Fundamentos DevOps

O DevOps como metodologia para uma melhor integração entre equipes de desenvolvedores e operadores.

Prof. Brunelli Gabriel Cupello

1. Itens iniciais

Propósito

Compreender o funcionamento da metodologia DevOps, como cultura na engenharia de software, é essencial ao profissional que quer implementar um mecanismo de eficiência do ciclo de planejamento, desenvolvimento e automação de um serviço, fazendo com que os desenvolvedores tenham entre si agilidade, integração e cooperação.

Objetivos

- Identificar as bases da cultura DevOps.
- Reconhecer o modelo da entrega contínua.
- Reconhecer os processos de entrega contínua, integração contínua e implantação contínua.
- Reconhecer o princípio das 3 maneiras.

Introdução

A primeira parte deste estudo é entender que o DevOps é um processo cíclico com etapas dependentes umas das outras, e também saber que a produção de um produto não é mais fordista (em que os trabalhadores não se comunicam e não têm ciência da parte/etapa do trabalho da outra pessoa), mas também é compartilhada de forma contínua, criando nas empresas microssistemas autônomos e comunicantes entre si.

É um dos princípios de engenharia de software que ajudam na entrega de softwares e nos guiam na **entrega contínua**, buscando manter a eficiência e qualidade, por meio da automatização e nos proporcionando uma redução nos gastos e confiabilidade. Podemos separar esses princípios em seis: responsabilidade, automatização, versionamento, qualidade, release e melhora contínua.

Continuous integration (integração contínua), *continuous delivery* (entrega contínua) e *continuous deployment* (implantação contínua) são práticas DevOps amplamente utilizadas durante os últimos anos. No entanto, ainda existe confusão sobre o que são essas práticas e suas diferenças e não há necessidade de complicar algo que é fácil. Todas são metodologias que facilitam a implantação de processos e ferramentas que possibilitam uma rápida entrega de versões de software. IC, CD e IC são como uma cadeia de evolução da biologia.

Tendo essa imagem na mente, integração contínua é o início de tudo, como se fosse o Big Bang, é o *head start* das metodologias. A mais importante é a que dá mais valor ao negócio. Daí ela evolui para algo mais mutável para atender o valor que a IC agrega para o produto, que é a entrega contínua. Dela, com o intuito de dar mais segurança, vem a implantação contínua. Falaremos de cada uma das metodologias, entenderemos como elas são estruturadas e como podemos utilizá-las para otimizar o desenvolvimento e a entrega do software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Conceituação

O que é DevOps?

Fundamentalmente, o DevOps preza o desenvolvimento objetivo de software para o que mais agrega valor ao negócio, isto é, manter a produção com um foco para o que os clientes e usuários utilizam constantemente e entendem como indispensável e com um ritmo constante.

Comentário

DevOps sugere validar de forma contínua o que está sendo produzido e planeja para que em nenhum momento do processo o ideal seja perdido. Tal prática agrega muito valor ao negócio, pois quanto mais recorrente for a validação, maiores serão a eficiência e eficácia do produto.

O que é cultura DevOps?

A cultura DevOps é um conjunto de ações cujos pilares são acelerar a entrega do produto, trazer transparência para os processos, dinamismo no desenvolvimento do produto, escalonamento de demandas, integração e entrega contínua (CI/CD) e a segurança do que é desenvolvido. Tais ações têm como base a comunicação efetiva intratimes e intertimes.

> É realmente importante que os funcionários da empresa aceitem, reconheçam e pratiquem o conjunto de ações e condutas que precisam ser incorporadas para que a estrutura do DevOps seja implementada na empresa.

As equipes precisam ser compostas de trabalhadores de diversas áreas do domínio de Tecnologia da Informação, visando ter uma melhor comunicação intra e intertimes na corporação.

Exemplo

Imagine uma equipe de tradutores composta por pessoas com diferentes formações em línguas antigas. Quando aparece algum trabalho de uma língua morta para traduzir, eles têm uma chance maior de conseguir resolver a demanda, da mesma forma que uma equipe multidisciplinar de TI conseguirá atender de forma mais eficaz e efetiva os problemas que surgirem durante o caminho.

Os pilares da cultura DevOps fazem muito sentido quando destrinchamos a importância de cada um deles. Obvserve!

Pilares da cultura DevOps

A cultura DevOps se sustenta em alguns pilares. Entre eles, destacam-se:

Velocidade na entrega

É bem compreendida pela necessidade de desenvolver mudanças, adaptações e entregas com a agilidade indispensável para garantir o resultado, sem perder o dinamismo que é exigido pelo mercado para que as empresas e os colaboradores que demandam esse serviço convertam-se, a fim de atender e proporcionar a melhor experiência do produto para o cliente final.

Entrega rápida

Dialoga perfeitamente bem com a velocidade (nos códigos, nas interações, mudanças etc.). A necessidade de fazer o produto chegar mais rápido para o cliente gera satisfação e a empresa se mostra eficiente, tornando-se mais competitiva no mercado, tendo em vista que os clientes começam a visualizar vantagens em lançamentos mais rápidos e contínuos. Podemos resumir a diferença entre entrega rápida e velocidade na entrega da seguinte forma: uma entrega que levaria meses para ser feita, agora é dividida em várias pequenas entregas que resultam na mesma entrega final.

Escalonamento de tarefas do time

Necessidade de cada membro da equipe precisar assimilar muito bem seu papel para com a empresa, outros times da empresa e seu próprio time. Dessa forma, as reuniões mensais, semanais e até as diárias servem para acertar detalhes, ditar prioridades, alinhar expectativas de entrega, definir o papel que cada pessoa do time precisa exercer naquele momento ou talvez em longo prazo. Existem vários tipos de metodologias ágeis para aplicar à cultura DevOps, entre elas Kanban e Scrum são utilizadas pelos times. Precisa ser levada em consideração a maturidade do time e sua dinâmica para escolher da melhor forma as metodologias capazes de resolver os problemas da equipe e atender às regras de negócio.

Faz parte da implementação da cultura DevOps o método de integração contínua. A cultura DevOps permite aos desenvolvedores desempenhar mudanças no código (de melhoria, correção ou apenas de rotina) com frequência e com uma maior certeza de não ocorrência de erros nas atualizações. A escolha de um software de integração contínua/entrega contínua (continuous integration/continuous delivery) é de grande importância para o tempo de vida do sistema de um pipeline de CI/CD. É parte desde o projeto até o gerenciamento e a implementação, permitindo entregar os produtos e serviços para o cliente final de forma ininterrupta e mais ágil.

Saiba mais

Como atualmente nem todos os aspectos do ciclo DevOps são automatizados, existe a possibilidade de se subir algum código contendo erros ou durante o seu ciclo de utilização podem aparecer bugs, pois estamos lidando, também, com seres humanos.

Pensando nisso, existe outro pilar, que é o da **segurança e transparência**, em que todos do time que estão envolvidos no processo de desenvolver software em determinado produto possuem uma responsabilidade compartilhada, logo, se houver alguma intercorrência, a resolução da questão torna-se muito mais rápida e cirúrgica, sem deixar que o problema cresça e complique outras partes do código.

Implementação da cultura DevOps

> Como e por que implantar a cultura DevOps nas empresas?

Quando a empresa não pratica os pilares da cultura DevOps, podem ocorrer resistências na sua implementação, o que é bastante normal. Tendo isso em vista, existe um esquema que valoriza os principais pontos e tornaria mais fácil a implementação da cultura DevOps nas empresas:

- entender o que é a filosofia DevOps;
- planejar a adoção da cultura DevOps;
- facilitar a familiarização dos membros dos times com a cultura DevOps;
- fornecer explicação detalhada das práticas DevOps, dos processos de união entre e intratimes;
- fornecer treinamentos iniciais, durante o processo e depois de implementado para manter a cultura DevOps sempre no padrão de excelência.

Com esses pontos atendidos, torna-se possível a implementação da cultura DevOps para uma real mudança na forma de pensar e agir, bem como na padronização dos processos internos, fornecendo métricas para que as lideranças da empresa tenham a possibilidade de avaliar se o método está sendo eficiente.

Saiba mais

É muito raro uma empresa aplicar essa cultura entre os seus colaboradores e não quantificar resultados positivos, pois a cultura DevOps traz muitos benefícios e, depois de implementada e solidificada nos times, necessita apenas de um treinamento regular e métricas de avaliação constantes. Ressalta-se a necessidade de profissionais com uma boa qualificação da área de TI para que a adoção da cultura seja feita da maneira correta.

Influência e benefícios da cultura DevOps nas empresas

Usar a cultura DevOps estimula o trabalho em equipe e a colaboração, possibilitando solucionar possíveis intercorrências. Para ter uma boa aceitação pela equipe e empresa, a cultura DevOps deve seguir práticas de convivência que estão intimamente conectadas com os pilares que foram descritos anteriormente e, para isso, precisa manter constâncias em tais práticas.

Entre elas, evidencia-se a comunicação entre todos os envolvidos, automação, integração, colaboração, entregas contínuas, testes contínuos, monitoramento, correção e definição do tempo para todas as etapas de produção, das quais separamos por estudo, planejamento, desenvolvimento, testes, implantação, operação e monitoramento, para definir a melhor metodologia aplicável.

A cultura DevOps é uma cultura de automatização de processos de infra e desenvolvimento de software, que surgiu com a dificuldade da troca de informações pelos diferentes métodos aplicados em cada equipe de trabalho. Foi fundamental a criação de um novo método que unificasse todo o projeto visando acabar com as diferentes linguagens usadas pelos times, facilitando o processo e melhorando o produto (agregando mais valor ao mesmo). A própria palavra “DevOps” deixa tudo bem claro quando a separamos e temos as abreviações “Dev”, de *development*, que tem como objetivo o desenvolvimento de software, e “Ops”, de *operations*, com o objetivo de implementação dos softwares.

> Por que a cultura DevOps é benéfica?

Os benefícios que a cultura DevOps proporciona para uma empresa são a criação de uma infraestrutura sólida, que permite um desenvolvimento constante de todas as operações sem perda de tempo em diversas comunicações entre equipes, com metodologias de produção distintas e, ainda, segurança, transparência e corresponsabilidade se algumas das etapas anteriores falharem, estimulando, dessa forma, um eficaz e eficiente ambiente de trabalho cooperativo.

Podemos listar alguns dos principais benefícios:

- mais agilidade na comunicação do desenvolvimento e infraestrutura;
- economia de recursos;
- mais velocidade produção;
- melhor qualidade;
- redução de erros;
- soluções com melhor desempenho e estabilidade;
- melhor ambiente de trabalho.

A unificação de linguagem e de times permite a troca de informações entre equipes. A comunicação gera uma compreensão e um senso de cooperação, agilizando as interações entre setores sem a necessidade de perda de tempo na troca de metodologia de cada área. Com o ganho de tempo, podemos direcionar e melhorar a eficiência no desenvolvimento, reduzindo prazos e corrigindo erros com mais eficácia. Envolver mais pessoas no projeto reduz a possibilidade de aparecimento de erros. Essa otimização cria uma melhoria no desempenho e estabilidade para todos na empresa, proporciona alta moral dos envolvidos e, assim, o rendimento empresarial aumenta, pois o valor agregado do produto cresce com todas essas condutas.

A equipe de TI é composta por diferentes profissionais com *soft skill* (habilidades básicas com pessoas), que entregam um trabalho bastante equilibrado, uma força de protocooperação. Se não existir esse equilíbrio da boa relação e troca de informações, toda a cadeia da cultura DevOps pode ser comprometida, desde um projeto com erros de desenvolvimento até erros operacionais.

Em tese, parece fácil mudar todo um ciclo que se desenvolve de determinada forma, porém, os reais benefícios da implementação dessa cultura só farão jus a toda dificuldade de implantação se houver uma real aceitação dos times. A empresa precisa incorporar de forma intensa todos os pilares.

O que é DevOps

A seguir, confira os principais pontos sobre o assunto.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Soluções ágeis na cultura DevOps

Existem algumas formas que podem proporcionar eficiência na cultura DevOps. O uso de alguns métodos já consolidados no meio pode garantir soluções mais ágeis e de otimização com os métodos Scrum e Kanban, que objetivam alcançar melhor gerenciamento de projetos, reduzindo o tempo e o custo de produção.

Scrum

A seguir, assista ao vídeo com os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Scrum é uma metodologia de gerenciamento de projetos amplamente utilizada por equipes de desenvolvedores de softwares que demonstra um modelo **Agile** com uma abordagem concentrada em produção de versões contínuas, voltada ao interesse de melhorar o feedback, ao mesmo tempo que faz uma avaliação de mercado. Pode produzir **Product Backlog**, que é dividido em algumas partes, os **Sprints**, sem que causem problemas nas atividades subsequentes. Os Sprints possuem um ciclo de médio de tempo de produção de, aproximadamente, um mês.

Etapas da metodologia Scrum.

Os Sprints ainda são compostos por muitas subdivisões, chamadas de Sprints Backlog, compostas por tarefas específicas e possuem um ciclo curto, que dura alguns dias. Por isso, reuniões devem acontecer diariamente para acompanhar o desenvolvimento de cada equipe e se adaptar da melhor maneira possível, a fim de impedir o atraso de produção; definimos isso com Daily Scrum. No final de um ciclo de um Sprint, ocorre uma reunião periódica, o Sprint Review Meeting, com o intuito de apresentar o que foi efetuado e dar feedback do trabalho desse ciclo e concluir a etapa daquele Sprint.

Ao iniciar um novo ciclo de Sprints, é necessária uma reunião para definir as prioridades dos Sprints, o Sprint Planning Meeting, uma reunião para determinar melhor forma de planejamento, ordem de prioridade e tempo de produção do Product Backlog, colocando em prática todo os métodos de Scrum, sempre visando à agilidade do trabalho, o que demonstra uma forma muito flexível de se adequar à maneira única de cada projeto e de remanejar recursos, com os Daily Scrum. Pode-se perceber que o acompanhamento em cima do desenvolvimento do projeto é constante, então, essas reuniões precisam ser objetivas e rápidas.

Kanban

A seguir, assista ao vídeo com os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

É uma metodologia aplicada em um sistema virtual de gestão de trabalho, que procura direcionar serviços em um fluxograma de trabalho. É um sistema pautado para ordenar o workflow e torná-lo mais eficiente. Dividida em trabalhos por tipos e níveis de dificuldade, cada tarefa é posta em colunas com cartões para determinar sua condição, assim, tendo controle e limitando a quantidade de itens que podem ser atribuídos por cada etapa de produção.

Seu objetivo tem como fundamento o desenvolvimento de produtividade e organização das entregas, definindo uma produção constante e progressiva, tornando um fluxo de trabalho bem definido e de maior qualidade. Surge com um método de tornar os profissionais mais autônomos em organizar e definir seu fluxo de trabalho, determinando maior produtividade e mantendo constante qualidade de produção, com um ambiente de trabalho familiar a todos os envolvidos e dando a sensação de conforto.

Verificando o aprendizado

Questão 1

Quais premissas são verdadeiras com relação à cultura DevOps?

I. A cultura DevOps não precisa se estender por todo o ambiente corporativo, desde o desenvolvimento de software integrado pela equipe de desenvolvedores até a equipe de operadores.

II. A troca de informações e a transparência são fundamentais para reduzir a perda de tempo útil e ocioso, tais ações quando colocadas em prática possibilitam mudanças por toda a estrutura empresarial.

III. É imprescindível velocidade na entrega, que pode ser bem compreendida pela necessidade de performar mudanças, adaptações e entregas com a agilidade indispensável para garantir o resultado sem perder o dinamismo.

A

I.

B

II.

C

III.

D

II e III.

E

I e III.

A alternativa D está correta.

A cultura DevOps foi desenvolvida para reduzir a distância entre desenvolvedores e operadores, e assim tornar o ambiente de trabalho mais transparente e agilizar a troca de informações, por isso, torna-se imprescindível a velocidade em todos os processos sem perda de qualidade.

Questão 2

Quais duas características podem ser encontradas em um ambiente de trabalho que utiliza a cultura DevOps?

A

Equipes descentralizadas e muita troca de comunicação.

B

Equipe unificada e pouca troca de informações.

C

Método ágil e longos períodos entre atualizações.

D

Equipe unificada e muita troca de informações.

E

Método ágil e baixa comunicação.

A alternativa D está correta.

A cultura DevOps proporciona a unificação de equipes, causando a elevada troca de informações e melhoria na comunicação, tornando-se um método ágil e com contínuas atualizações em seu desenvolvimento.

A entrega contínua

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A entrega contínua consiste em uma prática de desenvolvimento de software baseada na automação. Esse processo permite que os desenvolvedores possam, por exemplo, aplicar alterações que serão enviadas a um repositório de código automaticamente, depois de serem testados, eliminando os resultados falhos e garantindo a confiabilidade do processo.

Por isso, é de extrema importância que se tenha responsabilidade com o que é criado em relação ao código ou produto, pois essa criação será implantada na produção.

É preciso que haja uma clara visão da produção, de seu propósito, do atendimento da demanda. Além da construção bem embasada de um código, é necessário deixar possibilidade de mudança para atender demandas **just in time**.

> Afinal, o que é o modelo just in time?

O modelo de gestão JIT propõe que a produção seja feita de acordo com a demanda, então, a empresa vai produzir naquele momento somente os produtos necessários, nas quantidades necessárias. E esse processo se repete em cada etapa da produção.

Como todo tipo de modelo, a entrega contínua possui regras ou princípios para seu bom funcionamento. Veremos a seguir cada um deles:

Responsabilidade

Precisa-se ter responsabilidade com o que é criado. Todo tipo de software é criado para uma determinada finalidade, que em grande parte é agregar valor a algo ou "alguém", e esse alguém é o usuário. Todas as escolhas precisam ser feitas com essa ideia em mente. Antes de entender as regras de negócio de um sistema, antes de escolher as funcionalidades de um sistema, escolher as plataformas de hardware ou os processos de desenvolvimento, é de grande importância questionar:

> “Isso que eu estou fazendo contribuirá com algum significado ao sistema/produto?". Se a resposta for não, não faça! E basicamente, todos os outros se conectam nele.

KISS

Termo abreviado de “***Keep It Simple, Stupid!***”, que tem o sentido de “Deixe isso simples, não complique!”. Esse termo se encaixa de forma sistemática na montagem de um software. Muitos fatores precisam ser levados em consideração antes de decidir qualquer coisa quando trabalhamos em um projeto. Toda arquitetura deve ser a mais simples possível, porém, não deve ser reducionista.

Comentário

O sistema precisa ser facilmente compreendido e mantido. Isso não significa que ele não deve ser arrojado e elegante. Desenvolvedores têm a tendência de cometer o erro nesse princípio, pois confundem simplicidade com “gambiarra”; os projetos mais simples tendem a ser os mais bonitos. Na maioria das vezes, é preciso fazer várias pequenas ligações sofisticadas para se obter uma versão visual e funcionalmente mais simples, tornando o software mais fácil para manutenção e correção dos erros.

Mantenha a visão no que é essencial

O sucesso depende de uma visão clara. Sem foco no objetivo, o projeto torna-se redundante. À falta de uma integridade de conteúdo, todo software corre o risco de se transformar em um quebra-cabeça mal conectado, com peças erradas e incompletas, como se os projetos fossem incompatíveis, unidos por pontes inadequadas. Toda a falta de conectividade compromete a visão de arquitetura do sistema de software podendo até destruir estruturas corretas. Ter um profissional responsável por organizar as conexões da arquitetura ajuda a reforçar a segurança e o êxito do projeto.

Sempre produzir para atender necessidades

É raro um sistema de qualidade industrial ser construído para ser utilizado sem que se tenha integração. A maior parte do tempo não vai haver apenas um programador no projeto. Com esse fato em mente, sempre especifique como, implemente especificando o porquê e projete levando em conta que outra pessoa além de você irá mexer nesse código.

Saiba mais

Existe muito mercado para qualquer projeto que será feito, logo, é necessário que o produto seja bem explicado e seu objetivo fique bem claro para os usuários. Basicamente, projete se preocupando com quem no futuro irá cuidar (ampliar, modificar, melhorar, consertar etc.) do seu código. Facilitando o trabalho de todas as pessoas que usam seu código, você agrega valor ao seu projeto.

Produzir algo durável e adaptável a mudanças

Quando compramos uma geladeira, que é um eletrodoméstico amplamente usado no cotidiano, procuramos uma marca boa, durável, com bastantes peças para manutenção e uma boa rede autorizada. Tudo isso agrega um valor muito alto no produto por causa de todas essas “seguranças”.

> O sistema deve ter as mesmas “qualidades” da geladeira, ele deve ser bom, durável e adaptável às mudanças que são necessárias no ambiente computacional de hoje.

No contexto em que estamos inseridos, plataformas e softwares tornam-se obsoletos praticamente em meses. No entanto, os verdadeiros sistemas de “qualidade industrial” são idealizados para durar bem mais.

Para essa idealização ocorrer da forma correta, desde o início do projeto o desenvolvedor precisa pensar no futuro, desde seu início, o projeto deve ser preparado para mudanças. Programar pensando em problemas futuros é uma boa saída para evitar intercorrências que poderiam inutilizar o sistema e assim tornar a vida dele mais curta, diferentemente do que buscamos.

Dica

Cuidado para não se perder. Como assim? Tentar se programar para todas as coisas que podem acontecer não é viável e aumenta desnecessariamente o custo de produção.

Esteja preparado para ser reutilizado

Conseguir fabricar um sistema do zero que seja bom em aceitar atualizações para não se tornar obsoleto é sem dúvida um grande desafio. A reutilização do código poupa tempo e trabalho no projeto (aumentando seu valor).

> Utilizar novamente os códigos em projetos de software tem sido a grande "sacada" da programação orientada a objetos. Todavia, o retorno do investimento despendido no projeto não é algo imediato.

Para aproveitar os benefícios da **programação orientada a objetos** ou até mesmo a convencional, são necessários um bom planejamento e uma "boa visão" para intercorrências que possam vir a acontecer. Planejar com uma boa margem de tempo é essencial para a reutilização ser aproveitada da melhor forma, para reduzir o custo e agregar um valor maior aos componentes antigos e aos novos que estão sendo implementados.

programação orientada a objetos

Paradigma de linguagem de programação muito utilizado pelos desenvolvedores.

Pense sobre o assunto e pense muito!

Tudo que for feito no projeto deve ser pesquisado, estudado com muito cuidado para que o conhecimento se solidifique na cabeça e para que o caminho feito para chegar até determinado lugar seja lembrado.

> Refletir, estudar e registrar são os melhores meios para evitar erros ou cálculos ruins de projetos, pois a maioria das intercorrências acontecem quando não pensamos adequadamente no que foi feito.

Se todas essas regras forem contempladas no desenvolvimento de softwares, o valor agregado ao produto aflorará e se solidificará, tornando-se confiável, equilibrando a produção e a demanda, e rendendo bons frutos.

Verificando o aprendizado

Questão 1

Entre as opções a seguir, selecione a que representa melhor a finalidade da automação na entrega contínua.

A

Mostrar avanços tecnológicos em sua produção.

B

Evitar falhas e encontrar uma forma melhor de execução.

C

Produzir um teste bem-sucedido.

D

Tornar um código ou produto confiável por meio da eliminação de versões falhas.

E

Agilizar processos burocráticos e lentos.

A alternativa D está correta.

A automação na entrega contínua tem como finalidade eliminar as versões testes que não estejam prontas para produção, eliminando resultados falhos e garantindo produtos confiáveis.

Questão 2

Qual é o resultado mais esperado quando o modelo de entrega contínua abarca o modelo do just in time?

A

Alta produtividade e agilidade.

B

Larga produção, mantendo alta qualidade, eficiência e redução de custos.

C

Baixo orçamento com equipes reduzidas.

D

Produção de acordo com o necessário de demanda e quantidade.

E

Equipes reduzidas com alta eficiência de produção.

A alternativa D está correta.

O modelo just in time propõe que a produção seja feita de acordo com a demanda, ou seja, vai buscar equilíbrio na cadeia de produção entre operações e fluxo de demanda.

Vamos começar!

Integração contínua, entrega contínua e implantação contínua

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A prática da integração contínua (CI – continuous integration)

O processo de integração contínua

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A prática de CI (*continuous integration* ou integração contínua) é uma das mais importantes para a cultura DevOps e é a primeira a ser implementada. A *continuous integration* ou integração contínua possibilita aos desenvolvedores validarem seus códigos de forma automática para se certificarem de que eles estejam sem erros e sem incoerência, que geram inconsistência com os códigos previamente implantados no sistema. Existem algumas situações que precisam ser observadas, veja:

Primeira Situação

Em um processo de CI, a primeira coisa que deve ser observada é se o código-fonte da aplicação está disponível em um sistema de versionamento (subversion, GitHub, CSV ou GitLab). O versionamento é uma metodologia classificatória, utilizada por programadores, para o controle e o acompanhamento de alterações em um software.

Segunda Situação

É preciso observar se os desenvolvedores estão trabalhando nesse código e criando de forma contínua novas versões das funcionalidades. O CI tem o dever de garantir que os códigos novos que estão sendo desenvolvidos não possuam erros, dentro da margem do aceitável, e que não tenham conflito com o que já está escrito e rodando o código.

É bem normal que haja problema na questão da integrabilidade quando adicionamos coisas novas no projeto. Tais possíveis problemas, na maioria das vezes, são descobertos apenas depois da fase de homologação em que não foram automatizadas e que levam muito tempo para serem concluídas ou até mesmo para serem implantadas em produção. O processo de *continuous integration* tem como objetivo reduzir tais riscos e melhorar o tempo das validações.

Funcionamento do CI

O processo de integração contínua tem como início característico um programador concluir o desenvolvimento de uma funcionalidade (que envolve a escolha da metodologia ágil) e quando submete o código ao repositório da aplicação (processo conhecido como commit, que significa guardar ou enviar dados ou códigos para armazenamento em um sistema de controle de versão). No momento atual, os **testes unitários** são feitos de forma automatizada por alguma ferramenta de automação para garantir que não tenhamos códigos com erro.

Testes unitários

Os testes unitários têm como função validar uma funcionalidade isolada de uma aplicação sem levar em conta as outras partes que poderiam estar relacionadas. São usados para validar a menor parte passível de teste de uma aplicação.

> É realizado o TDD (*test-driven development* ou desenvolvimento orientado a teste), uma metodologia que preza por iniciar a programação com codificação de um teste que torne válida a funcionalidade que era esperada. Somente após a checagem, o desenvolvedor inicia o desenvolvimento do software que, quando finalizado, terá que atender às regras do teste criado anteriormente.

O teste sempre apresentará erro (tem o costume de ser chamado de "estágio vermelho") até que o código seja aprovado pelo teste e esteja preparado (que é chamado de "teste verde"). Além disso, o código deve ser revisto (fatorado) para eliminar inconsistências. Esse processo é conhecido como **ciclo TDD**.

Ciclo TDD.

Depois de o código passar pelo teste unitário, ele estará apto para aplicação em testes de validação da qualidade do código e, assim, provocar a necessidade dos desenvolvedores a revisarem seus códigos.

Os **testes de integração** precisam ser aplicados de forma automática para atestar que a nova adição no sistema não irá comprometer o restante da aplicação. Tais testes devem ser rodados toda vez que uma nova parte do código precisar se relacionar com as outras partes da aplicação. Já para as aplicações que usam

tipos de linguagens que precisam ser compiladas, o novo código é compilado previamente (processo chamado build) de forma automática e disponibilizado em um servidor (que costuma se chamar “servidor de desenvolvimento ou integração”), no qual são feitos os testes de integração.

Testes de integração

São os que validam como um código em desenvolvimento se comporta quando integrado nas outras partes da aplicação que já foram desenvolvidas.

Saiba mais

Existem diversas ferramentas que podem auxiliar na automatização do build, tais como o Ant e Maven para códigos Java, e Nat e MsBuild para Net.

Como o processo de CI tem que ser ininterrupto, caso o servidor de integração não esteja disponível, ele deverá ser criado de forma contínua durante o processo de validação. Pode também criar os servidores quando houver necessidade de uso, excluindo quando os testes estiverem terminados. Para a criação automática de servidores, é aconselhável usar algumas ferramentas de provisionamento (tais como Ansible ou Puppet) ou alguns servidores imutáveis do tipo contêineres (como o Docker).

O programador recebe notificações sobre o sucesso ou não das checagens. Ele deve, de forma ideal, submeter o seu código pelo menos duas vezes ao dia ao processo de CI, para não correr o risco de gastar um tempo desnecessário desenvolvendo um código que irá conflitar com o restante previamente criado.

Entrega contínua (continuous delivery – CD)

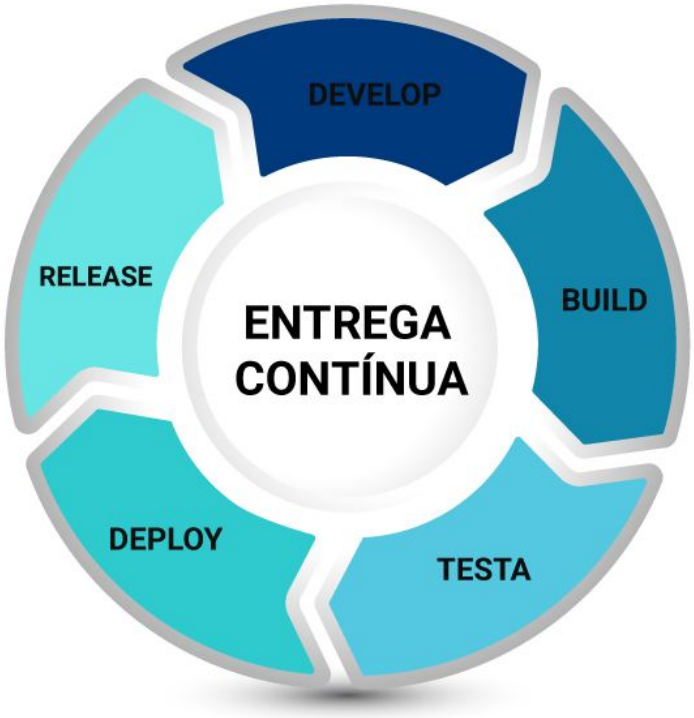
O processo de entrega contínua

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Na integração contínua, todas as checagens e a compilação são realizadas em um ambiente de desenvolvimento com a meta de otimizar os processos de confecção de softwares. O máximo da criação ainda não pode ser utilizado pelo cliente, logo, o valor agregado ao produto não chega aos usuários. Para um novo release (liberação) entrar em produção, é necessário fazer testes de aceitação, que são complementares às checagens feitas na fase de CI e essenciais para avaliar se um software está de acordo com as necessidades para seguir em produção.



Processo de entrega contínua.

Os testes de aceitação fazem parte de uma fase da implantação que aprova se um novo código está de acordo com as regras de negócio e com o sistema vigente. Eles verificam e certificam se os critérios de uma história foram contemplados. Os testes podem checar tanto as partes funcionais como as que não funcionam. Os não funcionais são capacidade, desempenho, capacidade de modificação, disponibilidade, segurança e usabilidade.

A entrega contínua é uma fase complementar ao CI, em que são realizados testes de aceite em ambientes antes da produção e feitas organizações finais para que um código novo seja aprovado para ser encaminhado para produção, após aprovação.

Funcionamento do CD

Depois da conclusão do processo de CI, começa a implementação de um novo código em um ambiente de homologação (comumente conhecido como staging), no qual os testes de aceitação são feitos. Nessa etapa, tem-se o costume de incluir testes de interface (**Selenium** por exemplo), que de maneira automática executam a validação de funcionalidades esperadas, mas também podem adicionar testes de desempenho, segurança e outros. Os testes de aceitação manuais e complementares são comumente realizados por uma equipe. Além disso, eles também devem ser evoluídos de forma contínua até o ponto de cercarem a maior parte das validações necessárias, sobrando poucas tarefas para as fases com verificações finais e manuais.

Selenium

Conjunto de ferramentas de código aberto usado para testar interfaces de forma automatizada.

A cada nova história terminada, ela poderá ser submetida ao processo de CD, ou apenas ao processo de CI para formar um bloco de atualizações a ser lançado de uma única vez em um release, que, a partir daí, seguiria para CD. Tal estratégia dependerá do plano de lançamento e da disponibilidade da empresa de executar novos testes. O ideal é que as modificações cheguem o mais rápido possível ao cliente.

Comentário

Os servidores usados nos ambientes de homologação devem ser os mais próximos possíveis dos usados em produção e devem ser disponibilizados para uso assim que houver a necessidade (de forma automática) para que o processo seja ininterrupto. É de suma importância a escolha de infraestrutura ágil para gerir de forma automatizada a estrutura, de modo que não haja interrupções. Em alguns projetos, essa etapa é utilizada para versionar novos componentes que já foram compilados e validados.

Avisos são enviados sobre cada etapa do processo. Dessa maneira, a equipe recebe pequenas porções de trabalho continuamente, tornando mais fácil o processo de validação. Os desenvolvedores recebem feedbacks sem interrupção sobre eventuais erros nos seus novos códigos, tornando mais rápida a resposta aos problemas, além de lembrar constantemente de detalhes do código que produziram. Usuários que recebem releases continuamente podem fornecer melhores feedbacks, aumentando o valor agregado do sistema por ele estar de acordo com as necessidades do cliente e do mercado de forma sempre atualizada.

A entrega contínua do deployment (implantação) em produção não é automática, mas a forma como é feita, sim. Como assim? Todo o sistema é automático, porém, depende da aprovação humana para ser aprovado e rodado.

Implantação contínua (continuous deployment)

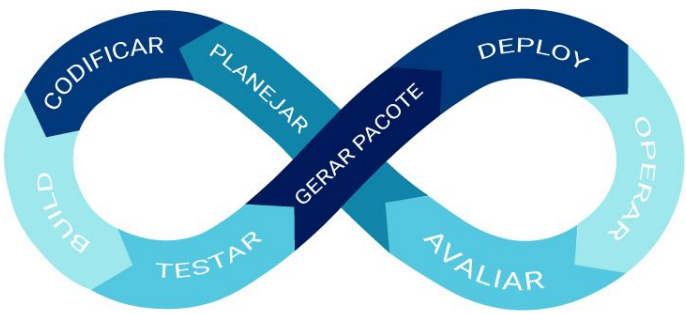
O processo de implantação contínua

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A implantação contínua é a continuação da entrega contínua. Logo que o programador decide que seu código está pronto e aprova o deploy, são feitas todas as checagens previstas nas regras anteriores, e, se não houver nenhum erro, um novo código é disponibilizado automaticamente no ambiente de produção.



Processo de implantação contínua.

Funcionamento da implantação contínua

A prática da implantação contínua não é aplicada em toda a empresa. Ela foi pensada principalmente, mas não apenas, para o desenvolvimento de aplicações para internet. Existem muitas regras de negócio rígidas e necessidades que precisam ser atendidas quanto a testes manuais que impeçam o uso dessa metodologia. No entanto, é aconselhável que, quando as empresas tenham a oportunidade (tempo, funcionário e verba), apliquem a IC, pois ela promove mudanças positivas e profundas na organização.

Comentário

Como o desenvolvedor tem a capacidade de colocar suas mudanças em produção de forma automática, o programador tende a se preocupar mais com as questões referentes à qualidade e confiabilidade do código. A permanência de disponibilidade das pequenas alterações em produção aumenta a quantidade e a frequência dos feedbacks (que passam a ser mais pontuais e específicos) dos usuários, e o valor agregado ao produto também aumenta.

A implantação contínua deve adotar ferramentas de vigilância no ambiente de produção para garantir como as alterações influenciaram o uso dos recursos disponíveis. Por exemplo, fazer uso das métricas que acabam sendo disponibilizadas, como a qualidade do acesso, volume do tráfego, tempo de resposta, entre outras.

Verificando o aprendizado

Questão 1

1) Quais das afirmativas a seguir são verdadeiras?

I - Continuous integration (integração contínua), continuous delivery (entrega contínua) e continuous deployment (implantação contínua) são práticas DevOps amplamente utilizadas durante os últimos anos.

II - A prática de CI (continuous integration ou integração contínua) é uma das menos importantes para a cultura DevOps e nunca é aplicada.

III - Testes unitários têm como função validar uma funcionalidade isolada de uma aplicação sem levar em conta as outras partes que poderiam estar relacionadas. São usados para validar a menor parte passível de teste de uma aplicação.

A

I e II.

B

II e III.

C

I e III.

D

II.

E

Todas as anteriores.

A alternativa C está correta.

A afirmativa II está errada, pois a prática de CI é uma das mais importantes para a cultura DevOps e quase sempre é aplicada.

Questão 2

Os testes de integração são usados para

A

validar a compatibilidade de um código novo com o sistema vigente.

B

iniciar a etapa da integração contínua.

C

automatizar o software.

D

validar a o software usado antes de subir algum código.

E

fazer a vigilância de erros no software.

A alternativa A está correta.

A letra A está correta, pois os testes de integração são os que validam como um código em desenvolvimento se comporta quando integrado nas outras partes da aplicação que já foram desenvolvidas.

Vamos começar!

Princípio das 3 maneiras

A seguir, confira os principais pontos sobre o assunto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

DevOps e o princípio das 3 maneiras

O **princípio das 3 maneiras** faz parte também das práticas DevOps, pois fala sobre valores e filosofias que constroem os processos do ciclo. Entender o Three Ways ou princípio das 3 maneiras é solidificar ainda mais a cultura DevOps, por meio da atuação das práticas na empresa. A seguir, trataremos de cada maneira e esclareceremos as etapas que precisam ser seguidas para a sua efetiva aplicação nos times.

Primeira maneira

A primeira maneira possibilita **aumentar a agilidade de um fluxo de trabalho** dos desenvolvedores para o cliente. Além disso, também coloca em foco o desempenho de todo o organismo, que é diferente do que é pregado nos times (em apenas um silo). Como o fluxo de todo um sistema é aumentado, dando agilidade aos processos, o valor agregado do produto também aumenta, pois as mudanças chegam mais rápido para o usuário final.

> Como funciona?

Como todo projeto, a primeira maneira tem uma sequência de ações que precisam ser feitas para que a implementação aconteça com solidez onde for implantada. Veja a sequência de ações:

Primeira

Fazer o trabalho do que é escrito visível, pois assim conseguimos identificar melhor as prioridades e ter em vista mais facilmente os objetivos.

Segunda

Limitar o trabalho em andamento (WIP), dividindo as tarefas de forma ordenada sequencialmente, otimizando o trabalho e diminuindo os efeitos negativos das multitarefas.

Terceira

Reduzir o tamanho dos lotes de trabalho, criando um fluxo de tarefas mais simples e fluido em forma de lotes pequenos.

Quarta
Reduzir o número de transferências, prezando que os projetos não mudem muito de equipe e quando for preciso mudar, que seja passado o trabalho por uma documentação ou por automatização.

Quinta
Identificar e reduzir as restrições, para assim diminuir os problemas, evitar gargalos de produção e, para maior agilidade, eliminar desperdícios, visando proporcionar aumento de rendimento, competitividade e valor agregado ao negócio.

Quando tal sequência de ações é aplicada, espera-se não replicar um defeito conhecido na próxima etapa da programação do código, não atualizar partes do sistema que iram se tornar obsoletas, aumentar o fluxo de forma segura da produção e sempre colocar como meta um profundo conhecimento do sistema.

Segunda maneira

A segunda maneira demanda a **amplificação dos feedbacks** para reduzir a recorrência de problemas ou possibilitem a detecção e recuperação de dados com mais facilidade e de forma mais ágil, visando sempre à segurança. Além disso, os feedbacks constantes ajudam os desenvolvedores a entender melhor o perfil do cliente, atendendo às demandas da melhor forma possível.

> Como funciona?

Para contemplar os princípios, temos um conjunto de ações que possibilitam sua aplicabilidade. Veja a seguir a sequência de ações:

Primeira
Trabalhar com sistemas complexos, onde nada mais justo é prezar pela sua segurança com o código. Para isso, é necessário ter um bom gerenciamento, revelando os problemas o mais precocemente possível, compartilhando o saber com todos os presentes na produção do código e sempre treinando alguém para o trabalho de prevenção de saídas ou demissões no time.

Segunda
Ver os problemas à medida que eles ocorram, tendo como meta aumentar o fluxo de informações para criar uma boa infraestrutura.

Terceira
Documentar para construir novos conhecimentos: para resolver algum erro no código, temos que estudar, e tal movimento deve ser documentado para prevenir futuros problemas. Esse movimento diminui exponencialmente o custo e o esforço para consertar problemas, impede que se inicie vários projetos ao mesmo tempo (o que poderia causar erros no sistema).

Quarta
Manter as tomadas de decisão nas mãos dos desenvolvedores: sabemos que a necessidade de automação é importante para os processos, porém, quando tudo é decidido por máquinas, algo pode sair errado e tal comando pode se perpetuar por todo o sistema.

Terceira maneira

Depois de incorporar muito bem as práticas da primeira e segunda maneira, poderemos inovar. A terceira maneira cria **sistemas de trabalho mais seguros** e que se adaptam ao perfil do cliente rapidamente, entregando melhor e mais rápido para o mercado, dessa forma ganhando da concorrência e aumentando o valor agregado ao produto. A terceira maneira também estimula a implementação de culturas na empresa e a disseminação do conhecimento.

> Como funciona?

Nessa etapa, precisamos estimular o aprendizado organizacional dentro da empresa e fazer com que este seja tão enraizado, que todos os times pensem da mesma forma (com relação aos processos), e que a comunicação se faça de uma única maneira, evitando, assim, desentendimentos, sempre estimulando os líderes das equipes a exercerem uma liderança generativa, e não burocrática ou patológica. Veja a sequência a seguir:

Primeira
Estimular a constante melhoria nos projetos. Descobertas no âmbito local sendo distribuídas de forma global, cada time tem uma experiência e cada uma delas deve ser compartilhada para converter o conhecimento do “acaso” a um conhecimento perene que se torna de grande valor para a empresa, deixando assim um “livro” de erros, acertos, caminhos e decisões que são melhores para o sistema.

Segunda
Injetar uma constância e resiliência no trabalho com código, a fim de deixar o código menos frágil, visando aumentar o valor agregado ao produto, pois ele está mais seguro (desde o planejamento até a sua entrega ao usuário).

Terceira
Enraizar na empresa, times e desenvolvedores todas as etapas e tudo o que se é discutido. Assim, teremos líderes aptos a difundir esse conhecimento na empresa, estimulando os subordinados a crescerem e levarem essa cultura para todos os desenvolvedores.

Verificando o aprendizado

Questão 1

Sobre o princípio das 3 maneiras, marque a alternativa correta.

A

Não está incluído nas práticas DevOps.

B

Não aborda como lidar com sistemas complexos.

C

Coloca em foco o desempenho de todo o organismo, diferentemente do que é pregado nos times (em apenas um silo).

D

Estimula os líderes das equipes a exercerem uma liderança burocrática ou patológica, e não generativa.

E

Coloca em foco o desempenho de parte do organismo.

A alternativa C está correta.

Nessa etapa, precisamos estimular o aprendizado organizacional dentro da empresa e fazer com que este seja tão enraizado que todos os membros dos times pensem da mesma forma (com relação aos processos), que a comunicação se faça de um único modo, evitando, assim, desentendimentos. É preciso estimular sempre os líderes das equipes a exercerem uma liderança generativa, e não burocrática ou patológica.

Questão 2

Por que a segunda maneira demanda a amplificação dos feedbacks?

A

Porque visa estimular a implementação de culturas na empresa e a disseminação do conhecimento por meio dos feedbacks.

B

Porque possui uma sequência de ações que precisam ser feitas para que a implementação aconteça com solidez onde for implantada.

C

Porque o feedback diminui exponencialmente o custo e o esforço para consertar problemas, impede que se inicie vários projetos ao mesmo tempo, o que poderia causar erros no sistema.

D

Porque reduz a recorrência de problemas e possibilita a detecção e recuperação de dados com mais facilidade e de forma mais ágil, visando sempre à segurança e ajudando os desenvolvedores a entenderem melhor o perfil do cliente.

E

Porque o feedback aumenta o custo para consertar problemas.

A alternativa D está correta.

Os feedbacks geram uma aproximação do usuário final, não deixando o produto sair demais do padrão esperado pelo cliente.

Considerações finais

No decorrer dos anos, as metodologias e práticas relacionadas ao desenvolvimento de software passaram por várias mudanças. Em 2009, apareceu um movimento que ficou conhecido como DevOps, cujo objetivo principal é elaborar uma cultura de colaboração entre as equipes de desenvolvimento e operadores, com o intuito de fazer entregas de software, funcionando em produção de forma ágil, segura e estável.

Vimos como a cultura DevOps é importante para a colaboração entre funções como desenvolvimento e operações. É uma cultura sem culpa e de alta confiança, na qual as pessoas são livres para experimentar. A cultura DevOps é cercada de regras, princípios e metodologias, com forte vocação para a melhoria contínua.

Considerando todos esses benefícios, podemos dizer que a escolha da adoção da cultura DevOps não é apenas algo estratégico para a área de Tecnologia. DevOps é estratégico também para as áreas de negócio, pois, além de trazer práticas que melhoram a qualidade e a assertividade do software, aprimoram também a experiência para o cliente final, possibilitando que as empresas se mantenham cada vez mais competitivas no mercado e entreguem novas funcionalidades em uma agilidade cada vez maior.

Podcast

Para encerrar, ouça os principais pontos sobre este estudo.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Leia o texto **Fundamentos Internacionais do Scrum Master**, de Steen Lerche-Jensen, publicado na Scrum Academy, em maio de 2019.

Referências

BARROS, A. R. C. **Avaliação da Cultura Organizacional para a Implantação de DevOps**. 2018. Dissertação (Mestrado em Gestão do Conhecimento e Tecnologia da Informação) – Universidade Católica de Brasília, Brasília, 2018. Consultado na Internet em: 05 out. 2022.

DAVIS, J.; DANIELS, R. **Effective DevOps**: Building a Culture of Collaboration, Affinity, and Tooling at Scale. Massachusetts: O'Reilly, 2016.

HOOKER, D. **Seven principles of software development**. Wiki. C2. Consultado na Internet em: 05 out. 2022.

HUTTERMANN, M. **DevOps for developers**: integrate development and operations, the agile way. New York: Apress, 2012.

KIM, Gene *et al.* **The DevOps Handbook**: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Portland: It Revolution Press, 2016.

KNIBERG, H. **Kanban and Scrum**: making the most of both. San Francisco: QCon, 2009. Consultado na Internet em: 05 out. 2022.