dimensões	Ti Tradicional	DevOps
Planejamento e organização	Tamanho do lote	Grande	Micro
	Organização	Silos centralizados	Células dedicadas
	Agendamento	Centralizado	Decentralizado e contínuo
Desempenho e cultura	Lançamento	Evento de alto risco	Sem evento
	Informação	Disseminada	Acionável
	Cultura	Não falhe	Falhar cedo
Medidas	Métrica	Custo e capacidade	Custo, capacidade e fluxo
	Definição de “feito”	“Eu fiz o meu trabalho”	“Está pronto para implantar”

Fonte: (KAPADIA, 2015).

Planejamento e organização

Tamanhos de lote, de maior para menor

Por ter crescido baseando-se no método cascata, a maioria das empresas de TI tradicional tende a se tornar grande. Os lançamentos também são caros e problemáticos, o time de operações permite algumas janelas por período para que sejam efetuados os lançamentos de *software*. Como resultado, segundo Kapadia (2015), as organizações de desenvolvimento maximizam a produtividade, planejando grandes projetos, que envolvem muito código, empacotados (às vezes às pressas) em uma versão e atolados na produção.

Em uma organização DevOps, entende-se que os lotes grandes são arriscados e difíceis de coordenar e os lotes pequenos são fáceis de se entender, testar com mais rigor e rápido de corrigir.

Organização

De silos centrados em habilidades para células dedicadas

O agrupamento em silos centrados em habilidade, criados pela TI tradicional, beneficiam-se de economias em escala. Quando um novo recurso precisa ir adiante, ele passa por três ou quatro silos antes que o cliente receba a solução, gerando um ciclo que gasta, em média, 80% de seu tempo.

Em uma organização DevOps, seus silos são células, formadas por equipes funcionais cruzadas, dedicadas e focadas em apenas um único aplicativo. Essas células, ao serem formadas, criam uma equipe autossuficiente, constituídas por desenvolvedores, testadores, analistas de negócios, operadores. A ideia de passar de um estágio para outro promove o treinamento, a integração e concentra a equipe no objetivo final.

Quanto aos benefícios de utilização e escala, as regras para resposta afirmam que para cada ¼ de redução no tempo do ciclo, a produtividade melhorará duas vezes e os custos operacionais em 20% (STALK, 1990).

Agendamento

Centralize para descentralizar e contínuo

Uma organização tradicional de TI mantém centralizados seus agendamentos, pois, como os recursos são agrupados, os projetos geralmente clamam pelo acesso à infraestrutura. Para organizar, investe em sofisticados sistemas de planejamento de programação e, embora esses sistemas sejam bastante sensíveis, eles são imprecisos e levam muito tempo para serem gerenciados, tornando-se, muitas vezes, um problema a ser resolvido.

Em uma organização DevOps, o agendamento é planejado no nível celular, ou seja, local da célula. A combinação entre tamanhos de lote menores, equipes

dedicadas e processos automatizados torna o agendamento mais simples de operar:

- » A previsão é limitada a um futuro muito próximo, no qual as equipes têm uma visão melhor.
- » Não há luta pelo tempo das pessoas, uma vez que é uma equipe dedicada.
- » Não há espera pela infraestrutura, pois ela já foi definida e provisionada automaticamente.
- » Elimina atrasos de *loop* demorados para o gerenciamento de escalonamentos e decisões.

Performance e cultura

Release: transforme um evento de alto risco em um não evento

Em uma organização de TI tradicional, liberar *software* para produção é um “**evento**” de alto risco. Os problemas e o combate a incêndios são constantes. O processo é rigidamente gerenciado, governado pelos níveis mais altos e requer participação de todas as partes da organização. Não é incomum que a TI configure salas de guerra equipadas 24 × 7 por semanas antes e depois do lançamento.

As organizações de DevOps tornam o lançamento de *software* em um “**não evento**” o máximo possível. O risco é reduzido diariamente, integrando códigos, automatizando os testes, garantindo que todos os ambientes estejam sincronizados, reduzindo o tamanho dos lotes etc. No DevOps, o código de um estágio só é promovido para outro depois de se ter certeza de que se vai trabalhar na produção. Assim, eliminando todos os possíveis problemas sobre janelas de lançamento, e eles são capazes de mover novas funcionalidades para produção em um ritmo muito mais rápido.

Informação

O foco está em disseminação

Tanto a organização de DevOps quanto a organização tradicional de TI geram e compartilham muita informação.

Na organização de TI tradicional, as informações são geradas por especialistas, com a equipe de operações, por exemplo, normalmente publicadas em relatórios extensos, obedecendo a um fluxo que primeiro passa para a aprovação da gerência, para depois ser compartilhado para outros gerentes e especialistas. Frequentemente, devido ao massivo número de informações, esses relatórios simplesmente não são lidos. Portanto, conclui-se que essa informação existe, é compartilhada, é mal consumida e dificilmente é tomada alguma decisão baseando-se por ela.

Dentro de uma organização de DevOps, a própria célula é quem trata de gerenciar as informações, ou seja, são coletadas apenas informações necessárias para serem consumidas localmente. Por serem processadas dentro da equipe, eliminam intervalos no trajeto entre a aprovação e a tomada de decisão. Como resultado, sua leitura é rápida, e o *feedback* é muito mais rápido.

Cultura

Não falha vs. falha precoce

Por ser totalmente avessa aos riscos, a TI tradicional tem sua prioridade em não causar danos ao negócio. Por isso o investimento em processos burocráticos para a prevenção de falhas é imenso. E apesar de tanto investimento, o histórico de projetos entregues com atraso, retrabalho, problema de infraestrutura, é muito alto.

Uma organização de DevOps também é avessa a riscos, mas ela também entende que a falha é inevitável. Por sua cultura estar voltada a processo de entrega contínua, integração diária, lotes pequenos, estrutura celular, automação etc., favorece e reforça o objetivo de “falhar cedo e recuperar rapidamente”.

A métrica

Custo e capacidade de custo, capacidade e fluxo

A formação de silos na TI tradicional está diretamente ligada ao modelo de medição: custo e capacidade, ou seja, o quanto pode ser feito com o mínimo de dinheiro. Foi a partir dessa métrica que as terceirizações cresceram nas últimas décadas. Entretanto, para aplicativos legados e estáveis, esse modelo funciona,

porém, quando novos aplicativos foram surgindo, foi necessário acrescentar mais uma variável: o tempo. Completando, assim, esse modelo para o desenvolvimento de *software* tradicional.

No DevOps, foi adicionado o “fluxo”. Ele força uma organização a analisar o seu resultado até o fim do ciclo, identificar áreas de desperdício, calcular o tempo produtivo real, quantificar a qualidade e se concentrar nas atividades que agregam mais valor.

Definição de feito

“Eu fiz o meu trabalho” vs. “está pronto para implantar”

Outra fonte de medição entre o TI tradicional e o DevOps está no conceito de “trabalho feito”. Na TI tradicional, o especialista faz apenas sua parte, e seu principal e único objetivo é entregá-lo, que, em sua essência, é cumprir o prazo de entrega. Essa cultura leva a alguns maus hábitos que, certamente, vão impactar a qualidade do produto.

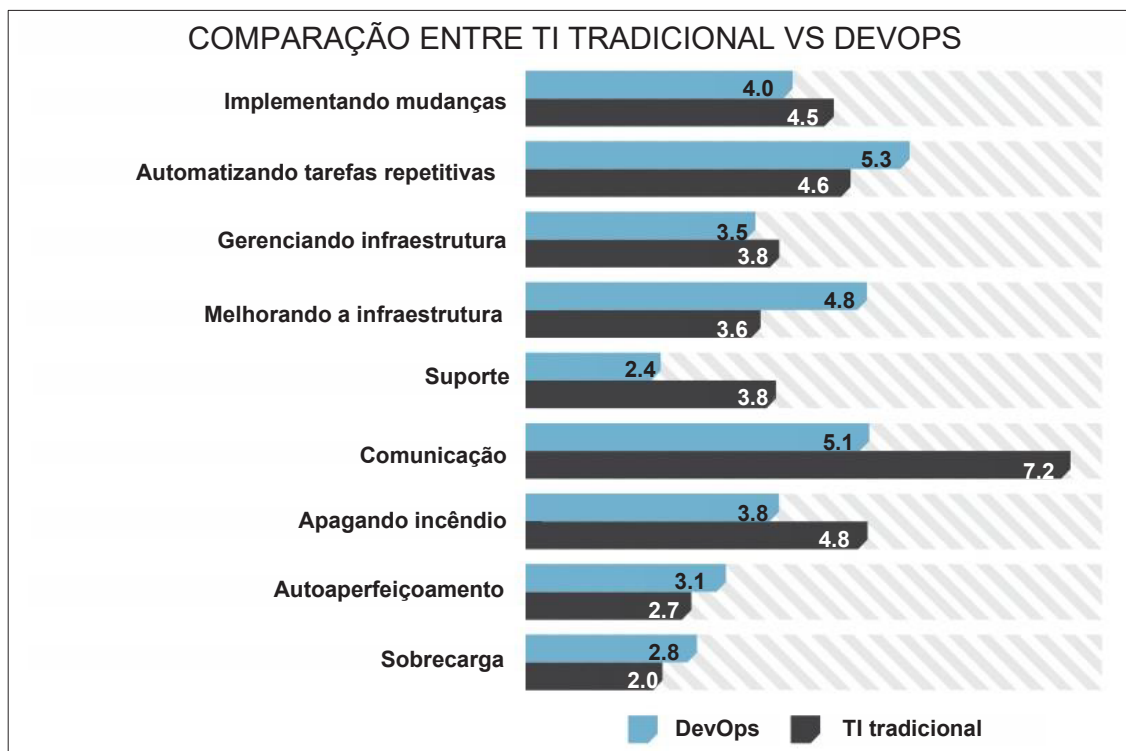
A organização de DevOps, com sua estrutura celular, cria equipes mais funcionais, em que cada elemento é responsabilizado por uma única tarefa, criando uma definição para o que está realmente feito, ou seja, trazer um produto de qualidade para o cliente.

Segundo Morgan (2016), as equipes de DevOps gastam menos tempo apagando incêndios e com suporte. Com isso, uma equipe de DevOps ocupa seu tempo melhorando a infraestrutura contra falhas. Testes contínuos, alertas, monitoramento e *feedback* são colocados em prática para que as equipes DevOps possam reagir de maneira rápida e eficaz.

As equipes tradicionais de TI têm quase duas vezes mais chances de precisar de mais de 60 minutos para se recuperarem, enquanto as recuperações em menos de 30 minutos são 33% mais prováveis para as equipes de DevOps.

Veja na figura 7 a porcentagem entre a TI tradicional e a DevOps

Figura 7. TI tradicional x DevOps.



Fonte: <https://www.slideshare.net/ZeroTurnaround/traditional-it-ops-vs-dev-ops-devops-days-ignite-talk-by-oliver-white>.

Resumindo, as equipes de DevOps fazem mais e resolvem os problemas mais rapidamente, pois gastam mais tempo melhorando as coisas, menos tempo consertando as coisas, recuperando-se de falhas mais rapidamente e liberando os aplicativos mais de duas vezes mais rápido que os Ops de TI tradicionais.

CAPÍTULO 4

Do *Lean* ao DevOps

Em seu artigo, Steve Mezak (2018), menciona cronologicamente o início e ascensão do conceito DevOps. O artigo pode ser acessado em: <https://devops.com/the-origins-of-devops-whats-in-a-name>.

Antes de 2007

Antes de 2007, ocorreram vários eventos que originariam as primeiras menções do DevOps. O estabelecimento da manufatura enxuta, como melhores práticas na indústria, foi marcado pelo método Toyota de fabricação e seu conceito de otimização de processos.

Esses conceitos eram:

- » **Mantenha o estoque no mínimo.** Produção enxuta significa estoque mínimo disponível: matérias-primas e estoque acabado.
- » **Minimize a fila de pedidos.** Idealmente, os pedidos recebidos devem passar imediatamente para o modo de atendimento.
- » **Maximize a eficiência no processo de fabricação.** A reengenharia de processos e a automação aprimorada se unirão ao objetivo de produzir bens o mais rápido possível.

Na tecnologia da informação, os métodos tradicionais de desenvolvimento do *software* em cascata já estavam perdendo espaço para os métodos ágeis. Nesse mesmo tempo, os serviços de infraestrutura de TI, a computação em nuvem, a infraestrutura (IaaS) e a plataforma (PaaS) como serviço estavam se tornando ofertas muito atrativas. Também, um novo conceito de integração contínua, o método *Booch*.

Criado por *Grady Booch*, em 1991, o método *Booch* ajuda a projetar sistemas usando o paradigma de objeto. Abrange as fases de análise e design de um sistema orientado a objetos. O método define modelos diferentes para descrever um sistema e suporta o desenvolvimento iterativo e incremental de sistemas.

2008 – Patrick Debois

Em 2008, o belga e gerente de projetos, Patrick Debois, foi designado pelo governo para ajudar na migração do Data Center. Em meio às suas atividades, estavam tarefas que relacionavam as equipes de desenvolvimento e as equipes de operações. Ele, então, percebeu um muro enorme que separava ambas as equipes e a falta de coesão entre os métodos de aplicações e infraestrutura.

2008 – Agile Conference

Em 2008, na *Agile Conference*, em Toronto, Andrew Schafer publicou uma oferta para moderar uma reunião chamada de “*Birds of a Feather*” para discutir o tópico “*Agile Infrastructure*”. Apenas uma pessoa apareceu para discutir o assunto: *Patrick Debois*. Suas discussões e compartilhamento de ideias avançaram o conceito de “administração de sistemas ágeis”. Nesse mesmo ano, Debois e Shafer formaram um grupo de Administradores de Sistemas Ágeis no Google, com sucesso limitado.

2009 – O’Reilly Conference e Flickr

Em 2009, na O’Reilly Conference, dois funcionários da Flickr, John Allspaw, vice-presidente sênior de operações técnicas, e Paul Hammond, diretor de engenharia, fizeram uma apresentação que teve um toque dramático, já que Allspaw e Hammond representariam a interação entre representantes de desenvolvimento e operações durante uma implementação típica de *software*, juntamente com todas as acusações e culpas que acontecem, como por exemplo: “Não é o meu código, são as suas máquinas!”.

A apresentação fez com que o único caminho racional para o desenvolvimento de aplicativos e atividades de operações fosse transparente e totalmente integrado. Com o tempo, esta apresentação alcançou status lendário. Incapaz de comparecer pessoalmente, Debois assistiu à apresentação por vídeo. Ele foi inspirado e formou sua própria conferência chamada *Devopsdays* em Ghent, na Bélgica. A essa altura, o termo “DevOps” havia oficialmente desembarcado nos livros de história.

2010 – Devopsdays USA

Em 2010, o DevOps desembarca nos Estados Unidos, onde é realizada a *Devopsdays*, na Califórnia, logo após a conferência anual *Velocity*. Enquanto esse material está sendo produzido, exatamente em 3 de agosto de 2019, estava sendo realizada a Devopsday em Goiânia, Brasil (<https://devopsdays.org/events/2019-goiania/welcome/>).

2013 – The Phoenix Project

Em 2013, foi publicado o livro “*The Phoenix Project*”, escrito por Gene Kim, Kevin Behr e George Spafford. Este romance fictício conta a história de um gerente de TI envolvido em uma situação aparentemente sem esperança, já que ele é encarregado de salvar um projeto de desenvolvimento de comércio eletrônico de missão crítica que saiu dos trilhos.

Seu misterioso mentor, um membro do conselho mergulhado nas disciplinas de manufatura enxuta, orienta o personagem principal em novas formas de pensar sobre TI e, segundo Kim *et al.* (2018, p. 4), “Devops e suas práticas representam muitos movimentos filosóficos e gerenciais, pois, embora cada empresa tenha desenvolvido esses princípios independentes, a convergência DevOps mostra uma imensa progressão de pensamentos e conexões improváveis, principalmente da manufatura física e da liderança de fluxo da TI.”

Embora muitos autores cite que o DevOps seja derivado do Lean, da Toyota, muitos veem como continuação do movimento ágil. Na verdade, Devops conta com conhecimento de Lean, teorias das restrições, sistema Toyota de produção, engenharia da resiliência, organização de aprendizado, cultura de segurança, fatores humanos e muitos outros.

CALMS

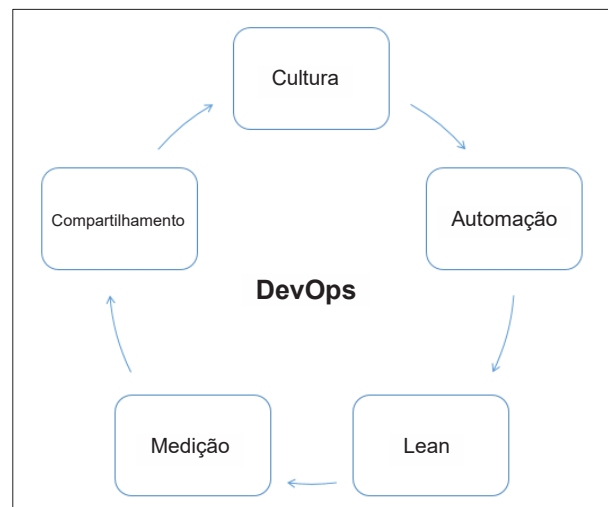
O CALMS faz parte da cultura DevOps e é uma estrutura conceitual de integração de equipes, funções, equipes de desenvolvimento e operações. Frequentemente sua estrutura é utilizada para avaliar a maturidade, ajudando gerentes a avaliar se sua organização está pronta ou não para DevOps.

Os cinco pilares da estrutura do CALMS para DevOps são:

- » Cultura – *Culture*;
- » Automação – *Automation*;
- » Lean – *Lean*;
- » Medição – *Measure*;
- » Compartilhamento – *Sharing*.

A sigla CALMS é creditada a Jez Humble, coautor de *The DevOps Handbook*.

Figura 8. Pilares do CALMS – DEVOPS.



Fonte: Elaboração própria do autor.

Cultura

É a base cultural do Devops! Está ligada ao relacionamento interpessoal e à responsabilidade de se ater como os profissionais se relacionam, se organizam, quais canais e metodologias ágeis são adequados. O componente “Cultura” também está relacionado ao investimento em tecnologias.

Em uma organização de Devops, há muito tempo se estabelece que a adoção de tecnologia deva ser para suprir uma necessidade comercial, em vez de investir em tecnologia por si só. O ROI previsto, associado à automação de um processo, deve ser defendido em toda a equipe de DevOps. Tudo isso junto forma a cultura organizacional de desenvolvimento e operação, é um ponto que requer maior atenção na transformação digital da empresa. Uma cultura ruim influencia diretamente a qualidade dos resultados entregues.

Automação

Automatizar o máximo de tarefas possíveis! Esse é o lema de uma cultura DevOps, na qual se automatiza desde o fluxo de criação até a entrega contínua, evitando erros humanos, documentando e padronizando cada passo do processo. Isso resulta em um evento, dentro do DevOps, chamado *Pipeline*.

A Netflix é bem conhecida como um estudo de caso exemplar para a automação do DevOps. Entre outras coisas, ela pode implantar atualizações de código e *software* em questão de minutos em toda a sua enorme infraestrutura, muitas vezes ao longo do dia.

Lean

O Lean vai ajudar uma organização a identificar gargalos, a aperfeiçoar fluxos, a desenhar melhor as etapas de entrega de valor da empresa. Isso porque de nada adianta os membros da equipe visualizarem o trabalho em andamento (WIP), limitarem os tamanhos dos lotes, gerenciarem os comprimentos das filas, automatizarem tudo e terem uma excelente cultura organizacional, se não souberem quais são os gargalos no processo. Com os processos mapeados, é possível entender e identificar quais pontos estão tomando mais tempo ou causando retrabalho.

Medição

Depois de todas as etapas implementadas, mensurar o ambiente e gerar *feedback* é importante para entender o que realmente acontece com sua solução. Além de gerar uma base sólida de conhecimento, cria previsibilidade sobre possíveis incidentes ou desastres que possam vir a surgir. Logs e *dashboards* são insumos suficientes para analisar falhas e gerar melhorias constantemente.

É imprescindível mensurar os diversos níveis que sua aplicação possa ter, desde *containers* e servidores, até regras de negócio, para garantir que sua aplicação esteja cumprindo os requisitos mínimos de disponibilidade.

Compartilhamento – *Sharing*

Existem canais de comunicação fáceis de usar que estimulam a comunicação contínua entre o desenvolvimento e as operações.

Compartilhar, em muitos aspectos, se sobrepõe à cultura. É aqui que avalia se todas as equipes diferentes do DevOps estão trabalhando juntas e se comunicam. Mas compartilhar nem sempre é fácil. Mesmo em equipes separadas, é fácil cair na armadilha de querer realizar seu próprio trabalho sem necessariamente pensar no time tanto quanto deveria.

A convergência DevOps

“Muitos acreditam que DevOps está usufruindo de uma incrível convergência de movimentos de administração em que todos estão se reforçando mutuamente e podem criar uma poderosa coalisão para transformar o modo com que organizações desenvolvem e entregam produtos e serviços de TI” (KIM *et al.*, 2018, p.354).

Essa convergência está baseada nos seguintes movimentos:

Movimento Lean

O Movimento Lean começou nos anos 1980 como uma tentativa de codificar o sistema Toyota de produção, com a popularização de técnicas de mapeamento de fluxo de valor, quadros kanban e manutenção produtiva total.

Movimento ágil

Iniciado em 2001, o Movimento Ágil foi criado por 17 dos principais pensadores de desenvolvimento de *software*, com o objetivo de transformar métodos leves, em movimento mais amplo que pudesse enfrentar processos de desenvolvimento de *software* pesados como o desenvolvimento em cascata e RUP.

Movimento entrega contínua

Com base na disciplina do Desenvolvimento de *build*, teste e integração contínua, foi ampliado o conceito de entrega contínua que incluía um “*pipeline* de implementação” para garantir que códigos e infraestrutura estivessem sempre em um estado implementável e que todo código inserido no repositório fosse implementado na produção.

Movimento Toyota Kata

Descreve rotinas gerenciais invisíveis e o pensamento por trás do sucesso da Toyota com melhoria contínua e adaptação e como outras empresas desenvolvem rotinas e pensamentos semelhantes.

Movimento Lean StartUp

Eric Ries escreveu *A Startup Enxuta*, codificando as lições que aprendeu na IMVU, uma *startup* do Vale do Silício. Eric também codificou práticas e termos relacionados, incluindo produto viável mínimo, o ciclo construir-medir-aprender e muitos padrões técnicos de implementação contínua.

Movimento Lean UX

Em 2013, Jeff Githel escreveu *Lean UX, applying lean principles to improve user experience*, que codificava como melhorar o *front-end* impreciso e explicava como donos de produto podem enquadrar hipótese comercial, experimento e ganhar confiança, antes de investirem tempo e recursos.

Movimento computação reforçada

Em 2011, Joshua Corman, David Rice e Jeff Willians examinaram a aparente futilidade de tornar aplicativos e ambientes seguros tardiamente no ciclo de vida. Em resposta, eles criaram uma filosofia chamada “Computação reforçada”, que tenta enquadrar os requisitos não funcionais de estabilidade, escalabilidade, disponibilidade, sobrevivência, sustentabilidade, segurança, suportabilidade, manuseabilidade e defensibilidade.

CAPÍTULO 1

Lean

Muitas literaturas iniciam seu estudo sobre Lean pelo histórico de que seu conceito surgiu logo após a Segunda Guerra Mundial com a Toyota. Porém, bem antes da fundação da empresa, em 1867, nascia *Sakichi Toyoda*, exatamente quando o Japão passava por um processo de modernização. *Sakichi* viveu em uma aldeia de camponeses e logo cedo modernizou o tear manual de sua mãe e, com 24 anos, mudou-se para Tóquio, montando um negócio de teares, patenteando seu produto.

Em 1893, *Sakichi* casa, tem um filho, chamado *Kiichiro Toyoda*, e volta pra terra natal, onde continua desenvolvendo teares. Em 1896 criou um mecanismo de segurança que permitia travar imediatamente o tear caso alguma falha ocorresse durante seu funcionamento. Nascia nesse momento o conceito de automação, de onde viria a ser o sistema Toyota de Produção e Automação. A empresa Mitsui fecha, portanto, um contrato para a comercialização dos teares *Toyoda*. Nesse instante, as máquinas criadas por *Toyoda* custavam um décimo dos teares fabricados na Alemanha e um quarto dos fabricados em França.

Algumas guerras também interferiram na comercialização dos teares *Toyoda*. Em 1894, a guerra entre o Japão e China gerou uma crise, e a resseção que seguiu fez com que *Sakichi* se dedicasse novamente ao aperfeiçoamento das suas máquinas. E dez anos mais tarde, a guerra entre a Rússia e o Japão fez com que a procura de algodão aumentasse e, com ela, a venda de teares *Toyoda*.

Em 1907, *Sakichi* funda a empresa *Toyoda Loom Works*. Três anos mais tarde, viaja para os Estados Unidos da América, se encanta pela complexidade do mundo do automóvel e, de volta ao Japão, o empresário vende os direitos da empresa de teares à britânica *Platt Brothers* e funda a *Toyoda Spinning and Weaving Co. Ltd*, encarregando ao seu filho *Kiichiro* os investimentos da empresa de automóvel. *Sakichi* morre em 1930, e o seu filho inicia o seu trabalho no desenvolvimento de motores de combustão a gasolina.

Em 1937 Kiichiro produz o primeiro protótipo de automóvel e estabelece as bases para fundar a Toyota Motor Compagny Ltd. Após a Segunda Guerra Mundial, com a economia japonesa arrasada, a Toyota define sete tipos de desperdícios e adota uma estratégia para os eliminar. Esse conceito de eliminação de desperdícios tornou-se a base do Sistema Toyota de Produção, trazido para o ocidente com o nome de Lean Manufacturing.

O sistema de produção de veículos da Toyota Motor Corporation é uma maneira de tornar as coisas que às vezes são chamadas de “sistema de manufatura enxuta” ou “sistema Just-in-Time (JIT)”, e já são bem conhecidas e estudadas em todo o mundo. Esse sistema de controle de produção foi estabelecido com base em muitos anos de melhorias contínuas, com o objetivo de tornar os veículos encomendados pelos clientes da maneira mais rápida e eficiente, a fim de entregar os veículos o mais rapidamente possível.

O Toyota Production System (TPS) foi estabelecido com base em dois conceitos: “jidoka” (que pode ser traduzido livremente como “automação com um toque humano”), quando ocorre um problema, o equipamento para imediatamente, impedindo a produção de produtos defeituosos; e o conceito “Just-in-Time”, no qual cada processo produz apenas o que é necessário para o próximo processo em um fluxo contínuo.

Com base nas filosofias básicas de jidoka e Just-in-Time, a TPS pode produzir de forma eficiente e rápida veículos de qualidade sonora, um de cada vez, que satisfaçam totalmente os requisitos do cliente.

O TPS e sua abordagem de redução de custos são as fontes de força competitiva e vantagens exclusivas para a Toyota. Aperfeiçoar completamente essas forças é essencial para a futura sobrevivência da Toyota. Usaremos essas iniciativas e desenvolveremos nossos recursos humanos para fabricar carros cada vez melhores, que serão valorizados pelos clientes

Lean Thinking

É um conceito que se refere às atividades que as empresas possam implementar para reduzir desperdícios em seus negócios e processos fabris. Seu objetivo é dar aos clientes o produto que eles realmente contrataram ou encomendaram.

Lean Thinking é o modo como as organizações podem eliminar todas as coisas que tornam seus procedimentos de produção mais pesados e pouco eficazes,

permitindo que sejam mais produtivos de maneira mais simples e fácil, reduzindo custos e tempos. Ela não busca apenas a redução de resíduos, mas também a satisfação das necessidades dos clientes da melhor maneira possível. Essa filosofia procura produzir não apenas algum produto ou serviço, mas fabricar esses elementos de acordo com o que eles precisam e desejam.

Aplicando os princípios do Lean

Ao longo do tempo, para se obter uma estrutura de criação e produção eficiente e eficaz, muitas organizações utilizaram os cinco princípios do Lean, observando que seus gerentes conseguiam descobrir as ineficiências ligadas aos processos e assim podiam oferecer um melhor produto aos seus clientes. Os princípios encorajam a criação de um melhor fluxo nos processos de trabalho e o desenvolvimento de uma cultura de melhoria contínua.

Ao praticar todos os cinco princípios, uma organização pode permanecer competitiva, aumentar o valor entregue aos clientes, diminuir o custo de fazer negócios e aumentar sua lucratividade.

Definir valor

Para entender melhor o primeiro princípio que define o valor do cliente, é importante entender qual é o valor. Valor é o que o cliente está disposto a pagar, e descobrir suas necessidades é fundamental, pois nem sempre os clientes conseguem demonstrar o que querem ou, simplesmente, não sabem. Para isso existem muitas técnicas quantitativa e qualitativa, como entrevistas, pesquisas, informações demográficas e análises da Web que podem ajudar a decifrar e descobrir o que os clientes consideram valioso. Essas técnicas podem ajudar a descobrir o que os clientes querem, como querem que o produto ou serviço seja entregue e o preço que eles oferecem.

Mapeie o fluxo de valor

Nesta etapa, o objetivo é usar o valor do cliente como um ponto de referência e identificar todas as atividades que contribuem para esses valores, eliminando desperdícios que podem ser classificados como:

- » Sem valor e necessário: que pode ser reduzido ao máximo;
- » Sem valor e desnecessário: deve ser eliminado totalmente.

Reduzindo e eliminando processos ou etapas desnecessárias, você pode garantir que os clientes obtenham exatamente o que desejam e, ao mesmo tempo, reduzindo o custo de produção desse produto ou serviço.

Crie fluxos contínuos

Segundo Pereira (2010), essa etapa exige uma completa mudança na cultura da organização e da empresa. É necessário abandonar a ideia de divisão da empresa por departamentos, como sendo a melhor solução de tomada de decisão. O efeito imediato da criação de fluxos contínuos pode ser sentido na redução de tempos de criação de novos produtos, de processamento de pedido e em quantidades de estoques. O aumento da capacidade de resposta da empresa faz com que esta reaja de imediato, a qualquer flutuação do mercado e necessidades dos clientes.

Estabeleça a tração – *pull system*

O objetivo de um sistema baseado em *pull* permite a entrega e fabricação *just-in-time*, em que os produtos são criados no momento em que são necessários e apenas nas quantidades necessárias. Os sistemas baseados em *pull* são sempre criados a partir das necessidades dos clientes finais. Seguindo o fluxo de valor, você pode garantir que os produtos produzidos sejam capazes de satisfazer as necessidades dos clientes.

Prosseguir a perfeição

Todos os resíduos foram eliminados ou evitados por meio dos passos anteriores. A quinta etapa de buscar a perfeição é o mais importante entre todos eles. Segundo Pereira (2010), a procura do aperfeiçoamento contínuo em direção a um estado ideal deve conduzir e encaminhar todos os esforços da empresa, em processos bem definidos, nos quais todos os participantes e membros da cadeia tenham um conhecimento bem claro e profundo do processo como um todo, podendo participar e procurar continuamente melhores formas de criar valor.

Isso faz com que o pensamento enxuto e a melhoria contínua de processos façam parte da cultura organizacional.

Os três Mu

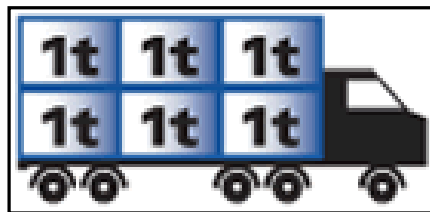
Para evitar o desperdício, é preciso chegar à capacidade produtiva igual ao que foi pedido para ser produzido. Esse conceito significativo é formado por três

componentes diferentes; Mura, Muri e Muda. Esses três elementos são focados na redução de todas as “coisas” que não agregam valor a um produto ou serviço específico e devem trabalhar juntos para obter os resultados desejados.

Muri

Seu significado é excessivo ou sobrecarga.

Figura 9. Sobrecarga.

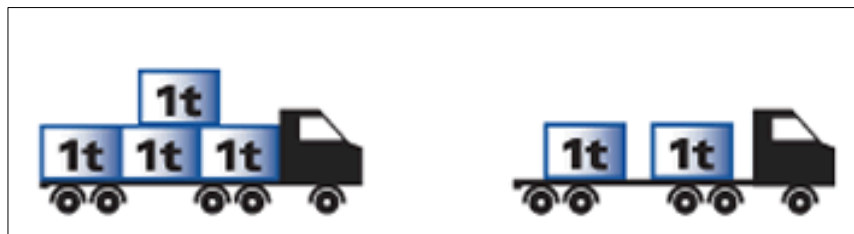


Fonte: LEI (2018).

Mura

Esta palavra japonesa significa irregularidade ou uniformidade.

Figura 10. Irregularidade.

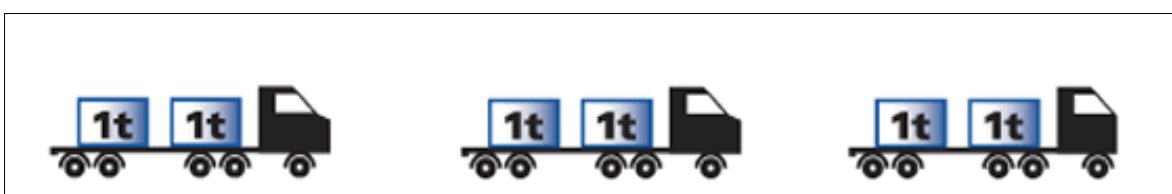


Fonte: LEI (2018).

Muda

Esta palavra japonesa significa desperdício.

Figura 11. Desperdício.



Fonte: LEI (2018).

A figura 12 mostra como foram evitadas as três possibilidades de desperdícios:

Figura 12. Evitando o desperdício.



Fonte: LEI (2018).

CAPÍTULO 2

DevOps e Kanban

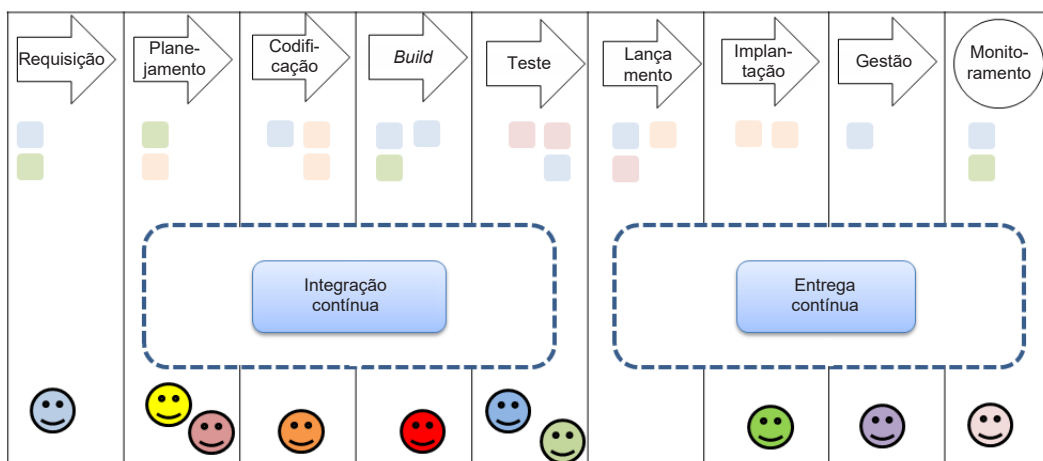
DevOps e Kanban

A técnica Kanban surgiu no final da década de 1940 como a abordagem da Toyota à manufatura e engenharia. Kanban é a palavra japonesa que significa “sinal visual” ou “cartão”. A natureza altamente visual do sistema permitiu que as equipes se comunicassem mais facilmente sobre o trabalho necessário e seus prazos. Também padronizou sugestões e processos refinados, o que ajudou a reduzir o desperdício e maximizar o valor.

A aplicação do Kanban ao trabalho de conhecimento, influenciado não apenas pelo Sistema Toyota de Produção, mas também pelo pensamento Lean, começou em 2005. Os princípios centrais de Kanban são os mesmos em indústrias como desenvolvimento de *software* e gerenciamento de recursos humanos como na indústria:

No DevOps, a principal vantagem do Kanban é o incentivo oferecido para as equipes se concentrarem na melhoria do fluxo do sistema. Sua maturidade proporciona boas entregas, facilidade no lançamento de produtos, correções de defeitos etc. Esse recurso do Kanban o torna bastante adequado aos requisitos contínuos de distribuição e distribuição do DevOps. A visualização do fluxo de trabalho garante a combinação de diferentes atividades desde o desenvolvimento, integração, testes, implantação até o monitoramento de aplicativos. Essa ferramenta, ao longo do tempo, propiciará uma evolução das equipes de *Dev e Ops*, para uma única equipe de trabalho e fluxo único.

Figura 13. Exemplo de fluxo em Kanban.



Fonte: Elaboração própria do autor, adaptada de <https://d30s2hykpf82zu.cloudfront.net/wp-content/uploads/2018/10/kanban-for-devops1.png>.

A visibilidade proporcionada pelo Kanban garante que todos saibam em qual cada etapa estará um trabalho, o que precisa ser corrigido e para onde deve fluir. Com isso, fica muito clara a condição do status. A prioridade é considerada com apenas uma visualização no quadro, e isso garante que todas as equipes saibam o que está alinhado às metas da empresa. O trabalho bloqueado é imediatamente percebido se houver problema, os tomadores de decisões podem imediatamente intervir e descobrir qual decisão tomar e não havendo custo de *feedback* atrasado.

O sistema *pull* garante que a equipe gaste seu tempo com um número reduzido de mudanças, seguindo o jargão: “pare de iniciar e comece a terminar”, garantindo que o fluxo seja constante, pois a mudança frequente de contexto gera perda de produtividade.

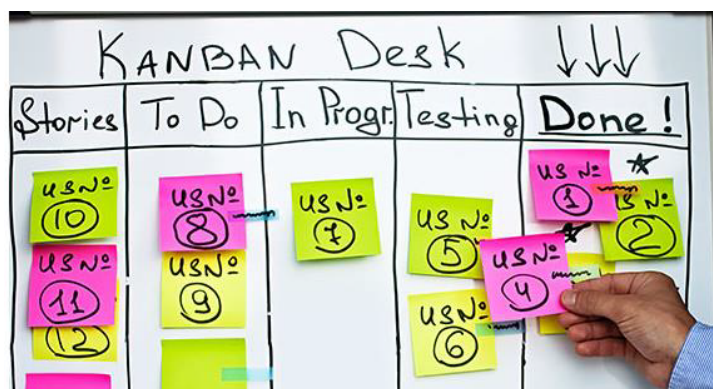
Quadro Kanban

Quando uma organização DevOps adota um quadro Kanban, estará adotando a melhor ferramenta para a visualização dos trabalhos. Ele é usado por qualquer equipe que usa o Kanban para gerenciamento visual de seu trabalho e para melhorar a entrega de produtos e serviços em termos de previsibilidade, qualidade e desempenho.

Um quadro Kanban pode ser físico ou eletrônico. Ele consiste em uma ou mais raia e várias colunas para representar o processo de fluxo de trabalho que é usado para gerenciar e representar um “sistema Kanban virtual” usado para modelar o processo e rastrear o trabalho de conhecimento que está sendo feito por sua equipe.

Você pode começar com um quadro Kanban físico em um quadro branco ou talvez algumas grandes janelas de vidro ou divisórias, em seu escritório, lisas para colar adesivos. Quanto maior e mais visível eles são, melhor.

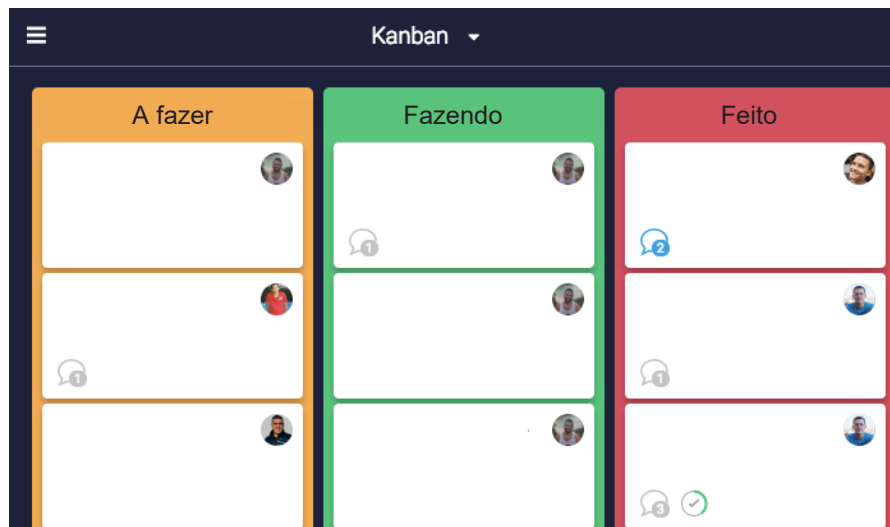
Figura 14. Quadro Kanban físico.



Fonte: <https://netproject.com.br/blog/wp-content/uploads/2016/12/quadro-kanban.jpg>.

Os quadros de Kanban também podem ser digitais, como o do *software* Trello. Os Kanbans digitais permitem que as equipes trabalhem remotamente, podendo ser acessadas por aplicativos em nuvem.

Figura 15. Kanban virtual.

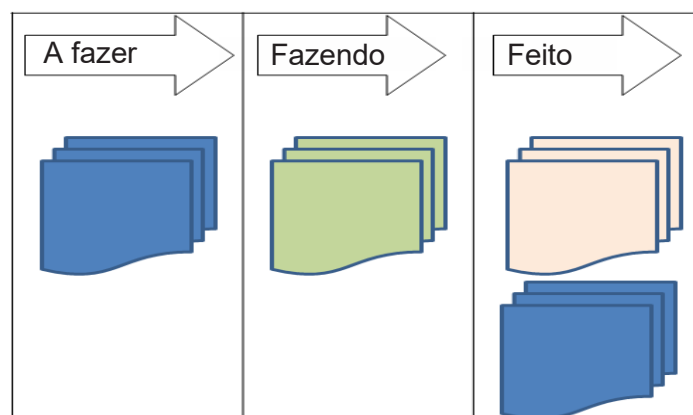


Fonte: https://support.monday.com/hc/article_attachments/360000272939/image.png.

Elementos em um quadro kanban

Comece com o que você tem! Essa é uma premissa básica para se iniciar em kanban. Não importam as diversas variedades de kanban encontradas no mercado e em *softwares* próprios para isso. O importante é que você comece de alguma forma certa. O quadro com três colunas é a forma mais básica e possui as colunas: **A fazer**, **Fazendo** e **Feito**. Outro componente de seu quadro deverá ser os cartões que representarão os itens de trabalho.

Figura 16. Quadro kanban.



Fonte: Elaboração própria do autor.

Segundo Rehkopf (2019), uma das primeiras coisas que você notará sobre um quadro kanban são os cartões visuais (adesivos, *tickets* ou outros). As equipes kanban escrevem todos os seus projetos e itens de trabalho em cartões, geralmente um por cartão. Para equipes ágeis, cada cartão pode encapsular uma história de usuário. Uma vez no quadro, esses sinais visuais ajudam os companheiros de equipe e as partes interessadas a entender rapidamente no que a equipe está trabalhando.

As colunas representam uma tarefa específica que define o fluxo de trabalho. Os cartões fluirão pelas colunas ou fluxos, rumo à conclusão.

Os WIPs (*work in progress*) são o número máximo de cartões que podem estar em uma coluna em um determinado momento. Uma coluna com um limite de WIP de três não pode ter mais de três cartões nela. Esses limites de WIP são críticos para expor gargalos no fluxo de trabalho e maximizar o fluxo. Os limites de WIP fornecem um sinal de aviso antecipado de que você se comprometeu com muito trabalho REHKOPF

CAPÍTULO 3

Corda de Andon

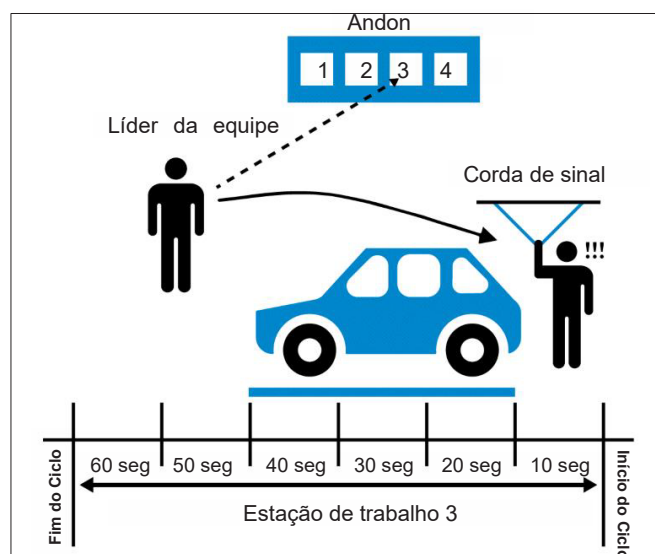
Outro princípio de fabricação Lean e ferramenta usada para resolver problemas no *pipeline* de produção é o Andon Cord. Esta ferramenta vem diretamente do Toyota Production System (TPS). Criado pela Toyota, o Andon Cord é um método de qualidade Jidoka projetado para sinalizar aos colegas e à gerência que há um problema na linha de montagem. Se o problema persistir após um tempo determinado, toda a produção fica parada até que a equipe possa encontrar uma solução.

Puxar o Andon Cord não é apenas resolver um problema. Como parte do processo de melhoria contínua, oferece uma oportunidade adicional de construir qualidade no sistema de produção – ou no *pipeline* de desenvolvimento. A produção para a melhoria começa.

Embora o conceito de desenvolvimento de *software* seja mais desafiador, os princípios devem permanecer os mesmos.

Na Toyota, se algo está errado, não é desnecessariamente escalada a cadeia de comando. Qualquer trabalhador na linha de montagem pode, e deve, puxar o Cordão Andon. Isso sinaliza ao líder da equipe que há um problema, se o trabalhador e o líder da equipe não puderem encontrar uma resolução em um período de tempo predefinido, a linha de montagem será interrompida.

Figura 17. Simulação da Corda de Andon.



Fonte: Kim *et al.* (2018, p.363).

Durante o desenvolvimento, quando um problema é detectado, será discutido abertamente, porém o projeto avança independentemente, e os problemas continuam no processo.

Um exemplo para utilizar o Andon Cord pode ser os testes automatizados dentro do *pipeline* de desenvolvimento, sinalizando os problemas para a equipe. Se um teste falhar, o líder da equipe e o desenvolvedor devem tentar resolver o problema. De preferência, dentro de um prazo acordado. Se eles não puderem resolver o problema, o desenvolvimento é totalmente parado. A equipe pode trabalhar o problema até que ele resolva o código e limpe o *pipeline*.

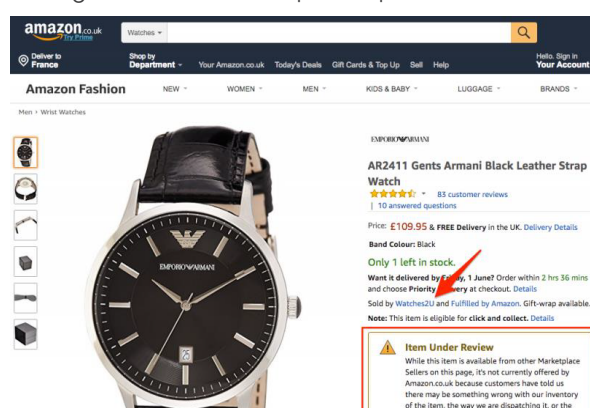
Os testes devem abranger todas as etapas do desenvolvimento de *software*, desde a qualidade e o desempenho do código até a conformidade e segurança. Desta forma, quando um teste falha e a produção para até ser resolvida, há certeza de que pelo menos o evento não se repetirá.

Corda de Andon e a Amazon

A Amazon adotou em seu *e-commerce* o princípio da Andon Cord para seu inventário. Segundo Bayard (2019), cordas de andon são acionadas quando a Amazon tem dúvidas sobre seu inventário no local. O cabo pode ser puxado por um funcionário que recebe uma reclamação do cliente ou por um algoritmo que vê uma taxa anormal de retornos (muitas pessoas retornam produtos quando percebem que houve descontos após a compra). As dúvidas são geralmente sobre problemas de etiquetagem incorreta ou produtos defeituosos (um produto defeituoso foi fabricado).

A razão pela qual a Amazon faz isso é para ter certeza de que não vai enviar mais produtos defeituosos. Ela quer checar que o problema é um alarme falso ou um caso isolado e não há problema com o restante do lote.

Figura 18. Produto bloqueado por Andon Cord.



Fonte: https://blueboard.io/resources/wp-content/uploads/2018/05/Pasted_Image_30_05_2018_11_53-1024x629.png.

Corda de Andon e Netflix Chaos Monkey

A Netflix projetou o Chaos Monkey para testar a estabilidade do sistema aplicando falhas por meio da terminação aleatória de instâncias e serviços dentro da sua arquitetura. Após a migração para a nuvem, o serviço da Netflix dependia da Amazon Web Services e precisava de uma tecnologia que mostrasse como o sistema respondia quando os componentes críticos de sua infraestrutura de serviços de produção eram removidos. Intencionalmente, causar essa falha única eliminaria quaisquer pontos fracos em seus sistemas e os direcionaria para soluções automatizadas que tratariam de forma graciosa de futuras falhas desse tipo.

O Chaos Monkey ajudou a impulsionar a Chaos Engineering como uma nova prática de engenharia. A Chaos Engineering é uma abordagem disciplinada para identificar falhas antes de se tornarem interrupções. Seu objetivo é identificar problemas ocultos que possam surgir na produção. Ao testar proativamente como um sistema responde às condições de falha, é possível identificar e corrigir falhas antes que elas se tornem públicas devido a interrupções.

A Chaos Engineering permite validar o que você imagina estar ocorrendo realmente em seus sistemas. Ao realizar as menores experiências possíveis, você pode medir quebras de sistemas que você mesmo provocou, a fim de aprender a construir sistemas mais resilientes.

Em 2011, a Netflix anunciou a evolução do Chaos Monkey com uma série adicional de ferramentas conhecida como The Simian Army. Inspirados pelo sucesso de sua ferramenta Chaos Monkey original, que visa desabilitar aleatoriamente instâncias de produção e serviços, a equipe de engenharia desenvolveu ferramentas adicionais, construídas para causar outros tipos de falhas e induzir condições anormais do sistema.

Por exemplo, a ferramenta Latency Monkey introduz atrasos artificiais na comunicação cliente-servidor, permitindo que a equipe da Netflix simule a indisponibilidade do serviço sem realmente reduzir o serviço.



Andon Cord, DevOps, Safety Culture, NASA, Amazon, Netflix

<https://twitter.com/realgenekim/status/655045231314792448>.

Principles Of Chaos Engineering

<http://principlesofchaos.org/?lang=ENcontent>.

Chaos Community

<http://chaos.community/>.

Chaos Engineering

<https://www.infoq.com/articles/chaos-engineering/>.

Chaos & Intuition Engineering at Netflix

<https://www.youtube.com/watch?v=Q4nniyAarbs>.

CAPÍTULO 4

Mapa de fluxo de valor

O Mapeamento de Fluxo de Valor (*Value Stream Mapping*) é uma técnica para analisar, projetar e gerenciar o fluxo de materiais e informações necessárias para levar um produto a um cliente. Ele utiliza fluxogramas e simbologia próprias para ilustrar, analisar e aprimorar as etapas necessárias para entregar um produto ou serviço. Uma parte fundamental da metodologia Lean, o VSM, revisa o fluxo das etapas do processo e as informações desde a origem até a entrega ao cliente, para identificar e eliminar o desperdício. Tradicionalmente, ele tem sido usado na manufatura, mas os princípios são realmente muito úteis em mais áreas do que linhas de montagem, como iniciativas de TI e operações.

A lista a seguir é uma sugestão de como o VSM pode ser utilizado por organizações DevOps:

- » Ajuda a identificar gargalos e pontos problemáticos;
- » Gerencia erros e defeitos;
- » Cria maior visibilidade e rastreabilidade durante todo o ciclo;
- » Elimina processos redundantes e dispendiosos;
- » Fomenta a colaboração multifuncional;
- » Revela oportunidades para automação;
- » *Feedback* mais rápido e integrado;
- » Fornece clareza de contexto e processo com dados e recursos visuais;
- » Enfatiza os resultados e os KPIs.

Em uma organização DevOps, as metodologias de VSM e Lean são adaptadas para ações específicas, como mover o trabalho entre as equipes para criar entregas de produtos de valor e relatórios de incidentes. O DevOps VSM é uma representação visual unificada de como a TI e os negócios criam, implantam e gerenciam fluxos de trabalho, começando com o SDLC e passando pela garantia de qualidade, liberação e operações.

As métricas em uma organização DevOps parecerão um pouco diferentes dos usos tradicionais. No entanto, ainda é possível usar as métricas sugeridas, simplesmente adaptadas ao seu próprio domínio. Aqui está uma rápida visão geral dos três componentes principais:

Valor adicionado (VA)

O tempo de valor agregado é a quantidade de tempo que uma equipe realmente gasta trabalhando no projeto.

Prazo de execução (LT)

Lead time é o tempo que a pessoa leva pra efetuar uma tarefa

% Completo / preciso (% C/A)

Porcentagem baseada em informações de “completo e preciso” no momento que não precisa de retrabalho.

Para criar um VSM, as seguintes etapas poderão ser seguidas:

Mapeie o VSM e Devops

Identifique uma fatia de valor comercial no produto a ser desenvolvido. Isso conduzirá o caminho mais fácil para as próximas etapas.

Identifique o desperdício

O VSM ajuda a identificar os desperdícios excessivos de um projeto. Nomear o tipo de desperdício em seu processo é útil para orientar o tipo de melhorias necessário. O desperdício pode ocorrer em grupos de pessoas, processos específicos, tecnologia ou mesmo em uma combinação.

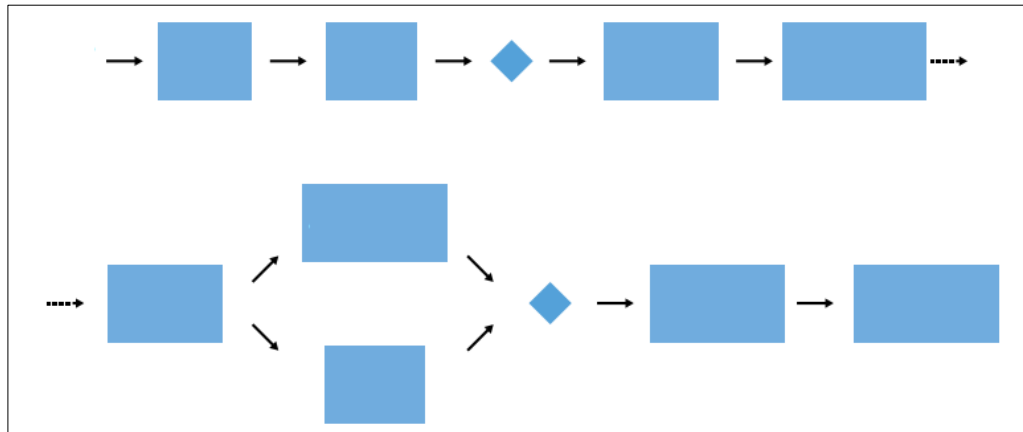
Capacitação da equipe

Capacite uma equipe experiente que possa garantir a confecção do mapa em tempo hábil.

Linha do tempo

Crie uma linha do tempo ou *timeline* de processos

Figura 19. Mapa de fluxo de valor.



Fonte: Kim *et al.* (2018, p. 65).

Comunicação

Comunicar e receber *feedback* em tempo real de toda equipe e partes interessadas do DevOps. As informações devem ser atualizadas e compartilhadas com controle de acesso personalizável.

Benefícios do mapeamento do fluxo de valor

- » A redução ou eliminação de desperdícios melhorarão os resultados das empresas. Como trabalho secundário, é possível descobrir a fonte dos problemas de desperdícios.
- » Uma vez que os fluxos desnecessários são identificados como parte dos visualizadores de VSM, suas equipes podem conscientemente melhorar o comportamento, a cultura, a comunicação e a colaboração.
- » As equipes priorizam com base na perspectiva do cliente.

Desafios do mapeamento do fluxo de valor

Cuidado com o excesso de VSM que poderá ser um desperdício em si mesmo, se não formos cuidadosos.

Veja como evitar algumas armadilhas:

- » Como em toda boa prática de planejamento de projetos, o nível de esforço para conduzir um VSM deve ser equilibrado com o valor potencial e a economia, assim como o retorno de investimento (ROI).
- » Envolver pessoas experientes na condução do VSM, já que o processo de mapeamento pode ser multifuncional e complexo.
- » O medo e a incerteza nas mentes das pessoas são sintomas comuns quando um VSM está sendo conduzido, e o processo de identificação de resíduos, portanto, pode ser intenso.
- » O segredo é começar devagar no aprendizado, pois muitos podem não captar a ideia, e o avanço será muito lento.
- » Não se apresse em usar gráficos, ferramentas e símbolos profissionais imediatamente. Primeiro, desenhe com um lápis ou use um quadro branco e obtenha a ideia.

O mapeamento do fluxo de valor usa um conjunto de símbolos exclusivos para visualizar um processo. Os principais são:

Processo

Um processo é representado com um retângulo e a palavra “Processo”. Um processo geralmente representará os processos coletivos de um departamento inteiro.

Figura 20. Processo.

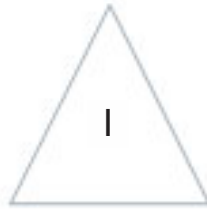


Fonte: <https://wcs.smartdraw.com/value-stream-map/img/process-symbol.jpg>.

Inventário

Um triângulo com um “I” representa a troca de inventário durante o processo.

Figura 21. Inventário.

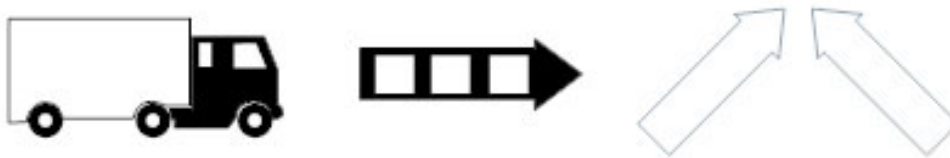


Fonte: <https://wcs.smartdraw.com/value-stream-map/img/inventory-symbol.jpg?bn=1510011148>.

Envio

Uma remessa de matérias-primas de fornecedores é representada com setas largas em branco.

Figura 22. Envio.



Fonte: <https://wcs.smartdraw.com/value-stream-map/img/shipment-symbol.jpg>.

Fornecedor e cliente

Fornecedores e clientes compartilham o mesmo símbolo, que se parece com uma representação geométrica abstrata de uma fábrica. Um fornecedor marcará o início de um processo e será localizado à esquerda do fluxo de valor, enquanto um cliente geralmente é encontrado como a última etapa, na extremidade direita do mapa do fluxo de valor (SmarDraw).

Figura 23. Fornecedor ou cliente.



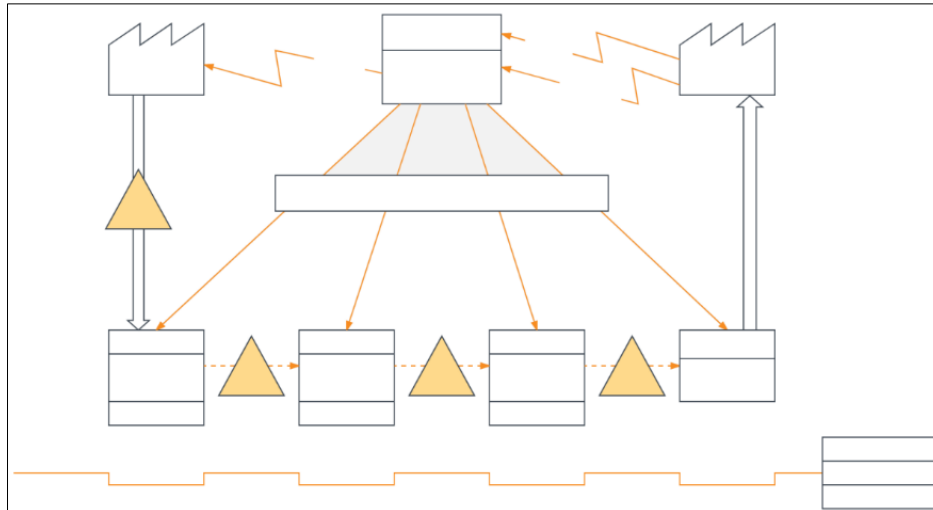
Fonte: <https://wcs.smartdraw.com/value-stream-map/img/supplier-customer-symbol.jpg>.

Existem ferramentas que auxiliam o desenvolvimento de um mapa de fluxo de valor. Essas ferramentas são hospedadas em nuvem, o que proporciona o acesso muito facilitado em equipes móveis.

Ludichart

É uma ferramenta que pode ser acessada em <https://www.lucidchart.com/pages> e possui uma versão gratuita para um usuário.

Figura 24. Exemplo de mapeamento de fluxo de valor.

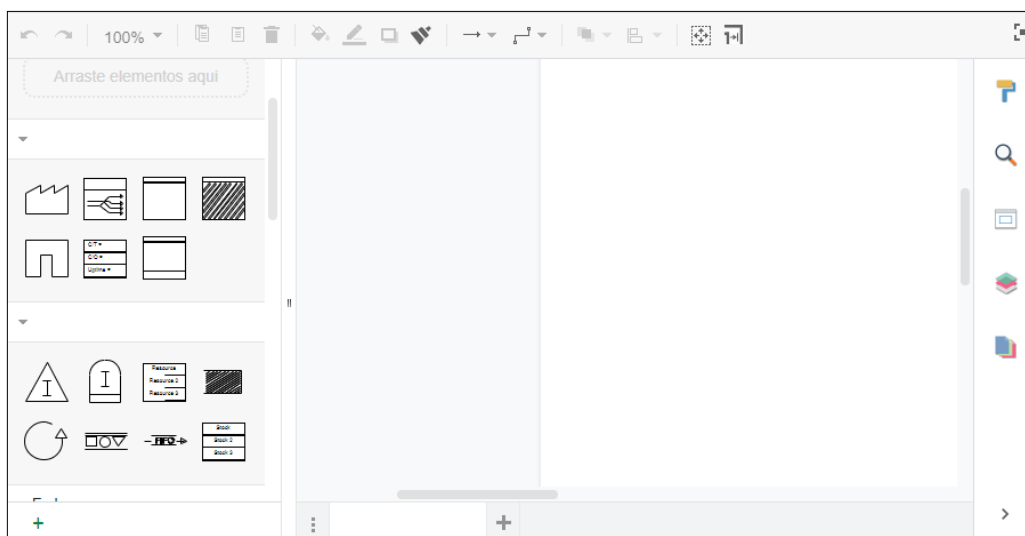


Fonte: <https://d2slcw3kip6qmk.cloudfront.net/marketing/pages/chart/examples/valuestreammappingexample.svg>.

Visual Paradigm

Uma das mais completas ferramentas para desenvolvimento de diversos tipos de diagramas. Também possui uma versão gratuita, e diversos tipos de diagramas podem ser feitos. Pode ser acessada em: <http://visual-paradigm.com>.

Figura 25. Área de trabalho.

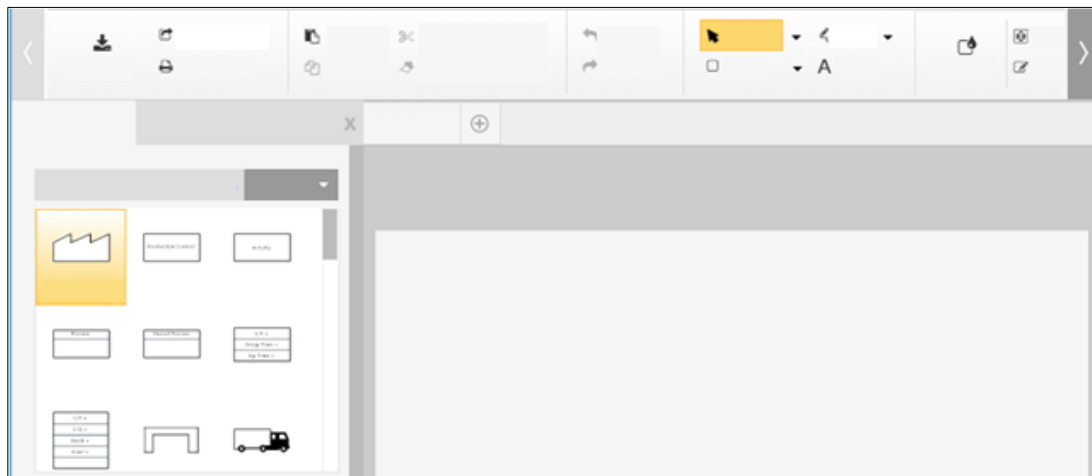


Fonte: Ferramenta visual-paradigm: elaboração própria do autor.

SmartDraw

A SmartDraw também é uma ferramenta de desenvolvimento de diversos diagramas, porém possui uma versão online e uma instalável.

Figura 26. Área de trabalho.



Fonte: Ferramenta SmartDraw: elaboração própria do autor.

CAPÍTULO 1

Cultura colaborativa

Como mencionado anteriormente, DevOps é um processo, um movimento, uma filosofia cultural. Uma empresa poderá ter adotado os melhores processos ou as melhores ferramentas, porém se as pessoas não estiverem alinhadas com a cultura DevOps, o gestor não conseguirá levar adiante os processos de implantação. Portanto, construir essa cultura é a peça fundamental.

Construir uma cultura de DevOps requer que os líderes da organização trabalhem com suas equipes para criar um ambiente e uma cultura de colaboração e compartilhamento. Os líderes devem remover quaisquer barreiras impostas à cooperação.

Se uma equipe de desenvolvimento, responsável por cuidar do sistema até sua entrega, assume a responsabilidade de cuidar de um sistema ao longo de sua vida útil, ele pode ter a real noção do que é o trabalho da equipe de operações e identificar maneiras de simplificar a implantação e a manutenção.

Por outro lado, quando a equipe de operações compartilha a responsabilidade das metas de negócios de um sistema, ela pode trabalhar mais de perto com os desenvolvedores para entender melhor as necessidades operacionais de um sistema e ajudá-los.

Quando iniciamos o trabalho de introduzir Devops, atribuímos a alguém ou a uma equipe o papel de DevOps. Os membros do TI tradicional veem como um impacto muito forte e é denominado um “antipadrão”. Fazer isso perpetua os tipos de silos que o DevOps pretende quebrar e impede que a cultura e as práticas de DevOps se espalhem e sejam adotadas pela organização como um todo.

Estabelecendo uma cultura de aprendizado

Segundo Kim *et al.* (2018, p. 275), “um dos pré-requisitos para uma cultura de aprendizado é quando ocorrem acidentes e a resposta é vista como “justa”, pois, quando são vistas como injustas, isso atrapalha as investigações de segurança, causando medo em vez de atenção, tornando as organizações mais burocráticas a cuidadosas.”

Reunião Post-mortem sem culpa

A reunião de post-mortem deve ser agendada logo quando um acidente ocorre, pois, caso contrário, pode haver esquecimentos de fatores importantes para que futuras soluções sejam escritas.

Segundo Kim *et al.* (2018, p. 278), “durante a reunião, frases como “poderia ter” são declarações contrafactuais e devem ser evitadas, pois enquadram o problema em termos de coisas imaginadas e não em termos de realidade. Um dos resultados é que pessoas frequentemente se culparão por coisas fora de seu controle ou sua habilidade.”

Usar chats para capturar conhecimento organizacional

“Essa ideia foi difundida para GitHub, e o objetivo foi colocar ferramentas de automação, no meio das conversas, em salas de bate-papo, criando transparência e documentações. Também criaram um *bot* para interagir com a equipe de Ops. Com isso, todos viam o que estava acontecendo. Engenheiros, no primeiro dia de trabalho, poderiam aprender rapidamente, aumentando a colaboração” (KIM *et al.*, 2018, p. 290).

Expandindo DevOps para a organização

Um esforço inicial muito grande será realizado, independentemente do escopo, entretanto o marketing positivo sobre qualquer pequeno avanço será fundamental. Para isso, é muito importante dividir as etapas a serem realizadas em ciclos incrementais.

Segundo Kim *et al.* (2018, p. 59), “à medida que o sucesso vai sendo gerado, ganhamos o direito de expandir o escopo de nossa iniciativa DevOps e

descrevemos uma lista de fases ideais para construirmos uma coalisão e base de apoio:”

- » Encontrar inovadores concentrando esforços em equipes que realmente querem ajudar e simpatizantes da jornada DevOps.
- » Construir massa crítica e silenciosa trabalhando com equipes mais receptivas e evitar batalhas políticas perigosas que poderiam prejudicar a iniciativa.
- » Identificar oponentes que são os detratores influentes e conhecidos que têm mais propensão a resistir aos esforços.

Obter ótimos resultados integrando operações na rotina do desenvolvimento

Tendo em mente que o objetivo é gerar resultados voltados ao mercado e entregar valor aos clientes, pode haver um obstáculo quando a equipe de Operações é centralizada.

Segundo Kim *et al.* (2018, p. 93), “é possível gerar mais resultados voltados ao mercado, integrando recursos de Ops nas equipes de Dev, tornando-as mais produtivos e eficientes.”

Tornar o trabalho de Ops visível no quadro kanban

É normal a equipe de desenvolvimento utilizar o quadro kanban para demonstrar visualmente o status de seus trabalhos. Entretanto o trabalho de operações normalmente não é colocado no quadro. Segundo Kim *et al.* (2018, p. 104), “como operações também fazem parte do fluxo de valor do produto, devemos colocar seu trabalho no quadro. Isso permite um melhor monitoramento, acompanhamento e integração.”

Criar repositório único, de verdade, para o sistema inteiro

O controle de versão e repositório de código-fonte já é um padrão entre desenvolvedores individuais e equipes. Ele registra mudanças de arquivos, textos

que podem ser arquivos de configuração, recursos ou código-fonte. Ao juntar todos os arquivos no controle de versão, ele se torna um repositório de verdade. Segundo Kim *et al.* (2018, p. 115), “o controle de versão serve para todos em nosso fluxo de valor, incluindo QA, Operações, InfoSec e desenvolvedores. Existe uma brincadeira entre os desenvolvedores que até os Hubs de repositórios de código e controle de versões online são considerados redes sociais.”

Construir um conjunto de teste de validação automatizado, rápido e confiável

Para evitarmos cenários dos quais não conseguiremos, em tempo hábil, descobrir quem incluiu uma alteração que prejudicou o funcionamento do sistema, precisamos de testes rápidos e automatizados que o execute dentro do sistema, toda vez que alguma alteração for efetuada.

Segundo Kim *et al.* (2018, p. 131), os testes normalmente caem nas seguintes categorias:

- » Teste unitário: testam métodos, classes ou funções isoladamente, garantindo ao desenvolvedor que seu código funcione conforme projetado.
- » Teste de aceitação: testam o aplicativo como um todo para garantir que um nível mais alto de funcionalidade aja conforme o projetado. Seu objetivo é provar que o aplicativo faz o que o cliente pretende.
- » Teste de integração: garante que o aplicativo interaja corretamente com outros aplicativos ou serviços de produção e é realizado sempre que os testes unitários e de aceitação já foram realizados.

Mentalidade de crescimento

Segundo Guckenheimer (2017), as equipes de DevOps aplicam uma mentalidade de crescimento. As equipes tornam as crenças explícitas, criam uma possibilidade hipotética do impacto para obter bons resultados e implementar as hipóteses como se fossem experimentos. Fazem uso massivo da telemetria para reunir evidências e observar resultados. No momento em que as evidências diminuem a hipótese, esses resultados tornam-se oportunidades para falhar rapidamente ou coletar o aprendizado do experimento. Quando as evidências suportam hipóteses, a equipe usa a oportunidade para perseverar ou dobrar as ações que levam à melhoria.

Otimização de tempo médio entre falhas

Na transição para o DevOps, as equipes mudam sua prioridade de otimizar o tempo médio entre falhas (*mean time Between failures*) para o tempo de mitigação (*mean time to mitigation*) e o tempo médio para remediar (*mean time to resolve*). Ao contrário do passado, quando longos processos eram projetados para impedir mudanças que poderiam levar a problemas no campo, as equipes de DevOps enfatizam a capacidade de se mover rapidamente, entender o impacto e reagir rapidamente (GUCKENHEIMER, 2017).

Definição de KPIs

A cultura colaborativa de DevOps tem em suas características a comunicação, o *feedback*, automatização etc. Saber utilizar métricas e *KPIs* é muito importante para que os acontecimentos sejam previstos antes que ocasionem um estrago maior. Existem várias formas de medir o caminho em que o *software* ou o *hardware* estão nos levando. O artigo do site AlertOps (2018) sugere uma base de alertas:

Tempo médio para detectar (MTTD)

O tempo médio de detecção (MTTD) refere-se ao tempo médio necessário para descobrir um problema. Ele mede o período entre o início de uma indisponibilidade do sistema, o mau funcionamento do serviço ou qualquer outra atividade geradora de receita e a quantidade de tempo que uma equipe de DevOps precisa para identificar esse problema.

Tempo médio de falha (MTTF)

O tempo médio de falha (MTTF), também conhecido como “tempo de atividade”, é o tempo médio que um sistema defeituoso pode continuar sendo executado antes de falhar. O tempo começa quando ocorre um defeito sério em um sistema e termina quando o sistema falha completamente. O MTTF é usado para monitorar o status de componentes do sistema não reparáveis e analisar quanto tempo um componente executará no campo antes que ele falhe.

Tempo médio entre falhas (MTBF)

O tempo médio entre falhas (MTBF) é uma métrica de confiabilidade e disponibilidade. Ele é usado para medir a capacidade de um sistema ou

componente de executar suas funções exigidas sob condições estabelecidas por um determinado período de tempo.

Tempo médio de resolução (MTTR)

O tempo médio de resolução (MTTR) refere-se ao tempo necessário para corrigir um sistema com falha. Também é conhecido como tempo médio para resolução. É uma medida do tempo médio que uma equipe de DevOps precisa para reparar um sistema inativo após uma falha.

Estudo de caso: criação de métricas self-service no LinkedIn

O LinkedIn foi criado em 2003 para ajudar os usuários a se conectarem em suas redes em busca de oportunidade de trabalho e, em novembro de 2015, tinha mais de 350 milhões de membros.

Em 2011, seu diretor de engenharia, Prachi Gupta, escreveu que lá no LinkedIn eram muito enfatizados a telemetria, os KPIs e a visualização de gráficos para monitorar as falhas antes que elas comessem a acontecer. No entanto, em 2010, mesmo gerando um volume muito grande de dados, era extremamente difícil de analisá-los. Foi então que Eric Wong deu início à criação de um novo mecanismo: o InGraphs. Depois de sua criação, muitas outras funcionalidades foram adicionadas, até que, finalmente, Gupta pode dar sua conclusão que, a eficácia do sistema de monitoramento ganhou destaque quando a funcionalidade de monitoramento do InGraphs, começou a mostrar tendências decrescentes, ligadas a um provedor de e-mail quando o mesmo só percebeu que havia algum problema, após a equipe de suporte entrar em contato.

“O *InGraphs* teve tanto sucesso que os gráficos em tempo real são apresentados em destaque nos escritórios de engenharia da empresa” (KIM *et al.*, 2018, p. 209).



Mais imagens da sala de monitoramento e painéis de controles podem ser acessadas em: <https://engineering.linkedin.com/32/eric-intern-origin-ingraphs>.

Figura 27. Tela de monitoramento do InGraphs.



Fonte: <https://engineering.linkedin.com/32/eric-intern-origin-ingraphs>.

CAPÍTULO 2

Benefícios

Em um artigo publicado na DevopsDigest, intitulado DevOps Prediction (12/2015), visualizado através do link <https://www.devopsdigest.com/2016-devops-predictions-1>, acessado em 8/11/2019, por meio de um webinar, vários CIOs de empresas que fornecem os diversos serviços para a implementação do DevOps relataram as principais vantagens e benefícios de se implementar DevOps. As experiências com DevOps relatadas sempre serão aquelas vividas por cada situação, cada projeto, cada organização e, portanto, poderemos obter alguns relatos diferentes para a mesma situação.

Satisfação para o cliente

Um exemplo é o conceito de satisfação ao cliente após encomendar um produto desenvolvido por uma empresa DevOps. Aruna Ravichandran VP de produto DevOps e marketing de soluções, CA Technologies (<http://ca.com>), menciona que o principal objetivo do DevOps é fornecer *software* de maior qualidade aos usuários finais em um ritmo mais rápido, gerando benefícios em torno de uma melhor experiência do cliente e maior oportunidade de receita.

O resultado final, certamente será um produto de qualidade alinhado com o que foi pedido. Sua base colaborativa é oferecida ao cliente para melhorar a experiência com o usuário e aderir à motivação de melhoria contínua.

Já Andreas Grabner, estrategista de tecnologia da Dynatrace (<https://dynatrace.com/>), afirma que, comercialmente, o cliente estará se mantendo à frente da concorrência, oferecendo produtos de melhor qualidade, entregas mais precisas e rápidas. Ao introduzir uma cultura focada em colaboração com múltiplos ciclos de *feedback* entre Dev, Ops e equipes de negócios, as organizações podem identificar problemas no *pipeline* de desenvolvimento, enviar menos códigos ruins, reduzir o tempo gasto em combate a incêndios, melhorar o tempo médio de reparo e finalmente entregar uma melhor experiência do cliente. O DevOps permite que você cumpra a promessa não dita aos consumidores que não consideram o desempenho digital um luxo, mas esperam isso em nosso mundo hiperconectado.

Stefan Schneider, gerente de Marketing, SevOne, acredita que a razão mais importante para adotar o DevOps do ponto de vista de negócios é obter a capacidade de ser proativo na geração de qualidade de experiência para seus usuários. Como isso afeta seu resultado final? Ter uma organização mais proativa pode melhorar constantemente os produtos e ferramentas em que seus usuários confiam, economiza seu tempo, reduz o custo das operações e gera maior receita pelo maior engajamento do cliente, redução de revisões e melhor controle de custos e cronogramas.

Quebrando os silos

Vimos em capítulos anteriores que um dos principais objetivos do DevOps era criar equipes celulares e desmontar os silos criados pela TI tradicional. Jason Bloomberg, presidente da Intellyx (<https://intellyx.com>), e Samir Ibradžić, chefe de Infraestrutura e Sistemas da Midokura(<https://midokura.com/>), acreditam que a vantagem estratégica mais importante é provar que as abordagens auto-organizacionais podem quebrar com sucesso os silos que resultam de modelos organizacionais hierárquicos. O Manifesto Ágil apregooou a auto-organização para pequenas equipes, mas estendendo-a por meio das linhas organizacionais será a mais importante vantagem final para o DevOps ao longo do tempo.

Alinhando TI aos negócios

Um dos principais objetivos do DevOps é criar empatia em todas as equipes de uma organização, de modo que o valor comercial seja trazido para o primeiro plano de todos os departamentos e essa linha de pensamento significa maior eficiência na inovação de produtos e serviços e agilidade de mudar em um centavo com os mercados e a concorrência. Jason Hand DevOps Evangelista, VictorOps (<https://victorops.com/>).

Segundo Ashish Kuthiala, Diretor Sênior, Estratégia e Marketing, DevOps, Hewlett Packard Enterprise(<https://hpe.com/us/en/home.html>) e Andrew Phillips VP da DevOps Strategy, Xebialabs (<https://xebialabs.com/>), as equipes de negócios e de TI estão igualmente entusiasmadas com o DevOps, pois esse conceito ajuda as duas equipes a trabalharem juntas de forma mais colaborativa e eficaz para expandir os negócios e, ao mesmo tempo, reduzir custos. O DevOps coloca à sua disposição um conjunto de ferramentas, práticas e ideias que você pode usar para resolver qualquer problema de negócios principal.

Agilidade

A razão mais importante pelas quais as empresas mais bem-sucedidas adotaram o DevOps é que elas compartilham a necessidade de operar como pequenas empresas e, ao mesmo tempo, entregar resultados de grandes empresas. Numa época em que a distância entre interrupção e morte (pense no PayPal vs. Lehman Brothers) pode ser tão pequena quanto alguns recursos lançados algumas semanas antes, a inovação, impulsionada pela agilidade e velocidade, tornou-se imperativo comercial – não técnico. Dan Turchin VP de Produto, BigPanda (<https://www.bigpanda.io/>).

A principal vantagem da adoção do DevOps no ambiente de negócios atual é a agilidade nos negócios. À medida que a taxa de mudança para os negócios acelera, as empresas são menos capazes de prever para onde os negócios estão indo. O principal imperativo estratégico torna-se responder por meio de agilidade e modularidade por meio de DevOps e TI adaptável. Quando você não sabe o que o futuro reserva, é importante ter a infraestrutura pronta para responder às necessidades dos negócios em tempo real. Paola Moretto Fundadora e CEO, Nouvola (<https://nouvola.com/>).

A agilidade é o motivo mais importante para adotar o DevOps. A velocidade de comercialização e a iteração rápida são essenciais para competir em *software*, caso contrário, os concorrentes superarão o desenvolvimento de seu produto e ganharão o mercado. Para competir, as equipes de engenharia devem adotar o DevOps para capacitar as equipes e reduzir o atrito, a fim de aumentar a agilidade e a velocidade para o mercado. Kevin Fishner Chefe de Operações Comerciais da HashiCorp (<https://www.hashicorp.com/>).

Tempo rápido para a produção

Quando você usa a metodologia ágil do DevOps, a TI trabalha diretamente com os usuários de negócios e fornece exatamente o que eles precisam – e nada mais. Ao focar as necessidades de negócios, primeiro, vemos que os projetos são concluídos e passam para a produção rapidamente. Estrategista de Aliança de Kent Erickson, Zenoss (<https://www.zenoss.com/>).

O motivo para a adoção do DevOps está atingindo o limite de escalabilidade no processo de integração e liberação orientado para humanos. Todos nós fizemos muito bem sem DevOps com lançamentos trimestrais. Não podemos liberar a produção várias vezes ao dia com procedimentos manuais. Baruch Sadogursky Developer Advocate, JFrog (<https://jfrog.com/>).

Para fornecer a prestação de serviços em velocidade sem precedentes, as organizações precisam incorporar as práticas de DevOps. O maior valor que o DevOps oferece é a velocidade, garantindo uma entrega mais rápida de recursos. Os principais aspectos são implantações automatizadas e integração contínua, garantindo que menos erros sejam cometidos e que a produtividade seja aumentada. Richard Whitehead Chefe Evangelista, Moogsoft (<https://www.moogsoft.com/>).

Por fim, o DevOps permite que as empresas ofereçam *software* de missão crítica para a produção muito mais rápido – às vezes, de 10 a 100 vezes mais rápido. Para qualquer empresa que aproveite qualquer tipo de *software*, esse tipo de melhoria pode significar sucesso ou fracasso. No futuro, quando mais empresas implementarem com sucesso o DevOps, não haverá mais espaço para a falha na adoção do DevOps. Embora existam outras vantagens para o DevOps, uma vantagem de velocidade de 10x a 100x é a única que conta. Joan Wrabetz CTO, QualiSystems (<https://www.quali.com/>).

Estabilidade e software de qualidade

Para organizações de TI que adotaram o DEVOPS, todos são responsáveis por criar e executar um aplicativo que funcione conforme o esperado. Toda a equipe compartilha os mesmos objetivos em relação à qualidade: desempenho, experiência do usuário, estabilidade, segurança e até tempo para comercializar. Os princípios do DEVOPS garantem que tudo seja monitorado, todas as mudanças são conhecidas e todos entendem como cada lançamento afeta a qualidade. Gerardo Dada VP, Marketing e Estratégia de Produto, SolarWinds (<https://www.solarwinds.com>).

O DevOps oferece um número inigualável de oportunidades: ele não apenas fornece *software* mais rapidamente, mas também quase sempre leva a melhor estabilidade e qualidade geral. Se isso é uma questão de o DevOps ser muito bom, ou as organizações e processos que acabam sendo muito ruins, é difícil dizer. Kelly Looney Gerente Regional de Consultoria, Estratégia DevOps, Skytap (<https://www.skytap.com/>).

O DevOps oferece maior estabilidade e confiabilidade, pois ciclos de lançamento mais rápidos e mais frequentes nos permitem identificar e resolver problemas imediatamente. Isso significa que os desenvolvedores e outras pessoas da empresa têm mais tempo para se concentrar em melhorias e inovações que contribuem para o resultado final. Joe Alfaro VP de Engenharia, Sauce Labs (<https://saucelabs.com/>).

O argumento mais forte para o DevOps é que isso faz com que as pessoas trabalhem melhor, fazendo com que elas usem chapéus diferentes. Os desenvolvedores que são obrigados a operar o *software* que escrevem, criam um software mais fácil de operar, mais confiável e, em última instância, melhor para os clientes e para o sucesso dos negócios. Sven Dummer Diretor Sênior de Marketing de Produtos, Loggly (<https://www.loggly.com/>).

À medida que os processos automatizados de gerenciamento de versão e de controle de qualidade são implantados com mais frequência, as organizações descobrem uma maneira diferente de planejar e priorizar os atrasos. O lado comercial não é mais obrigado a esperar por janelas de lançamento trimestrais e pode esperar mudanças semanais diárias de hora em hora. Isso elimina a pressão dos lançamentos em relação ao *release stuffing*, permitindo aplicar medidas de controle de qualidade mais cautelosas, resultando em melhor experiência do usuário. Ivo Mägi Co-fundador e Chefe do Produto, Plumb (<https://plumb.io/>).



Mapeando marcos e métricas do DevOps. Gene Kim – Aruna Ravichandran

<https://www.devopsdigest.com/gene-kim-charting-devops-milestones-and-metrics>.

Devops: a cultural rethink. Jason Bloombergs.

<https://www.devopsdigest.com/devops-a-cultural-rethink>.

Making Virtual Test Networks Available to the Whole DevOps Team. Frank Puranik's

<https://www.devopsdigest.com/making-virtual-test-networks-available-to-the-whole-devops-team-0>.

Incredible Opportunities Offered by the API Economy. Sven Hammar

<https://www.devopsdigest.com/incredible-opportunities-offered-by-the-api-economy>.

10 Visions of Development's Future. Malia Powers

<https://www.devopsdigest.com/10-visions-of-developments-future>.

Too Many Alerts from Too Many Tools Put Customers at Risk Dan Turchin

<http://www.apmdigest.com/bigpanda-state-of-monitoring-report-2016>.

The Latest Trends in Developer Productivity. RBaruch Sadogursky

<https://www.devopsdigest.com/the-latest-trends-in-developer-productivity>.

Sandboxes - Uber Containers for Enterprise DevOps. Joan Wrabetz

<https://www.devopsdigest.com/sandboxes-uber-containers-for-enterprise-devops>.

CAPÍTULO 3

Estrutura de equipes

Construindo uma equipe

Como vimos em capítulos passados, DevOps foi feito para quebrar certas barreiras culturais nas organizações, dentre elas os silos das equipes. Para que a cultura do DevOps possa ser implementada e haja uma colaboração efetiva, diferentes organizações podem precisar de diferentes estruturas de equipe.

A coisa mais importante a lembrar quando você constrói uma equipe é que uma equipe, no sentido mais básico, é um grupo de pessoas que trabalha junto para atingir um objetivo comum. Em uma equipe de DevOps, o objetivo é geralmente a entrega de um produto ou um serviço que faz parte de um produto.

Qual é a melhor estrutura?

Se fizermos essa pergunta diretamente a qualquer profissional que tenha implementado DevOps, a resposta deve ser algo como: não há um modelo existente que se encaixe em todas as organizações. Na verdade, a implementação de DevOps é um projeto, e uma das características principais de um projeto é que ele precisa ser único. Portanto, cada caso terá suas particularidades e comportamentos.

Segundo Skelton (2013), é útil caracterizar um pequeno número de modelos diferentes para estruturas de equipe, algumas das quais se adequam melhor a determinadas organizações do que outras. Ao explorar os pontos fortes e fracos dessas estruturas de equipe (ou topologias), podemos identificar a estrutura de equipe que pode funcionar melhor para as práticas de DevOps em nossas próprias organizações, levando em conta a Lei de *Conway*, que diz o seguinte: “Qualquer organização que projete um sistema (definido de maneira ampla) produzirá um design cuja estrutura é uma cópia da estrutura de comunicação da organização.”

Segundo Kim *et al.* (2018, p. 77), o experimento do Dr. Conway foi realizado com duas equipes em que uma das equipes, compostas por oito pessoas, iria fazer um compilador COBOL; e outra, composta por três pessoas, ia fazer um compilado ALGOL. Sua observação foi que o compilador COBOL resultante executa o processo de compilação em cinco etapas; e o ALGOL, em três.

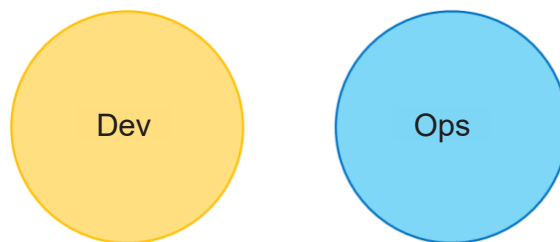
Explicando a lei de Conway, o modo de como iremos organizar nossas equipes terá impacto diretamente no *software* que iremos produzir.

No endereço <https://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/>, Matthew Skelton cita trechos de seu livro, que foi lançado em Setembro de 2019, juntamente com Manuel Pais, algumas topologias para time de DevOps.

Antitipo A: silos separados

Essa topologia é a mais comum, encontrada sempre que as organizações buscam as práticas de DevOps.

Figura 28. Silos separados.

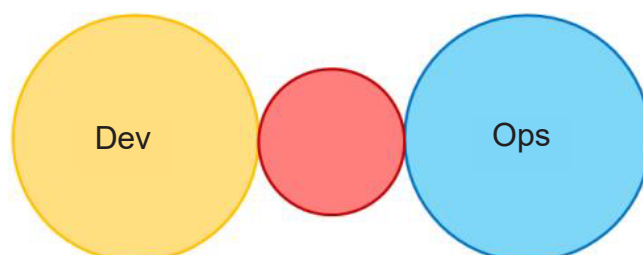


Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide2.png>.

Antitipo B: silo de DevOps separado

Resulta em um gerente ou executivo decidindo que o time precisa começar uma equipe de DevOps. Quando isso acontece, a tendência é a dos membros formarem rapidamente outro silo, mantendo o Dev e o Ops distantes.

Figura 29. Dev, DevOps e Ops.



Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide3.png>.

Antitipo C: “Nós não precisamos de DevOps”

Segundo Matthew (2013), essa topologia é suportada por uma combinação de ingenuidade e arrogância de desenvolvedores e gerentes de desenvolvimento, especialmente ao iniciar novos projetos ou sistemas. Assumindo que o Ops é agora uma coisa do passado, os desenvolvedores subestimam a complexidade e importância das habilidades e atividades operacionais e acreditam que podem passar sem eles, ou apenas cobri-los em horas de folga.

Figura 30. Nós não precisamos de DevOps.



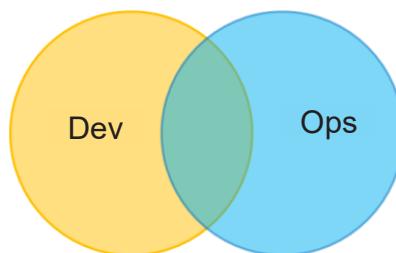
Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide4.png>.

Depois de velicarmos os antitipos, podemos conhecer o lado bom de algumas topologias, citadas por Matthew (2013):

Tipo 1: colaboração suave

Uma colaboração harmoniosa entre as equipes de desenvolvimento e de operações, cada uma se especializando naquilo que é necessário, mas também compartilhando quando necessário.

Figura 31. Colaboração suave.

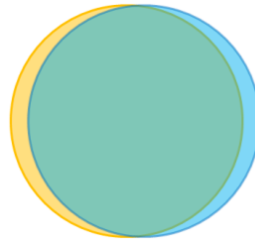


Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide5.png>.

Tipo 2: totalmente incorporado

As pessoas de operações são totalmente incorporadas nas equipes de desenvolvimento de produtos. Há tão pouca separação entre Dev e Ops que todas as pessoas estão altamente focadas em um propósito compartilhado.

Figura 32. Totalmente incorporado.

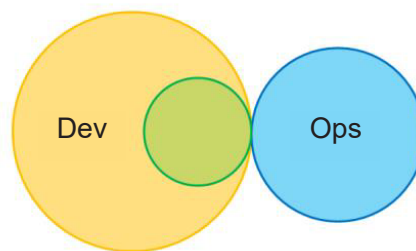


Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide6.png>.

Tipo 3: Infraestrutura como serviço

Voltada para organizações com um departamento de operações de TI bastante tradicional, que não pode ou não mudará rapidamente o suficiente, e para organizações que executam todos os seus aplicativos na nuvem pública.

Figura 33. IaaS.

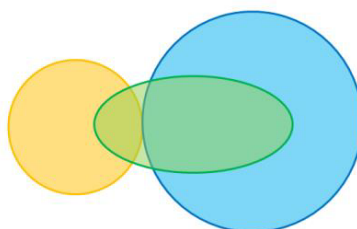


Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide7.png>.

Tipo 4: DevOps-as-a-Service

Algumas organizações podem não ter as finanças, a experiência ou a equipe para assumir a liderança nos aspectos operacionais do *software* que produzem. A equipe de desenvolvimento pode então entrar em contato com um provedor de serviços para ajudá-los a criar ambientes de teste e automatizar sua infraestrutura e monitoramento.

Figura 34. DevOps as services.



Fonte: <https://matthewskelton.files.wordpress.com/2013/10/slide8.png>.

Identificar as equipes que dão suporte ao fluxo de valor

Segundo Kim *et al.* (2018), Esse estágio inclui identificar todos os membros responsáveis por trabalhar em conjunto para gerar valor aos clientes, que em geral inclui:

- » **Dono do produto:** a voz interna da empresa que define o próximo conjunto de funcionalidade no serviço.
- » **Desenvolvimento:** equipe responsável por desenvolver funcionalidade na aplicação dos serviços.
- » **QA:** equipe responsável pela existência de *loops* de *feedback* para garantir que o serviço funcione como desejado.
- » **Operações:** equipe frequentemente responsável por manter o ambiente de produção e ajudar a garantir que os níveis de serviço sejam atingidos.
- » **InfoSec:** equipe responsável pela segurança de sistemas de dados.
- » **Gerentes de releases:** as pessoas responsáveis por administrar e coordenar a implementação da produção e os processos de releases.
- » **Executivo de tecnologia ou gerente de fluxo de valor:** o responsável por garantir que o fluxo de valor satisfaça ou exceda os requisitos do cliente para o fluxo de valor global.

Características da equipe

Além das topologias, as equipes precisam possuir características da cultura DevOps e gerenciamento de projetos.

Tamanho das equipes

Segundo Kim *et al.* (2018, p. 91), a lei de Conway ajuda a projetar os limites das equipes, estimula a manter pequeno seu tamanho e possui quatro efeitos importantes:

- » Equipe possuirá entendimento claro e compartilhado do sistema que está trabalhando.

- » Limitação da taxa de crescimento do produto ou serviço.
- » Descentralização do poder e possibilita a autonomia.
- » Revezamento na liderança de equipes para que não ocorram impactos catastróficos.

Diversidade

Segundo Gajda (2019), uma maneira comum de pensar a diversidade de habilidades é pensar em equipes multidisciplinares que reúnem designers, desenvolvedores, testadores, operações e assim por diante. Cada vez mais, no entanto, muitas organizações estão construindo equipes de desenvolvedores *full-stack*, em que cada pessoa deve ser multidisciplinar. Alcançar a diversidade de mentalidade e cultura é mais difícil do que alcançar a diversidade de habilidades. Em qualquer atividade de equipe, algumas pessoas tendem a se mover rapidamente.

Autonomia

Ter equipes autônomas significa que as equipes são responsáveis por descobrir a melhor forma de realizar o trabalho que precisa ser feito e tomar decisões por si mesmas.

Colocação

Na maioria das vezes, é preferível ter uma conversa cara a cara com um especialista no assunto, em vez de ler um artigo. Quando os membros da equipe são colocados, eles gastam menos tempo escrevendo e lendo e-mails. A colocação melhora a comunicação e a eficiência.

Produtividade

A minimização de distrações concentra-se em observar como as equipes gastam seu tempo e, em seguida, capacitar os membros da equipe e seus gerentes a agir para reduzir o ruído, permitindo que eles se concentrem nas tarefas críticas em mãos. Os benefícios são enormes: as equipes relatam maior satisfação no trabalho, os desenvolvedores podem fazer muito mais do que se estivessem lidando com interrupções constantes, a qualidade do código desenvolvido também é maior porque a linha de pensamento de um desenvolvedor não é interrompida.

Transparência

A transparência cria confiança entre as equipes virtuais e dissolve os limites entre elas. A transparência no local de trabalho significa operar de uma maneira que facilite para todos na sua organização ver o que está acontecendo.

Diversão

Quando os funcionários se divertem no local de trabalho, eles aproveitam seu trabalho e produzem melhores resultados. Os gerentes em ambientes de DevOps se esforçam para criar um ambiente desafiador, criativo e divertido para os funcionários e para si mesmos. Para obter mais informações sobre como criar um ótimo ambiente de trabalho, consulte diversão no local de trabalho.

CAPÍTULO 4

Etapas

No primeiro capítulo foi mencionado que a literatura DevOps do livro *The Phoenix Project* está baseada em três etapas ou maneiras. Como o DevOps se trata de uma cultura e não existe uma receita de implementações bem-sucedida, todo material relacionado ao assunto se apoia em boas práticas, assim como outros guias de gerenciamento de projetos, como o PMBOK, por exemplo.

Sendo assim, detalharemos as três etapas e suas fases de implementação.

Fluxo de valor tecnológico

A primeira etapa deve garantir que o trabalho flua do desenvolvimento para operações de maneira suave, para que o valor seja entregue o mais rápido possível ao cliente.

Segundo Kim *et al.* (2019, p. 15), “aumentando esse fluxo tornando o trabalho visível, reduzindo o tamanho dos lotes e os intervalos de trabalhos (WIP), incorporando qualidade, evitando defeitos, isso faz com que se acelere a fluidez pelo fluxo de valor tecnológico, reduzindo o tempo de execução exigido para atender às demandas.”

Tornando o trabalho visível

A diferença da visibilidade para a manufatura é que o fluxo de valor tecnológico é invisível, pois, ao contrário de processos físicos, não é fácil descobrir onde o trabalho está sendo obstruído, atrasado ou parado.

Para suprir essa necessidade, foi adotado o quadro kanban, já estudado no Capítulo 6. Segundo Kim *et al.* (2018, p. 17), “ao colocar todo o trabalho de cada núcleo em filas, tornando-os visíveis, todos os interessados poderão priorizá-lo mais facilmente no contexto dos objetivos globais.”

Limitar o trabalho em andamento

Quando utilizamos o quadro kanban, podemos colocar um limite para que o trabalho seja mais bem gerenciado. O limite será definido em comum acordo

entre a equipe de desenvolvimento para cada etapa ou coluna. Por exemplo, pode-se definir que deverão ser colocados apenas três cartões na coluna de testes, ou seja, não poderá nenhum outro ser adicionado enquanto não for removido um cartão da referida coluna.

Segundo Kim *et al.* (2018, p. 17), “limitar o WIP também pode facilitar ver os problemas que impedem a conclusão do trabalho. Por exemplo, quando limitamos o trabalho, descobrimos que talvez ficaremos sem fazer nada, pois estamos esperando outra pessoa terminar e retirar o cartão.”

Embora seja tentador iniciar outra tarefa, uma iniciativa melhor seria tentar descobrir o que está causando o atraso e oferecer ajuda.

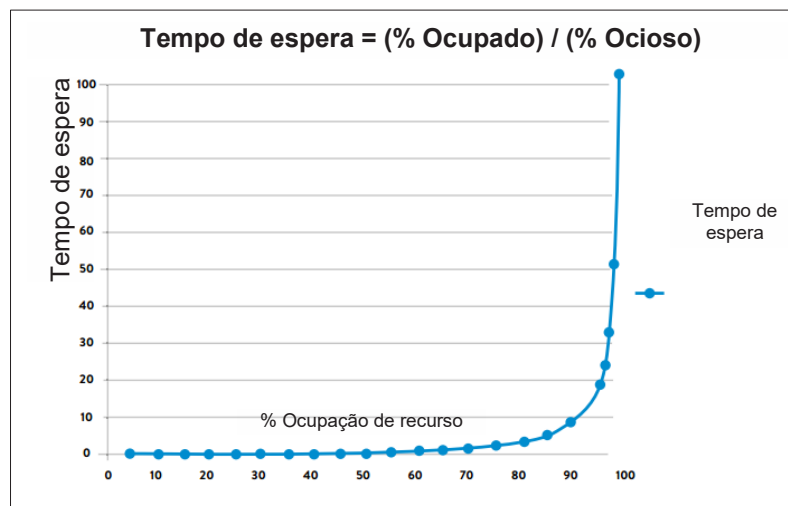
Reduzir o tamanho dos lotes

A redução de lotes também proporciona uma transição suave do fluxo, já lotes grandes resultam em altos níveis de WIP e alta variação de fluxo e o resultado são tempos de execução longos e qualidade ruim.

Redução de numero de transferências

Segundo Kim *et al.* (2018, p. 21), sempre que o trabalho passa de uma equipe para outra, são exigidos todos os tipos de comunicação: solicitação, especificação, sinalização, coordenação, priorização, agendamento, eliminação de conflitos, teste e verificação. Cada etapa é uma fila de potencial atraso e, para diminuir esse problema, é feita uma redução de transferência ou automatização de partes significativas do trabalho ou uma reorganização de equipes.

Figura 35. Tamanho da fila e tempos de espera como função da porcentagem de utilização.



Fonte: Kim *et al.* (2018 p. 360).

Princípios do *feedback*

“O princípio do *feedback* está relacionado com a segunda etapa dos princípios do DevOps. Ela descreve os princípios que possibilitam *feedback* rápido, constante e recíproco, da esquerda para direita, em qualquer estágio do fluxo de valor. Seu objetivo é criar um sistema de trabalho mais seguro e resiliente” (KIM *et al.*, 2018, p. 27).

Normalmente, o trabalho de desenvolvimento e operações ocorre dentro de ambientes críticos, complexos e, caso haja qualquer problema, as consequências são catastróficas. Por isso o fluxo de informação deve ser rápido, e qualquer problema deve ser transformado em aprendizado e não causa para punição e acusação.

Ver problemas quando eles ocorrem

Segundo Kim *et al.* (2019, p. 30), “o objetivo é criar *loops* de *feedback* antecipados para que cada estágio possa detectar imediatamente algum problema, criar telemetrias para ver como os componentes de *software* se comportam. Os *loops* não são apenas para detectar problemas, mas sim informar de como evitá-los no futuro.”

Aglomerar e resolver problemas para construir novo conhecimento

Kim *et al.* (2018, p. 31) “citam que o objetivo da aglomeração é controlar problemas antes que se espalhem e diagnosticá-los para que não ocorram novamente. É nessa situação que normalmente se utiliza a corda de Andon.”

Qualidade

Para que se possa entregar um produto de valor ao cliente, problemas devem ser corrigidos o quanto antes e dentro de sua área de controle. Fazendo isso, estaremos incentivando cada membro da equipe a ser responsável pela qualidade e pela segurança. A tomada de decisão fica dentro da equipe e não para executivos distantes. Isso não só melhora o resultado como acelera o resultado.

Princípios do aprendizado contínuo e experimentação

Estão focados na criação de uma cultura de aprendizagem contínua e experimentação, ou seja, preparam o ambiente para a criação de conhecimentos individuais que serão transformados em conhecimento da equipe organizacional. No fluxo de valor tecnológico, a punição e o medo por erros cometidos não existem. Seu objetivo é criar uma cultura de alta confiança e reforçar que todos são aprendizes e precisam assumir riscos.

Aprendizado organizacional e cultura de segurança

Kim *et al.* (2018, p. 39) “citam a pesquisa do Dr. Wetrum (2004), sendo o primeiro a observar a cultura organizacional da segurança e desempenho, do qual definiu três culturas:”

- » Organizações patológicas são caracterizadas por medo e ameaças;
- » Organizações burocráticas são caracterizadas por regras e processos departamentais e territoriais;
- » Organizações produtivas são caracterizadas por compartilhamento e busca ativo de informações.

Institucionalizar a melhoria do trabalho diário

Existem duas situações em que a melhoria de aprendizado pode encontrar certos obstáculos: as equipes não querem ou não estão motivadas a melhorar, e os processos tendem a se perder, a se degradar ao longo do tempo.

No fluxo de valor tecnológico, segundo Kim *et al.* (2018, p. 40), “o trabalho é melhorado quando se reserva um tempo para pagar dívidas técnicas, corrigir defeitos, refatorar códigos e melhorar áreas problemáticas dos códigos ou dos ambientes. Como resultado, todos encontram e corrigem problemas em suas áreas de controle o tempo todo, como parte do trabalho diário.”

Transformar descobertas locais em melhorias globais

Esse é um princípio que, pelo fato de o DevOps trabalhar com equipes celulares, as descobertas sempre são realizadas localmente. Entretanto, eles devem estabelecer mecanismos que possibilitem ao restante da organização usar e aproveitar o conhecimento.

Kim *et al.* (2018, p. 42) “relatam que, nesse caso, quando alguém for fazer um trabalho similar, o fará com a experiência acumulada e coletiva de todos na organização que já fizeram o mesmo trabalho.”

Líderes reforçam a cultura do aprendizado

É normal esperar que um líder conduza os trabalhos e experimentos de suas equipes. Segundo Kim *et al.* (2018, p. 45), “o método iterativo conduz todos os processos de melhorias internos, além de realizar experimentos para garantir que os produtos realmente ajudem os clientes, internos e externos, a atingir seus objetivos. O líder ajuda a pessoa que está fazendo experimentos com perguntas tais como:”

- » Qual foi seu último passo e o que aconteceu?
- » O que você aprendeu?
- » Qual sua condição agora?
- » Em qual obstáculo você está trabalhando agora?
- » Qual seu próximo passo?
- » Qual o resultado esperado?
- » Quando podemos verificar?

Estudo de caso: programação em pares substituindo revisão de códigos incompletos na Pivotal Labs

Em sua apresentação no Enterprise DevOps Summit 2015, Elisabeth Hendrickson, vice presidente de engenharia da Pivotal Labs, descreveu como existiam dois métodos de revisão de códigos aceitos na empresa: programação em pares ou um processo de revisão de código, gerenciado pelo Gerrit (<https://www.gerritcodereview.com/>).

O problema encontrado no Gerrit foi que o processo de revisão de códigos demorava uma semana para os desenvolvedores receberem a revisão necessária. Para corrigir esse problema e eliminar os atrasos, eles acabaram dismantelando toda a revisão feita pelo Gerrit e, no lugar, entrou a programação em pares para implementar mudanças de códigos no sistema. Com isso, reduziram de semanas para horas o tempo necessário para ter código revisado.

CAPÍTULO 1

Conceito de infraestrutura

Ambiente de desenvolvimento

O ambiente de desenvolvimento é o local onde estão instaladas todas as ferramentas que os desenvolvedores utilizarão para produzir códigos, integrar e atualizar seus códigos com demais membros do projeto. Nesse ambiente, normalmente estão instalados uma IDE, que é uma ferramenta que facilita a digitação de códigos, bibliotecas necessárias para fazer o aplicativo funcionar, emuladores, servidores web, servidores de banco de dados etc. Ou seja, o aplicativo deverá funcionar no ambiente de desenvolvimento com todas as suas funcionalidades.

Esse ambiente deverá ser o mesmo para toda a equipe de desenvolvimento, desde as versões de bibliotecas até versões de ferramentas de apoio. Nesse momento, existe um problema para a equipe de infraestrutura resolver, pois, certamente, haverá sistemas operacionais diferentes em suas variadas versões. Porém, para contornar esse problema, existem as máquinas virtuais e/ou *containers*.

Máquinas virtuais

Máquinas virtuais são ambientes computacionais simulados em arquivos executados dentro de outro ambiente computacional. Nelas, é possível criar recursos como processador, memória RAM e disco. Esse formato pode ser tanto um arquivo local tanto em nuvem. Para a infraestrutura de desenvolvimento, normalmente são ambientes criados e preparados localmente, com as ferramentas e dependências necessárias e depois disponibilizada uma cópia para os demais desenvolvedores.

Por simular um computador inteiro, com todos os recursos, incluindo o sistema operacional, o arquivo fica muito grande e de difícil transporte. As máquinas virtuais mais utilizadas são a VirtualBox e VMWare.

É bem simples instalar a VirtualBox, basta entrar no site do fabricante, escolher a última versão para o sistema operacional desejado e escolher o velho método: Avança, avança e finaliza. Depois, para criarmos um ambiente com um sistema operacional Linux, por exemplo, é só baixar a distribuição desejada. Em nosso exemplo, baixaremos o Ubuntu (<https://ubuntu.com/download/desktop>). O contrário também é possível, ou seja, utilizar uma máquina virtual Windows de dentro de um sistema operacional Linux.

Após executar a instalação, abra o VirtualBox e escolha a opção do Menu, Máquinas, Novo. Vamos configurar, na primeira tela, o sistema operacional que iremos virtualizar. Neste caso, escolhermos o Ubuntu.

Lembrando que a escolha do sistema operacional deverá ser organizada entre a equipe de desenvolvedores por questões de preferências e disponibilidades de ferramentas.

Figura 36. Configuração do SO.

← Criar Máquina Virtual

Nome e Sistema Operacional

Escolha um nome descritivo para a nova máquina virtual e selecione o tipo de sistema operacional que você pretende instalar nela. O nome que você escolher será utilizado pelo VirtualBox para identificar esta máquina.

Nome:

Pasta da Máquina:

Tipo:

Versão:

Fonte: Elaboração própria do autor.

Em seguida, escolheremos o tamanho da memória RAM.

Figura 37. Escolha da memória RAM.

Tamanho da memória

Selecione a quantidade de memória (RAM) em megabytes que será alocado para a máquina virtual.

O tamanho recomendado para memória é de **1024MB**.

4 MB 2048 MB 8192 MB

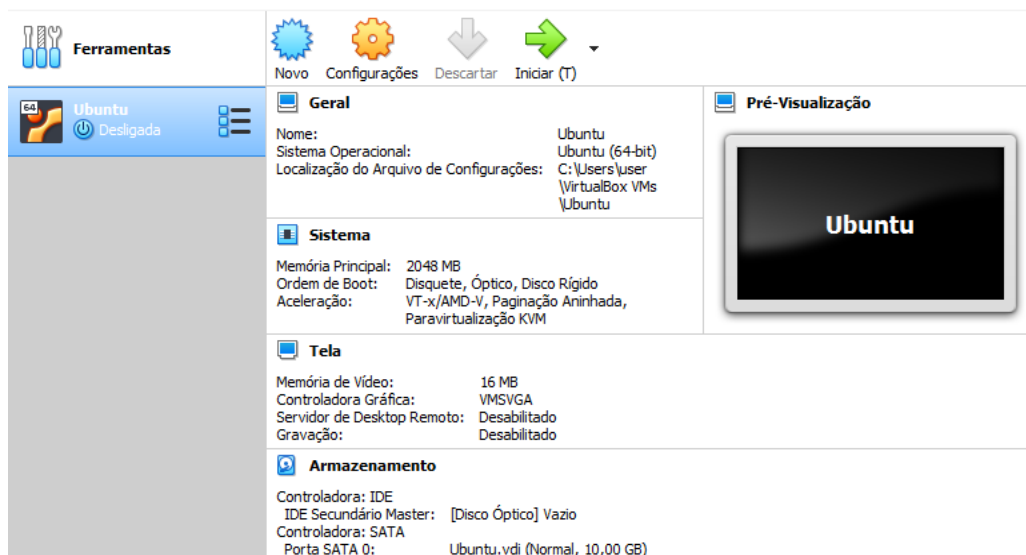
Fonte: Elaboração própria do autor.

Para a criação do disco, escolha as opções:

- » Criar um disco virtual agora e próximo;
- » VDI – *Virtual disk image* e próximo;
- » Dinamicamente alocado e próximo;
- » Tamanho do arquivo de disco. Deixe como está e próximo;

Depois de instalado, aparecerá a seguinte tela:

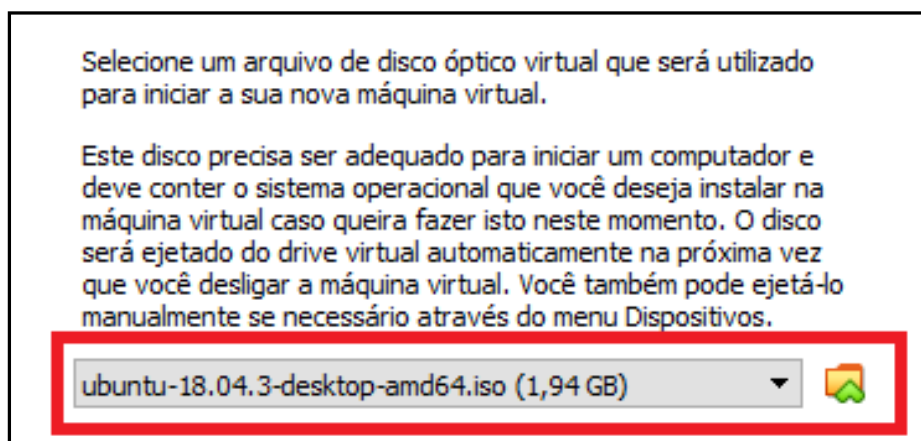
Figura 38. Instalação pronta do VirtualBox.



Fonte: Elaboração própria do autor.

Clique em iniciar para escolher a imagem baixada do Ubuntu:

Figura 39. Imagem selecionada.



Fonte: Elaboração própria do autor.

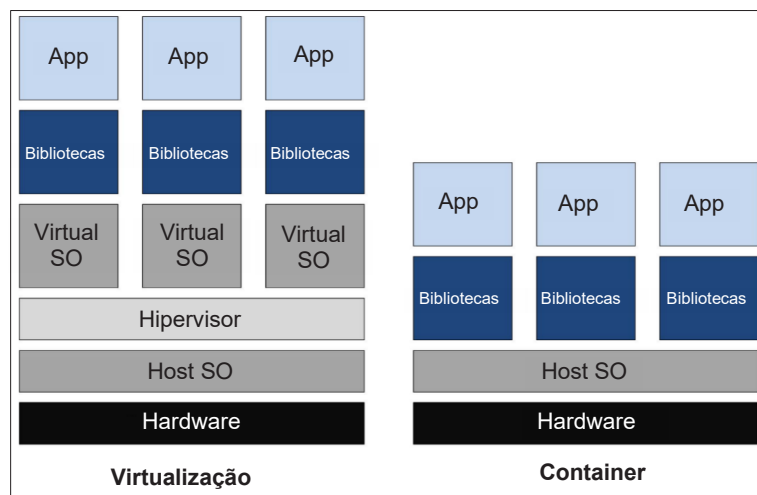
Na sequência, é só prosseguir com a instalação padrão do Ubuntu e instalar as principais ferramentas. Para distribuir aos demais desenvolvedores, depois de instalado, basta gerar um arquivo de transporte pelo menu Arquivo, Exportar *Appliance*.

Containers

Na virtualização, nós temos um ambiente inteiro isolado dentro de um único arquivo. Nos *containers*, são utilizados recursos e bibliotecas de kernel compartilhados, ou seja, dentro de um ambiente podemos executar aplicativos como web Server, banco de dados, em diversas versões, porém sua instalação ficará dentro de um *container*, compartilhando os recursos principais do computador, como memória RAM, processador e sistema operacional.

Container é o nome dado para a segregação de processos no mesmo kernel, de forma que o processo seja isolado o máximo possível de todo o resto do ambiente. Em termos práticos, são File Systems, criados a partir de uma “imagem”, e que podem possuir também algumas características próprias.

Figura 40. Arquiteturas de máquinas virtuais e *containers*.



Fonte: <https://www.mundodocker.com.br/wp-content/uploads/2015/06/lxc-vm.jpg>.

O *container* mais utilizado é o Docker. É uma ferramenta que se apoia em recursos existentes no kernel, inicialmente Linux, para isolar a execução de processos. As ferramentas que o Docker traz são basicamente uma camada de administração de *containers*, baseada originalmente no LXC. Alguns isolamentos possíveis:

- » Limites de uso de memória;
- » Limites de uso de CPU;

- » Limites de uso de I/O;
- » Limites de uso de rede;
- » Isolamento da rede (que redes e portas são acessíveis);
- » Isolamento do file system;
- » Permissões e Políticas;
- » Capacidades do kernel.

Containers Docker empacotam componentes de *software* em um sistema de arquivos completo, que contém o necessário para a execução: código, *runtime*, ferramentas de sistema – qualquer coisa que possa ser instalada em um servidor. Isto garante que o *software* sempre irá executar da mesma forma, independentemente do seu ambiente.

Podemos ainda encontrar imagens prontas que facilitam o trabalho. Uma imagem Docker é a materialização de um modelo de um sistema de arquivos, modelo este produzido por um processo chamado *build*. Essa imagem é representada por um ou mais arquivos e pode ser armazenada em um repositório. Existem várias imagens prontas em <https://hub.docker.com>.

Ambiente de produção

O ambiente de produção é o destino de entrega de uma aplicação. Nele, deverá ser instalado todo recurso necessário para rodar a aplicação, porém sem as ferramentas de desenvolvimento. Normalmente são instalados os servidores, bibliotecas e dependências. O ambiente de produção poderá ser instalado em servidores locais ou em nuvens.

Ao longo do tempo, a necessidade de se criar ambientes de infraestrutura mais ágeis, com custo reduzido e disponibilidade em tempo integral, fez com que empresas como a Amazon, Google, *MicroSoft*, *IBM* e inúmeras espalhadas pelo mundo, criassem ambientes para que a disponibilização de aplicações fosse mais rápida possível.

Não bastasse a agilidade na criação de uma estrutura computacional completa, surgiram empresas, como a *Heroku*, que já disponibilizam *containers* prontos, com todas as ferramentas necessárias para fazer os *deploys* e atualização de dependências.

A prática de DevOps não está relacionada somente à utilização de ferramentas. As ferramentas devem garantir que as etapas de DevOps sejam cumpridas. Com isso, poderá haver mais de uma ferramenta no mercado, efetuando as mesmas funcionalidades, agradando os mais variados tipos de clientes e necessidades.

Também é muito comum existir empresas dispostas a pesquisar, testar e listar tendências de ferramentas que podem ser utilizadas na prática do DevOps. Segue uma lista de ferramentas DevOps para 2019, feita pela *Raugun*: <https://raygun.com/blog/best-devops-tools/>.

Gradle

Apache Ant e Maven dominaram o mercado de ferramentas automatizadas de construção por muitos anos, mas Gradle apareceu em cena e sua popularidade tem crescido constantemente desde então. A melhor coisa sobre Gradle são *builds* incrementais, já que eles economizam uma boa quantidade de tempo de compilação. De acordo com as medições de desempenho, Gradle é até 100 vezes mais rápido que o Maven.

Git

É uma das ferramentas mais populares de DevOps, amplamente utilizada em toda a indústria de *software*. O Git permite-lhe acompanhar o progresso do seu trabalho de desenvolvimento. Você pode salvar versões diferentes do seu código-fonte e retornar a uma versão anterior quando necessário.

Jenkins

É a ferramenta de automação DevOps para muitas equipes de desenvolvimento de *software*. É um servidor de código aberto que permite automatizar os diferentes estágios do *pipeline* de entrega. A principal razão para a popularidade do Jenkins é o seu enorme ecossistema de *plugins*. Atualmente, oferece mais de 1.000 *plugins*, portanto integra-se a quase todas as ferramentas de DevOps, do *Docker* ao *Puppet*. Com o Jenkins, você pode configurar e personalizar seu *pipeline* de CI/CD de acordo com suas próprias necessidades.

Bamboo

É a solução de servidor CI/CD da Atlassian que possui muitos recursos semelhantes ao Jenkins. Ambas são ferramentas populares de DevOps que

permitem automatizar seu *pipeline* de entrega, desde as construções até a implantação. No entanto, enquanto Jenkins é *open source*, o Bamboo é pago.

Docker

O Docker tornou o uso de *containers* popular no mundo da tecnologia, principalmente porque possibilita o desenvolvimento distribuído e automatiza a implantação de seus aplicativos. Ele isola os aplicativos em *containers* separados, tornando-os portáteis e mais seguros. Os aplicativos do Docker também são independentes do sistema operacional e da plataforma. Você pode usar *containers* do Docker em vez de máquinas virtuais, como o VirtualBox.

Kubernetes

É uma plataforma de orquestração de *containers* que leva a utilização de *containers* para o próximo nível. Funciona bem com o Docker. Com o Kubernetes, você não precisa amarrar seus aplicativos em *containers* em uma única máquina. Em vez disso, você pode implantá-lo em um *cluster* de computadores. O Kubernetes automatiza a distribuição e o agendamento de *containers* em todo o *cluster*.

Puppet

É uma plataforma de gerenciamento de configuração entre plataformas. Ele permite que você gerencie sua infraestrutura como código. Ao automatizar o gerenciamento da infraestrutura, você pode entregar o *software* com mais rapidez e segurança.

Nagios

É uma das ferramentas mais populares de monitoramento de DevOps gratuitas e de código aberto. Ele permite que você monitore sua infraestrutura para encontrar e corrigir problemas. Com o Nagios, você pode manter registros de eventos, interrupções e falhas.

Raygun

A ferramenta DevOps da Raygun ajuda você a diagnosticar problemas de desempenho e rastreá-los de volta à linha exata de código, função ou chamada de API. A ferramenta APM também se encaixa bem com o fluxo de trabalho de gerenciamento de erros da Raygun.

CAPÍTULO 2

Pipelines

No DevOps, as organizações trabalham de maneira mais ágil para acompanhar os requisitos de negócios em constante mudança. Organizar *pipelines* de integração, *pipelines* de entrega e de implantação contínua é uma tarefa fundamental, pois seu sucesso garante os princípios básicos do DevOps.

Pipelines são sequências planejadas para atingir determinado objetivo. Assim sendo, antes de se criar qualquer *pipeline*, é necessário um planejamento adequado. Esse planejamento só será possível se a organização estiver em pleno DevOps, pois é necessária total colaboração de ambos os times.

Pipeline de integração contínua

Atualmente, os desenvolvedores possuem mecanismos que produzem códigos muito rápidos, seja digitando, seja planejando. As metodologias ágeis realmente conseguiram organizar as equipes em plena produção. Aliada a essa produção está a possibilidade de se trabalhar em locais fisicamente separados, aumentando muito a quantidade de membros em um mesmo projeto. Imaginem agora todos esses desenvolvedores “juntando” seus códigos, várias vezes em um dia, em um produto que tem que funcionar adequadamente. Pois é essa a missão da integração contínua.

Integração contínua (CI) é um processo contínuo de combinar código-fonte de diferentes fontes (tipicamente de desenvolvedores ou equipe) em um único aplicativo e, então, executar um conjunto de tarefas automatizadas de teste no aplicativo resultante.

Segundo Azeri (2019), ao praticar a integração contínua, os desenvolvedores integram seu código em uma ramificação principal de um repositório comum. Em vez de criar recursos isoladamente e enviar cada um deles no final do ciclo, um desenvolvedor pode contribuir com seus códigos para o repositório, várias vezes em qualquer dia. A ideia principal do CI é reduzir os custos de integração fazendo com que os desenvolvedores façam isso com mais frequência e mais cedo do que o normal.

Será nesse momento que o desenvolvedor descobrirá, na prática, conflitos entre o código novo e o existente. Apoiado em um dos princípios do DevOps, que é descobrir erros o quanto antes, as correções dos conflitos serão mais fáceis e mais dispendiosas de se realizar. O atrito durante as tarefas de integração dependerá dos testes e de suas automatizações.

Pipeline de entrega contínua

A sequência, logo após a integração contínua, é a entrega contínua, pois o processo de entrega de *software* é automatizado ainda mais para permitir implementações fáceis e confiáveis na produção a qualquer momento. Um processo de entrega contínua, bem elaborado e com alta maturidade, sempre terá uma base de código implantável.

Com a entrega contínua, a liberação de *software* se torna uma rotina suave e sem nenhuma surpresa desagradável que coloque em risco o produto final, deixando as equipes mais tranquilas com as tarefas de desenvolvimento, melhorias e aprendizados diários, confiantes que poderão criar uma versão pronta para a produção e que possa ser implantada a qualquer momento.

Segundo Kim *et al.* (2018, p. 111), “para criar o *pipeline* de entrega, é necessário pipeline de implantação central no qual a equipe automatiza os processos de teste e implantação. Esse *pipeline* poderá ser criado em uma ferramenta que executa um conjunto progressivo de sequências de testes.”

Em cada segmento do *pipeline*, poderá haver um teste crítico e, nesse caso, o *pipeline* comunicará a equipe. Se não houver nenhum problema, o processo continuará no próximo conjunto de testes, com testes sucessivos, resultando em promoção automática para o próximo segmento no *pipeline*. O último segmento no *pipeline* implantará a construção em um ambiente equivalente de produção. O resultado é uma construção confiável, implantável e verificável em um ambiente de produção real.

Pipeline de implantação contínua

Seguindo para a sequência final, a implantação contínua estende a entrega contínua para que o *software* implante automaticamente após passar em todos os testes, não havendo a necessidade de uma pessoa decidir quando entrará em produção. Claro, houve uma pessoa para configurar quando esse *software* entraria em produção.

A última etapa em um sistema com implantação contínua implantará automaticamente todos os componentes e pacotes de construção que forem bem-sucedidos na saída do *pipeline* de entrega. Essas implantações automáticas podem ser configuradas para distribuir rapidamente componentes, recursos e correções para os clientes e fornecer clareza sobre o que foi empurrado para a produção (AZERI, 2019).

As organizações DevOps se beneficiam da implantação contínua, pois a capacidade dos usuários fornecerem um *feedback* é muito mais rápida sob as novas implantações. Isso vale para os recursos que, por algum motivo, se tornaram obsoletos ou desnecessários depois da análise de requisitos.

No entanto, como os recursos estão sendo entregues rapidamente aos usuários, qualquer defeito que se torne evidente deve ser tratado imediatamente ou a equipe corre o risco de ficar sobrecarregada com a tentativa de corrigir os erros mais recentes e liberar novos recursos (AZERI, 2019).

Estudo de casos

No primeiro capítulo, foram citadas algumas empresas que oferecem serviços que auxiliam na cultura DevOps, e uma delas é a Amazon, que possui em seu portfólio várias ferramentas. Uma delas é a AWS Code Pipeline. Como a própria descrição no site, visualizado através do link <https://aws.amazon.com/pt/codepipeline/>. Acessado em 8/11/2019, diz:

O AWS CodePipeline é um serviço gerenciado de **entrega contínua** que ajuda a automatizar pipelines de liberação para oferecer atualizações rápidas e confiáveis de aplicativos e infraestruturas. O CodePipeline automatiza as fases de compilação, teste e implantação do processo de liberação sempre que ocorre uma mudança no código, de acordo com o modelo de liberação que você definiu.

Figura 41. AWS CodePipeline.



Fonte: https://d1.awsstatic.com/Products/product-name/diagrams/product-page-diagram_CodePipeline.7b8dd19eb6478b7f6f747d936c2f0b0b66757bbf.png.

No site do fabricante, podemos contar com dois depoimentos para termos a ideia do resultado da implementação de uma ferramenta como essa.

O primeiro caso é da empresa Lululemon Athletica, que atua no *e-commerce* e utilizou uma ferramenta de entrega contínua. O depoimento completo pode ser acessado através do link <https://aws.amazon.com/pt/solutions/case-studies/lululemon-athletica>. Acesso em 8/11/2019:

A Lululemon Athletica pode criar ambientes de desenvolvimento em minutos em vez de dias, automatizar seu ambiente e permitir integração e implementação contínuas com a AWS. A empresa canadense vende roupas inspiradas no yoga e outras roupas em mais de 350 locais em todo o mundo. A empresa executa os seus ambientes de desenvolvimento e teste, bem como um aplicativo móvel futuro, na Nuvem AWS.

O segundo caso foi da empresa A 3M Health Information Systems (3M HIS), com sede Salt Lake City, Utah, podendo ser acessado pelo link <https://aws.amazon.com/pt/solutions/case-studies/3M-health-information-systems/>, acessado em 8/11/2019. É uma das maiores provedoras mundiais de *software* para o setor de saúde: “O processo de provisionamento de servidor levava 10 semanas ou mais, mas foi reduzido a minutos. Isso nos dá a flexibilidade para oferecer suporte para os eventos de escalabilidade que vivenciamos várias vezes por dia”.

Além das arquiteturas de *hardware* e *data centers*, existem muitas ferramentas que podem ser utilizadas para esse propósito:

Pipeline de monitoramento

Nos processos ligados às boas práticas de DevOps, vimos a importância dos *pipelines* de integração, entrega e implantação. Entretanto, ao passarmos de um estágio para outro, deixamos, propositadamente, para falar do processo mais importante que, sem ele, não teríamos os avisos necessários: *pipeline* de monitoramento.

O processo de monitoramento é bem mais antigo que o próprio DevOps, porém sua utilização evoluiu pegando carona na onda DevOps.

Na integração, entrega e implantação contínua, o monitoramento possui um papel fundamental, que é o de fazer o fluxo de valor chegar até o final do ciclo. Qualquer

mensagem deve ser configurada para avisar, de alguma forma, os membros das equipes, sejam de desenvolvimento ou operações.

Outra situação que deve ser monitorada e que não foi relacionada é a segurança. Seja sobre vulnerabilidades na infraestrutura, seja nas lógicas de programação, a segurança deverá estar com os alertas bem configurados.

Alguns exemplos de alertas que podem ser emitidos pelo monitoramento, podem estar relacionados tanto para o ambiente de desenvolvimento, testes ou produção. Entretanto o excesso de informações também poderá ser ruim, pois os usuários poderão simplesmente ignorar, caso recebam muitos alertas.

CAPÍTULO 3

Mudanças

Certos serviços de TI precisam que os alicerces da Segurança da Informação tenham níveis altos de confidencialidade, integridade e disponibilidade, CIA (confidentiality, integrity, availability). Tais serviços são sofrem muitas alterações e geralmente precisam de uma abordagem muito formal para gerenciamento de mudanças, a fim de ajudar a garantir que cumpram os seus requisitos. Eles são frequentemente sujeitos a requisitos legais ou regulamentares, de modo que todas as mudanças precisam ser totalmente aditáveis.

Entretanto outros serviços de TI podem precisar de níveis mais baixos de CIA, porém um número maior de agilidade, para resolver rapidamente a evolução das necessidades e expectativas do cliente. Esses serviços muitas vezes são diretamente visíveis aos clientes finais.

Processo de aprovação de mudanças

Segundo Kim *et al.* (2018, p. 333), “quase toda organização de TI, de qualquer tamanho significativo, terá processos de gestão de mudança, que são principalmente controles para reduzir riscos de operações e segurança. O gerente de conformidade e os gerentes de seguranças dependem dos processos de gestão da mudança para requisitos de conformidade e normalmente exigem evidências de que todas as alterações foram adequadamente autorizadas.”

Se os *pipelines* de implementação forem construídos corretamente, sem falhas e com risco baixo para que as implementações sejam efetuadas de modo seguro, essas alterações não precisarão passar por processos morosos de aprovações de mudanças manuais, pois há confiança em controles de testes automatizados.

Essa é uma etapa para fazer o que for preciso para garantir que possamos integrar segurança e conformidade em todo o processo do gerenciamento de mudanças existente.

Segundo Kim *et al.* (2018, p. 335), políticas eficazes de gestão da mudança reconhecerão que existem diferentes riscos associados a diferentes tipos de

alterações e que todas essas alterações são tratadas de formas diferentes. Esses processos estão definidos no ITIL, que dividem as alterações em três categorias:”

- » Alteração padrão: são mudanças de risco menor, que seguem um processo estabelecido e aprovado, também podendo ser pré-aprovadas.
- » Alterações normais: mudanças de riscos mais altos, que exigem revisão e aprovação da autoridade que concordou com elas.
- » Alterações urgentes: são mudanças de emergência, e como consequência, potencialmente de alto risco, que devem ser colocadas na produção imediatamente. Um objetivo importante das práticas de DevOps é otimizar o processo de alteração normal, de modo também conveniente para alterações de emergências.

Reclassificação de alterações padrão para mudanças de baixo risco

Idealmente, em um *pipeline* de implantação confiável em vigor, já teremos a reputação de implementação rápida, confiável e não dramática. Nesse ponto, devemos buscar concordância de operações e das autoridades de mudança relevantes de que nossas alterações mostraram ser de risco baixo o suficiente para serem definidas como alterações padrão, pré-aprovadas. Isso nos permite implementar na produção, sem necessidade de mais aprovação, embora as mudanças ainda devam ser corretamente registradas.

Não é apropriado gerenciar mudanças para todos esses serviços da mesma maneira. Você deve criar um conjunto de categorias bem definidas e atribuir cada serviço a uma categoria. Por exemplo em seu artigo, que pode ser acessado pelo link <https://www.gartner.com/en/information-technology/role/applications-leaders>. Acesso em 8/22/201, a Gartner definiu três categorias comuns de sistemas de TI.

- » Sistemas de registro: apoiam dados críticos de uma organização e transações. Eles normalmente têm uma baixa taxa de mudança e podem estar sujeitos a exigências regulatórias.
- » Sistemas de diferenciação: empresa única de apoio ou indústria de capacidades. Eles precisam de reconfigurações bastante frequentes para atender crescentes necessidades de negócios ou clientes.
- » Sistemas de Inovação: são muitas vezes construídos ou modificados numa base *ad hoc* para atender aos novos requisitos ou oportunidades.

O que fazer quando as mudanças são classificadas como alterações normais

As mudanças que não podem ser classificadas como alteração padrão serão consideradas alterações normais e exigirão aprovação de pelo menos um subconjunto do CAB, antes da implementação. Nesse caso, nosso objetivo ainda é garantir que possamos implementar rapidamente, mesmo que não seja de forma totalmente automatizada.

Reduzir a dependência da separação de tarefas

Por décadas, usamos separação de tarefas como um dos principais controles para reduzir o risco de fraude ou erros no processo de desenvolvimento de *software*. Essa tem sido a prática, na maioria dos SDLCs, para exigir que alterações de desenvolvimento sejam submetidas a um bibliotecário de códigos, o qual examina e aprova a alteração, antes que as operações de TI a promovam para a produção.

Garantir documentação e prova para auditores de conformidades

À medida que as organizações tecnológicas adotam cada vez mais os padrões de DevOps, há mais tensão do que nunca entre o TI e auditoria. Esses novos padrões de DevOps desafiam o pensamento tradicional sobre auditoria, controles e mitigação de riscos.

Minimizar o número de aprovadores necessário para cada mudança

“Há um velho ditado: se todo mundo é responsável por alguma coisa, então ninguém é responsável. Isso vale principalmente para a aprovação das solicitações de mudança. Eu vi organizações nas quais existem 20 ou mais pessoas em uma reunião CAB, e qualquer pessoa pode rejeitar uma alteração, mas ninguém está autorizado a aprová-lo. Quando a mudança falhar, ninguém assumirá a responsabilidade para entender por que a decisão errada foi tomada a fim de melhorar as coisas para o futuro. Seria muito melhor nesses casos envolver muito menos pessoas na tomada de decisão” (KIM *et al.*, p. 345).

Existem situações em que cada solicitação de mudança tem exatamente um aprovador. Informações fornecidas no pedido de mudança canalizam automaticamente o pedido para o aprovador de mudança apropriado, que era então responsável por consultar outras pessoas, conforme necessidade para garantir que fizeram uma boa decisão.

O efeito disso foi que as pessoas realmente se importavam com as mudanças aprovadas. Eles tomaram muito cuidado para rever as alterações e garantir que teriam sucesso antes de as aprovarem, mas sem a sobrecarga burocrática de uma reunião com 20 pessoas.

“Normalmente, o aprovador da mudança entraria em contato com uma gama de pessoas com experiências diferentes para pedir sua entrada antes de tomar uma decisão, para, então, assumir a responsabilidade pela decisão final. Outra vantagem dessa abordagem foi que permitiu que o aprovador da mudança consultasse uma ampla gama de partes interessadas do que participar de uma reunião CAB típica, ajudando a garantir que uma decisão equilibrada levou em conta as opiniões dos clientes e usuários, bem como de TI pessoal e de gestão” (KIM *et al.*, p. 346).

Estudo de caso: mudanças de estrutura automatizada na Salesforce

A Salesforce, em 2007, possuía mais de 59 mil clientes empresariais processando milhões de transações por dia. Nessa época, contudo, sua capacidade de desenvolver e lançar novas funcionalidades para seus clientes parecia paralisada. Em 2006, iriam fazer quatro *releases* importantes para os clientes, mas fizeram apenas um. Como resultado, os recursos entregues foram diminuindo; e os *releases* importantes, aumentando.

As coisas começaram a mudar quando tornaram a engenharia de qualidade uma tarefa de todos, independentemente de fazer parte do desenvolvimento, operações ou Infosec. Para isso, testes automatizados em todos os estágios foram injetados. Outro ponto fundamental foi a transformação das mudanças na infraestrutura em mudanças padrão, exigindo bem menos ou nenhuma aprovação do CAB.

“Com esse procedimento, eles não apenas integraram seus processos de DevOps no processo de gestão de mudanças, mas também deram motivação para alterar o processo de motivação” (KIM *et al.*, 2018, p. 340).

CAPÍTULO 4

Segurança da informação na integração, entrega e implantação

Quando a cultura DevOps chega ao InfoSec, as objeções começam a surgir, principalmente se for um silo fora do Desenvolvimento e Operações. Mesmo com essas objeções, o DevOps é a melhor maneira de integrar segurança da informação no trabalho do fluxo de valor tecnológico.

Segundo Aielo (2019), como parte do esforço do desenvolvimento de *software*, é necessário entender e resolver os riscos. A segurança, por sua importância, não pode ser simplesmente acrescentada ao sistema no final do processo de desenvolvimento. O DevOps fornece a estrutura necessária para que os sistemas sejam projetados e desenvolvidos levando a segurança em consideração desde o início do ciclo de vida, pois irá ajudar a solucionar muitos dos riscos de segurança referentes à criação de qualquer sistema tecnológico complexo.

Integrar segurança em demonstrações de iteração e desenvolvimento

Segundo Kim *et al.* (2019, p. 316), “o objetivo é ter equipes de recurso engajadas no InfoSec o mais cedo possível e não deixar para o final. A melhor maneira de fazer isso é convidá-las para a demonstração de produto ao final de cada intervalo de desenvolvimento.”

“Quando InfoSec faz parte da equipe, mesmo que tenha pouca informação, o *software* ganha contexto comercial de que precisa para tomar melhores decisões baseadas em riscos. Além de auxiliar as demais equipes e saber o que é necessário para atingir a conformidade”, concluem Kim *et al.* (2019, p. 316).

Integrar segurança no controle de defeitos

Integrar segurança nos repositórios de código-fonte













Um dos grandes propósitos de se garantir a segurança em repositório é validar as bibliotecas de códigos de terceiros. Existem também arquivos de configuração

que, por um descuido, poderão ser distribuídos sem qualquer prevenção. Algumas recomendações poderão ser estabelecidas para facilitar a criação de padrões de segurança nos ambientes.

Integrar segurança no *pipeline* de implementação

Segundo Kim *et al.* (2018, p. 319), “essa etapa dos testes de segurança será automatizada ao máximo possível, para que sejam executados com todos os outros pipelines de implementação. O objetivo é oferecer aos Dev e Ops *feedback* o mais rápido possível, quando seu trabalho oferecer potenciais quebras de segurança ao *software* e ao ambiente.”

Figura 42. Jenkins executando teste de segurança.

Jenkins					
S	T	Nome	Último êxodo	Última falha	Última duração
		Varredura da análise estática	7 dias 1h - #2	N/A	6,3s
		Verificação de vulnerabilidades			
		Download e teste de unidade	7 dias 1h - #2	N/A	32s
		Varredura com OWASP ZAP	7 dias 1h - #2	N/A	4min3s
		Início	7 dias 1h - #2	N/A	5min3s
		Varredura de Vírus	7 dias 1h - #2	N/A	4min3s

Fonte: Kim *et al.* (2018, p. 320).

Proteger o aplicativo

Normalmente, os testes do time de desenvolvimento focam na funcionalidade da lógica correta dos aplicativos, denominada muitas vezes de “caminho feliz”. Já o pessoal de QA e InfoSec focam no “caminho triste”, ou seja, quando tudo dá errado, principalmente as funcionalidades relacionadas à segurança.

Alguns testes, citados por Kim *et al.* (2018, p. 321) foram relacionados:

- » **Análise estática:** é realizada em ambiente que não é tempo de execução, normalmente em *pipeline* de implementação.
- » **Análise dinâmica:** é o contrário da análise estática, ou seja, são testes feitos em um programa em operações.

- » **Varredura de dependências:** teste estático, feitos dentro de um *pipeline* de implementação que envolve inventariar todas as dependências de binários e executáveis.
- » **Integridade de código-fonte:** são efetivações assinadas e seus *hashs* gravados no serviço de registro centralizado.

Referências

AIELO, B.; SACHS, L. **Assegure a segurança de informações robusta e efetiva com o DevOps**. Disponível em: <<https://ibm.com/developerworks/br/library/d-robust-effective-information-security-devops/index.html>>. Acesso em: 2 ago. 2019.

AZERI, I. **What is CI/CD?** Disponível em: <<https://www.mabl.com/blog/what-is-cicd>>. Acesso em: 1 ago 2019.

BAYARD, V. **Amazon Andon Cord. BlueBoard**. Disponível em: <<https://blueboard.io/resources/wp-content/uploads/2019/03/Andon-Cord-FAQ-BlueBoard.pdf>>. Acesso em: 23 jul. 2019.

GAJDA, K. **Build a DevOps culture and team**. Disponível em: <https://www.ibm.com/cloud/garage/practices/culture/practice_building_culture/>. Acesso em: 1 jul. 2019.

GREMLIN. **Chaos Monkey Guide for Engineers**. Disponível em: <<https://res.cloudinary.com/gremlin/image/upload/v1558365189/ChaosMonkey-PDF.pdf>>. Acesso em: 21 jul. 2019.

GUCKENHEIMER, S. **Whats is Devops Culture?** 2017. Disponível em: <<https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops-culture>>. Acesso em: 20 jul. 2019.

HAND, J. **The top 10 DevOps myths debunked**. 2015. Disponível em: <<https://techbeacon.com/contributors/jason-hand>>. Acesso em: 10 ago 2019.

KAPADIA, M. **Comparing DevOps to traditional IT: Eight key differences**. 2015. Disponível em: <<https://devops.com/comparing-devops-traditional-eight-key-differences/>>. Acesso em: 10 ago. 2019.

KIM, G.; HUMBLE, J.; DEBOIS, P.; WILLIS, J. **DevOps – como obter agilidade, confiabilidade e segurança em organizações tecnológicas**. Rio de Janeiro: Alta Books: 2018.

MORGAN, J. **DevOps vs Traditional IT**. 2016. Disponível em: <<https://www.missioncloud.com/blog/devops-vs-traditional-it/>>. Acesso em: 10 ago 2019.

PEREIRA, C. A. S. **Lean Manufacturing**: Aplicação do conceito a células de trabalho. Covilhã, outubro de 2010.

REHKOPF, M. **What is a kanban board?** Disponível em: <<https://www.atlassian.com/agile/kanban/boards>>. Acesso em: 1º jul. 2019.

SKELTON, M. **What team structure is right for devops to flourish?** 2013. Disponível em: <<https://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/>>. Acesso em: 1 jun. 2019.

STALK, G.; HOUT, T. M. **Competing Against Time**: How Time-Based Competition is Reshaping Global Markets. New York: Free Press, 1990.

WERKEMA, M. C. C. **Lean Seis Sigma** – Introdução às Ferramentas do Lean Manufacturing. Belo Horizonte: Werkema Editora, 2006. 120p. Versão 4

WESTRUM, R. A typology of organisational cultures. **BMJ Quality & Safety**, v. 13, n. 2, 2004, p. 22-27. Disponível em: <https://qualitysafety.bmj.com/content/13/suppl_2/ii22.full>. Acesso em: 21 jul. /2019.

WOMACK, J. LEI - Lean Enterprise Intitute (2008). **Making things better through lean thinking and practice**: Muda, Mura, Muri. <https://www.lean.org/womack/DisplayObject.cfm?o=743>. Acesso em 8/11/2019.

Sites

<<https://aws.amazon.com/pt/devops/what-is-devops/>>.

<<https://www.digite.com/kanban/kanban-board/>>.

<[https://www.lean.org.br/conceitos/117/sistema-toyota-de-producao-\(toyota-production-system---tps\).aspx](https://www.lean.org.br/conceitos/117/sistema-toyota-de-producao-(toyota-production-system---tps).aspx)>.

<<https://devops.com/using-calms-to-assess-organizations-devops/>>.

<<https://gaea.com.br/conheca-a-incrivel-historia-do-devops/>>.

<<https://techbeacon.com/devops/top-10-devops-myths-debunked>>.

<<https://www.slideshare.net/ZeroTurnaround/traditional-it-ops-vs-dev-ops-devops-days-ignite-talk-by-oliver-white>>.

<<https://www.gartner.com/en/documents/2987231>>.

REFERÊNCIAS

<<https://www3.dbmaestro.com/blog/devops-myths-facts>>.

<<https://puppet.com/resources/whitepaper/2014-state-devops-report>>.

<<https://blog.estabil.is/calms-conceito-devops/>>.

<https://www.ibm.com/cloud/garage/practices/culture/practice_building_culture/>.

<<https://gaea.com.br/mainframe-e-devops-entenda-tudo-sobre-essa-relacao-aqui/>>.

<<https://www.lucidchart.com>>.

<<https://www.devopsdigest.com/devops-advantages-1>>.

<<https://www.smartdraw.com/value-stream-map/>>.

<<https://visual-paradigm.com>>.

<http://www.melconway.com/Home/Conways_Law.html>.

<<https://blueboard.io/resources/wp-content/uploads/2019/03/Andon-Cord-FAQ-BlueBoard.pdf>>.

<<https://github.com/Netflix/chaosmonkey>>.

<<https://medium.com/netflix-techblog/tagged/chaos-monkey>>.

<<https://res.cloudinary.com/gremlin/image/upload/v1558365189/ChaosMonkey-PDF.pdf>>.

<<https://github.com/dastergon/awesome-chaos-engineering>>.

<<https://caylent.com/devops-deming-andon-cord>>.

<<https://global.toyota/en/company/vision-and-philosophy/production-system/>>.

<<https://speakerdeck.com/garethr/battle-tested-code-without-the-battle>>.

<<https://www.slideshare.net/Couchbase/couchbase-at-linkedin-couchbase-connect-2014>>.

<<https://success.docker.com/article/dev-pipeline>>.

<https://support.monday.com/hc/article_attachments>.

<<https://netproject.com.br>>.

< <https://aws.amazon.com/pt/devops/what-is-devops>>.

<<https://www.atlassian.com/br/devops#!team-morale>>.

<<https://www.redhat.com/pt-br/topics/devops>>.

<<https://azure.microsoft.com/pt-br/overview/what-is-devops>>.

<<https://www.ibm.com/br-pt/cloud/devops>>.