Integração Contínua DevOps

A metodologia de integração contínua DevOps e seu envolvimento com a cultura, automação e design de plataforma de software. O valor agregado ao produto com o aumento da capacidade de resposta às mudanças, entrega de serviços rápidos, mas de alta qualidade.

Prof. Brunelli Gabriel Cupello

Propósito

Criar ou integrar um software e investigar *bugs* rapidamente, além de melhorar sua qualidade, reduzindo o tempo de validação, possibilitando o lançamento de novas atualizações, bem como um componente de automação, como um serviço de criação e um componente cultural.

Objetivos

- Identificar o funcionamento e os benefícios da implantação da integração contínua.
- Analisar os requisitos necessários para a implantação da integração contínua em uma empresa.
- Reconhecer quais são as práticas sugeridas ao implementar a integração contínua DevOps.
- Identificar o que são e como abordar os testes de integração dentro da cultura DevOps.

Introdução

A integração contínua é uma forma de automatizar um pacote de alterações de código de variados contribuidores em um projeto único de software. Também é reconhecida como uma prática primária recomendada de DevOps, termo utilizado para descrever uma prática de engenharia de software que tem como intuito aproximar e unir os times de desenvolvimento (Dev) com os times de operação de software (Ops).

DevOps torna os desenvolvedores capazes de mesclar frequentemente as alterações de código em um repositório central onde builds e testes são realizados. Pode-se dizer que a metodologia DevOps possui abordagens que aceleram os processos entre as ideias do que será desenvolvido e sua implementação.

Para que a implementação do DevOps seja eficaz e resolutiva, é imprescindível haver comunicação entre os membros das equipes, além de um trabalho colaborativo, provisionamento flexível e a capacidade de atender às demandas dos clientes sem perder sua qualidade.

Nessa metodologia, a criação de códigos é feita por uma equipe de desenvolvedores em um ambiente padrão. Esses desenvolvedores têm estreita colaboração com a equipe de operações de TI com o intuito de fomentar a compilação de programas de software, fazer testagens e solucionar possíveis problemas, mantendo a confiabilidade. Um sistema de controle de versão de código-fonte é o ponto crucial do processo de integração contínua, sendo complementado também com verificações como testes de qualidade de código automatizados, ferramentas de análise do estilo da sintaxe, entre outras.

Considerando os recursos conseguidos com a metodologia DevOps por meio do autosserviço e da automação, a assistência que o DevOps traz para o gerenciamento de ambientes e as possibilidades de sucesso que as organizações conseguem ter no alinhamento de suas equipes em relação a processos, ferramentas e responsabilidades, é necessário compreender essa metodologia para incorporá-la no desenvolvimento de softwares trazendo agilidade e alinhamento.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O que é a integração contínua?

Neste vídeo apresentaremos os principais passos do processo de integração contínua com uma breve descrição do que é feito em cada um deles.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A integração contínua, também conhecida como *continuous integration* (CI), tem a capacidade de construir um conjunto enorme de melhorias contínuas e incrementais nos processos, indo desde a construção (*build*) até a publicação em ambientes para teste, homologação ou produção. Nada mais é do que a integração de um código ao projeto principal cada vez que ele for alterado ou quando forem criadas novas funcionalidades. A integração contínua DevOps faz a junção de equipes de desenvolvimento e operações que trabalham juntas, por meio de ferramentas e práticas.

Processo de integração contínua.

O objetivo é encontrar possíveis *bugs*, retirando-os com o intuito de melhorar o funcionamento e as funcionalidades do software. A integração contínua permite que vários códigos sejam integrados e criados, alterando-os ao mesmo tempo, o que não era possível antes. Até que as alterações estivessem prontas para o lançamento de uma nova versão de um software ou aplicativo, demorava-se muitos meses e o usuário final, caso tivesse problemas no uso de um aplicativo, como o Whatsapp, por exemplo, permanecia com ele assim.

Comentário

Para que a integração contínua seja implantada, é necessário que as equipes adotem novas formas de trabalhar, se aproximando dos usuários, para compreender melhor as necessidades deles.

Essas equipes de operações devem adicionar ao seu dia a dia requisitos de manutenção, como conjunto de ferramentas para controle de versão, integração contínua, testes automatizados e inspeção da qualidade do código. Devem também alinhar as necessidades do cliente ao software que está sendo criado, unindo princípios essenciais a um ritmo mais rápido e maior qualidade do que os usados em desenvolvimento de softwares tradicionais.

Para que todo esse processo flua bem, os membros da equipe, independentemente do tamanho dela, precisam implementar a integração contínua.

Funcionamento da integração contínua

A integração contínua por DevOps está sempre atenta à redução de riscos à aplicação. Por isso, funciona como uma **técnica** de **desenvolvimento de software** na qual as mudanças de código em um aplicativo são liberadas de forma automática no ambiente de produção, sendo esta automação guiada por testes predefinidos. Assim que se percebe a alteração em um código, o servidor seleciona e testa o código de sua aplicação. Porém, se algo sai fora do esperado, automaticamente uma ferramenta notifica os desenvolvedores para que assim possam corrigir o erro rapidamente.

Comentário

Essa função do próprio sistema de “avisar” aos desenvolvedores sobre o erro de um código, chamada de visibilidade das métricas de qualidade, faz com que a equipe se preocupe mais com o desenvolvimento, com vistas a facilitar a manutenção do código.

Por meio dos testes de aceitação, podemos dizer que a integração contínua também funciona como uma ferramenta de comunicação, pois dá o resumo completo do real estado do projeto em andamento. Por isso, é capaz de simplificar e acelerar a entrega do produto/software, contribuindo com a **automação do processo**, automatizando a publicação da última versão da aplicação ou autorizando a operação com simples toque no botão. Após as novas atualizações passarem nos testes, o sistema envia as atualizações diretamente para os usuários do software, como funciona, por exemplo, nos sistemas Android e IOS de smartphones.

Implementando a integração contínua

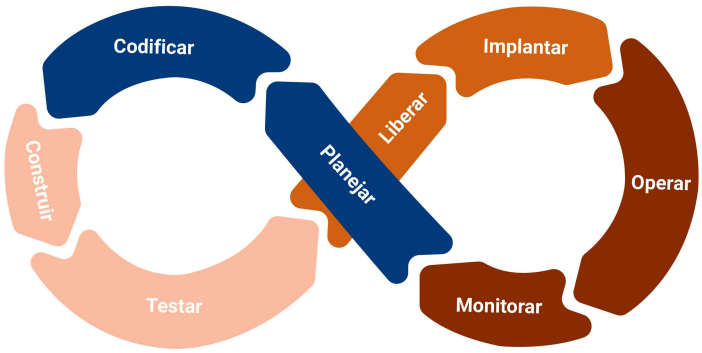
Neste vídeo, abordaremos os principais passos do processo de implementação da integração contínua com uma breve descrição do que é feito em cada um deles.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Passos para a Implementação da integração contínua por DevOps

A **IC** é uma ferramenta muito útil para a criação de softwares em equipe, pois é capaz de melhorar a produtividade, ao passo que automatiza o trabalho. Apesar de ser uma metodologia nova, vem sendo amplamente divulgada e, por isso, sua adoção nas empresas está sendo acelerada. Mas como tudo na vida exige processos, com a implementação do DevOps não seria diferente. Para a sua implementação, é necessário ter atenção e uma sequência de passos a fim de evitar erros, observe a seguir:



Passos para manutenção do Ciclo DevOps.

Primeiramente, é essencial mudar a cultura do ambiente de trabalho. É imprescindível uma equipe multidisciplinar, visto que o DevOps envolve tanto a área de tecnologia da informação como a área de desenvolvimento. A análise de ferramentas e de processos não será suficiente se não houver também a abordagem colaborativa.

Outro ponto é a adoção de metodologias ágeis por parte dos gestores, de modo que sua abordagem seja voltada à integração dos times e resultados. Os funcionários da empresa, que tomam lugares de liderança das equipes, devem ter a mesma visão de planejamento de atividades focadas em resultados, automação e integração. Além disso, as metodologias ágeis precisam ser analisadas por meio de ciclos de desenvolvimento de softwares para que os processos sejam acelerados.

De acordo com matéria publicada no site **InfraNews Telecom**:

Dica

A parte técnica suporta a gestão de pessoas e vice-versa. Se a equipe está preparada para a implementação do DevOps e sua manutenção, então o processo será benéfico.

As automações também são pontos cruciais dentro da metodologia DevOps, pois se não forem feitas corretamente, podem afetar tanto a estrutura da empresa como o próprio cliente. Desse modo, antes que qualquer ferramenta seja ativada para sua função, é importante fazer diversos testes, como os de regressão, aceitação, integração e outros. A finalidade é garantir que o ciclo automatizado esteja alinhado tanto ao desempenho da empresa como aos desejos do cliente.

Por último, destaca-se o uso de metas e métricas, as quais devem ser traçadas antes mesmo do início do projeto. Dessa forma, evita-se a geração de dados de baixo valor ao negócio, ao mesmo tempo que torna possível a visualização do objetivo final no momento específico que está acontecendo.

DevOps e a containerização

Com a necessidade de entrega rápida e de qualidade, surge um novo conceito em TI:

> A containerização, que nada mais é do que a permissão de compartilhamento de tecnologias e dependências por meio de aplicações entre as equipes.

Essas equipes atuam em diferentes partes do processo de desenvolvimento dentro da empresa e, mesmo assim, são capazes de transitar entre os ambientes de desenvolvimento, homologação e produção de um

software. Essa prática de “**colocar em containers**” facilita a criação de máquinas virtuais e é capaz de mudar os ambientes e as entregas de softwares, produtos e funcionalidades no mundo da tecnologia da informação.

Atenção

O papel do container no DevOps é de proteger minimamente a aplicação com o seu próprio sistema operacional.

Por meio da containerização, os desenvolvedores compartilham entre os colegas de equipes de operações as imagens e dependências. Assim, é possível que todos vejam as evoluções do software, por exemplo, e eliminem os pequenos impasses.

São exemplos de containers o **Docker** e o **Snaps**. O Docker, que é uma plataforma de código aberto, é mais famoso e usado na maioria das aplicações, sendo capaz de criar imagens que podem ser compartilhadas entre diferentes ambientes rapidamente em questão de segundos. Além disso, os ambientes também podem ser copiados automaticamente, promovendo uma fácil integração entre desenvolvimento, homologação e produção.

A containerização trouxe benefícios para a metodologia DevOps, pois foi capaz de deixar de lado a utilização de máquinas virtuais e servidores de aplicação e agora permite uma escalabilidade fácil, assumindo a responsabilidade de responder ao mercado e às suas mudanças de forma ágil.

Comentário

Essa resposta ocorre por meio da organização do sistema em partes separadas. Assim, é possível manter a produção de softwares ou funcionalidades para uma aplicação, pois a modularização permite a coexistência de várias esteiras de produção ao mesmo tempo, atuando em etapas diferentes do processo de produção, como: desenvolvimento, homologação e produção de um produto.

Antigamente, com o uso dos servidores de aplicação e das máquinas virtuais, não era praticável realizar escalonamentos, executar Deploys nem entregar com a rapidez e facilidade que se entrega hoje com o uso do Docker. Com o uso da containerização, cria-se um ambiente isolado e mais leve para rodar os programas, havendo ainda a promoção de reversão de versões em caso de problemas.

Se um aplicativo foi atualizado e alguma de suas funções veio alterada, é possível avançar o aplicativo, lançando uma nova versão. Há ainda a possibilidade de reverter essa função fácil e rapidamente. Esse aumento da eficiência no desenvolvimento de softwares acaba sendo considerado como um fator motivador para a manutenção do uso de DevOps e da containerização nas empresas.

Benefícios da integração contínua

Benefícios da integração Contínua do Desenvolvimento de Software

Neste vídeo, destacaremos como o processo de integração contínua beneficia o desenvolvimento de software e quais as métricas que podem auferir esses benefícios.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O uso da metodologia de Integração Contínua de DevOps aborda meios para a aceleração de processos que vão desde a criação de um projeto até a implantação dele em um ambiente. Reflita, sozinhas as partes funcionam muito bem. Porém, juntas pode ser que não trabalhem da mesma forma. Isso traz um atraso considerável, uma vez que um ou mais membros da equipe precisa entender o que está acontecendo e corrigir esses problemas para que o sistema completo possa ser dado como concluído. É daí que vem talvez a **principal vantagem** da integração contínua:

Atenção

Integrar todas as partes constantemente durante o desenvolvimento, para que, na conclusão do software, ele funcione como um todo, sempre com a preocupação de satisfazer o usuário final.

Esse projeto pode ser um novo recurso de software, a solicitação de aprimoramento ou uma correção de **bug**.

Para tanto, é necessário que haja a criação da cultura de colaboração e confiança. A equipe que usa a integração contínua de DevOps de alto desempenho deve manter a responsabilidade compartilhada, o trabalho colaborativo, a empatia entre os membros, a transparência, o feedback mais rápido, o provisionamento flexível e a escalabilidade.

Comentário

Escalabilidade nada mais é do que fazer a empresa crescer atendendo às demandas dos clientes sem perder as qualidades que lhe agregam valor e é alcançada por meio do autosserviço e da automação. Com essas atribuições, todas as partes do projeto/software são integradas e o gargalo de tempo é diminuído.

Desse modo, as equipes lançam versões com mais frequência e com qualidade e estabilidade maiores, sendo capazes de liberar versões com uma frequência 208 vezes maior e 106 vezes mais rápido do que equipes de baixo desempenho, conforme o relatório "**2019 State of DevOps**", da DORA.

A inclusão de ciclos automatizados de teste e revisão diminui o tempo de resposta a incidentes, aumentando a velocidade e melhorando a confiança da equipe. Esses ciclos são ferramentas que impulsionam a automação e os processos novos. Por meio deles, as equipes podem aumentar a produtividade e lançar com mais frequência e com menos contratempos.

A equipe com ciclo de feedback mais rápido, ou seja, com comunicação aberta, é aquela que se destaca, minimizando o tempo de inatividade e resolvendo itens rapidamente, evitando tensões e frustrações entre equipes de desenvolvimento e operações, corrigindo incidentes e desbloqueando pipelines de lançamento mais rápido, visando à maior satisfação do cliente.

Atenção

Para tal finalidade, é necessário planejamento, pois, por meio de estabelecimento de processos e priorização clara, as equipes gerenciam melhor os imprevistos e continuam focando no que já havia sido planejado. Assim, também é possível reduzir complexidade de problemas, erros e ainda diminuir o tempo médio de recuperação (MTTR) quando há incidentes e interrupções.

Principais erros cometidos na prática de DevOps

Neste vídeo, destacaremos quais os principais erros cometidos pelos profissionais em relação ao processo de integração contínua e a própria prática do DEVOPS.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vejamos a seguir um compilado dos principais erros associados ao DevOps.

1

Cultura erroneamente divulgada

A cultura DevOps vem sendo erroneamente divulgada como vaga de emprego em algumas plataformas de contratação. No entanto, vale ressaltar que, para ocupar uma “vaga de DevOps”, o especialista deverá estar ciente que assumirá as funções de desenvolvedor de software e do engenheiro de suporte por um salário mensal único.

2

Contratação de especialista

Contratar um especialista em DevOps e não mudar a estrutura de desenvolvimento dentro da empresa não trará bons resultados. É necessário abrir mão dos métodos tradicionais e obsoletos, contratar especialistas e mudar a estrutura organizacional e de trabalho para que, de fato, a metodologia DevOps seja implementada e dê resultados.

3

DevOps não é continuação

DevOps não é a continuação das metodologias ágil ou lean, apesar de alguns erroneamente o denominarem assim. Essa confusão ocorre por conta de o DevOps ter surgido do desenvolvimento ágil e por este estabelecer relações com os clientes, alinhando seus pedidos com o produto final e lançando o software o mais rápido possível. O DevOps estende a produção ágil para o TI de forma geral, envolvendo organização, processo e cadeia de valor, assim como a tão falada mudança cultural na empresa, o que não ocorre no método ágil.

4 Implementação do DevOps

A implementação do DevOps vai além de uma entrega rápida para o cliente, seu principal objetivo é eliminar a fragilidade da infraestrutura, construindo sistemas antifrágeis nos pontos de instabilidade usados a favor do software.

5

DevOps não é ferramenta

DevOps não é apenas uma ferramenta para automatizar o processo de desenvolvimento de um produto. Na verdade, ele depende da disponibilidade e eficácia de ferramentas de automação, as quais podem ser reduzidas a um sistema de controle de versão, armazenando todos os códigos-fonte.

Verificando o aprendizado

Questão 1

O que é necessário para que a integração contínua seja implementada em uma empresa?

A

Elaborar um projeto principal.

B

Desenvolver uma funcionalidade nova no software.

C

Adotar operações, ferramentas e práticas.

D

Adotar novas formas de trabalhar por parte das equipes.

E

Realizar manutenção no software.

A alternativa D está correta.

Para que a integração contínua seja implantada, é necessário que as equipes adotem novas formas de trabalhar, se aproximando dos usuários, para compreender melhor as necessidades deles.

Questão 2

Sabe-se que a integração contínua funciona como uma técnica de desenvolvimento de software em que as mudanças de código em um aplicativo são liberadas automaticamente no ambiente de produção. Nesse contexto, é correto afirmar que essa automação é guiada por:

A

Testes predefinidos.

B

Trabalho colaborativo.

C

Servidores de aplicação.

D

Equipe multidisciplinar.

E

Manutenção do código.

A alternativa A está correta.

A integração contínua é uma técnica de desenvolvimento de software em que as mudanças de código em um aplicativo são liberadas automaticamente no ambiente de produção, e essa automação é guiada por testes predefinidos. Após as novas atualizações passarem nos testes, o sistema envia as atualizações diretamente para os usuários do software.

Controle de versão

Controle de Versão e DEVOPs

Neste vídeo, apresentaremos o que é controle de versão e porque esta ferramenta é um requisito para integração contínua.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Sistemas de controle de versão são softwares que servem para buscar e gerenciar as alterações em um código do software durante seu ciclo de desenvolvimento, ampliando a quantidade de implementações que dão certo.

Os objetivos do controle de versão são rastrear os arquivos e recuperar versões anteriores de maneira automatizada, para diminuir o tempo nessa tarefa e tornar o software mais eficiente e confiável, além de possibilitar a realização de configurações paralelamente, reduzindo o espaço gasto com o armazenamento.

Comentário

Esses softwares controlam as alterações realizadas ao longo do processo de desenvolvimento, independentemente da quantidade de desenvolvedores envolvidos. Esse controle é feito a partir da criação de versões instantâneas salvas em um repositório. Caso alguma alteração seja descartada e seja necessária para que o software funcione adequadamente, é possível recuperá-la por meio do gerenciamento das versões do código, onde o histórico de versões fica armazenado, sendo apresentado à equipe uma única versão por vez.

O histórico pode ser acessado em caso de consultas ou para retornar a uma versão mais estável do software, em caso de problemas com as atualizações. Assim, a cada alteração que for feita no código, será preciso atualizar sua versão no servidor. Desse modo, não é necessário que os desenvolvedores salvem cópias em seus computadores, evitando-se também erros ao se excluir uma versão errada culminando na perda do trabalho, por exemplo.

Também é possível que vários desenvolvedores façam modificações em partes diferentes do software ao mesmo tempo, sem uma interferir na outra. Todo o projeto deve estar em um único repositório, incluindo os códigos, testes, *scripts* de bancos de dados, de compilação e implantação, e tudo o que for necessário para criar, instalar, executar e testar sua aplicação. Independentemente do tamanho do projeto, esse controle deve ser feito, podendo ser escolhidos sistemas de controle de versão simples, leves e poderosos.

Os **dois tipos** de controle de versão mais utilizados são:

- Centralizado
- Distribuído

No controle de versão centralizado, existe um repositório central; por meio do *commit* e do *update*, os desenvolvedores se comunicam e fazem suas alterações. Funciona bem para equipes menores e que estão no mesmo lugar físico. O Subversion é um tipo de controle de versão centralizado.

Já controle de versão distribuído é mais utilizado em equipes com grande número de desenvolvedores que não precisam estar no mesmo espaço físico. Funciona com cada computador tendo o seu próprio servidor, que faz as operações automaticamente. A comunicação entre o servidor principal e as áreas de trabalho de cada desenvolvedor ocorre a partir dos comandos *push* e *pull*, que fazem a mesclagem das versões do software.

Compilação automatizada

Compilação Automatizada e DEVOPs

Neste vídeo, apresentaremos o que é a compilação automatizada e porque esta ferramenta é um requisito para integração contínua.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A compilação automatizada, cujas características principais correspondem à **repetição** e **escalabilidade**, consiste no tratamento e na preparação das informações utilizadas em análises, avaliações e consultas. Essas informações serão automatizadas em *scripts*, que são atividades variadas, ou seja, todos os processos que os desenvolvedores de software precisam realizar, como: criação de documentos e notas de lançamento, processos envolvendo os testes, compilação de códigos-fonte em código binário e o empacotamento desses códigos, entre outras.

Ao automatizar todas essas atividades, diminui-se tanto o tempo de desenvolvimento do software, como o risco de aparecimento de erros e o retrabalho, bem como o custo do processo do desenvolvimento do software. O processo de automatizar diminui o quantitativo de pessoal que faria as mesmas atividades só que de forma manual.

Atenção

Contrastar dados, desagregar elementos diferentes, juntar elementos semelhantes e categorizar informações são técnicas utilizadas para compilar dados. É uma vantagem competitiva no mundo de criação de softwares implantar essa automatização.

O processo de compilação deve ser executado a partir de uma linha de comando, podendo ser iniciado com um programa simples que compila a aplicação e roda os testes. Também pode ser usado um sistema de múltiplos estágios, no qual outros *scripts* fazem seu trabalho. É importante que o mecanismo escolhido seja praticável para qualquer um da equipe compilar, testar e instalar a aplicação de maneira automatizada a partir da tal linha de comando.

No que tange à gestão, a compilação automatizada é capaz de dar, em tempo real para o gestor, relatórios automatizados sobre o quantitativo de vendas e a arrecadação, quão produtiva está sendo uma equipe e quanto tempo cada membro dela leva em cada atividade realizada (visando sempre à diminuição desse tempo e ao aumento da qualidade de entrega), além de somar o banco de horas de cada funcionário (quando houver), por exemplo.

Comentário

Hoje em dia, com o avanço das tecnologias no setor de desenvolvimento de softwares, existe a robotic process automation (RPA), ou automação robótica de processos, em português. Essa automação consiste na criação de um software robô programado para compilar dados, possibilitando ainda que as apresentações e o relatórios sejam automatizados, conseguidos por meio da coleta de dados automatizados.

O **RPA** torna as empresas mais suscetíveis a se tornarem escaláveis, ou seja, a aumentar o nível de serviços de tecnologia para diversos cenários ou em diversas escalas, adaptando-os ao crescimento da empresa, culminando em uma gestão de informações mais eficiente.

Após a **compilação dos dados**, eles encontram-se separados por:

- Categorias
- Finalidades
- Característica

Uma vez separadas, facilitam a organização do processo de trabalho para o desenvolvedor.

Flins

Por ser uma prática, a IC necessita de um grande empenho e de disciplina da equipe, de modo que todos façam o check-in de nova funcionalidade no tronco de desenvolvimento em mudanças pequenas e incrementais e concordem que o ofício de maior predileção no projeto é consertar qualquer mudança que rompa a aplicação.

Verificando o aprendizado

Questão 1

Sobre os requisitos para a implementação da integração contínua, fazem parte:

I. Compilação automatizada.

II. Controle de versão.

III. Aceitação da equipe.

IV. Linha de comando.

V. Pequenos projetos.

A

V.

B

I e V.

C

I, II e V.

D

I e V.

E

I, II, III e IV.

A alternativa E está correta.

Para que a integração contínua seja implementada, é necessário que o projeto esteja em um único repositório, independentemente do tamanho do projeto. E mais: o processo de compilação deve ser feito com linhas de comando com mecanismo compatível com compilação, testagem e instalação da aplicação de maneira automatizada, além de ser necessária a aceitação da equipe.

Questão 2

Julgue as sentenças:

( ) Todo o projeto deve estar em um único repositório, incluindo códigos, testes, *scripts* de bancos de dados, de compilação e implantação.

( ) É importante que o mecanismo escolhido para compilação automatizada seja praticável para qualquer um compilar, testar e instalar a aplicação de maneira automatizada a partir da linha de comando.

( ) Todos da equipe devem concordar que o ofício de maior predileção no projeto é consertar qualquer mudança que rompa a aplicação.

A

V, V, V.

B

F, F, F.

C

V, F, V.

D

F, V, F.

E

V, V, F.

A alternativa A está correta.

Para que a integração contínua seja implementada, é necessário primeiramente que haja aceitação da equipe e que a compilação seja automatizada e aplicável a todas as etapas de produção do software.

Gerenciamento de projetos

Gerenciamento de Projetos e DEVOPs

Neste vídeo, abordaremos o que é gerenciamento de projetos e porque esta ferramenta é uma prática sugerida para integração contínua.

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

Agilidade é imprescindível no gerenciamento de projetos e desenvolvimento de software, ajudando as equipes a satisfazer os clientes com rapidez. As equipes ágeis entregam o trabalho em progressos menores e não precisam esperar por uma data única para um lançamento enorme. São avaliados frequentemente requisitos, planos e resultados, permitindo que as equipes respondam ao feedback e façam alterações conforme necessário.

Para que seja possível a realização desse gerenciamento, é necessário planejar e estabelecer um **fluxo de trabalho**, incluindo **quatro fases**:

- Fazer
- Em andamento
- Revisão de código
- Concluído

O projeto de grande escala deve ser **dividido** em **tarefas menores** e deve ser dado **feedback das mudanças** à equipe à medida que estas progridem. Além disso, também é necessário fazer um **planejamento**, **acompanhamento** e uma **medição** do desdobramento do trabalho por meio do Scrum e Kanban.

Switch to the left

Mudar para a esquerda, ou *switch to the left*, significa levar os testes para seus processos de desenvolvimento de código desde o início.

Ao invés de enviar várias alterações para a equipe de qualidade para cada teste em separado, uma variedade de testes é realizada ao longo do processo de codificação para que os desenvolvedores possam corrigir erros ou melhorar a qualidade do código enquanto trabalham na seção relevante da base de código. A prática de integração contínua e implementação sustenta a capacidade de mudar para a esquerda.

Ferramentas para criação e implementação da automação

A implantação de DevOps requer as ferramentas certas para cada fase do seu ciclo de vida, visando à possibilidade de melhorias de qualidade do software e da velocidade de entrega.

A integração e a entrega contínuas permitem mesclagem do código com regularidade no repositório principal. Em vez de verificar o código manualmente, a integração contínua automatiza esse processo, desde a organização feita em lotes em um momento específico até confirmações frequentes. O teste automatizado também é primordial para práticas bem-sucedidas de DevOps e pode incluir testes de ponta a ponta, de unidade, de integração e de desempenho.

Monitoramento do pipeline e dos apps do DevOps

Pipeline é um termo em inglês que, traduzido, significa **canalização**. Trazendo-o para o mundo virtual, o *pipeline* é uma "canalização" que segmenta dados e incrementa o rendimento de um sistema digital.

Segmentar os cálculos melhora a frequência de trabalho. A saída de um fluxo de dados implica a entrada de outros, assim as fases encadeiam-se como uma canalização, conseguindo agilizar o fluxo por meio desse *pipeline*, evitando interrupção da compilação ou que um teste com falha cause atrasos desnecessários.

> Além de trazer essa segurança, a automação otimiza o desenvolvimento.

O *pipeline* também pode ser introduzido na infraestrutura de produção e é de fácil instalação.

Observabilidade e feedback contínuo

Os **três pilares** da observabilidade são:

- Logs
- Rastreamento
- Métricas

Logs são registros gerados com datação de eventos ao longo do tempo de criação do aplicativo ou software, trazendo informações importantes sobre o funcionamento desse app ou software. Além disso, são gerados pela maioria dos componentes e aplicativos do sistema. Existem três formas de *logs*:

- Texto
- Estruturado
- Binário

Os *logs* de texto são os mais comuns. Os logs estruturados são os mais utilizados ultimamente e são emitidos no formato JSON. E os logs binários podem ser usados como *frontend* para a ferramenta *tcpdump*.

A estrutura dos rastreamentos, ou *tracing,* se parece muito com os logs de eventos capazes de mostrar para os engenheiros de software, ou a equipe de desenvolvimento no geral, o caminho percorrido por uma requisição ou pela estrutura dela, permitindo assim entender os efeitos das execuções desta dentro do aplicativo. Dessa forma, diz-se que os rastreamentos identificam pontos específicos dentro do aplicativo, como: *threads*, continuações, fronteiras RPC, chamadas de função, entre outros que, no caminho de uma requisição, podem representar um *thread* do sistema operacional ou um *fan-out*.

Comentário

As métricas são representações numéricas de dados medidos em intervalos de tempo. Elas podem usar modelagem matemática e análise preditiva para entender o comportamento de um sistema em espaços de tempo, podendo ser avaliados tanto no presente, como no futuro. As métricas incluem reserva ou uso de CPU/RAM, espaço em disco, conectividade de rede, dentre outros.

Podemos concluir que observabilidade nada mais é que usar as três fontes de informação em conjunto, a fim de descobrir e prever o funcionamento de um sistema complexo.

Já o feedback contínuo é uma conversa que pode ocorrer pessoalmente ou on-line, de modo recorrente. Garante não apenas que os membros da equipe tenham todas as informações necessárias para realizar o trabalho em tempo hábil, como também que os resultados de testes de código claros e completos sejam disponibilizados para os desenvolvedores rapidamente. Assim, a equipe fica ciente de quaisquer falhas de produção, deficiências de desempenho ou erros relatados, sendo capaz de otimizar a velocidade e a qualidade.

Verificando o aprendizado

Questão 1

O que é imprescindível nas práticas sugeridas de DevOps?

A

Agilidade no desenvolvimento do software.

B

Finalização imediata de todo o software.

C

Equipes lentas, mas qualificadas.

D

Data única para o lançamento do software contratado pelo cliente.

E

Equipes engessadas que não façam alterações frequentes no software.

A alternativa A está correta.

É imprescindível a agilidade, para que as equipes satisfaçam os clientes com rapidez, entregando o trabalho em progressos menores, ao invés de esperar por uma data única para um lançamento enorme.

Questão 2

O que é avaliado frequentemente no gerenciamento de projetos?

A

Somente requisitos.

B

Somente planos.

C

Somente resultados.

D

Requisitos, planos e resultados.

E

Requisitos e resultados.

A alternativa D está correta.

São avaliados frequentemente os requisitos, planos e resultados, permitindo que as equipes deem feedbacks e façam alterações no software conforme necessário.

Benefícios da implementação dos testes na cultura DevOps

Teste de Integração em DEVOPs

Neste vídeo, apresentaremos o que é teste de integração e porque esta ferramenta é importante para integração contínua.

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

Teste de software é um processo importante no desenvolvimento, sendo fundamental para verificar a correção, a qualidade e o desempenho. Por meio dos testes de software, os desenvolvedores identificam erros eventuais.

Por isso, poupam tempo e dinheiro através da redução de custos de desenvolvimento e manutenção de software. Além disso, garantem estabilidade para desenvolver novos recursos, visto que podem ser utilizados em recursos já entregues e também garantem que os softwares funcionem como o esperado, reduzindo a introdução de novos *bugs* e, portanto, aumentando a qualidade do produto.

Os testes reduzem custos de manutenção. Além disso, cada vez mais, as empresas entram na corrida para obter respostas mais rápidas para disponibilizar seus softwares aos seus clientes. Por esse motivo, metodologias ágeis, como o DevOps, vêm sendo amplamente utilizadas em empresas de diversas áreas. Essa implementação vem crescendo exponencialmente, como aponta a pesquisa da Faun Community, segundo a qual, até 2030, 90% das empresas terão adotado DevOps.

> Comentário
>
> Junto com essa metodologia, os testes estão repercutindo tanto na qualidade como na agilidade. Com o intuito de responder aos clientes, de forma cada vez mais rápida, os testes automatizados são os principais atores, uma vez que são capazes de aumentar em 66% a aceleração da velocidade de entrega e em 61% o aumento de qualidade. Também são capazes de diminuir em 48% os riscos de algo dar errado no software, aumentar em 46% a satisfação do cliente, aumentar em 34% a visibilidade do fluxo de valor para os usuários, diminuir os custos de TI em 34% e aumentar em 26% a garantia de conformidade e/ou governança.

Apesar do crescimento da procura por DevOps e seus benefícios quando aliados a testes, muitos ainda possuem a falsa impressão de que não se passa de “uma história da carochinha”, dizendo até que ele atrapalha o desempenho do desenvolvimento. E essa falsa sensação pode se dar por conta da falta de testes automatizados consistentes.

Introduzindo o uso de testes automatizados

Primeiramente, é importante a criação de metas para que a implantação seja feita gradualmente e a chance de sucesso aumente, e assim criar um conjunto de suítes básicas de testes para o sistema. Dessa forma, diminui-se a chance de se cometer erros em outros projetos já em andamento, e o operador consegue calcular as ferramentas escolhidas e o quão produtivo seu time é ao desenvolver testes.

Comentário

Outro ponto importante é a adoção de test driven development (TDD) no início do processo, pois ele “escreve” um teste que será reprovado e, em seguida, “escreve” um código que será aprovado por esse teste, podendo ser usado para a correção de bugs.

Assim, o teste que replique o *bug* deve ser criado. Em seguida, o teste vai quebrar, então, é o momento de fazer a correção do *bug* para, em seguida, o teste ser executado novamente. Esse processo evita que o mesmo *bug* ocorra futuramente, de modo que o TDD pode passar a ser implementado em outras etapas do desenvolvimento.

As etapas de produção de um software e a automação dos testes

Os testes podem ser manuais ou automatizados. No entanto, a automação é a cereja do bolo para o uso de DevOps, pois integra os processos entre as equipes de desenvolvimento de software e de TI e auxilia na criação, testagem e no lançamento de software com mais rapidez e segurança, além de disponibilizar recursos. A automação é necessária para que seja possível sua execução em cada alteração feita ao repositório principal, pois, dessa forma, é possível prever problemas e eliminá-los, não causando interrupções para a equipe.

Atenção

Ao automatizar o pipeline, mas não os testes, as mudanças aplicadas no software geram muito mais trabalho, por entrarem no sistema de forma rápida, mas com bugs, obrigando o desenvolvedor a retornar a versões anteriores, não sendo possível commitar a feature, pois o sistema e o build não chegam ao líder técnico. Como resultado, consegue-se um cliente insatisfeito ou satisfeito parcialmente.

Normalmente, as etapas seguidas para a formação/produção de um software são as seguintes:

- Plano
- Código
- Construir
- Testar
- Liberar
- Implantar
- Operar
- Monitorar

Todas essas fases passam por desenvolvedores e operadores. Mas, ao utilizar testes automatizados, além dessas fases, são acrescentadas ao processo as ferramentas e as práticas de DevOps que vão otimizar o processo. A base de tudo são, justamente, os testes unitários, funcionais, de interface gráfica e integrados, garantindo a qualidade.

Vale ressaltar que os testes automatizados são empregados não só em empresas que possuem alto grau de “maturidade” em DevOps, mas também são exigidos naquelas que começaram a empregar essa metodologia há pouco tempo. O que muda são os tipos de testes.

Exemplo

No caso das empresas que se enquadram em estágio inicial de maturidade (Initial), os testes utilizados são os testes unitários automatizados (automated unit tests) na linha “Tests &; Verifications”. É esse o primeiro passo dentro da metodologia DevOps que permite a geração de releases frequentes e o lançamento de versões com menores índices de erros nos scripts.

Ressalta-se que, sem a inserção desse primeiro nível de maturidade com os testes unitários automatizados, não é possível migrar para um próximo nível. Ademais, na falta desses testes, o processo se torna mais custoso e de baixa confiabilidade.

Práticas de automação que apoiam o DevOps

Vejamos cada uma das práticas a seguir:

ATDD (*acceptance tests drive design*)

É capaz de automatizar os testes de aceitação utilizando o *Cucumber*. Ele reduz a quantidade de trabalho investida e aumenta a qualidade do sistema.

Issue tracking

Associa um teste automatizado a uma *feature* específica. Assim, é possível ver quais testes estão sendo realizados, quais *features* cada um está testando e se o teste expressa o que a *issue* está descrevendo, entre outros pontos. Isso aumenta consideravelmente a qualidade do seu projeto, otimizando seus custos e o seu tempo de entrega.

Previsibilidade

Nada mais é do que trabalhar com planejamento, por meio da utilização de testes automatizados, tendo a noção de quantos e quais testes serão necessários para cada *feature*, a fim de ter como resultados uma menor incidência de *bugs* no geral e uma entrega mais rápida e eficaz, dentro da previsão de tempo estipulada, o que só é possível em equipes de alto rendimento.

Refactoring

É a melhoria do design do código, mantendo seu comportamento. Por meio do *refactoring,* também é possível mudar o código, diminuindo a chance de inclusão de um *bug*, sem receio de fragmentar o sistema.

Os tipos de testes e sua implementação

Neste vídeo, apresentaremos os principais tipos de teste e em que sequência devem ser implementados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Tipos de teste

Apresentaremos a seguir cada tipo de teste:

Testes de unidade

São os primeiros tipos de testes automatizados a serem realizados por meio de níveis de métodos. Têm propósito restrito e verificam o comportamento de métodos ou funções individuais. São as menores partes testáveis do software, ou seja, são usados em pequenos trechos e implementados na etapa de desenvolvimento. O programador tem respostas *in-time* de cada pequeno trecho que está programando.

Testes funcionais

Por meio deles, o sistema é testado de forma contrária aos requisitos de função do software, como se fosse uma contra-prova de função, ou o caminho inverso do que o software será capaz de fazer, varrendo grandes trechos do sistema e garantindo que o código não seja quebrado enquanto o desenvolvedor mexe em parte dele. Assim, são responsáveis por garantir a entrega fiel dos requisitos solicitados pelo cliente. São chamados também de *end-to-end*.

Testes de Interface do usuário

Também chamados de *graphical user interface*, interagem com o computador por meio do uso de imagens. Por exemplo, alguns testes da empresa Google solicitam que o usuário marque as imagens que contêm semáforos apresentadas em forma de mosaico na tela. Esses testes garantem que o aplicativo funcione bem desde a perspectiva do usuário.

Testes de integração ou testes integrados

São chamados de *black-box* e validam a integração de módulos ou sistemas, garantindo que vários componentes se comportem bem juntos, podendo envolver diversas classes, como: teste da integração com outros serviços. Sua execução é um pouco mais complexa, visto que dependem de ter pelo menos dois sistemas no ar concomitantemente. São usados no sistema como um todo, e não só na etapa de desenvolvimento. Neles, se usam técnicas de *mocking*.

Testes de aceitação

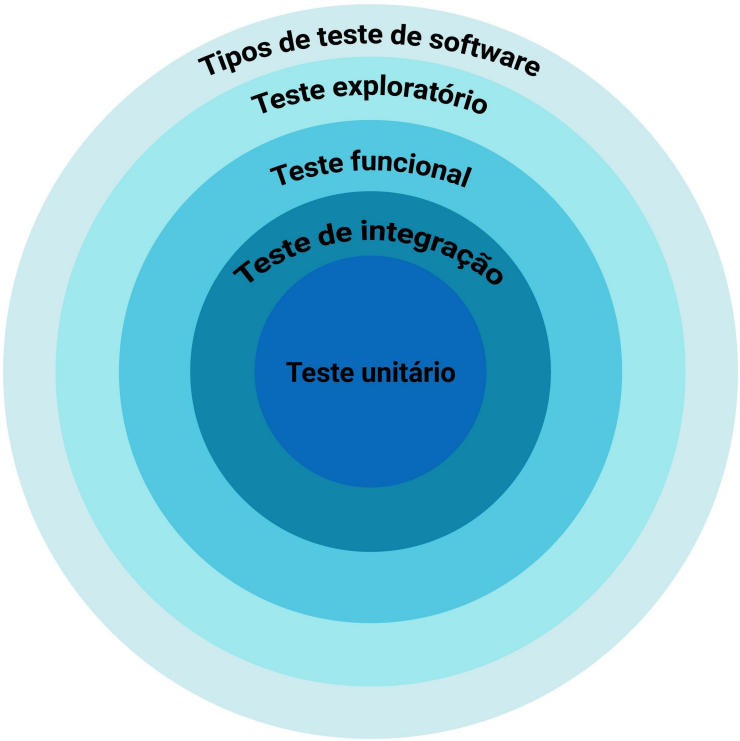
Parecidos com os testes de integração, mas se concentram nos casos de negócios em vez dos próprios componentes.

Esses testes funcionam de forma independente. Por exemplo, o teste funcional, que pode ser rodado independentemente de um teste de interface, que é algo pesado, mais demorado e de alto custo. No entanto, ao se pensar no sistema, sua arquitetura deve permitir a aplicação concomitante de testes.

Uma informação importante sobre os testes de integração e os funcionais é que, por conta de serem utilizados em etapas diferentes de produção do software, auxiliam na manutenção do design, mantendo o comportamento do software, sendo considerados eficientes para a prática de *refactoring*.

Sequências de testes a serem implementados

A imagem a seguir apresenta a sequência de testes de software para implementação.



Tipos de teste de software.

A seguir, falaremos a respeito de alguns tipos de testes:

Testes de módulos atômicos

Cada módulo é testado individualmente.

Testes de integração

São a montagem e integração dos módulos básicos compondo um pacote de software.

Testes de validação

São a aplicação de testes funcionais, baseados na especificação de requisitos funcionais, comportamentais e de desempenho. Os critérios de validação definidos após a fase de análise de requisitos devem ser testados.

Testes de sistema

O software validado é combinado com outros elementos do sistema. Esses testes verificam se todos os elementos se combinam adequadamente e se a função/desempenho global do sistema é conseguida.

Testes de aceitação

Consistem no processo de comparar o programa com seus requisitos iniciais e as necessidades dos usuários finais. É usualmente realizado pelo usuário final. Também denominado de Alfa-Teste, quando feito pelo desenvolvedor ou Beta-Teste quando feito pelo usuário final.

Pontos fundamentais para a Implementação dos testes automatizados

O primeiro ponto a ser avaliado é uma plataforma **forte**, capaz de gerar a automatização dos testes de forma eficiente. É fundamental estruturar essa plataforma, por meio de planejamento, para que seja possível dar o próximo passo e minimizar impactos significativos.

Comentário

É necessário continuar os processos a fim de obter o processo de entrega conforme esperado. Desse modo, a plataforma deve conter as funções de gravar os testes de unidades, fazer análise de qualidade dos testes viáveis a serem implementados e obter essas ações já citadas para criar uma compilação funcional do software após todos os testes.

Também é eficaz utilizar o *behaviour driven development* (BBD), ou desenvolvimento orientado por comportamento, que objetiva orientar os códigos por meio dos testes, focando nas linguagens e interações. O BBD é um tipo de método ágil e capaz de melhorar o software por meio dos testes, fazendo a integração dos pedidos solicitados pelo cliente no negócio com a linguagem de programação.

Assim, pode-se dizer que o BBD melhora a comunicação entre as equipes, assume o papel de compartilhador de conhecimento, reconhece a documentação dinâmica e assegura uma visão completa do processo de desenvolvimento, operações e testes do software, além de incentivar a colaboração contínua dos proprietários de produtos e desenvolvedores.

Atenção

Para que haja integração dos módulos, é usada preferencialmente a abordagem botton-up que segue as etapas de teste de módulos, os quais são agrupados em clusters e testados.

Os drives são removidos e, quando há necessidade, são realizados testes de funções de controle; posteriormente, é feita uma integração *top-down*, que consiste em aplicar a árvore de módulos primeiramente em profundidade e/ou em largura. Em seguida, são utilizados os *stubs* para os módulos ainda não testados e, após os testes, estes serão substituídos por módulos reais.

À medida que esse processo acontece, novos *stubs* são criados, até a chegada de módulos de nível mais baixo. A introdução de um novo módulo pode gerar erros, então os testes de regressão precisam ser aplicados para executar novamente testes já feitos, para que se tenha certeza de que as modificações feitas não produzam problemas no software.

Esses testes de regressão podem voltar-se tanto para as funções do software como para funções com potencial para terem sido abaladas por mudanças ou componentes modificados. No caso da abordagem de botton-up, o programa só funcionará após o último teste ter sido executado.

Comentário

A capacitação de equipes de garantia de qualidade também se faz útil, tendo em vista que esse grupo, dentro da produção do software, é o responsável por garantir a qualidade das soluções, estando diretamente ligado aos testes automatizados. No entanto, essa qualificação deve acontecer em outros setores também, pois os colaboradores precisam ter conhecimento sobre desenvolvimento de softwares, entre outras questões, atuando de forma considerável no ciclo de vida da solução.

Vale ressaltar ainda que, em alguns momentos, pode-se renunciar aos testes automatizados, sim! Desse modo, o processo é realizado de forma manual. Porém, a análise de quando usar os testes dessa forma deve ser feita pelo especialista em qual aplicação está sendo usada, entre outros.

Evitando erros na implementação dos testes automatizados

Conheça a seguir os **erros mais comuns** na implementação de testes automatizados em DevOps a fim de evitá-los:

- Os feedbacks lentos ou não confiáveis podem ocorrer se a automação não oferecer informações que norteiam uma intervenção, como no caso de uma quebra de *pipeline* de testes automatizados, em que razão da quebra não é confiável pelo feedback.
- Falta de integração do código com o repositório diariamente, acumulando o código local.
- Rodar os testes em ambientes compartilhados, local onde pode haver alteração por outras demandas, gerando testes não fidedignos e podendo causar perda de confiança dos times nos processos de automação.

As consequências dos erros desses testes podem gerar consequências para a equipe de DevOps, como: o aumento da carga de trabalho, junto com divisões dentro das equipes, havendo quebra de toda a estrutura da metodologia DevOps. Além de interrupções no processo e pouco espaço para inovações e dificuldade na detecção de erros dentro do software, atrasando a entrega para o cliente.

Verificando o aprendizado

Questão 1

Quais os benefícios da implementação de testes na cultura DevOps?

A

Identificar erros, poupar tempo e dinheiro e fazer manutenções no software.

B

Aumentar qualidade e gastos do desenvolvimento do software.

C

Identificar erros do software.

D

Fazer manutenções longas no software.

E

Diminuir gastos das manutenções do software.

A alternativa A está correta.

A implementação de testes na cultura DevOps é fundamental para verificar a correção, qualidade e desempenho. Por meio desses testes, os desenvolvedores identificam erros eventuais e poupam tempo e dinheiro quando reduzem custos de desenvolvimento e manutenção de software.

Questão 2

Sobre a implementação de testes na cultura DevOps, assinale a resposta certa.

( ) Aumentam a qualidade do software, aumentando a entrada de novos *bugs*.

( ) Ajudam os desenvolvedores a identificar erros.

( ) Garantem estabilidade para desenvolver novos recursos dentro do software.

A

V, V, F.

B

V, F, F.

C

F, V, V.

D

F, F, V.

E

V, F, V.

A alternativa C está correta.

O uso de testes dentro da cultura DevOps garante que os softwares funcionem como o esperado; reduz a introdução de novos *bugs*, aumentando a qualidade do produto, e assim os custos são diminuídos.

Considerações finais

A integração contínua DevOps é uma das chaves principais para a construção de softwares em equipe, pois automatiza o processo de criação de softwares por meio de mudanças de código em um aplicativo. Seu objetivo é otimizar tempo, agradar o cliente e o usuário final e gerar um software seguro e passível de ser atualizado, não perdendo a qualidade. No entanto, para que essa metodologia seja implementada, é fundamental que a equipe de desenvolvedores esteja disposta a trabalhar em conjunto, no qual todos cooperam em prol do resultado/produto, mantendo responsabilidade compartilhada e transparência para que seja possível um feedback mais rápido. Além disso, é necessária a implementação de testes automatizados em todo o processo de criação de um software, em que cada teste atua em um trecho diferente ou mais de um teste no mesmo trecho, a fim de evitar erros durante o desenvolvimento do software e após sua entrega para o cliente final.

Podcast

Para encerrar, ouça um resumo dos principais tópicos abordados.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Leia o livro **Accelerate: building and scaling high performing technology organizations**, que aborda como os altos níveis de confiança e colaboração resultam em tomadas de decisão de alta qualidade e níveis ainda mais altos de satisfação no trabalho.

Referências

AMAZON WEB SERVICES. **O que significa Integração Contínua.** Amazon web services, 2022.

AUTOMAÇÃO de Compilação. Wikipedia, 2020. Consultado na internet em: 25 out. 2022.

CINCO passos para implementar DevOps. InfraNews Telecom. Consultado na internet em: 25 out. 2022.

INTEGRAÇÃO Contínua em .NET. DevMedia, 2021. Consultado na internet em: 25 out. 2022.

INTRODUÇÃO ao DevOPs. RedHat, 2018. Consultado na internet em: 25 out. 2022.

MIRANDA JUNIOR, P. O. de. **Estratégias para teste**. Teses Tecnologia. Consultado na internet em: 25 out. 2022.

O AMBIENTE DevOps e a Containerização. Deal, 2019. Consultado na internet em: 25 out. 2022.

O QUE é pipeline. Conceito de, 2020. Consultado na internet em: 25 out. 2022.

O QUE uma empresa precisa para realizar a implementação da integração contínua. HNZ, 2021. Teses Tecnologia. Consultado na internet em: 25 out. 2022.

OS TRÊS pilares da observabilidade em sistemas distribuídos. Elven, 2021. Consultado na internet em: 25 out. 2022.

PITTET, S. Como configurar a integração contínua. Desenvolvimento de softwares. Atlassian, 2022. Consultado na internet em: 25 out. 2022.

VEJA como a compilação de dados automáticos pode ajudar sua empresa a crescer. Biti9. Consultado na internet em: 25 out. 2022.