Aula 00 - (Prof. Paolla Ramos e Raphael Lacerda)

TCE-PI (Auditor de Controle Externo - Área TI - Infraestrutura e Segurança) Desenvolvimento de Software - 2024 (Pós-Edital)

Autor:
Felipe Mathias, Paolla Ramos, Raphael Henrique Lacerda

20 de Outubro de 2024

Índice

APRESENTAÇÃO

PROF. PAOLLA RAMOS

FORMADA EM SISTEMAS DE INFORMAÇÃO PELA UNIVERSIDADE FEDERAL DE OURO PRETO,
PÓS-GRADUADA EM SISTEMAS DE INFORMAÇÃO
DIREITO TRIBUTÁRIO
DIREITO ADMINISTRATIVO
AUDITORA FISCAL ESPECIALISTA EM TI.

Olá, pessoal!! Meu nome é Paolla Ramos, sou Auditora Fiscal especialista em TI do ISS-Aracaju. Trabalhar nesse fisco incrível tem sido uma experiência fantástica!!

Pessoal, eu sou uma pessoa normal, assim como vocês. No início, achava que conquistar a aprovação em um concurso de alto nível era quase impossível, até que provei o contrário! Querem saber qual foi o segredo? Foi o hiper foco, galera! Não existe uma fórmula mágica, e eu nunca fui considerada "superinteligente" ou a primeira aluna na turma. No entanto, sempre fui MUITO DETERMINADA, PERSISTENTE.

A equipe de TI e eu estamos aprimorando nossas aulas de forma gradativa para oferecer o melhor conteúdo possível. Sabemos que o estudo pode ser complexo, especialmente por meio de livros eletrônicos, por isso, recomendo estudar em conjunto com as vídeo-aulas.

Além disso, informo que estamos trabalhando na atualização dos cursos neste exato momento! Estamos refazendo a formatação, adicionando questões e diagramas, entre outros aprimoramentos. Gradualmente, os cursos ficaram mais completos e aprofundados. E, para acompanhar as tecnologias mais recentes, novas videoaulas também estão a caminho.

Caso surja alguma demanda, não hesitem em contactar no fórum. Se preferirem, também podem entrar em contato pelo Instagram **@prof.paollaramos**. Eu amo ajudar os alunos e estou disponível para esclarecer quaisquer dúvidas que possam surgir.

A minha missão aqui é dar o meu melhor para ajudar cada um de vocês a conquistar a aprovação também! Podem contar comigo sempre que precisarem.

Então, minha ideia aqui é fazer o meu melhor para que você também consiga ser aprovado! Sempre que precisar, pode contar comigo. Meu instagram é:

ESTRATÉGIA FLASHCARDS

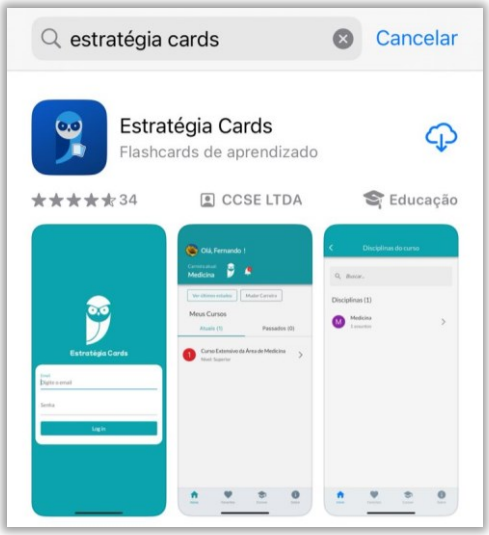
📚 Você tem dificuldade de estudar, memorizar e revisar os conteúdos que estuda em nossas aulas? Então nós temos a ferramenta perfeita para você!

Apresentamos o **Estratégia Cards**: app de flashcards que vai revolucionar sua forma de **estudar** e **revisar** conteúdos de provas de concurso público. Com nossa tecnologia inovadora e interface amigável, você dominará os tópicos mais complexos de maneira eficiente e divertida.

✨ Recursos do Estratégia Cards:

Curadoria de Flashcards	Flashcards criados e revisados por professores especializados em cada área, com qualidade e voltados para concursos públicos.
Flashcards Personalizados	Crie seus próprios flashcards, cobrindo os principais tópicos e matérias dos concursos públicos.
Repetição Espaçada	Técnica de aprendizagem que envolve revisar informações em intervalos crescentes para melhorar a retenção de longo prazo e combater o esquecimento.
Estatísticas Personalizadas	Visualize graficamente o percentual de acertos, erros ou dúvidas dos decks estudados.
Modo Offline	Estude em qualquer lugar, mesmo sem conexão à internet, fazendo o download dos decks.
Estudo por Áudio	*Está dirigindo ou fazendo esteira e quer continuar estudando?* Basta utilizar a opção "Escutar".
Decks Favoritos	Você pode escolher decks específicos como favoritos e visualizá-los em uma aba separada do app.
Opções de Estudo	Você poderá estudar todos os cards de um deck; ou apenas os que você errou; ou apenas os que você não estudou ainda; entre outras opções.

E como eu consigo baixar?



É muito fácil! Basta pesquisar por "Estratégia Cards" na loja oficial do seu smartphone.

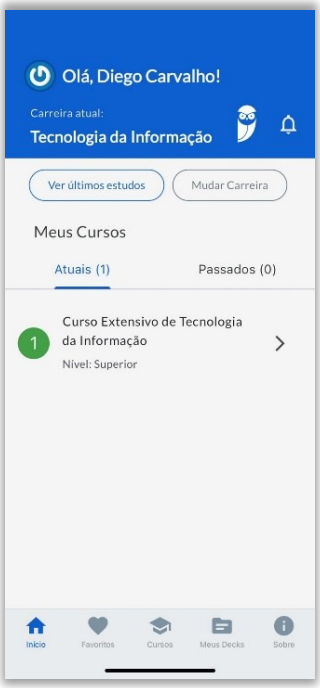
Se você tiver um Android, basta acessar a **Google Play**;

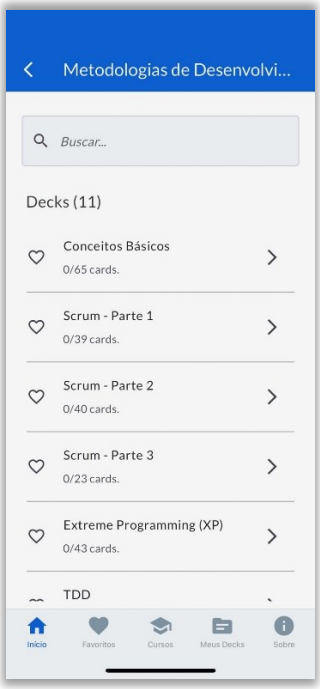
Se for tiver um iPhone, basta acessar a **App Store (iOS)**.

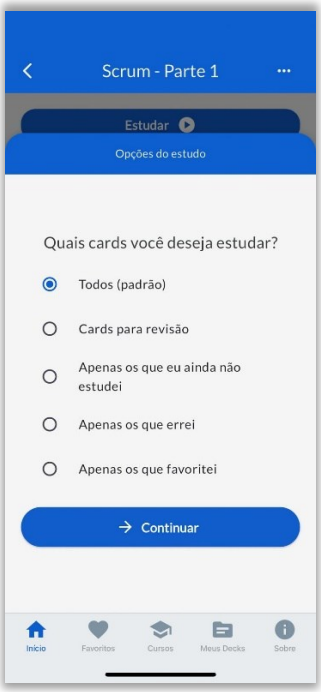
E para acessar?

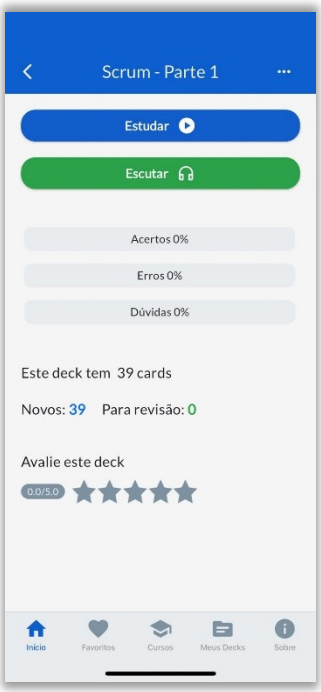
Para acessar, basta ter uma conta no Estratégia Concursos. Em seguida, utilize suas credenciais de login e senha para acessar o aplicativo. Por fim, acessa a carreira de Tecnologia da Informação.

Como utilizar o app:







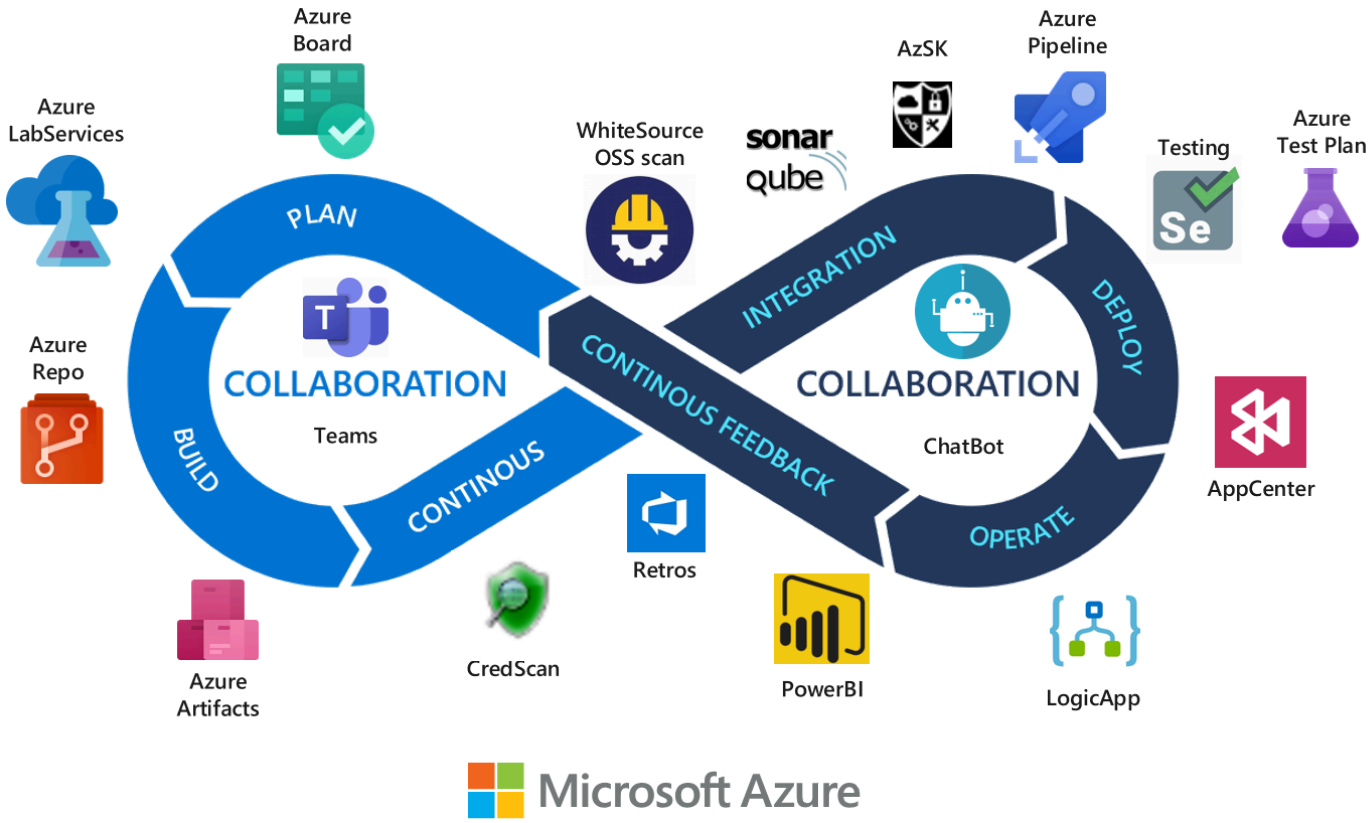


Sumário

Apresentação da Aula

Olá, pessoal! Hoje vamos falar sobre DevOps e como essa abordagem tem transformado a forma como desenvolvemos e entregam software. DevOps é uma cultura e conjunto de práticas que integra os times de desenvolvimento (Dev) e operações (Ops), buscando maior colaboração, automação e eficiência em todo o ciclo de vida do software.

Com o DevOps, a ideia é que os desenvolvedores e as equipes de operações trabalhem de forma conjunta desde o início do projeto até a sua implantação e manutenção. Isso significa que a colaboração é incentivada em todas as etapas do processo, permitindo uma comunicação mais eficiente e uma entrega mais rápida e confiável do software.



Uma das principais características do DevOps é a automação. Através de ferramentas e práticas automatizadas, é possível acelerar o desenvolvimento, testes, implantação e monitoramento do software. Por exemplo, é comum utilizar ferramentas de integração contínua (CI) e implantação contínua (CD) para automatizar os processos de construção, testes e implantação do software em diferentes ambientes.

Além disso, o DevOps também promove a cultura de feedback contínuo e melhoria. Isso significa que os times estão sempre buscando formas de aprender com as experiências, medindo e monitorando o desempenho do software em produção e realizando ajustes e melhorias constantes.

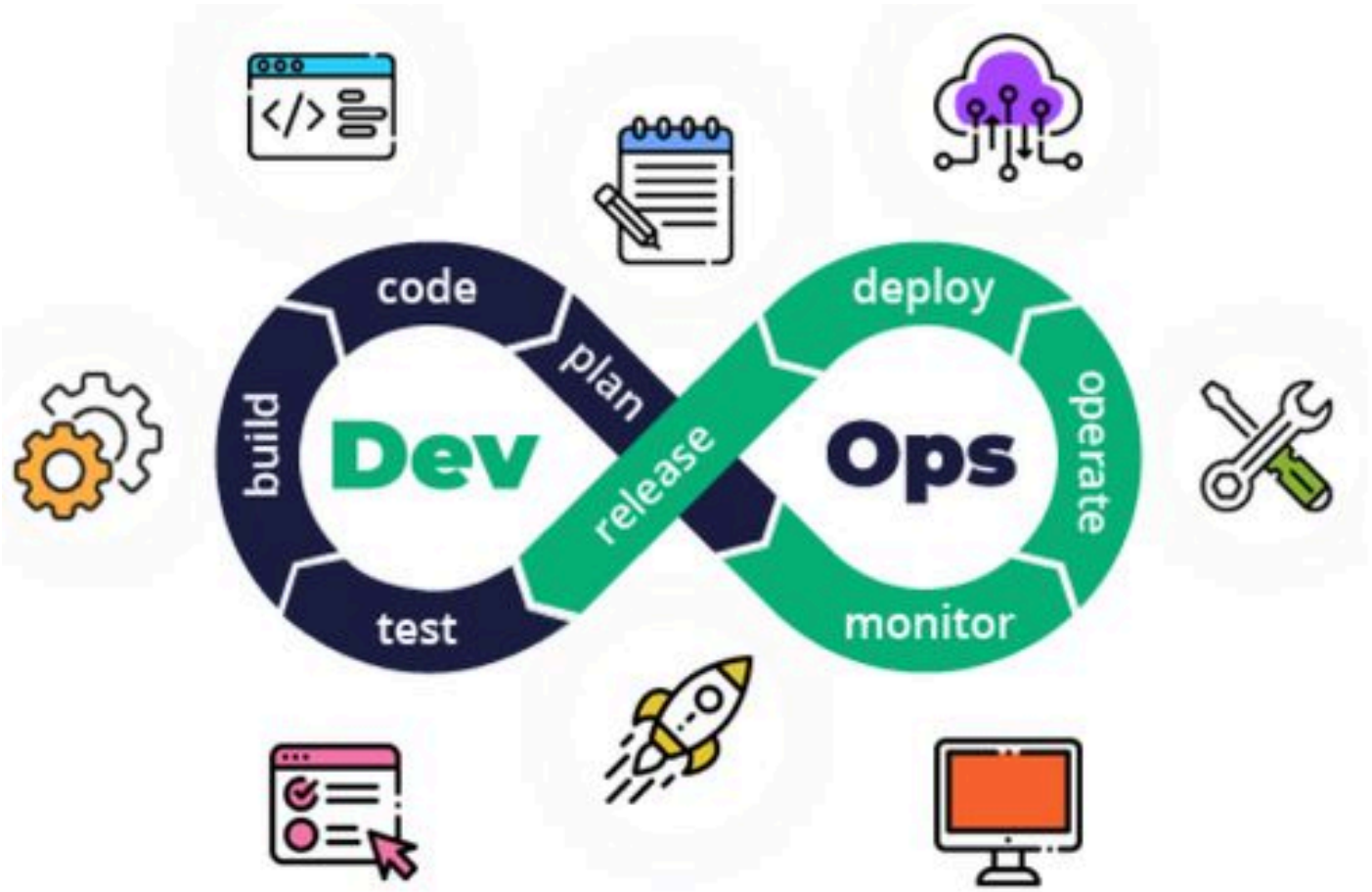
Uma das principais vantagens do DevOps é a capacidade de entregar software de forma mais rápida e frequente. Através da automação e colaboração, é possível reduzir o tempo entre o desenvolvimento de uma funcionalidade e sua disponibilização para os usuários. Isso permite que as empresas sejam mais ágeis, respondendo rapidamente às demandas do mercado e obtendo feedback mais cedo.

Outro benefício importante do DevOps é a maior estabilidade e confiabilidade do software. Ao integrar as equipes de desenvolvimento e operações, é possível antecipar e resolver problemas de forma mais eficiente, garantindo que o software seja implantado de maneira segura e confiável. Além disso, a automação dos processos reduz a chance de erros humanos e ajuda a manter a consistência entre os ambientes de desenvolvimento, teste e produção.

No entanto, implementar o DevOps não é apenas uma questão de adotar ferramentas e práticas específicas. É necessário promover uma mudança cultural dentro das equipes, incentivando a colaboração, a comunicação e a busca constante pela melhoria. Também é importante investir em treinamento e capacitação para que os membros das equipes possam adquirir as habilidades necessárias para trabalhar nesse novo ambiente colaborativo. Agora é hora de debruçar nos estudos e aprender sobre DevOps! 😃

DevOps

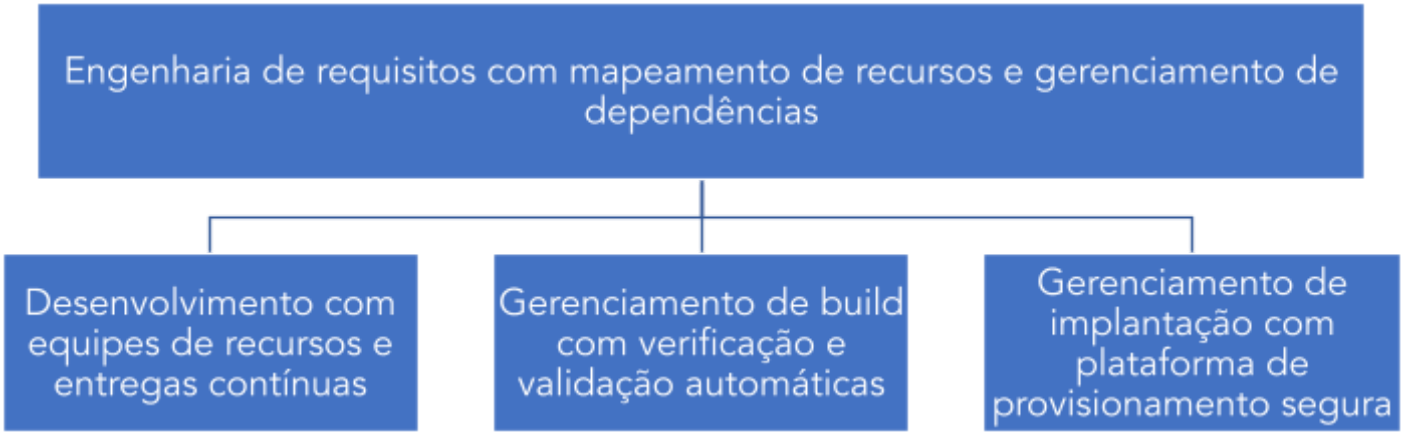
Conceitos Básicos



De acordo com Christof Ebert, DevOps é sobre desenvolvimento rápido e flexível e provisão de processos de negócios. Ele integra de forma eficiente o desenvolvimento, entrega e operações, facilitando uma conexão ágil e fluida desses silos tradicionalmente separados.

DevOps é uma abordagem que integra desenvolvimento e operações para tornar o processo de entrega de software mais eficiente. Em vez de ter equipes separadas trabalhando em suas próprias áreas, o DevOps promove a colaboração entre desenvolvedores, profissionais de qualidade e operações. Assim, DevOps integra os dois mundos do desenvolvimento e das operações, utilizando o desenvolvimento automatizado, implantação e monitoramento da infraestrutura. É uma mudança organizacional em que, em vez de grupos distribuídos em silos executando funções separadamente, equipes multidisciplinares trabalham em entregas contínuas de recursos operacionais. Essa abordagem ajuda a entregar valor de forma mais rápida e contínua, reduzindo problemas decorrentes da falta de comunicação entre os membros da equipe e acelerando a resolução de problemas.

DevOps é uma mudança cultural em que as equipes adotam uma cultura de engenharia de software, fluxo de trabalho e conjunto de ferramentas que elevam os requisitos operacionais ao mesmo nível de importância que a arquitetura, design e desenvolvimento. Os desenvolvedores que criam e executam o software têm uma maior compreensão dos requisitos e necessidades dos usuários. Os valores de uma cultura de DevOps incluem o aumento da transparência, a comunicação e a colaboração entre equipes. O esquema 1 mostra o processo genérico, que tem como objetivo integrar melhor os processos de negócios de desenvolvimento, produção e operações com a tecnologia adequada.



Em vez de permanecer com conceitos de processos altamente artificiais que nunca fluirão, as organizações estabelecem uma entrega contínua com pequenas atualizações. Empresas como Amazon e Google lideraram essa abordagem, alcançando tempos de ciclo de minutos. Obviamente, o tempo de ciclo alcançável depende das restrições ambientais e do modelo de implantação; um único serviço em nuvem é mais fácil de facilitar do que entregas reais de produtos reais.

DevOps pode ser aplicado a modelos de entrega muito diferentes, mas deve ser adaptado ao ambiente e à arquitetura do produto. Nem todos os produtos facilitam a entrega contínua, por exemplo, em sistemas críticos de segurança. No entanto, mesmo em ambientes restritos como esse, as atualizações podem ser planejadas e entregues rapidamente e de forma confiável, como mostra a evolução recente do software automotivo over-the-air. Além da entrega baseada em nuvem altamente segura, tais modelos de entrega exigem mudanças de arquitetura e hardware dedicados. Um exemplo é um controlador hot-swap em que uma metade está operacional e a outra metade constrói as próximas atualizações, que são trocadas para o modo ativo após procedimentos de segurança e verificação aprofundados. O DevOps para sistemas embarcados é mais desafiador do que para nuvem e serviços de TI, pois tenta combinar código e arquitetura legados com entrega contínua.

Princípios DevOps

Os princípios do DevOps, conforme enunciados por Jez Humble e David Harley, estão alinhados com a ideia de uma entrega de software mais eficiente e colaborativa. Vamos revisar cada um deles:

- Crie um processo repetível e confiável para entrega de software: Este princípio enfatiza a importância de ter um processo consistente e confiável para a entrega de software. A ideia é que a entrega não seja um evento traumático, mas sim algo rotineiro e previsível. Isso envolve ter etapas automatizadas, garantindo que a entrega seja tão simples quanto pressionar um botão.

- Automatize tudo que for possível: Esse princípio está intrinsecamente ligado ao primeiro. Ele defende que todas as etapas do processo de entrega de software devem ser automatizadas, desde a compilação do código até a configuração dos servidores e a implantação do sistema. A automação permite uma entrega mais rápida, consistente e livre de erros humanos.

- Mantenha tudo em um sistema de controle de versões: Esse princípio enfatiza a importância de manter todo o código fonte, arquivos de configuração, scripts, documentação e outros artefatos relacionados ao software em um sistema de controle de versões, como o Git. Isso facilita o rastreamento de alterações, o trabalho colaborativo e a possibilidade de retornar a versões anteriores do sistema, caso necessário.

- Se um passo causa dor, execute-o com mais frequência e o quanto antes: Esse princípio incentiva a antecipação de problemas e a resolução deles o mais cedo possível. Se um determinado passo do processo de entrega de software é complicado ou propenso a erros, é recomendado realizá-lo com mais frequência e de forma contínua. Um exemplo disso é a prática de Integração Contínua, em que o código é integrado regularmente para evitar problemas de integração no futuro.

- Concluído significa pronto para entrega: Esse princípio visa eliminar ambiguidades e garantir que um trabalho esteja verdadeiramente concluído antes de ser considerado pronto para entrar em produção. Isso envolve não apenas a implementação do código, mas também a realização de testes adequados, documentação completa e integração com outros sistemas, se necessário. A conclusão deve ter uma semântica clara, indicando que o trabalho está 100% pronto para ser entregue.

- Todos são responsáveis pela entrega do software: Esse último princípio destaca a importância da colaboração e da quebra de silos entre as equipes de desenvolvimento e operações. Todos os membros envolvidos no ciclo de vida do software são responsáveis pela entrega bem-sucedida do produto. Isso promove uma cultura de colaboração, comunicação eficiente e compartilhamento de responsabilidades.

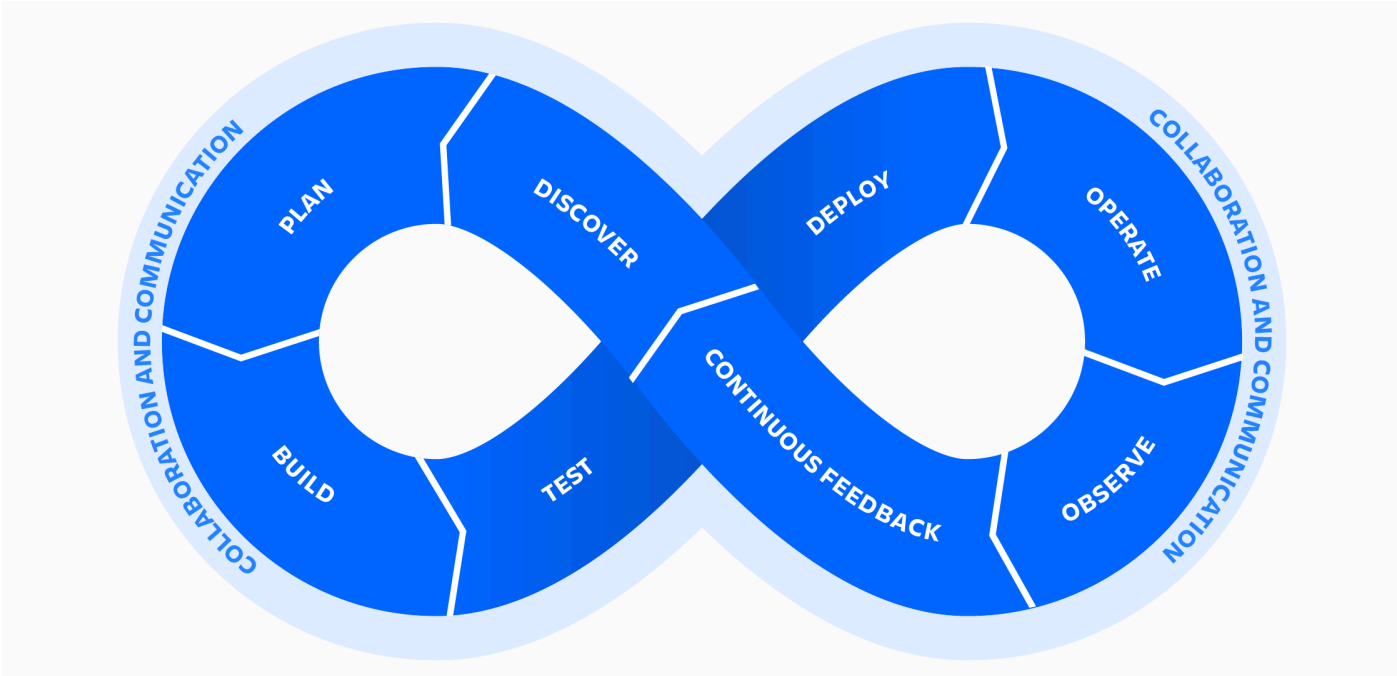
Ciclo de Vida

O ciclo do DevOps evoluiu para incluir a fase de "Discover" (Descobrir) no início, devido à importância de se compreender e alinhar os requisitos e metas do projeto desde o início do processo de desenvolvimento. Anteriormente, o ciclo do DevOps costumava começar diretamente com a fase "Plan" (Planejar), onde as equipes definiam as estratégias e planos para o desenvolvimento e entrega do software.

No entanto, reconheceu-se que a fase de "Discover" é fundamental para garantir que as necessidades do cliente sejam adequadamente identificadas e entendidas antes de iniciar o planejamento e a construção do software. Isso envolve pesquisas, análises e discussões para entender as expectativas dos usuários, as oportunidades de melhoria e os requisitos de negócio.

Ao introduzir a fase de "Discover" no início do ciclo do DevOps, as equipes têm a oportunidade de coletar informações valiosas, realizar análises e alinhar as expectativas de todas as partes interessadas envolvidas no projeto. Isso ajuda a evitar retrabalho, minimiza os riscos de desenvolver um produto que não atende às expectativas dos usuários e garante que o desenvolvimento seja direcionado para as necessidades reais do negócio.

O ciclo de vida do DevOps, de acordo com a Atlassian, é composto por sete fases que abrangem desde a concepção até o monitoramento contínuo. Vamos explorar cada uma dessas fases de forma mais didática:



1. Planejamento (Plan): Esta é a fase inicial do processo, na qual são definidos os objetivos, requisitos e cronograma do projeto. Os desenvolvedores e os operadores de infraestrutura trabalham juntos para garantir que os objetivos sejam alcançáveis e que a infraestrutura esteja preparada para suportar a nova versão do software.

2. Codificação (Code): Nesta fase, os desenvolvedores escrevem o código-fonte do novo software. O código deve ser escrito de acordo com as melhores práticas de desenvolvimento, de forma a ser seguro, eficiente e fácil de manter.
3. Construção (Build): Nesta fase, o código-fonte é compilado e empacotado em um formato executável. O processo de construção deve ser automatizado, de forma a garantir que o software seja construído sempre da mesma maneira.
4. Teste (Test): Nesta fase, o software é testado para garantir que atende aos requisitos e que não possui erros. Os testes devem ser realizados de forma automatizada, de forma a reduzir o tempo e o esforço necessários para testar o software.
5. Lançamento (Release): Nesta fase, o software é disponibilizado aos usuários. O lançamento deve ser feito de forma controlada, de forma a minimizar o impacto em caso de problemas.
6. Implantação (Deploy): Nesta fase, o software é implantado na infraestrutura de produção. O processo de implantação deve ser automatizado, de forma a garantir que o software seja implantado com segurança e eficiência.
7. Operação (Operate): Nesta fase, o software é operado e mantido em produção. Os operadores de infraestrutura são responsáveis por garantir que o software esteja funcionando corretamente e que seja protegido de ataques.
8. Monitoramento (Monitor): Esta é a fase em que o software é monitorado para garantir que esteja funcionando corretamente e que esteja protegido de ataques. O monitoramento pode ser realizado por meio de ferramentas de monitoramento, que coletam dados sobre o desempenho do software.

Além dessas fases, o ciclo de vida do DevOps enfatiza o feedback contínuo, que ocorre em todas as etapas do processo. Ele envolve a coleta de feedback dos usuários, das equipes de operações e dos stakeholders para impulsionar melhorias contínuas no software e no processo de desenvolvimento.

Deployment Contínuo

Com a Integração Contínua (CI), novo código é frequentemente integrado no ramo principal do projeto, mesmo que não esteja pronto para entrar em produção. Isso permite que outros desenvolvedores tenham conhecimento da existência desse código e evitem conflitos futuros de integração. Por exemplo, é possível integrar uma versão preliminar de uma tela com uma interface ainda em desenvolvimento ou uma versão de uma função com problemas de desempenho.

No entanto, existe um passo adicional na automação proposta pelo DevOps, chamado de Deployment Contínuo (Continuous Deployment ou CD). A diferença entre CI e CD é simples, mas impactante: com o CD, cada novo commit no ramo principal é rapidamente colocado em produção, em questão de horas, por exemplo. O fluxo de trabalho com o CD é o seguinte:

O desenvolvedor desenvolve e testa o código em sua máquina local. Ele realiza um commit e o servidor de CI executa novamente o build e os testes de unidade. Algumas vezes por dia, o servidor de CI realiza testes mais abrangentes nos novos commits que ainda não foram colocados em produção. Isso inclui testes de integração, testes de interface e testes de desempenho. Se todos os testes forem aprovados, os commits são imediatamente colocados em produção, e os usuários já podem interagir com a nova versão do código.
Algumas vantagens do Deployment Contínuo são:

Redução do tempo de entrega de novas funcionalidades. Em vez de esperar até que todas as funcionalidades estejam prontas para lançar uma nova versão, com o CD, as funcionalidades são liberadas assim que ficam prontas. Isso diminui o intervalo entre as releases, resultando em mais releases com um menor número de funcionalidades em cada uma delas.

Transformação das implantações em eventos não significativos. Não há mais um dia específico ou prazo final para lançar uma nova versão. Os prazos são uma fonte de estresse para os desenvolvedores e operadores de sistemas. Com o CD, as funcionalidades são implantadas assim que estiverem prontas, eliminando a pressão de prazos.

Com o CD, os desenvolvedores recebem feedback rapidamente, já que suas funcionalidades são colocadas em produção logo após serem concluídas. Isso evita que eles trabalhem por meses sem receber feedback sobre o sucesso ou falha de suas tarefas.

O favorecimento da experimentação e do desenvolvimento orientado por dados e feedback dos usuários é uma abordagem que visa aprimorar a criação e aprimoramento de produtos e serviços com base em informações concretas e observações reais. Com o CD, novas funcionalidades são rapidamente lançadas em produção, permitindo que os desenvolvedores recebam feedback dos usuários. Com base nesse feedback, é possível fazer ajustes na implementação ou até mesmo cancelar uma funcionalidade, se necessário.

No mundo real, várias empresas que desenvolvem sistemas web adotam o CD. Por exemplo, no Facebook, cada desenvolvedor faz em média 3,5 atualizações de software por semana, com uma média de 92 linhas de código adicionadas ou modificadas em cada atualização. Esses números demonstram que, para que o CD funcione bem, as atualizações de código devem ser pequenas.

Para isso, os desenvolvedores precisam ter a habilidade de dividir qualquer tarefa de programação, mesmo que complexa, em partes menores que possam ser implementadas, testadas, integradas e entregues rapidamente. Essa abordagem permite que o CD funcione de maneira eficaz.

Entrega Contínua

A entrega contínua no DevOps é uma abordagem que visa tornar o processo de entrega de software mais eficiente, confiável e rápida. Ela envolve a automação de várias etapas do ciclo de vida do desenvolvimento de software, garantindo que as alterações feitas no código sejam compiladas, testadas e implantadas de forma automatizada.

Primeiramente, a entrega contínua inclui a compilação automatizada do código-fonte. Isso significa que, assim que um desenvolvedor conclui uma alteração no código, o sistema automaticamente compila o código para criar uma versão executável do software.

Em seguida, a entrega contínua envolve a execução de testes automatizados. Diferentes tipos de testes, como testes unitários, testes de integração e testes de aceitação, podem ser realizados de forma automatizada para verificar se o software funciona corretamente e atende aos requisitos estabelecidos.

Além disso, a entrega contínua inclui a criação de artefatos de implantação. Esses artefatos podem ser pacotes de software, imagens de contêiner ou qualquer outro formato necessário para implantar o software em diferentes ambientes.

Ademais, a entrega contínua abrange a implantação automatizada do software em ambientes de teste, homologação e produção. Com base nas configurações e regras definidas, o sistema pode implantar o software de forma automatizada em diferentes servidores ou infraestruturas, garantindo que a versão mais recente do software esteja disponível para uso.

A entrega contínua é suportada por pipelines de entrega, que são fluxos de trabalho automatizados que orquestram todas essas etapas. Esses pipelines são configurados para serem acionados automaticamente quando uma alteração de código é detectada e executam todas as etapas necessárias até a implantação do software.

Em alguns tipos de sistemas, como IDEs, navegadores web, aplicativos móveis e sistemas embutidos em hardware, o Deployment Contínuo (CD) não é adequado. Isso ocorre porque nesses casos os usuários não desejam ser constantemente interrompidos com notificações de novas versões. Imagine se toda vez que você abrisse o navegador no seu computador, fosse notificado sobre uma nova versão disponível, ou se seu celular o informasse diariamente sobre uma atualização do aplicativo de rede social que você usa. Além disso, alguns desses sistemas requerem um processo de instalação que não é transparente para os usuários, ao contrário das atualizações automáticas em um sistema web.

No entanto, para lidar com essas situações, pode-se adotar uma abordagem mais suave chamada de Entrega Contínua (Continuous Delivery). Nesse caso, cada commit feito pelos desenvolvedores é considerado pronto para ser lançado em produção imediatamente. No entanto, a liberação efetiva para os usuários finais depende de uma autorização manual por parte de uma autoridade externa, como um gerente de projetos ou de releases. Essa decisão pode ser influenciada por fatores como estratégia da empresa ou demandas do mercado.

Na Entrega Contínua, cada alteração feita pelos desenvolvedores é considerada pronta para ser lançada imediatamente. Eles trabalham com essa mentalidade, mas a liberação para os usuários finais não acontece automaticamente. É necessária a aprovação de alguém que está no comando, como um gerente de projetos ou de lançamentos. Essa pessoa decide quando as alterações serão disponibilizadas aos usuários. Essa decisão pode ser influenciada por estratégias da empresa ou pelo mercado em que o sistema está inserido.

Para entender melhor, vamos diferenciar os conceitos: o Deployment é o processo de lançar uma nova versão do sistema para os usuários, tornando-a disponível para uso. Já o Delivery é o processo de disponibilizar uma nova versão do sistema para que seja liberada no deployment.

Dessa forma, a Entrega Contínua permite que as alterações sejam preparadas e prontas para serem lançadas a qualquer momento, mas a liberação efetiva para os usuários depende de uma aprovação manual. Isso permite um controle mais cuidadoso sobre as atualizações e considerações estratégicas antes de disponibilizá-las para os usuários finais.

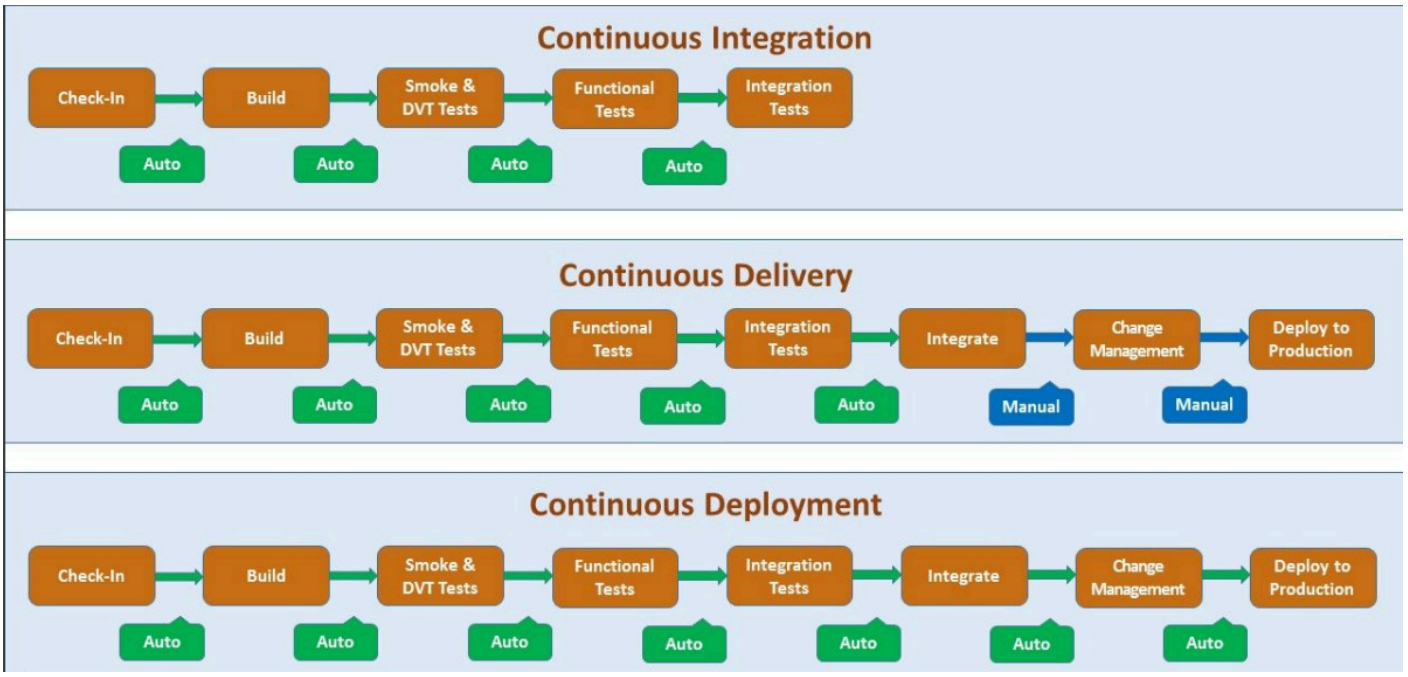
No mundo real, podemos observar diferentes frequências de lançamentos de novas versões em sistemas que não são baseados na web. Isso nos dá exemplos concretos de como as empresas lidam com as atualizações de seus produtos. Vamos ver alguns exemplos:

O Google, por exemplo, lança novas versões do navegador Chrome a cada seis semanas. Isso significa que a cada seis semanas os usuários podem esperar por atualizações e melhorias no navegador.

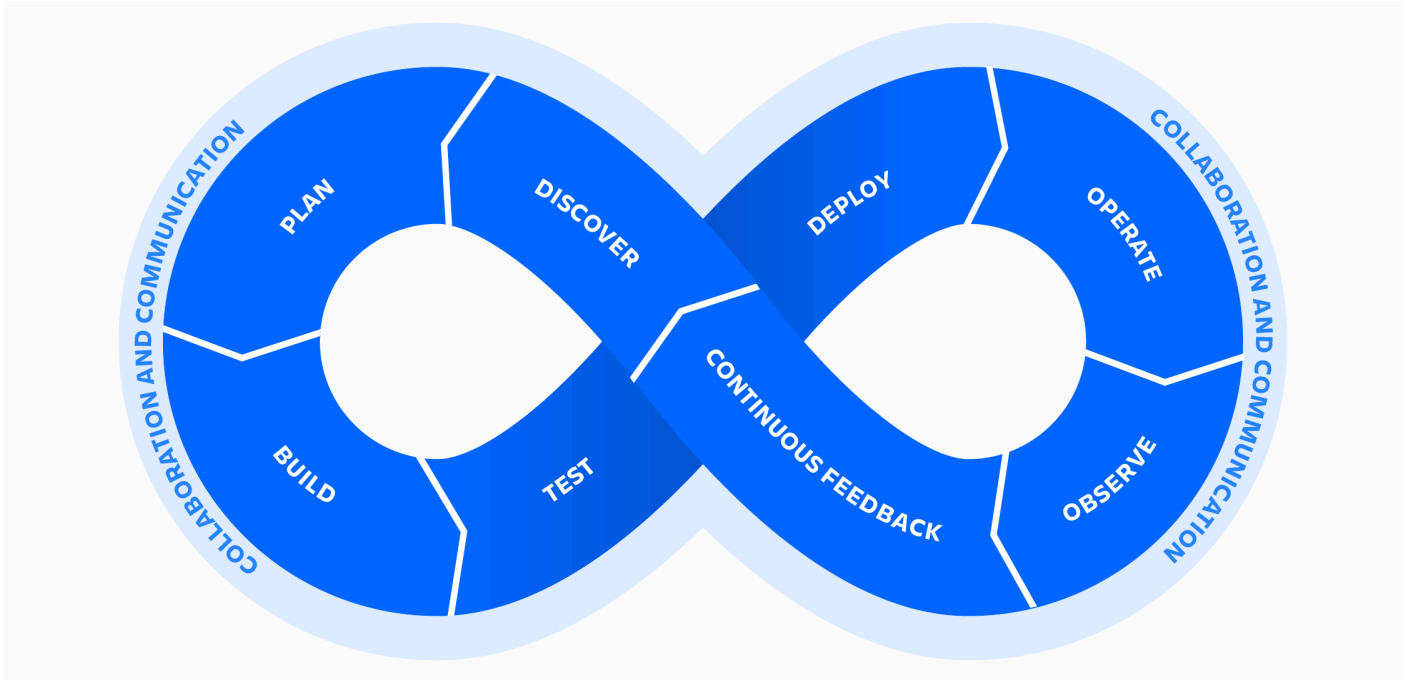
Anteriormente, a IDE Eclipse, que é uma ferramenta de desenvolvimento, costumava ter apenas uma nova versão por ano. No entanto, a partir de 2019, eles passaram a adotar um cronograma de lançamento mais frequente, com uma nova versão a cada 13 semanas. Isso permitiu que os desenvolvedores recebessem novas funcionalidades e melhorias com mais rapidez.

O Facebook, por sua vez, costumava atualizar sua versão para dispositivos Android a cada oito semanas. No entanto, recentemente eles reduziram esse tempo para apenas uma semana. Essa mudança tem como objetivo agradar os usuários, receber feedback mais rapidamente, manter os desenvolvedores motivados e se manter competitivo no mercado cada vez mais acelerado.

Esses exemplos mostram como as empresas estão se esforçando para lançar atualizações mais frequentes, a fim de melhorar seus produtos, atender às necessidades dos usuários e se manterem competitivas. Ao encurtar o tempo entre os lançamentos, eles podem garantir que novas funcionalidades sejam entregues mais rapidamente, permitindo a obtenção de feedback valioso e aprimorando constantemente suas soluções.



O DevOps é mais do que apenas equipes de desenvolvimento e operações trabalhando juntas. É mais do que ferramentas e práticas. O DevOps é uma forma de pensar, uma mudança cultural, em que as equipes adotam novas formas de trabalhar.



Na cultura de DevOps, os desenvolvedores se aproximam dos usuários, obtendo uma compreensão melhor dos requisitos e necessidades deles. As equipes de operações se envolvem no processo de desenvolvimento e adicionam requisitos de manutenção e necessidades do cliente. Ela também envolve aderir aos princípios essenciais a seguir, que ajudam as equipes de DevOps a oferecer aplicativos e serviços em um ritmo mais rápido e com maior qualidade do que as organizações que usam o modelo de desenvolvimento de software tradicional.

1. Automação: A automação é o uso de ferramentas e tecnologias para automatizar tarefas repetitivas e manuais ao longo do ciclo de vida do software. Isso inclui processos como compilação, teste, implantação e monitoramento. A automação é essencial para aumentar a eficiência, reduzir erros e acelerar o desenvolvimento e a entrega de software.

2. Colaboração ágil: A colaboração ágil é um princípio chave do DevOps que enfatiza a importância de equipes multidisciplinares trabalharem juntas de forma colaborativa e integrada. Isso envolve uma comunicação efetiva, compartilhamento de responsabilidades e conhecimento, e colaboração contínua ao longo do ciclo de vida do software. A colaboração ágil elimina silos organizacionais e promove a sinergia entre as equipes, resultando em uma entrega de software mais ágil e eficiente.

3. Produção contínua: A produção contínua, também conhecida como entrega contínua, é uma abordagem que busca entregar o software de forma consistente, confiável e repetível, pronta para ser colocada em produção. Isso envolve a automação do processo de implantação em ambientes de produção, permitindo implantações frequentes e seguras. A produção contínua reduz o tempo de espera para os usuários e permite um feedback mais rápido sobre o software.

4. Integração contínua: A integração contínua é uma prática em que as alterações de código são integradas de forma frequente e automatizada em um repositório compartilhado. O objetivo é identificar problemas de integração, conflitos e erros o mais cedo possível, garantindo que o software esteja sempre em um estado funcional. A integração contínua permite que as equipes trabalhem de forma colaborativa, integrando suas alterações de forma contínua e automatizada, o que reduz o risco de problemas decorrentes da integração tardia e melhora a eficiência do processo de desenvolvimento.

5. Ação centrada no cliente: A ação centrada no cliente é um princípio que destaca a importância de colocar o cliente no centro das atividades de desenvolvimento e operação de software. Isso envolve entender as necessidades e expectativas do cliente, coletar feedback regularmente e ajustar as prioridades e estratégias de desenvolvimento de acordo. Uma abordagem centrada no cliente garante que as equipes estejam desenvolvendo software que atenda às necessidades dos usuários finais e esteja alinhado com os objetivos do negócio.

Vamos analisar uma questão da FGV que aborda os princípios do DevOps. Nessa questão, é apresentado um cenário em que um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Além disso, é mencionado que todo o software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o objetivo de verificar vulnerabilidades antes de entrar em produção. Vamos analisar.

(FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto

de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

a) automação;
b) colaboração ágil;
c) produção contínua;
d) integração contínua;
e) ação centrada no cliente.

Comentários:

O gabarito é a letra A - automação. A falha no princípio da automação ocorre porque o processo descrito não é totalmente automatizado. Embora o Time de Desenvolvimento de Software (TDS) siga um protocolo automatizado para gerar, testar e combinar pacotes de software, há uma etapa em que é necessário fazer uma requisição formal ao Time de Operações (TO) para executar um conjunto de testes. Essa requisição formal indica que há uma intervenção manual no processo, o que vai contra o princípio da automação do DevOps e do DevSecOps.

As demais opções estão incorretas porque não estão diretamente relacionadas à falha mencionada. Colaboração ágil (opção B): Embora a colaboração entre o Time de Desenvolvimento de Software (TDS) e o Time de Operações (TO) seja mencionada, não há indicação de uma falha específica nessa colaboração.
Produção contínua (opção C): O texto menciona que os pacotes de software são combinados separadamente e passam por testes antes de entrar em produção. Portanto, não há uma falha específica no princípio da produção contínua.
Integração contínua (opção D): Embora o texto mencione que os pacotes de software são combinados, não há informações suficientes para determinar se há uma falha no princípio da integração contínua.
Ação centrada no cliente (opção E): O texto não menciona diretamente a relação com o cliente, portanto não é possível afirmar que há uma falha no princípio da ação centrada no cliente. (Gabarito: Letra A)

Pessoal, vimos que a situação descrita se refere ao princípio de "Automação" no contexto de DevOps. No caso apresentado, o Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. A automação é o uso de ferramentas e tecnologias para automatizar tarefas repetitivas e manuais ao longo do ciclo de vida do software. Nesse caso, o processo de geração, teste e combinação de pacotes de software é realizado de forma automatizada, o que contribui para aumentar a eficiência, reduzir erros e acelerar o desenvolvimento e a entrega de software.

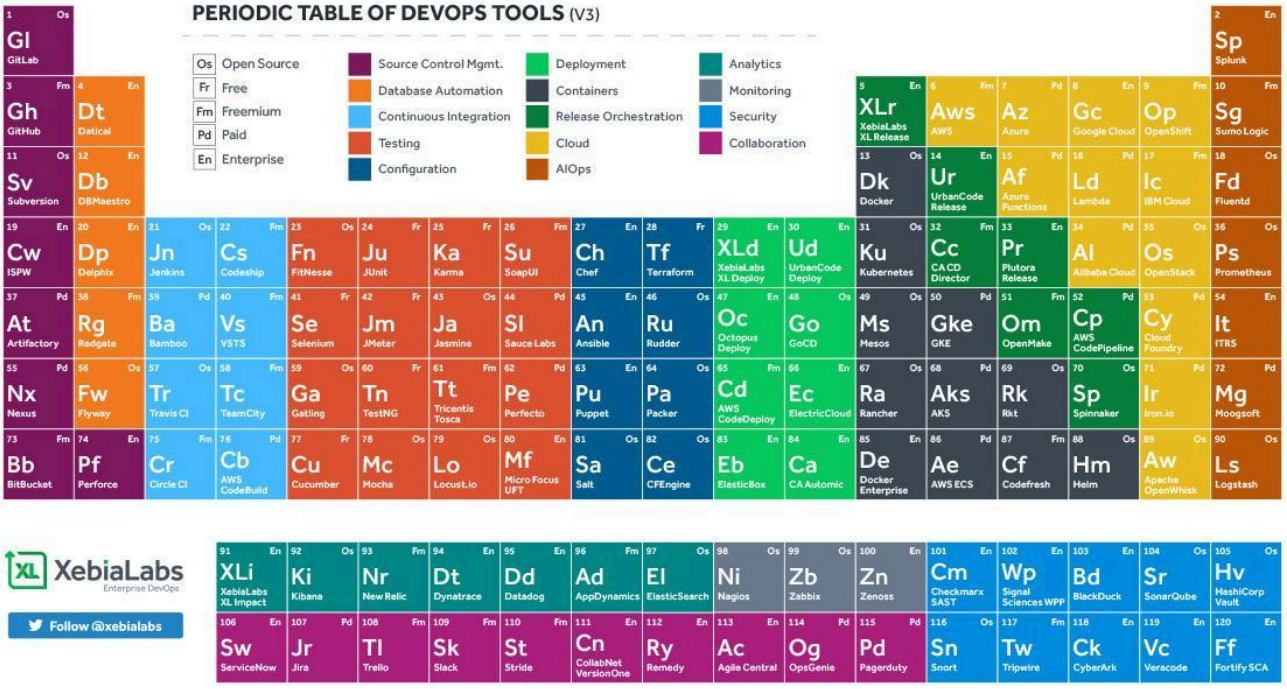
Além disso, é mencionado que o software combinado precisa passar por um processo de teste, no qual uma requisição formal é enviada ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes. Essa abordagem de automatizar o processo de teste também está alinhada com o princípio de automação, pois ajuda a garantir a qualidade do software de forma consistente e confiável.

Ferramentas de DevOps

A Tabela Periódica de Ferramentas DevOps[1] é um recurso abrangente que mostra várias ferramentas usadas no cenário DevOps, ela é SENSACIONALMENTE DIDÁTICA. Ela fornece uma visão categorizada dessas ferramentas. A Tabela Periódica de Ferramentas DevOps organiza AS ferramentas em diferentes categorias, facilitando a compreensão e escolha das corretas para tarefas específicas.

A tabela inclui 14 categorias, cada uma representando um aspecto diferente do DevOps. Essas categorias incluem Integração Contínua, Entrega Contínua, Testes, Gerenciamento de Configuração, Orquestração de Infraestrutura, Containerização, Monitoramento, Gerenciamento de Logs, Colaboração, Segurança, Gerenciamento de Banco de Dados, Gerenciamento de Versões e Automação de Builds.

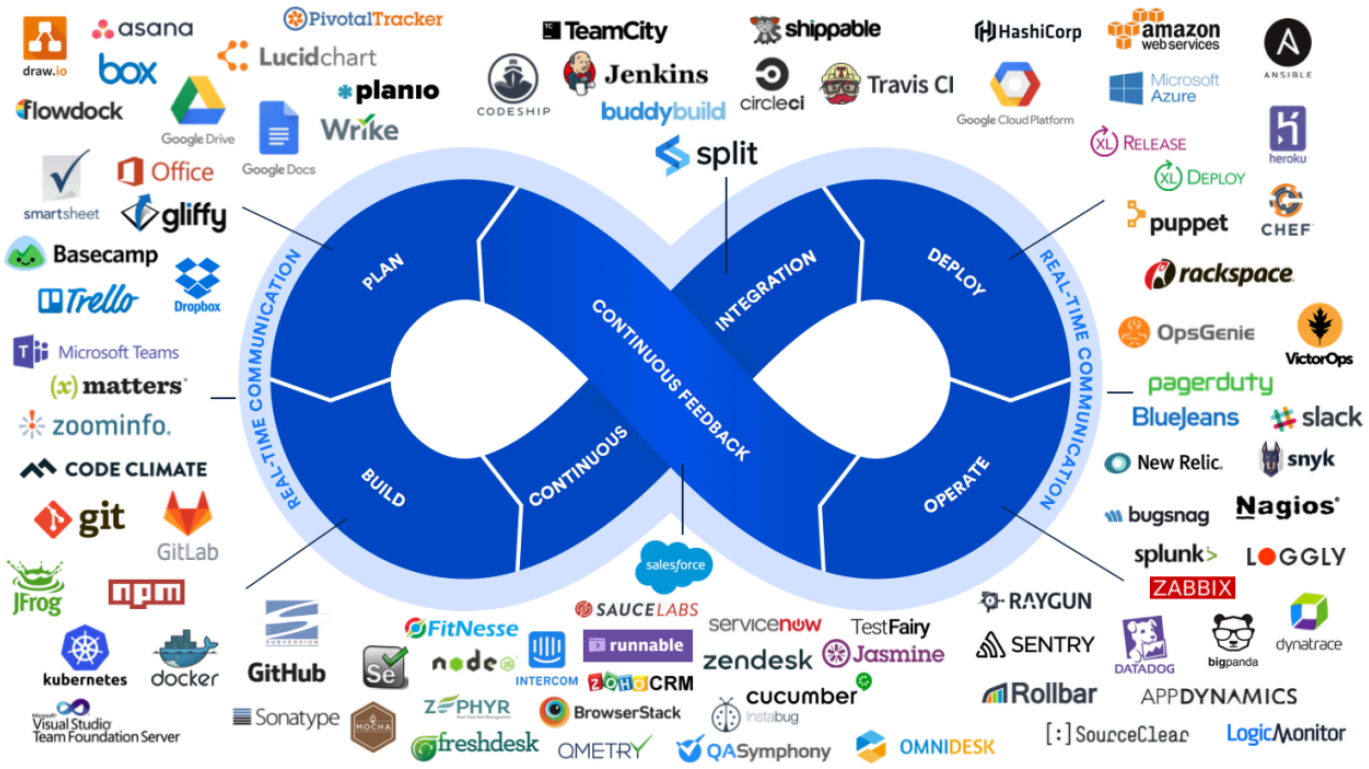
Dentro de cada categoria, você encontrará ferramentas populares amplamente usadas na indústria. A tabela fornece uma representação visual dessas ferramentas, ajudando os profissionais de TI a se manterem atualizados com as opções mais recentes disponíveis.



As ferramentas são fundamentais para automatizar o DevOps. Entregas de qualidade com ciclos curtos de tempo exigem um alto grau de automação. Portanto, escolher as ferramentas certas para o seu ambiente ou projeto é importante quando você adota o DevOps.

[1] https://xebialabs.com/wp-content/uploads/files/infographics/periodic-table-of-devops-tools-v3-1.pdf
https://xebialabs.com/periodic-table-of-devops-tools/

Pessoal, abaixo temos a lista mais completa que encontrei com as ferramentas possíveis de utilização no ciclo DevOps.



Colaboração

A etapa de colaboração no DevOps é uma das fases essenciais desse processo, que envolve a interação e cooperação entre as equipes de desenvolvimento e operações. Nessa etapa, as equipes trabalham em conjunto para compartilhar informações, alinhar objetivos, tomar decisões conjuntas e garantir uma comunicação eficiente ao longo de todo o ciclo de vida do software.

A colaboração no DevOps busca superar as barreiras tradicionais entre as equipes de desenvolvimento e operações, promovendo uma cultura de colaboração, transparência e responsabilidade compartilhada. Através dessa colaboração, as equipes podem identificar e solucionar problemas de forma mais rápida, garantir a entrega contínua de software de qualidade e responder de maneira ágil às necessidades dos usuários.

Nessa etapa, são utilizadas diversas ferramentas de colaboração, como JIRA, Trello, Basecamp, Asana, Microsoft Teams, Slack e muitas outras, que permitem a comunicação eficiente, o compartilhamento de informações, o gerenciamento de tarefas e a coordenação das atividades entre as equipes. A colaboração no DevOps é um princípio fundamental para promover uma cultura de trabalho colaborativa, em que os membros das equipes se apoiam mutuamente, compartilham conhecimentos, experiências e responsabilidades, visando alcançar os objetivos comuns de entrega contínua, qualidade e eficiência na produção de software. A colaboração é algo significativo para todas as aplicações. Uma aplicação ou software não é muito útil se for utilizado apenas para um único propósito. Ao invés disso, se o seu software colaborar com outros

softwares presentes no mercado, então ele se torna útil para ambos. Portanto, as principais ferramentas através das quais você pode colaborar com o seu software são as seguintes:

- JIRA: O JIRA é uma ferramenta de gerenciamento de projetos que permite o acompanhamento de tarefas, atribuição de responsabilidades, definição de prazos e monitoramento do progresso do projeto. Ele auxilia na organização das atividades e facilita a comunicação entre os membros da equipe.

- Trello: O Trello é uma ferramenta de gerenciamento de projetos baseada em quadros virtuais. Ele permite a criação de listas e cartões para representar tarefas e atividades. Os membros da equipe podem mover os cartões entre as listas para indicar o status atual de cada tarefa, facilitando o acompanhamento do progresso.

- Slack: O Slack é uma ferramenta de comunicação em equipe que oferece recursos avançados de mensagens instantâneas, chamadas de áudio e vídeo, compartilhamento de arquivos e integração com outras ferramentas. Ele permite que os membros da equipe se comuniquem de forma eficiente e colaborativa, facilitando a troca de informações e o trabalho em conjunto.

Gerenciamento de Código Fonte

Ao começarmos a desenvolver um aplicativo usando a abordagem DevOps, um dos primeiros passos é construir o código. Como cada aplicativo possui um código em execução que precisa ser atualizado conforme necessário, é importante gerenciar o código-fonte. As ferramentas de gerenciamento de código-fonte fornecem recursos para mostrar qual usuário fez alterações em determinado momento. As ferramentas mais populares nesse segmento são as seguintes:

- GitHub: O GitHub é uma plataforma de hospedagem de repositórios de controle de versão. Ele permite que os desenvolvedores colaborem em projetos, realizem o controle de versão do código-fonte e gerenciem as alterações feitas por diferentes membros da equipe. O GitHub oferece recursos de rastreamento de problemas, revisão de código e integração com outras ferramentas de desenvolvimento.

- Subversion: O Subversion (também conhecido como SVN) é um sistema de controle de versão centralizado amplamente utilizado. Ele permite que várias pessoas trabalhem em um projeto simultaneamente, rastreiem as alterações feitas no código e revertam para versões anteriores, se necessário. O Subversion oferece recursos de ramificação e mesclagem, facilitando o gerenciamento de diferentes fluxos de trabalho de desenvolvimento.

- Artifactory: O Artifactory é um repositório de artefatos que gerencia e armazena os componentes de software criados durante o desenvolvimento. Ele permite o gerenciamento centralizado de pacotes, bibliotecas, dependências e outros artefatos necessários para construir e implantar um aplicativo. O Artifactory oferece recursos de gerenciamento de versões, controle de acesso e integração com ferramentas de automação de build e implantação.

- Nexus: O Nexus é outro exemplo de repositório de artefatos usado para armazenar e distribuir componentes de software. Ele oferece recursos de gerenciamento de dependências, controle de acesso, geração de relatórios e proxy para repositórios remotos. O Nexus é amplamente utilizado na integração contínua e entrega contínua para garantir que as dependências do projeto estejam disponíveis e atualizadas.

- Bitbucket: O Bitbucket é uma plataforma de hospedagem de repositórios de controle de versão, semelhante ao GitHub. Ele permite que os desenvolvedores armazenem, colaborem e controlem as versões de seu código-fonte. O Bitbucket suporta tanto repositórios Git quanto Mercurial e oferece recursos como revisão de código, integração com outras ferramentas de desenvolvimento e recursos avançados de gerenciamento de equipe.

Automação de Banco de Dados

Bancos de dados desempenham um papel fundamental em qualquer tipo de aplicação. No entanto, é praticamente impossível para os desenvolvedores realizar tarefas administrativas nos bancos de dados com frequência. Portanto, a automação de banco de dados consiste na aplicação de processos de atualização e administração não supervisionados em bancos de dados. Com esse tipo de automação, é possível reduzir erros nas operações, melhorar a velocidade e aumentar a confiabilidade. Algumas das ferramentas populares utilizadas para esse propósito são as seguintes:

- Datical: O Datical é uma ferramenta de automação de banco de dados que auxilia no gerenciamento e na implementação de alterações no banco de dados de forma segura e eficiente. Ele permite que as equipes de desenvolvimento e operações colaborem no controle de versões do banco de dados, evitando erros e facilitando a entrega contínua.

- DBMaestro: O DBMaestro é uma ferramenta de gerenciamento de alterações de banco de dados que ajuda as equipes a controlar e gerenciar as mudanças feitas nos bancos de dados. Ele oferece recursos de controle de versão, rastreamento de alterações, implantação automatizada e conformidade com regulamentos de segurança.

- Delphix: O Delphix é uma plataforma de automação de dados que permite a criação de cópias virtuais de bancos de dados para fins de desenvolvimento, teste e análise. Ele ajuda as equipes a economizar tempo e recursos, fornecendo dados virtualizados sob demanda, sem a necessidade de cópias físicas dos bancos de dados.

- Redgate: A Redgate fornece uma variedade de ferramentas para o gerenciamento e a automação de bancos de dados, incluindo ferramentas para controle de versão, comparação e sincronização de esquemas, monitoramento de desempenho e segurança. Essas ferramentas ajudam as equipes a garantir a integridade e a eficiência dos bancos de dados.

- Flyway: O Flyway é uma ferramenta de migração de banco de dados que permite que as equipes gerenciem e versionem as alterações feitas nos esquemas dos bancos de dados. Ele fornece um sistema simples e poderoso para rastrear, aplicar e reverter as migrações do banco de dados, garantindo a consistência e a confiabilidade dos dados.

Integração Contínua

Imagine que você está trabalhando em um projeto de desenvolvimento de software juntamente com outros colegas de equipe. Cada um de vocês está trabalhando em diferentes partes do código, adicionando novas funcionalidades ou corrigindo bugs.

Agora, suponha que todos vocês trabalhem de forma isolada durante dias ou semanas sem compartilhar o código entre si. Quando chegar a hora de integrar todas as partes do código, vocês podem enfrentar grandes problemas. Pode haver conflitos entre as alterações realizadas por diferentes pessoas, o que torna difícil e demorado resolver esses conflitos manualmente.

É aí que entra a Integração Contínua. Essa prática sugere que, em vez de esperar até o final do projeto para integrar todo o código, vocês devem realizar integrações frequentes ao longo do processo de desenvolvimento. Isso significa que cada vez que um desenvolvedor terminar uma pequena tarefa ou implementar uma funcionalidade, ele deve integrar seu código ao repositório compartilhado o mais rápido possível. Dessa forma, vocês evitam acumular grandes quantidades de código não integrado. Quando o código é integrado com frequência, é mais fácil identificar e resolver problemas de integração. Além disso, as alterações feitas por diferentes desenvolvedores são combinadas regularmente, o que ajuda a detectar conflitos e erros mais cedo no processo, facilitando a sua correção.

Paolla, mas quão rápido é "integrar seu código ao repositório compartilhado o mais rápido possível"? A velocidade de integração do código ao repositório compartilhado na prática de Integração Contínua pode variar, mas o objetivo é realizar integrações frequentes e regulares para obter os benefícios dessa abordagem.

Kent Beck, um dos defensores do Extreme Programming (XP) e da Integração Contínua, sugere que os desenvolvedores devem integrar e testar seu código em intervalos menores do que algumas horas. Ele argumenta que a duração da tarefa de integração é imprevisível e pode levar mais tempo do que a tarefa original de codificação. Portanto, quanto mais tempo for necessário para integrar o código, maiores e mais imprevisíveis serão os custos associados. Outros especialistas, como Martin Fowler, mencionam que pelo menos uma integração por dia por desenvolvedor é um limite mínimo para considerar que uma equipe está aplicando a Integração Contínua.

Essas referências sugerem que a integração contínua deve acontecer com uma frequência significativa, mas não necessariamente a cada poucos minutos ou segundos. O principal objetivo é evitar longos períodos de trabalho isolado e garantir que as alterações de código sejam integradas e testadas em intervalos regulares, minimizando a chance de conflitos e problemas de integração. O tempo exato entre as integrações pode variar dependendo das necessidades e da complexidade do projeto, bem como das preferências da equipe. O importante é encontrar um ritmo que permita uma colaboração contínua, detecção precoce de problemas e resolução eficiente de conflitos. Vejamos essa citação de Kent Beck:

> Você deve integrar e testar o seu código em intervalos menores do que algumas horas. Programação em times não é um problema do tipo dividir-e-conquistar. Na verdade, é um problema que requer dividir, conquistar e integrar. A duração de

> uma tarefa de integração é imprevisível e pode facilmente levar mais tempo do que a tarefa original de codificação. Assim, quanto mais tempo você demorar para integrar, maiores e mais imprevisíveis serão os custos.

Você deve integrar e testar o seu código em intervalos menores do que algumas horas. Programação em times não é um problema do tipo dividir-e-conquistar. Na verdade, é um problema que requer dividir, conquistar e integrar. A duração de uma tarefa de integração é imprevisível e pode facilmente levar mais tempo do que a tarefa original de codificação. Assim, quanto mais tempo você demorar para integrar, maiores e mais imprevisíveis serão os custos.

A Integração Contínua também envolve a execução automatizada de testes em cada integração. Assim, vocês garantem que o código integrado seja testado continuamente, identificando possíveis problemas o mais cedo possível. Ao adotar a Integração Contínua, vocês reduzem os riscos de erros e conflitos tardios, economizando tempo e esforço. Além disso, ela promove uma colaboração mais estreita entre os membros da equipe, evitando que o trabalho seja desenvolvido em silos isolados.

Boas Práticas para Uso de CI

Quando falamos de CI, estamos nos referindo a um processo de desenvolvimento de software em que o código é constantemente atualizado e integrado na ramificação principal, também conhecida como "master". O objetivo é garantir que o código esteja sempre funcionando corretamente, sem erros ou bugs, mesmo com as atualizações mais recentes. Existem algumas práticas recomendadas para garantir o bom funcionamento da CI:

- Build Automatizado: O processo de build consiste em compilar todos os arquivos de um sistema, criando uma versão executável. Para a integração contínua, é fundamental que esse processo seja automatizado, ou seja, sem intervenção manual. Isso ajuda a garantir que o build seja rápido e eficiente, pois será executado repetidamente. Autores especializados sugerem que o tempo de execução de um build não ultrapasse 10 minutos.

- Testes Automatizados: Além de garantir que o código compile sem erros, é importante verificar se o sistema continua funcionando como esperado após cada atualização. Por isso, é recomendado ter uma ampla cobertura de testes automatizados, especialmente testes de unidade. Esses testes ajudam a identificar possíveis problemas e bugs nas funcionalidades do sistema, garantindo sua qualidade.

- Servidores de CI são ferramentas que auxiliam na implementação da integração contínua em um projeto de desenvolvimento de software. Eles desempenham um papel fundamental na automatização dos processos de build e testes.

Vamos entender passo a passo como os Servidores de CI funcionam:

1. Após um novo commit: Quando um desenvolvedor realiza um novo commit no sistema de controle de versões, como o Git, o servidor de CI é notificado.

2. Clone do repositório: O servidor de CI realiza uma cópia do repositório do projeto, isso é conhecido como "clonar o repositório". Ele faz isso para obter a versão atualizada do código em que o commit foi realizado.

3. Execução do build completo: Após clonar o repositório, o servidor de CI inicia o processo de build completo do sistema. Esse processo envolve a compilação e a construção de todos os arquivos do projeto, resultando em uma versão executável do software. O objetivo é garantir que o código seja compilado corretamente, sem erros.

4. Execução dos testes: Após o build completo, o servidor de CI executa todos os testes automatizados configurados para o projeto. Isso inclui testes de unidade, integração, aceitação e outros, dependendo das necessidades do projeto. O objetivo é verificar se o sistema continua com o comportamento esperado e se não houve introdução de erros.

5. Notificação do usuário: Após a execução do build e dos testes, o servidor de CI notifica o usuário sobre os resultados. Isso pode ser feito por meio de mensagens, e-mails, notificações no sistema, entre outros meios de comunicação. Essa comunicação serve para informar ao desenvolvedor sobre o status do processo de CI, indicando se o build e os testes foram bem-sucedidos ou se ocorreram falhas.

(CESPE – Petrobras – 2022). Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

Comentários:

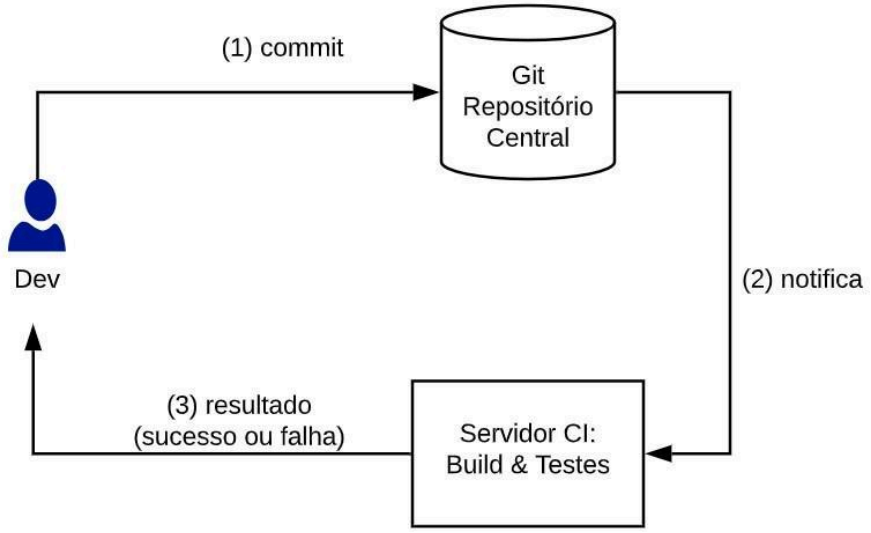
Na prática, muitas organizações utilizam sistemas de controle de versão, como o Git, para automatizar processos de DevOps. Esses sistemas permitem que equipes de desenvolvedores trabalhem em conjunto em um mesmo projeto, realizando alterações no código-fonte de forma simultânea. Com a automação, é possível evitar conflitos e preservar o histórico do código.

Um exemplo dessa automação ocorre na integração contínua. Sempre que são feitas alterações no repositório principal do código-fonte, os testes automatizados são executados. Isso garante que, a cada mudança realizada, os testes necessários sejam executados de forma rápida e eficiente.

Dessa forma, a automação dos testes permite que a equipe de desenvolvimento mantenha a qualidade do software, identificando erros e falhas de forma ágil. Além disso, ao automatizar os testes, é possível ter um feedback constante sobre o estado do código, garantindo que ele permaneça funcional e livre de problemas. Essa prática contribui para a eficiência do processo de desenvolvimento, uma vez que os testes são executados de maneira automática e contínua, sem a necessidade de intervenção manual a cada alteração no código-fonte. (Gabarito: Correto)

Os Servidores de CI são ferramentas poderosas que automatizam processos essenciais para garantir a qualidade do código e a estabilidade do sistema. Eles permitem a detecção precoce

de problemas e a identificação de erros antes que sejam integrados à ramificação principal do projeto.



O principal objetivo de um servidor de integração contínua é evitar a integração de código problemático, seja em relação ao processo de compilação ou ao comportamento do sistema. Quando ocorre uma falha no build, é comum dizer que o build quebrou. Às vezes, o build pode ter sido bem-sucedido na máquina do desenvolvedor, mas falha ao ser executado no servidor de CI. Isso pode acontecer se o desenvolvedor esquecer de fazer o commit de algum arquivo ou se houver dependências incorretas. Por exemplo, o código pode ter sido compilado e testado na máquina do desenvolvedor usando a versão 2.0 de uma biblioteca, enquanto o servidor de CI utiliza a versão 1.0.

Quando o servidor de CI notifica o desenvolvedor de que seu código não passou nos testes ou quebrou o build, é importante que ele pare o que está fazendo e corrija o problema imediatamente. Isso é crucial porque um build quebrado afeta o trabalho dos outros desenvolvedores, que não conseguirão mais compilar ou executar o sistema. Diz-se que a correção de um build quebrado é a maior prioridade em uma empresa de software. No entanto, em alguns casos, a solução pode ser simplesmente reverter o código para a versão anterior ao commit problemático.

A fim de manter a coerência com os princípios da Integração Contínua, é recomendado que os desenvolvedores aguardem o resultado do servidor de CI antes de avançarem para a próxima tarefa de programação. Isso significa que eles não devem iniciar a escrita de novo código até que tenham certeza de que o commit mais recente tenha passado pelo serviço de CI. Além disso, é importante evitar iniciar outras atividades significativas, como reuniões, almoços ou ir para casa, antes de receber o resultado do servidor de CI.

Existem várias opções de servidores de integração contínua disponíveis no mercado. Alguns são oferecidos como serviços independentes, sendo gratuitos para repositórios de código aberto, mas requerendo pagamento para repositórios privados de empresas. Portanto, se você possui um repositório aberto no GitHub, há mais de uma opção gratuita para ativar um serviço de CI nele.

Uma dúvida comum é se a Integração Contínua é compatível com o uso de branches. De acordo com a definição da Integração Contínua, a resposta é sim, desde que os branches sejam integrados frequentemente ao ramo principal (geralmente o "master"), preferencialmente diariamente. Em outras palavras, a Integração Contínua não é incompatível com o uso de branches, mas apenas com branches de longa duração. Por exemplo, Martin Fowler observa que

o uso de branches específicos para funcionalidades pode ser adotado em conjunto com a Integração Contínua, desde que esses branches sejam integrados regularmente.

> Na maioria das vezes, branches de funcionalidades constituem uma abordagem incompatível com CI. Um dos princípios de CI é que todos devem enviar commits para a linha de desenvolvimento principal diariamente. Então, a não ser que os branches de funcionalidades durem menos do que um dia, eles são um animal diferente de CI. É comum ouvir desenvolvedores dizendo que eles estão usando CI porque eles rodam builds automáticos, talvez usando um servidor de CI, após cada commit. Isso pode ser chamado de building contínuo e pode ser uma coisa boa. Porém, como não há integração, não podemos chamar essa prática de CI.

Programação em Pares

O Pair Programming, ou Programação em Pares, pode ser considerado uma forma contínua de revisão de código. Nessa prática, um desenvolvedor trabalha em conjunto com outro, sentando ao lado dele, revisando e colaborando na escrita de cada trecho de código. Assim como a Integração Contínua (CI), é recomendado utilizar o Pair Programming em conjunto com a CI. No entanto, seu uso não é obrigatório.

No Pair Programming, o código é revisado imediatamente durante a sessão de programação, garantindo que haja um controle contínuo de qualidade. Dessa forma, erros podem ser identificados e corrigidos rapidamente, antes mesmo do commit no repositório principal. Essa abordagem ajuda a minimizar os custos associados à revisão de código após a integração, quando os problemas podem ser mais complexos de solucionar.

No entanto, é possível realizar a revisão de código após o commit ser realizado no repositório principal. Nesse caso, os custos e esforços necessários para a revisão podem ser maiores, uma vez que o código já foi integrado e pode afetar outros desenvolvedores. Portanto, embora seja possível revisar o código após a integração, recomenda-se a adoção do Pair Programming para garantir uma revisão contínua e colaborativa, reduzindo os riscos de erros e melhorando a qualidade do código desde o início.

A Integração Contínua é o núcleo do Ciclo de Vida do DevOps, pois todos os membros de uma equipe coordenam seu trabalho com frequência. Cada integração é verificada por um processo automatizado para identificar a integração o mais rápido possível. Aqui, você só precisa lembrar que é necessário escolher um método confiável de correspondência para garantir que erros sejam encontrados muito mais cedo no pipeline de CI/CD. Algumas das conhecidas ferramentas de integração contínua são as seguintes:

- Jenkins: O Jenkins é uma ferramenta de integração contínua que automatiza o processo de construção, testes e implantação de software. Ele permite que as equipes de desenvolvimento integrem as alterações de código em um repositório compartilhado, onde são automaticamente construídas, testadas e implantadas em um ambiente de desenvolvimento ou produção.

- Codeship: O Codeship é uma plataforma de integração e entrega contínua (CI/CD) baseada na nuvem. Ele permite que as equipes de desenvolvimento automatizem o

processo de integração de código, construção, testes e implantação em diversos ambientes. Com o Codeship, é possível criar pipelines de entrega contínua para garantir a qualidade e eficiência do processo de desenvolvimento de software.

- AWS CodeBuild: O AWS CodeBuild é um serviço gerenciado da Amazon Web Services (AWS) que compila e testa o código-fonte de aplicativos de forma automatizada. Ele permite que as equipes de desenvolvimento construam, testem e implantem aplicativos de forma rápida e confiável na nuvem da AWS. O AWS CodeBuild é altamente escalável e integra-se perfeitamente com outras ferramentas e serviços da AWS.

Build (Construção)

Nessa fase do DevOps, as ferramentas devem suportar fluxos de trabalho rápidos. Aqui discutimos dois tipos de ferramentas. As ferramentas de construção ajudam a alcançar iterações rápidas, reduzindo tarefas manuais demoradas. As ferramentas de integração contínua (CI) mesclam o código de todos os desenvolvedores e verificam se há código com problemas, melhorando a qualidade do software.

Ferramentas de construção. A automação de construção é crucial para o ambiente de DevOps - tanto para projetos pequenos quanto grandes. Com o desenvolvimento ágil e fluido, as ferramentas de construção também se tornaram ferramentas para gerenciar o ciclo de vida do desenvolvimento de software e do serviço, o que envolve compilar código, gerenciar dependências, gerar documentação, executar testes ou implantar um aplicativo em diferentes ambientes.

O Apache Ant se tornou a ferramenta padrão para a construção de aplicativos, especialmente em projetos de código aberto quando é necessário compilar a partir das fontes. A sintaxe simples usada para definir as tarefas a serem executadas reflete a principal vantagem e desvantagem do Ant. É fácil de entender, mas é verboso até mesmo para tarefas simples, porque usa XML, que não é uma linguagem de programação procedural.

O Maven visa resolver alguns dos problemas do Ant. Ele também usa XML para definir o processo de construção. Nesse caso, as declarações definem a natureza do projeto em vez das tarefas que o constroem. O Maven usa o Project Object Model, que define uma maneira uniforme de construir sistemas. Isso tem a vantagem de definir um padrão no início de um projeto, mas às vezes causa inflexibilidade quando operações personalizadas precisam ser definidas. O gerenciamento de dependências também ajuda a automatizar o processo de construção, mas às vezes cria problemas devido às dependências externas automáticas.

Rake e Gradle tentam resolver os problemas do Ant e do Maven, oferecendo uma ferramenta baseada em linguagem de programação para construir aplicativos. Com o Rake, as linhas de código são escritas em Ruby; com o Gradle, a configuração usa uma linguagem específica de domínio (DSL) baseada em Groovy.

Ferramentas de integração contínua: A integração contínua (CI) é um conceito ágil fundamental. As ferramentas de CI integram o trabalho dos desenvolvedores o mais cedo e o mais frequentemente possível. Dessa forma, o sistema é constantemente testado. Adotar a CI nem sempre é fácil e direto. Por exemplo, no desenvolvimento de sistemas embarcados, os testes são

desafiadores porque nem sempre é possível construir e testar no hardware de destino e, portanto, deve ser simulado. Além disso, as limitações de hardware afetam a velocidade dos testes e, consequentemente, o tempo do ciclo.

O Jenkins é um sistema de código aberto baseado em Java e uma das ferramentas mais utilizadas, por isso possui muitas opções de plug-ins. Devido à sua popularidade, é fácil encontrar suporte em sua grande base de usuários. No entanto, sua interface de usuário está desatualizada e é menos atraente do que as interfaces das outras ferramentas.

O TeamCity, desenvolvido pela ITBrains, é baseado em Java e, portanto, tem bom suporte para Java, assim como o Jenkins. Ao contrário do Jenkins, ele também oferece bom suporte para .NET. Embora seja licenciado, possui uma edição gratuita para projetos pequenos.

O Bamboo é da Attlasian e se integra bem com outras ferramentas da Attlasian. Portanto, oferece vantagens se você já está usando ferramentas da Attlasian.

Implantação

Durante essa fase, a mudança mais importante é tratar a infraestrutura como código. Com essa abordagem, a infraestrutura pode ser compartilhada, testada e controlada por versões. Desenvolvimento e produção compartilham uma infraestrutura homogênea, reduzindo problemas e bugs devido a diferentes configurações de infraestrutura.

O DevOps promove a automação desde a aplicação até a infraestrutura. Em comparação com o provisionamento manual de infraestrutura, ferramentas de gerenciamento de configuração podem reduzir a complexidade do provisionamento e da manutenção de configurações de produção, permitindo a recriação do sistema de produção nas máquinas de desenvolvimento. Essas ferramentas são um grande facilitador do DevOps, especialmente à medida que a arquitetura passa de um bloco monolítico de software para uma abordagem de microsserviços.

Para alcançar a entrega contínua para sistemas embarcados, os dispositivos devem estar conectados, portanto, a comunicação máquina a máquina e uma arquitetura de Internet das Coisas são necessárias. A entrega contínua melhorará o tempo de lançamento no mercado e permitirá práticas ágeis com feedback rápido dos consumidores. No entanto, para aplicativos críticos em que a falha não é uma opção, como aplicativos médicos ou aeroespaciais, uma abordagem mais conservadora é preferível. As seguintes ferramentas ajudarão no ciclo de vida da aplicação.

O Ansible é o mais fácil de implementar, pois não requer a instalação de agentes nas máquinas clientes, pois utiliza o SSH (Secure Shell) para enviar as configurações. É uma ferramenta de código aberto baseada em Python, mas a configuração é codificada em arquivos YAML (YAML Ain't Markup Language), o que reduz a curva de aprendizado.

A abordagem do Puppet para o provisionamento de infraestrutura consiste em descrever o estado final desejado do sistema. Uma infraestrutura baseada no Puppet geralmente consiste em um servidor mestre com agentes em cada cliente. O Puppet é baseado em Ruby, mas usa uma DSL semelhante ao JSON (JavaScript Object Notation) para representar as configurações da máquina.

A abordagem do Chef é mais sobre escrever uma receita em uma DSL puramente baseada em Ruby que detalha as etapas necessárias para provisionar o sistema. Você pode usar o Chef como uma ferramenta cliente-servidor ou uma instalação isolada no modo "chef-solo".

Por fim, ao selecionar a tecnologia de orquestração, como o Docker, no contexto do DevOps, é importante levar em consideração se a infraestrutura de produção é baseada em virtualização ou se há intenção de migrar para tecnologias de containerização. O Docker é uma ferramenta que permite a automação de código, abrangendo tarefas como criação, teste e entrega ou implantação de software. Ele é amplamente utilizado para realizar integração contínua e entrega contínua (CI/CD) no processo de desenvolvimento de software.

Operações

Embora os pesquisadores tenham se concentrado bastante na construção e implantação, o DevOps também afeta as operações de infraestrutura. Portanto, você precisa de ferramentas para manter a estabilidade e o desempenho da sua infraestrutura.

Ferramentas de registro (logging). O registro às vezes é considerado como substituto da depuração, e os programadores são aconselhados a depurar em vez de registrar para não deixar rastros em um aplicativo. Mas do ponto de vista do DevOps, os rastros são necessários para o gerenciamento de aplicativos. A regra é registrar tudo e determinar o nível do registro (por exemplo, fatal, erro, aviso, info, trace ou debug). Para aplicativos não muito complexos, você pode usar as ferramentas padrão (em Java, é o pacote java.util.logging). Para projetos mais complexos, você pode usar estruturas (frameworks) como o Log4j ou a família de ferramentas Log4j para diferentes linguagens, embora as ferramentas mais conhecidas para essas tarefas sejam os sistemas de gerenciamento de logs.

A ferramenta Loggly oferece um serviço baseado em nuvem para coletar dados de registro de aplicativos, plataformas e servidores em tempo real usando padrões abertos como HTTP ou syslog. A instalação é simples e a criação de painéis personalizados de desempenho e DevOps.

Ferramentas de registro (logging).

O Loggly também fornece uma funcionalidade de pesquisa poderosa. Ele pode ser integrado ao New Relic, tornando-o eficaz para identificar e isolar problemas e encontrar soluções.

O Graylog2 é um analisador de registros de código aberto que permite armazenar e pesquisar erros de registro. Em sua interface da web fácil de usar, você pode criar painéis ad hoc e recursos fortes de alerta. Ele lida com uma ampla variedade de formatos de dados e pode ser personalizado com um número considerável de plug-ins.

Ferramentas de monitoramento.

As ferramentas de monitoramento permitem que as organizações identifiquem e resolvam problemas na infraestrutura de TI antes que eles afetem os processos de negócio críticos. Essas ferramentas monitoram aspectos do sistema, como carga da CPU, alocação de RAM, estatísticas de tráfego de rede, consumo de memória e disponibilidade de espaço livre em disco. Elas

acompanham continuamente a saúde do sistema e alertam os administradores quando detectam problemas, para que possam tomar ações corretivas.

O Nagios é uma conhecida ferramenta de código aberto para monitorar infraestruturas de TI, como estações de usuários finais, serviços de TI e componentes ativos de rede. Ele fornece uma interface web de fácil navegação com um painel interativo que inclui uma visão geral de alto nível de hosts, serviços e dispositivos de rede. Ele fornece gráficos de tendência e planejamento de capacidade que permitem que as organizações planejem atualizações de infraestrutura. Os plug-ins resolvem algumas das limitações da ferramenta, como a falta de descoberta automática de dispositivos e a dificuldade de configuração da ferramenta.

O New Relic, baseado em software como serviço, oferece uma interface poderosa para aplicativos web e monitora aplicativos móveis nativos para iOS e Android. Você pode monitorar aplicativos em execução em ambientes híbridos, em nuvem ou locais. Os painéis personalizáveis permitem que você integre recursos adicionais de monitoramento. Você pode gerar relatórios personalizados e definir alarmes de limite para receber notificações de eventos específicos.

O Cacti é um sistema baseado na web com uma interface intuitiva e fácil de usar. Sua configuração baseada em modelos permite um alto nível de personalização. Você pode criar gráficos, que podem ser exibidos no modo de visualização em árvore, além dos modos de lista padrão e de visualização prévia.

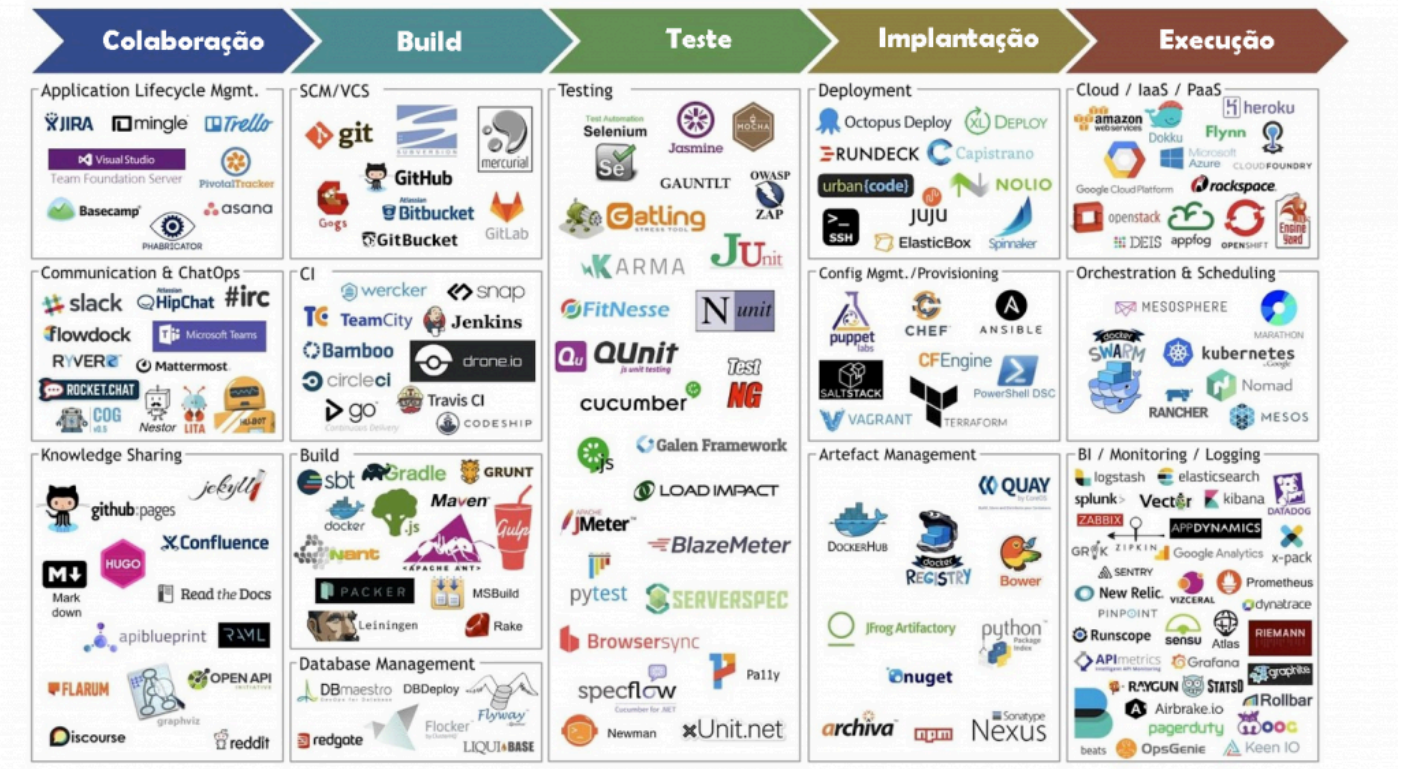
DevOps no Mundo Real

As indústrias de software e tecnologia da informação são voltadas para velocidade e eficiência. O DevOps surgiu como um paradigma para trazer produtos e recursos inovadores mais rapidamente ao mercado. Muitas tecnologias surgiram recentemente para suavizar a transição entre desenvolvimento e operações. Elas têm em comum práticas de entrega fluida, gerenciando assim a complexidade. Tecnologias para entrega rápida de aplicativos, como uma plataforma de busca ou venda, não se aplicam ao software embarcado. Mas certamente podemos aprender com essas abordagens e aplicá-las a outros domínios. Como mencionamos anteriormente, o surgimento de tecnologias de atualização rápida de software automotivo por meio de comunicação sem fio mostra que os princípios são transferíveis, levando em conta as rigorosas necessidades de segurança e continuidade funcional.

A nuvem e o desenvolvimento web têm sido adotantes precoces das práticas de DevOps e podem servir de guia para outros domínios. Por exemplo, a Amazon Web Services (AWS) oferece várias ferramentas para implementar a entrega contínua. Uma dessas ferramentas é o AWS Elastic Beanstalk, que suporta implantação contínua com uma abordagem fácil e, portanto, uma curva de aprendizado mais suave, embora com menos configurabilidade. O AWS OpsWorks oferece um caminho intermediário, no qual você pode codificar a infraestrutura; ele oferece integração com o Chef. Você também pode criar um modelo AWS CloudFormation, escrito em formato JSON, para provisionar infraestrutura de maneira repetível e controlar todos os componentes da infraestrutura na nuvem. Você pode usar o serviço AWS CodeDeploy para implantar aplicativos em várias máquinas virtuais (instâncias Elastic Compute Cloud) com um tempo de inatividade mínimo. Alternativamente, você pode usar o AWS CodePipeline. Esse serviço, lançado em 2015, integra compilação, teste e implementação.

Além disso, você pode usar outros componentes da AWS para oferecer suporte à entrega contínua. O AWS CodeCommit é um serviço de controle de origem gerenciado que hospeda repositórios privados. O AWS CloudWatch oferece uma infraestrutura de monitoramento e alerta que pode ajudar toda a equipe a trabalhar na confiabilidade e desempenho do aplicativo implantado.

Ao aproveitar essas ferramentas e práticas, as organizações podem adotar os princípios do DevOps e obter uma entrega de software mais rápida e eficiente em diversos domínios.



Testes

Após escrever o código, é muito importante verificar se tudo está funcionando corretamente. É aí que entram os testes de software. Eles são como um conjunto de verificações que garantem que o seu aplicativo está livre de erros e funciona como esperado.

Os testes são realizados para verificar se a aplicação possui erros e, caso sejam encontrados, corrigi-los. Os testes podem ser feitos de duas maneiras: manualmente, em que uma pessoa executa os testes de forma interativa, ou de forma automatizada, em que os testes são executados por meio de scripts ou ferramentas.

No contexto do DevOps, os testes integrados são fundamentais. Eles abrangem diferentes partes do sistema e têm o objetivo de garantir que todas elas funcionem corretamente em conjunto. Isso é importante porque, em um sistema complexo, diferentes componentes precisam interagir entre si, e os testes integrados garantem que isso aconteça sem problemas.

Para fazer esses testes, existem práticas como o Test-Driven Development (TDD) e o Behavior-Driven Development (BDD). Com o TDD, você escreve os testes antes mesmo de implementar o código. Isso ajuda a definir as expectativas e a criar um código mais confiável. Já o BDD se concentra em escrever testes de uma forma mais natural e legível, usando uma linguagem próxima ao negócio.

Uma vez que os testes integrados estão escritos, é possível automatizar sua execução. Isso significa que você pode criar um processo automatizado para executar os testes repetidamente, sempre que houver uma alteração no código. Isso é chamado de pipeline de integração contínua (CI). Quando o código é alterado, os testes são executados automaticamente, permitindo detectar rapidamente qualquer problema que possa ter surgido.

Além dos testes integrados, é importante também realizar outros tipos de testes. Os testes funcionais verificam se o aplicativo atende aos requisitos específicos de funcionalidade. Já os testes de integração garantem que as diferentes partes do sistema interajam corretamente entre si. Esses testes ajudam a verificar se o aplicativo funciona corretamente do começo ao fim, incluindo sua interação com outros componentes.

Quando todos esses tipos de testes estão integrados ao pipeline de CI, é possível ter um processo automatizado e contínuo de verificação da qualidade do aplicativo. Os testes são executados sempre que necessário, garantindo que o software seja testado de forma eficiente e identificando problemas o mais cedo possível. Existem também diferentes tipos de testes de software que podem ser realizados em diferentes estágios do desenvolvimento. Alguns desses testes incluem:

- Testes Unitários: São testes que verificam a funcionalidade de unidades individuais de código, como funções ou métodos. Eles são escritos pelos desenvolvedores e ajudam a garantir que cada parte do código esteja funcionando corretamente.

- Testes de Integração: São testes que verificam se as diferentes partes do sistema se integram corretamente entre si. Eles garantem que os componentes individuais funcionem em conjunto sem problemas.

- Testes de Sistema: São testes que verificam se a aplicação funciona corretamente como um todo, levando em consideração todos os seus componentes. Eles garantem que a aplicação atenda aos requisitos estabelecidos.

- Testes de Aceitação: São testes que verificam se a aplicação está pronta para ser aceita pelo cliente ou usuário final. Eles se concentram em validar se a aplicação atende aos critérios de aceitação definidos.

Existem, ainda, diversas ferramentas disponíveis para auxiliar nos testes de software. Algumas das ferramentas mais usadas incluem:

- JUnit: O JUnit é um framework de teste de unidade para a linguagem de programação Java. Ele permite que os desenvolvedores escrevam e executem testes automatizados para verificar se cada unidade de código (métodos, classes, etc.) funciona corretamente. O

JUnit é amplamente utilizado no desenvolvimento de software Java e é conhecido por sua simplicidade e eficiência.

- Karma: O Karma é uma ferramenta de execução de testes para aplicativos da web. Ele permite que os desenvolvedores executem testes em diferentes navegadores e plataformas, garantindo que o aplicativo seja compatível em todos os ambientes. O Karma é altamente configurável e suporta vários frameworks de teste, como o Jasmine.

- Jasmine: O Jasmine é um framework de teste de comportamento para JavaScript. Ele fornece uma sintaxe clara e concisa para escrever testes que verificam o comportamento esperado do código JavaScript. O Jasmine é amplamente utilizado no desenvolvimento de aplicativos web e é conhecido por sua legibilidade e facilidade de uso.

- TestNG: O TestNG é um framework de teste de unidade e integração para a linguagem de programação Java. Ele oferece recursos avançados, como a execução paralela de testes, relatórios detalhados e a capacidade de definir dependências entre testes. O TestNG é amplamente utilizado no desenvolvimento de software Java e é conhecido por sua flexibilidade e extensibilidade.

- Mocha: O Mocha é um framework de teste para JavaScript, tanto no navegador quanto no Node.js. Ele permite que os desenvolvedores escrevam testes simples e concisos usando uma sintaxe legível. O Mocha suporta a execução assíncrona de testes e oferece recursos como relatórios detalhados e geração de cobertura de código.

- Cucumber: O Cucumber é uma ferramenta de teste de comportamento que permite que os desenvolvedores escrevam testes em uma linguagem de fácil compreensão por pessoas não técnicas. Ele usa uma sintaxe baseada em linguagem natural chamada Gherkin, que descreve o comportamento do sistema em termos de cenários e passos de teste. O Cucumber é amplamente utilizado no desenvolvimento de software ágil e promove a colaboração entre desenvolvedores e partes interessadas não técnicas.

- Selenium: O Selenium é uma ferramenta de automação de testes para aplicativos da web. Ele permite que os desenvolvedores escrevam testes que interajam com o aplicativo como um usuário real, clicando em botões, preenchendo formulários e verificando resultados. O Selenium suporta vários navegadores e linguagens de programação e é amplamente utilizado no desenvolvimento de aplicativos web.

Gerenciamento de Configuração

O Gerenciamento de Configuração é um processo pelo qual você pode lidar com as alterações de forma sistemática. Esse processo garante que a integridade seja mantida o tempo todo e que o estado atual do sistema esteja em um estado conhecido e bom. As principais ferramentas utilizadas no gerenciamento de configuração são as seguintes:

- Ansible: O Ansible é uma ferramenta de TI de código aberto para gerenciar, automatizar, configurar servidores e, implantar aplicativos, a partir de uma localização central. Ele utiliza sua própria linguagem declarativa baseada em YAML para descrever as configurações do sistema e as tarefas que precisam ser executadas nos servidores. Essa linguagem de script

é fácil de entender e permite a automação de tarefas em várias máquinas. Com o Ansible, você pode automatizar várias tarefas, como a configuração de servidores, provisionamento de infraestrutura, implantação de aplicativos e muito mais. Ele facilita o gerenciamento e a manutenção da infraestrutura, permitindo que você defina as configurações desejadas e as aplique de maneira consistente em diferentes ambientes.

- Terraform: O Terraform é uma ferramenta de infraestrutura como código que permite criar, modificar e versionar a infraestrutura de forma declarativa. É uma ferramenta de software de infraestrutura como código de código aberto criada pela HashiCorp. Os usuários definem e fornecem infraestrutura de data center usando uma linguagem de configuração declarativa conhecida como HashiCorp Configuration Language ou, opcionalmente, JSON. Ele oferece suporte a várias plataformas e provedores de nuvem, permitindo a criação e gerenciamento eficiente de recursos de infraestrutura.

- Puppet: O Puppet é uma ferramenta de automação de TI que simplifica a administração de configurações e o gerenciamento de infraestrutura. É um utilitário para gerenciamento de configuração de código livre. Ele roda em muitos sistemas Unix compatíveis, bem como em Microsoft Windows; e inclui sua própria linguagem declarativa para descrever a configuração do sistema. Ele permite definir e gerenciar a configuração do sistema de maneira centralizada, garantindo a consistência e o controle da infraestrutura.

- Rudder: O Rudder é uma ferramenta de automação de TI que facilita o gerenciamento e a configuração de vários servidores. É um utilitário de gerenciamento de configuração e auditoria de código aberto para ajudar a automatizar a configuração do sistema em grandes infraestruturas de TI. O Rudder conta com um agente local leve instalado em cada máquina gerenciada. Ele oferece recursos avançados de controle de conformidade e auditoria, permitindo a implantação e manutenção eficiente de configurações em escala.

- Salt: O Salt é uma ferramenta de gerenciamento de configuração e automação que permite gerenciar e provisionar sistemas de forma eficiente. Ele oferece recursos avançados de automação e escalabilidade, tornando-o adequado para ambientes complexos e distribuídos.

Implantação

Depois que sua aplicação foi testada e está pronta para ser incorporada à produção, o próximo estágio é a implantação. Aqui, a aplicação é implantada no ambiente de produção usando diferentes ferramentas, dependendo do projeto ou da estrutura da aplicação. As principais ferramentas utilizadas para a etapa de implantação são as seguintes:

- O AWS CodeDeploy é um serviço fornecido pela Amazon Web Services (AWS) que permite automatizar as implantações de software em diferentes ambientes, como servidores da AWS, serviços de contêineres e servidores locais. Com o CodeDeploy, é possível realizar implantações de software de forma automatizada, eliminando a necessidade de processos manuais suscetíveis a erros. Isso agiliza o processo de implantação e ajuda a garantir a consistência e confiabilidade das implantações em diferentes ambientes de computação. O CodeDeploy é um serviço totalmente gerenciado pela AWS, o que significa que a infraestrutura subjacente e a escalabilidade são cuidadas

pela AWS, permitindo que as equipes se concentrem no desenvolvimento e na entrega de software de forma mais eficiente.

- GoCD é uma ferramenta de construção e lançamento de código aberto da Thoughtworks. Ela é usada no desenvolvimento de software para automatizar a entrega contínua de software. Com o GoCD, equipes e organizações podem automatizar todo o processo de construção, teste e lançamento de software, desde o momento em que o código é enviado até a implantação. A ferramenta oferece suporte a infraestrutura moderna e ajuda empresas a obter software de forma mais eficiente.

- Octopus é um pacote de software utilizado para realizar cálculos de teoria funcional de densidade (DFT) de Kohn-Sham e teoria funcional de densidade dependente do tempo (TDDFT). Ele emprega pseudopotenciais e grades numéricas de espaço real para simular sistemas unidimensionais, bidimensionais e tridimensionais. Com o Octopus, é possível calcular propriedades como polarizabilidades, suscetibilidades magnéticas, espectros de absorção e realizar simulações de dinâmica molecular usando os métodos Ehrenfest e Car-Parrinello.

Containers

Containers são um conceito que surgiu no mercado atual para construir aplicações. A containerização permitiu aos usuários construir a aplicação com a ajuda de microservices, em que todos os pacotes e bibliotecas necessários para o serviço são empacotados em um único contêiner. Algumas das ferramentas mais populares presentes no mercado atualmente são as seguintes:

- Docker: O Docker é uma plataforma de contêineres de software que utiliza virtualização de nível de sistema operacional para empacotar e distribuir aplicativos de forma isolada. Com o Docker, os aplicativos podem ser executados em um ambiente consistente e portátil em diferentes sistemas operacionais e ambientes de implantação. Ele oferece uma solução eficiente e escalável para a entrega de software em contêineres, permitindo que os aplicativos sejam executados de maneira isolada e com seus próprios recursos e configurações.

- Rancher: O Rancher é uma plataforma de gerenciamento de contêineres que oferece recursos avançados para implantar, gerenciar e orquestrar contêineres em ambientes de produção. Ele fornece uma interface amigável para gerenciar clusters de contêineres, facilitando a implantação e a escalabilidade de aplicativos baseados em contêineres.

- Azure Kubernetes: O Azure Kubernetes Service (AKS) é um serviço de orquestração de contêineres oferecido pela Microsoft Azure. Ele permite implantar, gerenciar e dimensionar facilmente aplicativos baseados em contêineres usando a plataforma Kubernetes. O AKS oferece recursos de automação e monitoramento para facilitar a implantação e a operação de clusters de contêineres no ambiente do Azure.

- Kubernetes: O Kubernetes é uma plataforma de orquestração de contêineres de código aberto que automatiza a implantação, dimensionamento e gerenciamento de aplicativos em contêineres. Projetado pelo Google e mantido pela Cloud Native Computing

Foundation, o Kubernetes permite a execução eficiente e resiliente de aplicativos baseados em contêineres, oferecendo recursos avançados como balanceamento de carga, autorecuperação e dimensionamento automático. Sua ampla adoção e diversas distribuições tornaram-no uma escolha popular para executar e gerenciar cargas de trabalho nativas da nuvem.

- AWS ECS: O Amazon Elastic Container Service (ECS) é um serviço de orquestração de contêineres totalmente gerenciado fornecido pela Amazon Web Services (AWS). Ele permite implantar, gerenciar e escalar aplicativos conteinerizados de forma eficiente, integrando-se perfeitamente ao ambiente da AWS. Com recursos avançados de segurança e a opção do Amazon ECS Anywhere, é possível executar workloads de contêineres tanto na nuvem quanto on-premises. O ECS oferece uma solução fácil de usar para executar aplicativos em contêineres, fornecendo recursos de implantação, dimensionamento e gerenciamento de clusters de contêineres de maneira eficiente. Sua integração com outros serviços da AWS simplifica ainda mais o desenvolvimento de aplicativos escaláveis e altamente disponíveis.

- Codefresh: O Codefresh é uma plataforma de CI/CD de próxima geração projetada para aplicativos modernos. Ele oferece automação desde o código até a nuvem, com builds ultrarrápidos e implantações Canary e Blue/Green GitOps. Equipes de DevOps de empresas como GoodRx, Monday.com e Deloitte dependem do Codefresh para construir e implantar seu software de maneira segura e escalável. O Codefresh é especialmente adequado para aplicativos baseados em contêineres, fornecendo ferramentas e integração com várias plataformas populares, facilitando o desenvolvimento e a implantação contínua desses aplicativos.

Orquestração de Lançamento

Conforme o nome sugere, a Orquestração de Lançamento é uma abordagem para automatizar, coordenar e gerenciar as etapas completas do pipeline de lançamento de software. Essas ferramentas ajudam a automatizar o seu pipeline de CI/CD e permitem que você aproveite ferramentas e práticas que você pode ter utilizado durante o desenvolvimento do seu software. Alguns dos softwares de orquestração de lançamento são os seguintes:

- O XebiaLabs XL Release é uma ferramenta poderosa de orquestração e automação de lançamentos que permite que as organizações gerenciem e controlem de forma eficaz o processo de implantação de software. Com o XL Release, as equipes podem coordenar e executar fluxos de trabalho complexos de lançamento, garantindo visibilidade, rastreabilidade e controle ao longo de todo o ciclo de vida do lançamento. Essa ferramenta simplifica o gerenciamento de lançamentos, fornecendo recursos necessários para realizar implantações de software eficientes e confiáveis.

- O UrbanCode Deploy é uma solução de implantação de aplicativos desenvolvida pela IBM que automatiza o processo de entrega contínua e implantação contínua (CI/CD) e oferece recursos robustos de visibilidade, rastreabilidade e auditoria. Essa ferramenta de orquestração permite a implantação eficiente de aplicativos em ambientes complexos, oferecendo recursos avançados de automação e gerenciamento de versões. Com o

UrbanCode Deploy, é possível realizar implantações rápidas e confiáveis de aplicativos em diversas plataformas e ambientes.

- O Spinnaker é uma plataforma de entrega contínua de código aberto, desenvolvida pelo Google, Netflix e outras empresas. Ele é projetado para simplificar a implantação de aplicativos em diversos ambientes e provedores de nuvem, permitindo uma entrega de software rápida e confiável. O Spinnaker oferece recursos avançados, como orquestração de implantação em vários estágios, gerenciamento de versões e integração com várias ferramentas e serviços. Com suporte para Kubernetes, Google Cloud Platform, AWS, Microsoft Azure e Oracle Cloud, o Spinnaker é uma opção versátil para organizações que buscam automação e agilidade na entrega de software.

- AWS CodePipeline é um serviço gerenciado pela Amazon Web Services (AWS) que oferece entrega contínua automatizada. Ele permite a automação do fluxo de trabalho de entrega de software, desde a integração do código até a implantação, proporcionando atualizações rápidas e confiáveis de aplicativos e infraestruturas. O CodePipeline integra-se perfeitamente a outros serviços da AWS, como o AWS CodeCommit, AWS CodeBuild e AWS CodeDeploy, oferecendo um pipeline completo de entrega contínua. Com o AWS CodePipeline, as equipes podem acelerar o processo de entrega de software, aumentar a eficiência e garantir a qualidade das atualizações.

Cloud

Cloud é o método de armazenar ou acessar seus dados pela internet, em vez de utilizar o seu próprio disco rígido. Atualmente, tudo é movido para a nuvem, executado na nuvem, acessado a partir da nuvem ou armazenado na nuvem. A aplicação ou o software que você desenvolve pode ser implantado na nuvem. Existem muitos provedores de nuvem no mercado atual, mas abaixo estão alguns provedores de nuvem populares que você pode considerar usar.

- Azure: O Azure é uma plataforma de computação em nuvem oferecida pela Microsoft. Ele fornece serviços de infraestrutura, como máquinas virtuais, armazenamento e redes, além de uma ampla gama de serviços gerenciados para desenvolvimento de aplicativos, análise de dados, inteligência artificial e muito mais. O Azure é conhecido por sua escalabilidade, confiabilidade e integração perfeita com outras ferramentas e serviços da Microsoft.

- AWS: A Amazon Web Services (AWS) é uma plataforma de computação em nuvem líder no mercado, oferecendo uma ampla variedade de serviços para ajudar empresas a construir e implantar aplicativos escaláveis e confiáveis. A AWS fornece serviços de infraestrutura, armazenamento, banco de dados, análise, inteligência artificial e muito mais. Ela é conhecida por sua flexibilidade, segurança e ampla presença global.

- Google Cloud: O Google Cloud é a plataforma de computação em nuvem oferecida pelo Google. Ele oferece serviços de computação, armazenamento, banco de dados, análise de dados, inteligência artificial e muito mais. O Google Cloud é conhecido por sua escalabilidade, desempenho e inovação, além de fornecer uma ampla gama de ferramentas e serviços para ajudar empresas a desenvolver, implantar e gerenciar aplicativos em escala.

- Lambda: O AWS Lambda é um serviço de computação sem servidor fornecido pela Amazon Web Services. Ele permite que os desenvolvedores executem código sem se preocupar com a infraestrutura subjacente. Com o AWS Lambda, é possível executar funções em resposta a eventos, como o upload de um arquivo para o armazenamento, uma solicitação HTTP ou uma atualização de banco de dados. O Lambda é conhecido por sua escalabilidade automática, pagamento por uso e fácil integração com outros serviços da AWS.

- IBM Cloud: O IBM Cloud é uma plataforma de computação em nuvem fornecida pela IBM. Ela oferece uma ampla gama de serviços, incluindo computação, armazenamento, banco de dados, inteligência artificial, análise de dados e muito mais. O IBM Cloud é conhecido por sua segurança, conformidade e capacidade de integração com ferramentas e serviços populares. Ele oferece opções flexíveis de implantação, incluindo nuvem pública, privada e híbrida, para atender às necessidades específicas das empresas.

AIOps

Operações de Inteligência Artificial ou AIOps é um termo amplo para análise de big data, inteligência artificial e outras tecnologias ou estruturas de aprendizado de máquina. Isso é usado para analisar os dados de um aplicativo utilizando conceitos como Big Data e Aprendizado de Máquina. Alguns dos tools mais comuns utilizados no mercado atual para AIOps são os seguintes:

Analytics

Analytics é usado para analisar os dados capturados por um aplicativo. Esse conjunto de ferramentas é usado principalmente para analisar e gerar relatórios inteligentes. Existem muitas ferramentas usadas para analisar os dados, mas poucas são muito populares na indústria DevOps. São elas:

- ITRS é um fornecedor líder de software de monitoramento e análise em tempo real para infraestrutura de TI, ajudando organizações a monitorar e gerenciar seus sistemas de TI e garantir alta disponibilidade e desempenho. Moogsoft é uma plataforma de AIOps que utiliza inteligência artificial e aprendizado de máquina para detectar e resolver incidentes e problemas de TI em tempo real. Logstash é um pipeline de processamento de dados de código aberto que ingere, processa e transforma dados de log para armazenamento e análise centralizados. Prometheus é um conjunto de ferramentas de monitoramento e alerta de código aberto projetado para ambientes nativos em nuvem, fornecendo uma solução flexível e escalável para coletar e analisar métricas. Fluentd é um coletor de dados de código aberto que permite a agregação e o processamento de dados de log de várias fontes, possibilitando o registro e a análise centralizados. Essas ferramentas desempenham um papel fundamental nas operações de TI modernas, ajudando as organizações a monitorar, analisar e solucionar problemas em sua infraestrutura e aplicativos de forma proativa e eficiente.

- Moogsoft é uma plataforma de análise de dados e gerenciamento de incidentes baseada em inteligência artificial. Ele usa algoritmos avançados para analisar grandes volumes de dados de log e métricas em tempo real, identificando padrões e anomalias que podem

indicar problemas ou incidentes. O Moogsoft ajuda as equipes de operações a identificar e responder rapidamente a problemas, minimizando o impacto nos serviços de TI.

- Logstash é uma ferramenta de ingestão e processamento de dados open source. Ela permite coletar, transformar e enriquecer dados de diferentes fontes em tempo real, preparando-os para análise e armazenamento posterior. O Logstash suporta várias fontes de dados, como logs de aplicativos, eventos de rede e bancos de dados, e permite a aplicação de filtros e transformações para extrair informações relevantes.

- Prometheus é um sistema de monitoramento de código aberto projetado para coletar e armazenar métricas de tempo real. Ele foi desenvolvido para ser altamente escalável e oferecer suporte a ambientes distribuídos. O Prometheus coleta métricas de diferentes fontes, como aplicativos, sistemas e serviços, permitindo a análise e visualização desses dados. Ele também possui recursos avançados, como alertas e gráficos interativos.

- Fluentd é um coletor de registros e sistema de processamento de dados. Ele permite coletar, filtrar e encaminhar registros de várias fontes para diferentes destinos, como bancos de dados, sistemas de armazenamento ou serviços de análise. O Fluentd suporta uma ampla variedade de fontes e destinos, permitindo a integração fácil com outras ferramentas e sistemas. Ele oferece flexibilidade na configuração de pipelines de dados e é conhecido por sua escalabilidade e desempenho.

Monitoramento

Quando a aplicação é lançada na produção, é essencial monitorar a aplicação para garantir que seu desempenho seja adequado, que ela seja carregada rapidamente, que todos os recursos e funcionalidades da aplicação estejam funcionando corretamente, entre outros fatores. Portanto, para monitorar continuamente as aplicações, você pode usar as seguintes ferramentas:

- Nagios é uma ferramenta de monitoramento de código aberto amplamente utilizada para monitorar a infraestrutura de TI e os serviços de rede. Ele verifica regularmente os componentes do sistema, como servidores, roteadores, switches e aplicativos, em busca de problemas de disponibilidade e desempenho. O Nagios oferece recursos como alertas em tempo real, geração de relatórios e capacidade de escalonamento para atender às necessidades de monitoramento de grandes ambientes.

- Zabbix é uma plataforma de monitoramento de código aberto que permite monitorar a disponibilidade e o desempenho de vários componentes de infraestrutura, como servidores, redes e aplicativos. Ele usa uma arquitetura distribuída para coletar dados de monitoramento em tempo real e oferece recursos avançados, como detecção de anomalias, visualizações personalizadas e geração de relatórios detalhados. O Zabbix também oferece suporte a alertas e integração com outras ferramentas de gerenciamento de TI.

- Zenoss é uma plataforma de gerenciamento de operações de TI que inclui recursos de monitoramento, descoberta automática de ativos, gerenciamento de eventos e análise de desempenho. Ele permite monitorar vários componentes de infraestrutura, como servidores, redes, dispositivos de armazenamento e aplicativos, e fornece uma visão

unificada e centralizada do ambiente de TI. O Zenoss também oferece recursos avançados, como análise preditiva, correlação de eventos e automação de tarefas de gerenciamento.

Segurança

Com o aumento do número de ameaças ou vulnerabilidades, garantir a segurança do aplicativo é um dos fatores mais importantes. Existem várias estratégias e tecnologias que você pode usar para proteger seu aplicativo contra diferentes tipos de ataques. No entanto, as principais ferramentas que você pode usar para garantir a segurança do seu aplicativo são as seguintes:

- SonarQube é uma plataforma de análise estática de código-fonte projetada para ajudar no gerenciamento da qualidade do código. Ele fornece um conjunto abrangente de ferramentas e recursos para analisar o código-fonte em várias linguagens de programação e identificar problemas de qualidade, como violações de boas práticas, bugs e vulnerabilidades de segurança. O SonarQube permite aos desenvolvedores detectar e corrigir problemas de código em um estágio inicial do ciclo de desenvolvimento, ajudando a melhorar a manutenção, a segurança e a eficiência dos aplicativos.

- Snort é um sistema de detecção de intrusões baseado em rede que monitora e analisa o tráfego em uma rede para identificar atividades maliciosas. Ele utiliza regras pré-configuradas para detectar e alertar sobre ameaças conhecidas, como ataques de negação de serviço, tentativas de invasão e atividades suspeitas. O Snort pode ser implantado em uma variedade de ambientes de rede e fornece recursos avançados, como análise de pacotes em tempo real, registro de eventos e integração com outros sistemas de segurança.

- Veracode é uma plataforma de segurança de aplicativos em nuvem que oferece análise estática e dinâmica de código para identificar e corrigir vulnerabilidades de segurança em aplicativos. Ele suporta várias linguagens de programação e fornece uma ampla gama de recursos de análise, como detecção de vulnerabilidades, análise de configuração de segurança, verificação de código malicioso e testes de penetração. O Veracode ajuda as organizações a melhorar a segurança de seus aplicativos, identificando e corrigindo falhas de segurança antes que elas sejam exploradas por atacantes.

Colaboração multidisciplinar

A colaboração multidisciplinar é uma característica fundamental da cultura DevOps. Ela envolve a quebra das barreiras tradicionais entre as equipes de desenvolvimento, operações e outras áreas envolvidas no ciclo de vida do software. Em vez de trabalhar de forma isolada, os profissionais dessas áreas colaboram de forma integrada, compartilhando conhecimentos, responsabilidades e objetivos comuns. Isso promove a comunicação eficiente, a troca de ideias e a resolução de problemas de forma ágil e colaborativa.

O teste manual no final do desenvolvimento é uma prática comum em muitos ambientes DevOps. Embora a automação de testes seja uma parte importante da abordagem DevOps, o teste manual também tem o seu lugar. Ele permite que os profissionais realizem testes exploratórios, identifiquem possíveis problemas que podem ter passado despercebidos e

validem a experiência do usuário final. O teste manual complementa os testes automatizados, proporcionando uma visão mais abrangente da qualidade do software.

A implantação contínua é outra característica marcante da cultura DevOps. Consiste em automatizar o processo de implantação de software, permitindo que as alterações sejam entregues ao ambiente de produção de forma rápida, frequente e confiável. Isso é alcançado por meio de práticas como integração contínua, entrega contínua e implantação contínua, nas quais as alterações são testadas e implantadas de maneira incremental e automatizada. A implantação contínua reduz os riscos, os gargalos e os atrasos associados às implantações tradicionais, permitindo que as equipes entreguem valor ao cliente de forma mais eficiente.

REFERÊNCIAS

Atlassian - Cinco princípios fundamentais de DevOps. Disponível em: https://www.atlassian.com/br/devops/what-is-devops.

Gene Kim, Jez Humble, John Willis, Patrick Debois. Manual de DevOps. Como Obter Agilidade, Confiabilidade e Segurança em Organizações Tecnológicas. Alta Books, 2018.

Jez Humble, David Farley. Entrega Contínua: Como Entregar Software de Forma Rápida e Confiável. Bookman, 2014.

Steve Matyas, Andrew Glover, Paul Duvall. Continuous Integration: Improving Software Quality and Reducing Risk. Addison-Wesley, 2007.

Valente, Marco Tulio. Engenharia de Software Moderna. 2022.

DevOps Lifecycle. Disponível em: https://www.educba.com/devops-lifecycle/

Ferramentas do DevOps Disponível em: https://www.atlassian.com/br/devops/devops-tools

DevOps Periodic Table. Disponível em: https://www.linkedin.com/pulse/devops-periodic-table-anurag-mor-1c

DevOps Periodic Table https://www.linkedin.com/pulse/devops-periodic-table-anurag-mor-1c

QUESTÕES COMENTADAS

DevOps

1. (CESPE – AGER-Mato Grosso– 2023). No que diz respeito aos conceitos de criptografia, à assinatura digital, aos conceitos utilizados em sistemas operacionais e às noções de DevOps, julgue o item seguinte.

   Colaboração multidisciplinar, teste manual no final do desenvolvimento e implantação contínua são algumas das características marcantes da cultura DevOps.

Comentários:

A cultura DevOps enfatiza a colaboração entre as equipes de desenvolvimento, operações e testes. Essa colaboração é necessária para garantir que o software seja desenvolvido, implantado e operado de forma eficiente e eficaz. O teste manual no final do desenvolvimento é uma prática tradicional, mas não é uma prática recomendada pelo DevOps. O DevOps enfatiza a automação de testes, pois permite que os testes sejam executados com mais frequência e eficiência. A implantação contínua é uma prática fundamental do DevOps. Ela visa garantir que as mudanças no software sejam implantadas em produção de forma rápida e segura.

Gabarito: Errado

---

2. (CESPE – AGER - Mato Grosso– 2023) Assinale a opção que apresenta exemplo de ferramenta que permite realizar automação de código — incluindo a execução de tarefas relacionadas à criação, ao teste e à entrega ou implantação de software — e, assim, realizar, no DevOps, integração contínua e entrega contínua (CI/CD).

   a) jenkins
   b) maven
   c) ansible
   d) puppet
   e) docker

Comentários:

A opção correta que apresenta um exemplo de ferramenta que permite a automação de código, incluindo a execução de tarefas relacionadas à criação, teste e entrega de software para integração contínua e entrega contínua (CI/CD) no contexto do DevOps é a opção A) Jenkins.

O Jenkins é uma ferramenta de automação de código aberto amplamente utilizada no DevOps. Ele permite criar pipelines de CI/CD, onde você pode automatizar as etapas de compilação, teste e implantação do software. Com o Jenkins, você pode configurar o fluxo de trabalho do seu projeto, realizar testes automatizados e implantar seu software de forma contínua e confiável. Ele é altamente configurável e suporta integração com várias ferramentas e serviços, tornando-o uma opção popular para automação no contexto do DevOps.

Já conhece os demais itens? Quer fazer uma revisão? Então vamos lá! O Maven é uma ferramenta de automação de build. Ele é frequentemente usado em projetos Java para gerenciar dependências, compilar o código-fonte, empacotar e distribuir o software. Com o Maven, você pode definir um arquivo de configuração chamado "pom.xml" que descreve as informações do projeto, as dependências necessárias e as etapas de compilação e empacotamento. O Maven automatiza o processo de build do projeto, tornando mais fácil para os desenvolvedores gerenciarem suas dependências e criar uma estrutura consistente para o código.

O Ansible é uma ferramenta de automação que permite gerenciar e configurar sistemas e infraestrutura de forma eficiente. Ele utiliza uma linguagem simples baseada em YAML para descrever as configurações e tarefas que precisam ser executadas nos servidores. Com o Ansible, você pode automatizar várias tarefas, como a configuração de servidores, provisionamento de infraestrutura, implantação de aplicativos e muito mais. Ele facilita o gerenciamento e a manutenção da infraestrutura, permitindo que você defina as configurações desejadas e as aplique de maneira consistente em diferentes ambientes.

O Puppet é uma ferramenta de gerenciamento de configuração que ajuda a manter a consistência e a configuração correta dos servidores e outras infraestruturas de TI. Com o Puppet, você pode descrever as configurações desejadas em um formato chamado "manifests" e o Puppet se encarrega de aplicar essas configurações em todos os servidores definidos. Ele automatiza tarefas como a instalação de pacotes, configuração de arquivos de configuração e garantia de que os servidores estejam de acordo com as políticas e padrões estabelecidos. O Puppet facilita o gerenciamento e a manutenção da infraestrutura em larga escala.

O Docker é uma plataforma que permite empacotar um aplicativo e suas dependências em um contêiner isolado. O contêiner é uma unidade autossuficiente que contém tudo o que o aplicativo precisa para ser executado, como código, bibliotecas e configurações. Com o Docker, você pode criar contêineres consistentes que podem ser executados em qualquer máquina que suporte Docker, independentemente do ambiente subjacente. Isso facilita a implantação consistente e repetível de aplicativos em diferentes ambientes, desde o desenvolvimento até a produção. O Docker é amplamente usado no contexto do DevOps para facilitar a entrega contínua, pois permite que os aplicativos sejam implantados de maneira rápida, fácil e confiável, garantindo a consistência entre os ambientes de desenvolvimento, teste e produção.

Gabarito: Letra A

3. (FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

a) automação;
b) colaboração ágil;
c) produção contínua;
d) integração contínua;
e) ação centrada no cliente.

Comentários:

O gabarito é a letra A - automação. A falha no princípio da automação ocorre porque o processo descrito não é totalmente automatizado. Embora o Time de Desenvolvimento de Software (TDS) siga um protocolo automatizado para gerar, testar e combinar pacotes de software, há uma etapa em que é necessário fazer uma requisição formal ao Time de Operações (TO) para executar um conjunto de testes. Essa requisição formal indica que há uma intervenção manual no processo, o que vai contra o princípio da automação do DevOps e do DevSecOps.

As demais opções estão incorretas porque não estão diretamente relacionadas à falha mencionada. Colaboração ágil (opção B): Embora a colaboração entre o Time de Desenvolvimento de Software (TDS) e o Time de Operações (TO) seja mencionada, não há indicação de uma falha específica nessa colaboração.
Produção contínua (opção C): O texto menciona que os pacotes de software são combinados separadamente e passam por testes antes de entrar em produção. Portanto, não há uma falha específica no princípio da produção contínua.
Integração contínua (opção D): Embora o texto mencione que os pacotes de software são combinados, não há informações suficientes para determinar se há uma falha no princípio da integração contínua.
Ação centrada no cliente (opção E): O texto não menciona diretamente a relação com o cliente, portanto não é possível afirmar que há uma falha no princípio da ação centrada no cliente.

Gabarito: Letra A

---

4. (CONSULPAM – ICTIM - RJ – 2023) Uma das carreiras em ascensão na área de tecnologia, é a de DevOps, responsável por acelerar a colocação da solução no mercado, manter a estabilidade e a confiabilidade do sistema, melhorar o tempo médio de recuperação, entre outras ações. Assinale a alternativa que descreve as palavras que formam o acrônimo DevOps.

a) Desenho e Operação.
b) Desenho e Otimização.
c) Desenvolvimento e Operação.
d) Desenvolvimento e Otimização.

Comentários:

O termo DevOps é um acrônimo formado pelas palavras "Desenvolvimento" e "Operação". Essas palavras representam as principais áreas envolvidas no trabalho de DevOps. O objetivo do DevOps é promover a integração e colaboração entre as equipes de desenvolvimento e operações de TI, buscando melhorar a eficiência, qualidade e velocidade no desenvolvimento e entrega de software. Assim, a alternativa correta é a letra C, "Desenvolvimento e Operação".

Gabarito: Letra C

---

5. (Instituto Consulplan – MPE-MG – 2023) Manifesto para o desenvolvimento ágil de software defende “indivíduos e interações acima de processos e ferramentas, software operacional acima de documentação completa, colaboração dos clientes acima de negociação contratual e respostas a mudanças acima de seguir um plano”. (Pressman e Maxim, 2021. P. 37.)
Considerando o exposto, analise as afirmativas a seguir.
I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir.
II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes.
III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!).

IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP.
V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato.

Está correto o que se afirma apenas em

a) I, II e V.
b) I, III e IV.
c) II, III e V.
d) III, IV e V.

Comentários:

Analisando as afirmativas: I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir. Essa afirmativa está incorreta. Embora o Scrum seja um framework ágil amplamente utilizado no desenvolvimento de software, as atividades metodológicas mencionadas não são exclusivas do Scrum. O Scrum é baseado em iterações chamadas de sprints, nas quais o trabalho é planejado, desenvolvido, testado e entregue de forma iterativa e incremental.

II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes. Essa afirmativa está correta. A Extreme Programming (XP) é uma metodologia ágil que inclui atividades de planejamento, projeto, codificação e testes como parte de suas práticas. O XP enfatiza a colaboração entre a equipe, o feedback contínuo e a entrega rápida de software funcional.

III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!). Essa afirmativa está correta. O projeto XP segue rigorosamente o princípio KISS, que enfatiza a importância de manter as coisas simples e evitar a complexidade desnecessária. O XP incentiva práticas de desenvolvimento simples e diretas para facilitar a compreensão e a manutenção do software.

IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP. Essa afirmativa está incorreta. Embora o Kanban e o XP sejam abordagens

ágeis, eles diferem em suas práticas e ênfases. As reuniões de equipe para o Kanban são focadas principalmente na visualização e no gerenciamento do fluxo de trabalho, enquanto o XP tem suas próprias reuniões específicas, como o planejamento do jogo (planning game), as reuniões de revisão e as reuniões de retrospectiva.

V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato. Essa afirmativa está correta. O DevOps combina desenvolvimento e operações, buscando integrar as equipes e os processos para acelerar a entrega de software. O fluxo de trabalho do DevOps envolve várias etapas que formam ciclos contínuos, incluindo desenvolvimento, testes, implantação e monitoramento, visando a entrega rápida e confiável do produto final. Portanto, a resposta correta é a alternativa c) II, III e V.

Gabarito: Letra C

---

6. (FUNDATEC – PROCERGS– 2023) O DevOps (desenvolvimento + operação) preza o estreitamento entre as áreas de desenvolvimento e infraestrutura através de ferramentas e metodologias, de modo que seja possível automatizar, monitorar, observar, testar e metrificar todas as etapas de desenvolvimento de software. Dentro dos processos de DevOps, que visam o aumento dessa qualidade e também a facilitação de colocar um projeto em produção, há um que é uma prática em que os times de desenvolvimento lançam novas funcionalidades de forma constante e automatizada. Quando uma nova funcionalidade é finalizada, automaticamente ela será disponibilizada no ambiente de testes e, posteriormente, no ambiente de produção e, em alguns casos, pode ir direto para produção. Assinale a alternativa que cita essa prática.

a) Design Patterns ou Padrões de Projeto.
b) Test Driven Development ou Desenvolvimento Orientado a Testes.
c) Continuous Delivery ou Entrega Contínua.
d) Behavior Driven Development ou Desenvolvimento Orientado ao Comportamento.
e) Continuous Integration ou Integração Contínua.

Comentários:

No contexto do DevOps, uma prática importante é a Entrega Contínua (Continuous Delivery). Essa prática consiste em lançar novas funcionalidades de forma constante e automatizada. Quando uma equipe de desenvolvimento finaliza uma nova funcionalidade, ela é automaticamente disponibilizada no ambiente de testes. Nesse ambiente, a funcionalidade passa por testes para garantir sua qualidade e funcionamento correto. Após ser aprovada nos testes, a funcionalidade pode ser promovida para o ambiente de produção, onde os usuários finais terão acesso a ela. A ideia por trás da Entrega Contínua é agilizar o processo de entrega de software, permitindo que as equipes entreguem novas funcionalidades de maneira rápida e eficiente. Em vez de lançar grandes atualizações em longos intervalos, a entrega contínua possibilita a disponibilização frequente de pequenas melhorias e recursos.

Gabarito: Letra C

---

7. (FUNDATEC – PROCERGS– 2023) No projeto em que você começará a trabalhar, você precisará de quatro engenheiros _____para programar o back-end (server) e front-end (client) da aplicação web. Além disso, precisará de mais um _____ para automatizar o deploy e a integração contínua da aplicação que será toda em AWS e GitHub Actions.

   Assinale a alternativa que preenche, correta e respectivamente, as lacunas do trecho acima.

   a) full-stack – engenheiro de dados
   b) full-stack – engenheiro DevOps
   c) de dados – analista de infraestrutura
   d) de dados – engenheiro DevOps
   e) full-stack – analisa de infraestrutura

Comentários:

No projeto em que você vai começar a trabalhar, serão necessários quatro engenheiros full-stack. Esses engenheiros serão responsáveis por programar tanto o back-end (parte do servidor) quanto o front-end (parte do cliente) da aplicação web. Eles devem ter habilidades para desenvolver todas as partes da aplicação.

Além dos engenheiros full-stack, será necessário mais um profissional: um engenheiro DevOps. Esse engenheiro terá a função de automatizar o processo de deploy (implantação) e a integração contínua da aplicação. Em outras palavras, ele será responsável por criar uma infraestrutura automatizada para que a aplicação seja implantada facilmente e de forma contínua. Ele utilizará ferramentas e tecnologias como AWS (Amazon Web Services) e GitHub Actions para realizar essa automação.

O engenheiro DevOps desempenha um papel importante na equipe, pois sua atuação permite que a equipe de desenvolvimento entregue o software de maneira mais eficiente, garantindo uma integração contínua suave e um ambiente de implantação estável.

Dessa forma, a resposta correta é a alternativa b) full-stack - engenheiro DevOps. Serão necessários quatro engenheiros full-stack para programar o back-end e front-end, e mais um engenheiro DevOps para automatizar o deploy e a integração contínua da aplicação, utilizando ferramentas como AWS e GitHub Actions.

Gabarito: Letra B

---

8. **(CESPE – BANRISUL – 2022)**. Julgue o item que se segue, acerca de DevOps.

> Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, construir, testar, codificar, operar, avaliar e relatar.

Comentários:

Pessoal, está errada a questão! Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, desenvolver, construir, testar (DEV)(em inglês: Plan, code, build, test), Release/ lançamento ou liberação (início da fase ops) implantar, operar, monitorar (em inglês: release, deploy, operate, monitor) (OPS). Esses processos representam as etapas do ciclo de vida do desenvolvimento e operação de software no contexto do DevOps.

Gabarito: Errado

---

9. **(CESPE – BNB – 2022)**. Considerando a figura a seguir, julgue o próximo item, acerca dosconceitos de DevOps.

A entrega contínua (CD) no DevOps é o processo de automatização que inclui a configuração e implantação de um aplicativo em um pipeline de produção, mas não abrange a compilação e o teste.

Comentários:

Errado. A entrega contínua (Continuous Delivery - CD) no DevOps abrange não apenas a configuração e implantação de um aplicativo, mas também a compilação e o teste automatizados. O objetivo da entrega contínua é garantir que o software seja entregue de maneira rápida, confiável e consistente, eliminando o trabalho manual e reduzindo o risco de erros humanos.

No contexto do DevOps, a entrega contínua envolve a automatização de todo o processo de desenvolvimento de software, desde o momento em que o código é escrito até o momento em que é implantado em produção. Isso inclui a compilação automatizada do código-fonte, a execução de testes automatizados para garantir a qualidade do software, a criação de artefatos de implantação e a implantação automatizada em ambientes de teste, homologação e produção.

A entrega contínua no DevOps é baseada na ideia de pipelines de entrega, que são fluxos de trabalho automatizados que movem o software através das diferentes etapas do processo de desenvolvimento e implantação. Esses pipelines são configurados para executar automaticamente todas as etapas necessárias, como compilação, teste, empacotamento e implantação, sem intervenção manual.

Gabarito: Errado

---

10. **(CESPE – Petrobras – 2022)**. Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

Comentários:

Na prática, muitas organizações utilizam sistemas de controle de versão, como o Git, para automatizar processos de DevOps. Esses sistemas permitem que equipes de desenvolvedores trabalhem em conjunto em um mesmo projeto, realizando alterações no código-fonte de forma simultânea. Com a automação, é possível evitar conflitos e preservar o histórico do código.

Um exemplo dessa automação ocorre na integração contínua. Sempre que são feitas alterações no repositório principal do código-fonte, os testes automatizados são executados. Isso garante que, a cada mudança realizada, os testes necessários sejam executados de forma rápida e eficiente.

Dessa forma, a automação dos testes permite que a equipe de desenvolvimento mantenha a qualidade do software, identificando erros e falhas de forma ágil. Além disso, ao automatizar os testes, é possível ter um feedback constante sobre o estado do código, garantindo que ele permaneça funcional e livre de problemas. Essa prática contribui para a eficiência do processo de desenvolvimento, uma vez que os testes são executados de maneira automática e contínua, sem a necessidade de intervenção manual a cada alteração no código-fonte.

Gabarito: Correto

---

11. (CESPE – TRT - 8ª Região (PA e AP) – 2022). No contexto de DevOps e DevSecOps, o Proxy reverso
[+conteúdo]
a) permite que diferentes servidores e serviços apareçam como se fossem uma única unidade, ocultando servidores atrás do mesmo nome.
b) não permite o balanceamento de carga para distribuir o tráfego de entrada, uma vez que essa tarefa é realizada nativamente por um firewall.
c) é um servidor que reside na frente de um ou mais clientes, interceptando solicitações internas e externas de servidores web.
d) garante que os clientes se comuniquem diretamente com um servidor de origem na Web.
e) não permite criptografar e descriptografar comunicações SSL (ou TLS) para cada cliente.

Comentários:

No contexto de DevOps e DevSecOps, o proxy reverso é uma ferramenta importante que desempenha um papel fundamental na arquitetura de sistemas. Vamos entender melhor! O proxy reverso permite que diferentes servidores e serviços sejam vistos como uma única unidade. Isso significa que, do ponto de vista do cliente, todos esses servidores estão ocultos por trás de um único nome. Essa abstração simplifica o acesso aos serviços e facilita a sua gestão.

Diferentemente de um firewall, que nativamente não realiza o balanceamento de carga, o proxy reverso pode ser configurado para distribuir o tráfego de entrada entre os servidores de forma balanceada. Isso ajuda a otimizar o desempenho e garantir que cada servidor não seja sobrecarregado com uma carga excessiva de requisições. Na prática, o proxy reverso é um servidor posicionado à frente de um ou mais servidores web. Ele intercepta as solicitações dos clientes, tanto internos quanto externos, e as encaminha para o servidor correto. Dessa forma, o cliente se comunica diretamente com o proxy reverso, que então direciona a solicitação para o servidor de origem apropriado.

Além disso, o proxy reverso também pode ser configurado para criptografar e descriptografar as comunicações SSL (ou TLS) para cada cliente, proporcionando uma camada adicional de segurança na transmissão de dados.

Gabarito: Letra A

12. (CESPE – BANRISUL – 2022). Julgue o item a seguir, relativos à gestão de configuração DevOps e CI/CD.

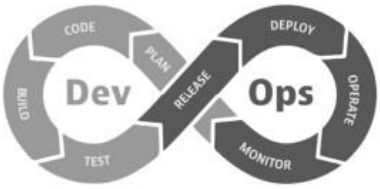
A integração contínua, a entrega contínua e a infraestrutura como código estão entre as melhores práticas de DevOps.

Comentários:

DevOps tem algumas práticas que ajudam na entrega de software de forma colaborativa e eficiente. Três delas são: integração contínua, entrega contínua e infraestrutura como código. A integração contínua é uma prática que envolve a integração frequente e automática de alterações no código de uma equipe de desenvolvimento. Isso significa que, sempre que um desenvolvedor faz uma alteração no código, essa alteração é integrada a um repositório compartilhado. Com a integração contínua, é possível identificar problemas de integração mais cedo, reduzindo o tempo necessário para detectar e corrigir erros. A entrega contínua é uma extensão da integração contínua. Ela envolve a automatização do processo de entrega de software para ambientes de teste ou produção. Com a entrega contínua, as alterações de código são testadas e disponibilizadas para os usuários finais de forma rápida e confiável. Isso permite um ciclo de feedback mais curto e uma resposta mais ágil às necessidades dos usuários. A infraestrutura como código (IaC) é uma prática que envolve a definição e gerenciamento da infraestrutura de um sistema de software por meio de código. Em vez de configurar manualmente servidores e recursos de infraestrutura, a infraestrutura é tratada como código, permitindo sua automação e reprodução consistente. O uso de ferramentas como o Git facilita a gestão e o versionamento do código de infraestrutura, permitindo que a infraestrutura seja tratada como uma parte integral do processo de desenvolvimento.

Gabarito: Correto

13. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



Com base nas etapas do DevOps, é correto afirmar que a ferramenta Jenkins está mais relacionada à etapa monitor que à etapa deploy.

Comentários:

O Jenkins está mais relacionado à etapa de integração contínua (CI) do DevOps, em que ocorre a compilação, teste e integração do código. Ele é utilizado para automatizar tarefas como compilação, teste e criação de artefatos de software. O Jenkins é uma ferramenta altamente personalizável e flexível, sendo amplamente utilizado devido à sua extensibilidade e suporte a plugins e integrações. Embora possa fornecer recursos para monitorar o processo de construção e entrega do software, seu principal foco está na etapa de integração contínua, não na etapa de monitoramento. Portanto, a afirmação de que o Jenkins está mais relacionado à etapa de monitoramento do que à etapa de deploy está incorreta.

Gabarito: Errado

---

14. (CESPE – TCE-RJ – 2022). Acerca de arquitetura de TI, DevOps e COBIT 2019, julgue o item subsequente.

Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, são relevantes as ferramentas para gerenciamento de controle de fontes, trabalho colaborativo e planejamento de projetos.

Comentários:

O item está correto. Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, como IDEs (Integrated Development Environments) e ferramentas de automação de testes, são relevantes as ferramentas para gerenciamento de controle de fontes (como sistemas de controle de versão), trabalho colaborativo (como plataformas de colaboração e comunicação) e planejamento de projetos (como ferramentas de gestão de projetos e rastreamento de problemas). Por exemplo, um sistema de controle de versão é uma ferramenta essencial para rastrear e gerenciar as alterações feitas no código ao longo do tempo, garantindo que as versões mais recentes estejam disponíveis para todos os membros da equipe. Plataformas de colaboração e comunicação também são importantes, pois permitem que os desenvolvedores compartilhem ideias, discutam problemas e trabalhem juntos de forma eficiente.

Além disso, ferramentas de gestão de projetos e rastreamento de problemas são úteis para organizar e acompanhar as tarefas do projeto, garantindo que o trabalho seja planejado e executado de maneira eficaz. Essas ferramentas desempenham um papel fundamental na promoção da colaboração entre os membros da equipe, no acompanhamento das mudanças no código, no gerenciamento de tarefas e no suporte às práticas ágeis e de integração contínua.

Gabarito: Correto

---

15. (CESPE – TRT - 8ª Região (PA e AP) – 2022). A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.
b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.
c) Os testes unitários são testes de caixa preta com cada função que compõe o software.
d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.
e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

Comentários:

No contexto de DevOps e DevSecOps, o Test Driven Development (TDD) é uma prática em que os testes unitários são criados antes mesmo da implementação do código de produção. Isso significa que os desenvolvedores escrevem os testes para orientar o desenvolvimento do código. O objetivo principal do TDD é melhorar a qualidade dos testes unitários, garantindo que eles cubram adequadamente o código.

Ao utilizar o TDD, os desenvolvedores primeiro escrevem os testes, especificando o comportamento desejado do sistema. Em seguida, eles implementam o código necessário para fazer esses testes passarem. Dessa forma, os testes atuam como um guia para a implementação do código.

Essa abordagem traz várias vantagens. Em primeiro lugar, ajuda a garantir que o código seja mais confiável, pois é desenvolvido com base em testes bem definidos. Além disso, o TDD contribui para a detecção precoce de problemas, uma vez que os testes são criados antes da implementação do código. Isso permite que os desenvolvedores identifiquem e corrijam possíveis erros mais cedo no processo de desenvolvimento.

Outro benefício importante é que o TDD facilita a manutenção e a evolução do código. Como os testes estão sempre presentes e são executados automaticamente, é mais seguro fazer alterações e adicionar novas funcionalidades, pois os testes garantem que as partes existentes do sistema não sejam afetadas negativamente.

Gabarito: Letra D

---

16. (CESPE – APEX Brasil – 2022). É(são) etapa(s) do DevOps

I Imersão.

II Ideação.

III Prototipação.

Assinale a opção correta.

a) Nenhum item está .
b) Apenas os itens I e II estão s.
c) Apenas os itens I e III estão s.
d) Apenas os itens II e III estão s.

Comentários:

A opção correta é a letra A: Nenhum item está correto.

As etapas mencionadas (Imersão, Ideação e Prototipação) não são etapas específicas do DevOps. Elas são mais comumente associadas a processos de desenvolvimento de produtos, como o Design Thinking, em que se busca entender as necessidades dos usuários, gerar ideias e criar protótipos para validar conceitos.

Por outro lado, o DevOps é focado na integração e colaboração entre equipes de desenvolvimento e operações de TI, com o objetivo de entregar software de forma contínua, eficiente e com qualidade. As principais etapas do DevOps incluem Integração Contínua, Entrega Contínua e Operação Contínua. Isso envolve práticas como automação de processos, monitoramento constante e feedback contínuo para garantir um ciclo de desenvolvimento ágil e efetivo.

Gabarito: Letra A

---

17. (CESPE – APEX Brasil– 2022). São tarefas do DevOps

o(a)I monitoramento (monitor).

II planejamento (plan).

III codificação (code).

Assinale a opção correta.

a) Apenas os itens I e II estão s.
b) Apenas os itens I e III estão s.
c) Apenas os itens II e III estão s.
d) Todos os itens estão s.

Comentários:

A opção correta é a letra D: Todos os itens estão corretos. No contexto do DevOps, todas as tarefas mencionadas - monitoramento, planejamento e codificação - são consideradas atividades importantes e fazem parte das responsabilidades do DevOps.

O monitoramento é fundamental para garantir a disponibilidade, desempenho e segurança dos sistemas em produção. Isso envolve a coleta de dados, análise de métricas e monitoramento contínuo para identificar problemas e tomar medidas corretivas.

O planejamento é necessário para definir objetivos, prioridades e estratégias de implantação do software. Isso inclui o planejamento de releases, a definição de cronogramas, a coordenação de recursos e a gestão de mudanças.

A codificação refere-se ao desenvolvimento e manutenção do código-fonte do software. Os profissionais de DevOps podem estar envolvidos na criação, integração e automação de pipelines de desenvolvimento, além de implementar práticas de integração contínua e entrega contínua.

Gabarito: Letra D

---

18. (CESPE – BNB– 2022). A respeito de DevOps, julgue o item subsequente.

A organização que investir em DevOps deve estar preparada para automatizar seus processos mediante a execução de scripts pré-definidos.

Comentários:

O item está correto. Quando uma organização decide adotar a abordagem do DevOps, é importante que esteja pronta para automatizar seus processos usando scripts pré-definidos.

Automatizar processos é uma parte fundamental do DevOps, pois permite que tarefas repetitivas e demoradas sejam executadas de forma rápida, consistente e confiável. Ao criar scripts pré-definidos, a organização pode especificar as etapas necessárias para realizar uma determinada atividade, como compilar e implantar software, configurar ambientes ou executar testes.

Esses scripts podem ser executados repetidamente, garantindo a consistência nos processos e reduzindo a possibilidade de erros humanos. Além disso, a automação proporciona maior agilidade, permitindo que a organização responda rapidamente a mudanças e demandas do mercado.

Ao investir em DevOps, é essencial que a organização esteja preparada para adotar a automação como parte integrante de seus processos. Isso significa que os colaboradores devem estar dispostos a aprender e usar ferramentas de automação, além de criar e manter os scripts necessários para automatizar as atividades relevantes.

Gabarito: Correto

---

19. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.

A ferramenta RedHat Ansible está mais relacionada à etapa deploy do que à etapa plan.

Comentários:

A ferramenta RedHat Ansible está mais relacionada à etapa de implantação (deploy) do que à etapa de planejamento (plan) no contexto do DevOps. O Ansible é uma ferramenta de automação de TI que permite realizar a implantação automatizada de aplicativos e configurações em diferentes ambientes de produção.

Com o Ansible, é possível criar playbooks que descrevem as tarefas e configurações necessárias para a implantação de sistemas. Esses playbooks podem ser executados para provisionar e configurar servidores, instalar pacotes, configurar redes, entre outras ações relacionadas à implantação de aplicativos.

Gabarito: Correto

---

20. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

O repositório de artefatos armazena artefatos de construção produzidos por integração contínua e os disponibiliza para implantação automatizada em ambientes de teste, preparação e produção.

Comentários:

O item está correto porque um repositório de artefatos desempenha um papel essencial no processo de integração contínua e implantação automatizada. Ele armazena os artefatos de construção, que são os resultados do processo de compilação, como pacotes de distribuição e arquivos WAR. Esses artefatos são produzidos pela integração contínua e são disponibilizados no repositório para serem implantados automaticamente em diferentes ambientes, como testes, preparação e produção. O repositório de artefatos centraliza o armazenamento desses artefatos, facilitando o acesso e a implantação em diferentes estágios do ciclo de vida do software.

Além disso, o repositório de artefatos oferece uma forma de rastrear e controlar as versões dos artefatos, garantindo que as versões corretas sejam implantadas nos ambientes apropriados. Ele também permite a automatização do processo de implantação, permitindo que as equipes de desenvolvimento e operações automatizem a implantação de artefatos em diferentes ambientes, reduzindo erros e garantindo consistência. Em resumo, o repositório de artefatos desempenha um papel fundamental na entrega contínua de software, armazenando e disponibilizando os artefatos de construção para implantação automatizada em ambientes de teste, preparação e produção.

Gabarito: Correto

---

21. **(CESPE – APEX Brasil – 2022)**. No DevOps, as mudanças feitas pelo desenvolvedor na soluçãode software, nas quais são feitos testes contra erros para depois serem enviadas a um repositório de versionamento de códigos, como o GitHub, representam a etapa de

a) deploy.
b) entrega contínua.
c) integração contínua.
d) operação.

Comentários:

A letra B é o gabarito correto porque a entrega contínua (continuous delivery) representa a etapa em que as mudanças feitas pelo desenvolvedor são testadas automaticamente e disponibilizadas em um repositório de código, como o GitHub. A entrega contínua visa minimizar o esforço na implantação de novos códigos, garantindo que as mudanças sejam testadas e prontas para serem implantadas em um ambiente de produção.

A entrega contínua tem como objetivo principal automatizar o processo de implantação, evitando atrasos causados por procedimentos manuais e garantindo uma entrega rápida e eficiente das aplicações aos clientes. Com a entrega contínua, as mudanças podem ser liberadas automaticamente para produção, sem a necessidade de intervenção manual, o que reduz a carga de trabalho das equipes de operações e acelera o ciclo de desenvolvimento.

Gabarito: Letra B

---

22. (FGV – TRT - 16ª REGIÃO (MA) – 2022) Considerando o DevOps e suas boas práticas,analise os itens a seguir:

I. Testes integrados são uma parte importante do processo DevOps. Esses testes devem levar em consideração as práticas de Test-Driven Development e Behavior-Driven Development, dessa forma a execução automática desses testes pode ser integrada ao pipeline de CI. No entanto, é importante integrar outros tipos de testes, como testes funcionais ou testes de integração, que permitem que o aplicativo seja testado funcionalmente do início ao fim com os outros componentes do seu ecossistema.
II. Recomenda-se automatizar apenas as tarefas críticas que envolvam poucas atualizações na implementação e nos testes dos aplicativos nas infraestruturas. Essas tarefas devem ser automatizadas em scripts que podem ser facilmente integradas e executadas em pipelines de CI/CD.
III. A construção de pipelines de CI/CD envolvem a escolha de ferramentas de DevOps adequadas pelas equipes considerando a natureza da empresa. E necessário levar em conta aspectos financeiros, avaliar entre ferramentas de código aberto e gratuitas e as proprietárias, que são mais ricas em recursos e suporte, mas exigem um investimento significativo.

Está correto apenas o que se afirma em

a) I e III.
b) I.
c) I e II.
d) II e III.
e) II.

Comentários:

A afirmação I está correta ao mencionar que os testes integrados são uma parte importante do processo DevOps. Esses testes garantem que as diferentes partes do sistema

funcionem corretamente em conjunto, permitindo testar funcionalidades do início ao fim, com integração aos outros componentes do ecossistema. Além disso, a afirmação menciona as práticas de Test-Driven Development (TDD) e Behavior-Driven Development (BDD), que são abordagens de desenvolvimento de software que incentivam a escrita de testes antes mesmo da implementação do código. A execução automática desses testes pode ser integrada ao pipeline de integração contínua (CI), garantindo que os testes sejam executados de forma automatizada e contínua.

A afirmação II está incorreta ao mencionar que apenas as tarefas críticas com poucas atualizações na implementação e nos testes dos aplicativos devem ser automatizadas. Na prática DevOps, a automação é um princípio fundamental, e recomenda-se automatizar o máximo possível de tarefas, independentemente de serem críticas ou não. Isso inclui tarefas como a compilação do código, execução de testes, implantação em ambientes de teste ou produção, provisionamento de infraestrutura, entre outras. A automação dessas tarefas é importante para garantir eficiência, consistência e agilidade no processo de entrega contínua (CI/CD).

A afirmação III está correta ao mencionar que a construção de pipelines de CI/CD envolve a escolha de ferramentas adequadas pelas equipes, considerando a natureza da empresa. É importante levar em conta aspectos financeiros ao escolher entre ferramentas de código aberto e gratuitas, assim como ferramentas proprietárias que possuem recursos e suporte mais abrangentes, mas geralmente exigem um investimento significativo. A escolha das ferramentas certas é fundamental para construir um pipeline de CI/CD eficiente e adaptado às necessidades da organização. Portanto, nosso gabarito é a opção A: I e III.

Gabarito: Letra A

---

23. **(FGV – CGU– 2022)** A equipe de redes de um órgão público está trabalhando para auxiliar no cumprimento das metas da equipe de desenvolvimento de sistemas do mesmo órgão e vislumbrou a possibilidade de utilização de DevOps. Para tal, a equipe de redes indicou a contratação de uma API em uma nuvem. A API indicada permite que os desenvolvedores e os administradores dos sistemas interajam com a infraestrutura de modo programático e em escala, evitando a instalação e a configuração dos recursos manualmente todas as vezes que precisam recriar um ambiente de desenvolvimento. Para essa atividade, a equipe de desenvolvimento utilizou a prática DevOps de:

a) comunicação e colaboração;

b) integração contínua;
c) entrega contínua;
d) microsserviços;
e) infraestrutura como código.

Comentários:

No cenário descrito, a equipe de redes do órgão público está colaborando com a equipe de desenvolvimento de sistemas para implementar o DevOps, uma abordagem que visa melhorar o processo de criação e gerenciamento dos ambientes de desenvolvimento. Para isso, eles propuseram a utilização de uma API em nuvem, que permite aos desenvolvedores e administradores interagir com a infraestrutura de forma programática, evitando a instalação e configuração manual dos recursos a cada criação de ambiente.

A prática de DevOps que está sendo aplicada nesse caso é a infraestrutura como código. Essa abordagem consiste em definir e gerenciar a infraestrutura utilizando scripts e configurações em vez de realizar as tarefas manualmente. Dessa forma, os desenvolvedores e administradores podem escrever o código que descreve a infraestrutura desejada e utilizar ferramentas que automatizam a criação e modificação dos recursos conforme necessário.

A infraestrutura como código traz diversos benefícios, como automação, consistência e escalabilidade. Ao tratar a infraestrutura como um código versionado, é possível acompanhar as alterações, realizar testes e manter um histórico das configurações. Isso torna o processo de criação e gerenciamento dos ambientes de desenvolvimento mais eficiente, confiável e facilita a colaboração entre as equipes de desenvolvimento e operações.

Gabarito: Letra E

---

24. (FGV – Senado Federal – 2022) Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes.

Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

Comentários:

A opção correta que descreve a ordem do ciclo de desenvolvimento orientado a testes é: e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar. Nesse ciclo, conhecido como "Red-Green-Refactor" (Vermelho-Verde-Refatorar), o processo se inicia escrevendo um código funcional que implemente uma funcionalidade ou um novo recurso. Em seguida, escreve-se um caso de teste automatizado que valida o comportamento esperado dessa funcionalidade. O ciclo de desenvolvimento orientado a testes (TDD - Test-Driven Development) é uma prática que busca aumentar a qualidade e a confiabilidade do código por meio de testes automatizados. A sequência do ciclo é:

- Escrever um caso de teste: O desenvolvedor começa escrevendo um caso de teste que descreve o comportamento desejado para uma funcionalidade específica. Esse caso de teste deve falhar inicialmente, pois a funcionalidade ainda não está implementada.

- Escrever um código funcional mínimo: Em seguida, o desenvolvedor escreve o código mínimo necessário para que o caso de teste passe. Nesse ponto, o teste estará em estado "verde", indicando que a funcionalidade está implementada corretamente.

- Refatorar o código: Após o teste estar verde, o desenvolvedor pode refatorar o código, fazendo melhorias estruturais, eliminando duplicações, otimizando o código, entre outras ações. É importante ressaltar que, durante a refatoração, os testes automatizados devem ser executados para garantir que as mudanças não impactem o comportamento esperado da aplicação.

Esse ciclo é repetido várias vezes, em pequenos incrementos, à medida que novas funcionalidades são adicionadas ou alterações são feitas no código existente. O objetivo é garantir que o código seja testado continuamente e que todos os testes passem antes de qualquer nova alteração ser feita.

Essa abordagem traz benefícios significativos, pois os testes automatizados fornecem uma base sólida para a confiabilidade do código, permitindo identificar e corrigir problemas com maior agilidade. Além disso, o desenvolvimento orientado a testes ajuda a melhorar o design do código, pois a escrita dos testes antes da implementação incentiva a pensar sobre a interface e o comportamento esperado da funcionalidade a ser desenvolvida. Isso resulta em um código mais modular, de fácil manutenção e menos propenso a erros.

Gabarito: Letra D

---

25. (FCC – PGE-MG– 2022) A transição de DevOps para DevSecOps requer a compreensão e utilização de técnicas e práticas específicas que podem garantir a segurança do software. Uma especialista em Engenharia de Software recomendou, dentre outras, as seguintes ferramentas e tecnologias para essa transição em uma empresa:
I. É usado para verificar o código sem realmente executá-lo. Este tipo de ferramenta ajuda a encontrar vulnerabilidades em potencial no código-fonte, evitando que ocorram várias vulnerabilidades do tipo zero-day. Common Weakness Enumeration (CWE) é uma das classificações de avisos mais comuns produzidos por estas ferramentas. CWE é uma lista oficial ou dicionário de pontos fracos de segurança comuns exploráveis por invasores para obter acesso não autorizado ao sistema.
II. Da mesma forma que as ferramentas que executam testes de caixa preta, estes analisadores dinâmicos podem identificar vulnerabilidades do programa, como injeções de SQL, estouros de buffer e similares.
III. Este tipo de ferramenta analisa o comportamento do aplicativo, implementando uma análise de segurança contínua, sendo uma das tecnologias de segurança usadas em tempo de execução.

Os itens I, II e III correspondem, correta e respectivamente, a

a) RASP (Runtime Application Self-Protection) – SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing).
b) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing).
c) SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing) – DAST (Dynamic Application Security Testing).

d) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – RASP (Runtime Application Self-Protection).
e) SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing) – RASP (Runtime Application Self-Protection).

Comentários:

A transição de DevOps para DevSecOps envolve a incorporação de práticas e ferramentas de segurança no processo de desenvolvimento de software. Para auxiliar nessa transição, a especialista em Engenharia de Software recomendou algumas ferramentas e tecnologias. Vamos explicar cada uma delas de maneira mais simples:

I. Ferramentas de SAST (Static Application Security Testing): Essas ferramentas verificam o código-fonte do software em busca de vulnerabilidades sem executá-lo. É como fazer uma análise minuciosa do código para encontrar possíveis falhas de segurança. Elas ajudam a identificar vulnerabilidades antes que elas possam ser exploradas por invasores. Um exemplo de classificação comum de avisos produzidos por essas ferramentas é a CWE (Common Weakness Enumeration), que lista pontos fracos de segurança conhecidos.

II. Analisadores dinâmicos: Essas ferramentas são capazes de identificar vulnerabilidades durante a execução do programa. Elas realizam testes em tempo real e podem encontrar problemas como injeções de SQL e estouros de buffer. É como ter uma pessoa observando o programa enquanto ele está sendo executado para detectar qualquer comportamento suspeito.

III. Ferramentas de RASP (Runtime Application Self-Protection): Essas ferramentas monitoram o comportamento do aplicativo em tempo de execução. Elas analisam continuamente o aplicativo em funcionamento para detectar possíveis vulnerabilidades. É como ter um guarda de segurança que observa o comportamento do programa e toma medidas para protegê-lo contra ameaças em tempo real.

Assim, a sequência correta para a transição para o DevSecOps é utilizar ferramentas de SAST para verificar o código-fonte, seguidas por ferramentas de DAST que analisam o programa em tempo de execução, e por fim, ferramentas de RASP que monitoram continuamente o comportamento do aplicativo para protegê-lo contra ameaças.

Gabarito: Letra E

26. (FCC – TRT - 17ª Região (ES)– 2022) Para construir um pipeline como código noJenkins, um analista utilizou um arquivo de texto simples conhecido como

a) Pipejenkins que especifica o encadeamento dos estágios do processo DevSecOps.
b) Scripted que especifica o enlace crítico entre as etapas primárias DevOps.
c) JenkinsFile que normalmente contém um código Groovy Domain Specific Language.
d) JenkinsGroovy que especifica o enlace crítico usando código DomainPline.
e) Declarative que especifica o enlace crítico usando código GroovyPline.

Comentários:

Ao construir um pipeline como código no Jenkins, utilizamos um arquivo de texto chamado JenkinsFile. Essa é a opção correta (letra C). O JenkinsFile é um arquivo especial que contém instruções escritas em uma linguagem de programação chamada Groovy Domain Specific Language (DSL). O Groovy é uma linguagem poderosa e flexível usada para escrever o código do JenkinsFile. Ele nos permite definir e configurar o pipeline de integração e entrega contínua (CI/CD) como código, em vez de usar a interface gráfica do Jenkins. O JenkinsFile nos permite definir o pipeline de forma declarativa, o que significa que descrevemos os estágios, as etapas e as condições de execução de forma clara e direta. Ele nos permite organizar e encadear os estágios do processo DevSecOps de maneira adequada, garantindo que cada etapa seja executada na ordem correta.

Gabarito: Letra C

---

27. (IBFC – MGS– 2022) Quanto aos conceitos básicos de DevOps, segundo a Amazon,analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F).

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) V - F - F
b) V - V - F
c) F - V - V

d) F - F - V

Comentários:

Analisando as afirmativas:

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
Essa afirmativa é Falsa. No DevOps, busca-se lançar versões de software de forma contínua e incremental, em ciclos curtos. O objetivo é ter entregas frequentes e rápidas, em vez de ciclos longos e lineares, para atender às necessidades dos usuários de maneira ágil.

( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
Essa afirmativa é Falsa. No DevOps, a colaboração e a integração são essenciais. As equipes de desenvolvimento e operações trabalham juntas, compartilhando responsabilidades e conhecimento. A ideia é quebrar as barreiras entre as equipes para alcançar uma abordagem colaborativa e eficiente.

( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.
Essa afirmativa é Verdadeira. DevOps vai além de ser apenas uma prática ou conjunto de ferramentas. É uma abordagem que combina filosofias culturais, práticas e ferramentas. Envolve uma mentalidade de colaboração, automação, integração contínua e entrega rápida de software.

Portanto, a sequência correta é: d) F - F - V. No DevOps, não são lançadas versões de software em ciclos longos e lineares, as equipes de desenvolvimento e operações trabalham juntas, e DevOps é uma combinação de filosofias culturais, práticas e ferramentas.

Gabarito: Letra D

---

28. (FUNDEP (Gestão de Concursos) – UFJF – 2022) Considere a sequência de comandos executados com sucesso em um repositório git para implementação de uma nova funcionalidade.

```
$ git branch cadastro-funcionario
$ git checkout cadastro-funcionario
...
$ git commit -a -m “Implementação do cadastro de funcionários”
```

$ git checkout master && git merge cadastro-funcionario
$ git push

Em relação à cultura DevOps e ao controle de versão, assinale a alternativa correta.

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
c) O comando de merge foi aplicado no branch cadastro-funcionario.
d) O comando de commit pode não ter registrado todasasmudanças realizadasnaimplementação.
e) O comando de push foi realizado no branch cadastro-funcionario.

Comentários:

Em relação à cultura DevOps e ao controle de versão, podemos afirmar o seguinte:

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
Essa afirmativa está incorreta. O comando git push é usado para enviar as alterações para o servidor remoto, incluindo o branch cadastro-funcionario. Portanto, a funcionalidade foi enviada ao servidor.

b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
Essa afirmativa está correta. O comando git branch cadastro-funcionario cria um novo branch chamado cadastro-funcionario. Esse é um exemplo de utilização de feature branch, que é uma prática comum em DevOps para desenvolver novas funcionalidades em um branch separado antes de mesclá-las ao branch principal.

c) O comando de merge foi aplicado no branch cadastro-funcionario.
Essa afirmativa está incorreta. O comando git checkout master && git merge cadastro-funcionario indica que o merge foi realizado no branch master, não no cadastro-funcionario. O git checkout master muda para o branch master, e o git merge cadastro-funcionario realiza o merge do branch cadastro-funcionario no branch master.

d) O comando de commit pode não ter registrado todas as mudanças realizadas na implementação.
Essa afirmativa está correta. O comando git commit -a -m “Implementação do cadastro de funcionários” registra as alterações feitas na implementação, mas é possível que nem todas as mudanças tenham sido incluídas. Isso ocorre porque o comando -a do git commit

somente registra alterações em arquivos que já estão sob controle de versão. Se houver novos arquivos ou alterações em arquivos não rastreados, eles não serão incluídos no commit.

e) O comando de push foi realizado no branch cadastro-funcionario.
Essa afirmativa está incorreta. O comando git push foi realizado após o merge no branch master, portanto, o push foi feito no branch master, não no cadastro-funcionario.

Portanto, a alternativa correta é: b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.

Gabarito: Letra B

---

29. (FEPESE – FAPESC– 2022) Muitas organizações têm adotado práticas de DevOpsno desenvolvimento de software.

Assinale a alternativa correta em relação ao assunto.
a) Baseado no modelo waterfall, DevOps é ideal para auxiliar equipes a manter um ritmo com modelos de desenvolvimento e entrega acelerados, tais como integração contínua e entrega para produção contínua (CI/CD).
b) Integração Contínua (Continuous integration - CI) é uma prática de desenvolvimento de software na qual desenvolvedores regularmente atualizam o seu código (realiza um commit) em um repositório compartilhado.
c) O uso de DevOps tem como objetivo criar uma cultura de colaboração entre as equipes de desenvolvimento e de operações que permite aumentar o fluxo de trabalho completado, reduzindo a quantidade de deploys, ao mesmo tempo aumentando a estabilidade e robustez do ambiente de produção.
d) A integração contínua (Continuous Integration - CI) foca na disponibilização de blocos de código completos para um repositório em intervalos regulares de tempo. Estes blocos de Código devem sempre estar em condições de serem executados para serem testados ou colocados em produção.
e) No uso de microservices, como uma arquitetura monolítica, CD permite aos desenvolvedores serem responsáveis por partes maiores e gerenciáveis do código que implementam funcionalidades individuais e trabalhar nestas em paralelo.

Comentários:

A alternativa correta é a letra B. A integração contínua (Continuous Integration - CI) é uma prática no desenvolvimento de software em que os desenvolvedores atualizam regularmente o seu código em um repositório compartilhado, também conhecido como repositório de controle de versão. Essa prática tem como objetivo principal integrar as alterações de código feitas por diferentes desenvolvedores de forma frequente, para detectar problemas de integração o mais cedo possível.

Ao realizar um commit (atualização do código) no repositório compartilhado, o código passa por um processo de integração automática, em que são executados testes automatizados para verificar se o código integrado não causa problemas de compatibilidade ou erros no sistema. Essa abordagem permite identificar e resolver rapidamente os problemas de integração, além de promover uma maior colaboração entre os membros da equipe.

A integração contínua é uma prática fundamental no contexto do DevOps, pois possibilita a entrega rápida e frequente de novas funcionalidades, mantendo a estabilidade e qualidade do software. Com a integração contínua, as equipes podem trabalhar de forma mais ágil, diminuindo os riscos de conflitos e problemas de integração tardia.

Gabarito: Letra B

---

30. (IBADE – SEA-SC – 2022) Em um modelo DevOps existe um método para entregar aplicações com frequência aos clientes, visando integração, entrega e implantação contínuas. Chamamos esse método de:

a) git
b) pipeline
c) flowchart
d) reentrante
e) CI/CD

Comentários:

A alternativa correta é a letra e) CI/CD. No modelo DevOps, a prática de integração contínua (CI) está relacionada à integração frequente e automatizada do código fonte em um repositório compartilhado. Isso permite que as equipes trabalhem de forma colaborativa, integrando suas alterações de forma contínua e automatizada.

Já a prática de entrega contínua (CD) envolve a automatização do processo de implantação do software em ambientes de produção. Com a entrega contínua, as alterações de software podem ser entregues rapidamente aos usuários, reduzindo o tempo de espera e permitindo um feedback mais rápido.

A combinação dessas práticas, CI/CD, é essencial no DevOps, pois permite uma entrega mais ágil, confiável e frequente de software, com a integração e implantação contínuas, mantendo a qualidade e a estabilidade do sistema.

Gabarito: Letra E

---

31. (UFAC – UFAC– 2022) O administrador DevOps acessou um servidor Linux da empresa, ao utilizar o comando “df” no “i” apontou 100% de uso. Julgue a informação recebida pelo console:

a) A memória swap atingiu sua capacidade máxima
b) A memória RAM não possui espaço disponível
c) O número de conexões máximas foi atingido
d) O sistema de arquivos raiz está cheio
e) Nenhuma das alternativas está correta.

Comentários:

Ao utilizar o comando "df" com a opção "i" em um servidor Linux e obter 100% de uso, a informação recebida pelo console indica que o número de inodes disponíveis no sistema de arquivos está totalmente utilizado. Isso significa que não há mais espaço disponível para armazenar informações sobre novos arquivos e diretórios.

Portanto, a resposta correta é a opção d) O sistema de arquivos raiz está cheio. Isso significa que o diretório raiz do sistema, que geralmente é montado em "/", está com sua capacidade de armazenamento esgotada. Isso pode afetar o funcionamento do servidor, causar problemas de desempenho e impedir a criação de novos arquivos.

Gabarito: Letra D

QUESTÕES COMENTADAS

DevOps

1. (CESPE – AGER-Mato Grosso– 2023). No que diz respeito aos conceitos de criptografia, à assinatura digital, aos conceitos utilizados em sistemas operacionais e às noções de DevOps, julgue o item seguinte.

   Colaboração multidisciplinar, teste manual no final do desenvolvimento e implantação contínua são algumas das características marcantes da cultura DevOps.

2. (CESPE – AGER - Mato Grosso– 2023) Assinale a opção que apresenta exemplo de ferramenta que permite realizar automação de código — incluindo a execução de tarefas relacionadas à criação, ao teste e à entrega ou implantação de software — e, assim, realizar, no DevOps, integração contínua e entrega contínua (CI/CD).

   a) jenkins
   b) maven
   c) ansible
   d) puppet
   e) docker

3. (FGV – PGM - Niterói – 2023). Um Time de Desenvolvimento de Software (TDS) segue um protocolo automatizado para gerar, testar e combinar pacotes de software gerados separadamente. Todo software combinado precisa passar por um processo que inclui uma requisição formal ao Time de Operações (TO) de um Centro de Dados para executar um conjunto de testes, com o intuito de verificar vulnerabilidades no software antes de entrar em produção. Considerando os conceitos de DevOps e DevSecOps, o TDS e o TO estão falhando no princípio:

   a) automação;
   b) colaboração ágil;
   c) produção contínua;
   d) integração contínua;
   e) ação centrada no cliente.

4. (CONSULPAM – ICTIM - RJ – 2023) Uma das carreiras em ascensão na área de tecnologia, é a de DevOps, responsável por acelerar a colocação da solução no mercado, manter a estabilidade e a confiabilidade do sistema, melhorar o tempo médio

de recuperação, entre outras ações. Assinale a alternativa que descreve as palavras que formam o acrônimo DevOps.

a) Desenho e Operação.
b) Desenho e Otimização.
c) Desenvolvimento e Operação.
d) Desenvolvimento e Otimização.

5. (Instituto Consulplan – MPE-MG – 2023) Manifesto para o desenvolvimento ágil de software defende "indivíduos e interações acima de processos e ferramentas, software operacional acima de documentação completa, colaboração dos clientes acima de negociação contratual e respostas a mudanças acima de seguir um plano". (Pressman e Maxim, 2021. P. 37.)
Considerando o exposto, analise as afirmativas a seguir.
I. Os princípios do Scrum são empregados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades metodológicas: planejar; codificar; construir; testar; e, distribuir.
II. A Extreme Programming (programação extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento; projeto; codificação; e, testes.
III. O projeto XP segue rigorosamente o princípio KISS (keep it simple, stupid!).
IV. As reuniões de equipe para o Kanban são semelhantes àquelas realizadas na metodologia XP.
V. O DevOps combina desenvolvimento (development) e operações (operations) e seu fluxo de trabalho envolve diversas etapas que formam ciclos contínuos até que o produto desejado exista de fato.

Está correto o que se afirma apenas em

a) I, II e V.
b) I, III e IV.
c) II, III e V.
d) III, IV e V.

6. (FUNDATEC – PROCERGS– 2023) O DevOps (desenvolvimento + operação) preza o estreitamento entre as áreas de desenvolvimento e infraestrutura através de ferramentas e metodologias, de modo que seja possível automatizar, monitorar,

observar, testar e metrificar todas as etapas de desenvolvimento de software. Dentro dos processos de DevOps, que visam o aumento dessa qualidade e também a facilitação de colocar um projeto em produção, há um que é uma prática em que os times de desenvolvimento lançam novas funcionalidades de forma constante e automatizada. Quando uma nova funcionalidade é finalizada, automaticamente ela será disponibilizada no ambiente de testes e, posteriormente, no ambiente de produção e, em alguns casos, pode ir direto para produção. Assinale a alternativa que cita essa prática.

a) Design Patterns ou Padrões de Projeto.
b) Test Driven Development ou Desenvolvimento Orientado a Testes.
c) Continuous Delivery ou Entrega Contínua.
d) Behavior Driven Development ou Desenvolvimento Orientado ao Comportamento.
e) Continuous Integration ou Integração Contínua.

7. (FUNDATEC – PROCERGS– 2023) No projeto em que você começará a trabalhar, você precisará de quatro engenheiros _____ para programar o back-end (server) e front-end (client) da aplicação web. Além disso, precisará de mais um _____ para automatizar o deploy e a integração contínua da aplicação que será toda em AWS e GitHub Actions.

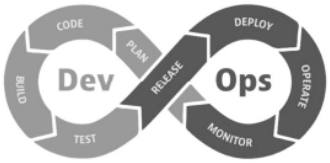
Assinale a alternativa que preenche, correta e respectivamente, as lacunas do trecho acima.

a) full-stack – engenheiro de dados
b) full-stack – engenheiro DevOps
c) de dados – analista de infraestrutura
d) de dados – engenheiro DevOps
e) full-stack – analisa de infraestrutura

8. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

Os processos envolvidos no DevOps são denominados, respectivamente, de planejar, construir, testar, codificar, operar, avaliar e relatar.

9. (CESPE – BNB – 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



A entrega contínua (CD) no DevOps é o processo de automatização que inclui a configuração e implantação de um aplicativo em um pipeline de produção, mas não abrange a compilação e o teste.

10. (CESPE – Petrobras – 2022). Julgue o item subsecutivo, relativos a DevOps e notação BPMN.

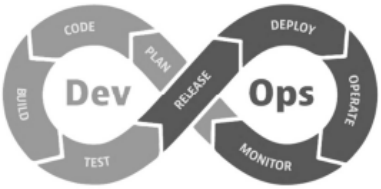
No DevOps, a integração contínua possui como uma de suas atividades a realização de testes; a fim de se obter os benefícios esperados convém automatizar os testes para poder executá-los para cada alteração feita no repositório principal.

11. (CESPE – TRT - 8ª Região (PA e AP) – 2022). No contexto de DevOps e DevSecOps, o Proxy reverso
[+conteúdo]
a) permite que diferentes servidores e serviços apareçam como se fossem uma única unidade, ocultando servidores atrás do mesmo nome.
b) não permite o balanceamento de carga para distribuir o tráfego de entrada, uma vez que essa tarefa é realizada nativamente por um firewall.
c) é um servidor que reside na frente de um ou mais clientes, interceptando solicitações internas e externas de servidores web.
d) garante que os clientes se comuniquem diretamente com um servidor de origem na Web.
e) não permite criptografar e descriptografar comunicações SSL (ou TLS) para cada cliente.

12. (CESPE – BANRISUL – 2022). Julgue o item a seguir, relativos à gestão de configuração DevOps e CI/CD.

A integração contínua, a entrega contínua e a infraestrutura como código estão entre as melhores práticas de DevOps.

13. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



Com base nas etapas do DevOps, é correto afirmar que a ferramenta Jenkins está mais relacionada à etapa monitor que à etapa deploy.

14. (CESPE – TCE-RJ – 2022). Acerca de arquitetura de TI, DevOps e COBIT 2019, julgue o item subsequente.

Em DevOps, além das ferramentas do ambiente de desenvolvimento integrado, são relevantes as ferramentas para gerenciamento de controle de fontes, trabalho colaborativo e planejamento de projetos.

15. (CESPE – TRT - 8ª Região (PA e AP) – 2022). A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.
b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.
c) Os testes unitários são testes de caixa preta com cada função que compõe o software.
d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.
e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

16. (CESPE – APEX Brasil – 2022). É(são) etapa(s) do DevOps

I Imersão.

II Ideação.

III Prototipação.

Assinale a opção correta.

a) Nenhum item está .
b) Apenas os itens I e II estão s.
c) Apenas os itens I e III estão s.
d) Apenas os itens II e III estão s.

17. (CESPE – APEX Brasil– 2022). São tarefas do DevOps o(a)

I monitoramento (monitor).

II planejamento (plan).

III codificação (code).

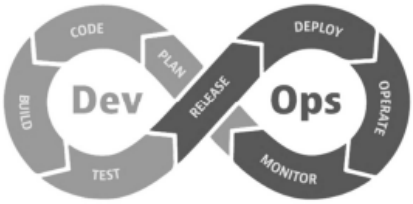
Assinale a opção correta.

a) Apenas os itens I e II estão s.
b) Apenas os itens I e III estão s.
c) Apenas os itens II e III estão s.
d) Todos os itens estão s.

18. (CESPE – BNB– 2022). A respeito de DevOps, julgue o item subsequente.

A organização que investir em DevOps deve estar preparada para automatizar seus processos mediante a execução de scripts pré-definidos.

19. (CESPE – BNB– 2022). Considerando a figura a seguir, julgue o próximo item, acerca dos conceitos de DevOps.



A ferramenta RedHat Ansible está mais relacionada à etapa deploy do que à etapa plan.

20. (CESPE – BANRISUL – 2022). Julgue o item que se segue, acerca de DevOps.

O repositório de artefatos armazena artefatos de construção produzidos por integração contínua e os disponibiliza para implantação automatizada em ambientes de teste, preparação e produção.

21. (CESPE – APEX Brasil – 2022). No DevOps, as mudanças feitas pelo desenvolvedor na solução de software, nas quais são feitos testes contra erros para depois serem enviadas a um repositório de versionamento de códigos, como o GitHub, representam a etapa de

a) deploy.
b) entrega contínua.

c) integração contínua.
d) operação.

22. (FGV – TRT - 16ª REGIÃO (MA) – 2022) Considerando o DevOps e suas boas práticas, analise os itens a seguir:

I. Testes integrados são uma parte importante do processo DevOps. Esses testes devem levar em consideração as práticas de Test-Driven Development e Behavior-Driven Development, dessa forma a execução automática desses testes pode ser integrada ao pipeline de CI. No entanto, é importante integrar outros tipos de testes, como testes funcionais ou testes de integração, que permitem que o aplicativo seja testado funcionalmente do início ao fim com os outros componentes do seu ecossistema.
II. Recomenda-se automatizar apenas as tarefas críticas que envolvam poucas atualizações na implementação e nos testes dos aplicativos nas infraestruturas. Essas tarefas devem ser automatizadas em scripts que podem ser facilmente integradas e executadas em pipelines de CI/CD.
III. A construção de pipelines de CI/CD envolvem a escolha de ferramentas de DevOps adequadas pelas equipes considerando a natureza da empresa. E necessário levar em conta aspectos financeiros, avaliar entre ferramentas de código aberto e gratuitas e as proprietárias, que são mais ricas em recursos e suporte, mas exigem um investimento significativo.

Está correto apenas o que se afirma em

a) I e III.
b) I.
c) I e II.
d) II e III.
e) II.

23. (FGV – CGU– 2022) A equipe de redes de um órgão público está trabalhando para auxiliar no cumprimento das metas da equipe de desenvolvimento de sistemas do mesmo órgão e vislumbrou a possibilidade de utilização de DevOps. Para tal, a equipe de redes indicou a contratação de uma API em uma nuvem. A API indicada permite que os desenvolvedores e os administradores dos sistemas interajam com a infraestrutura de modo programático e em escala, evitando a instalação e a configuração dos recursos manualmente todas as vezes que precisam recriar um ambiente de desenvolvimento. Para essa atividade, a equipe de desenvolvimento utilizou a prática DevOps de:

a) comunicação e colaboração;
b) integração contínua;
c) entrega contínua;
d) microsserviços;
e) infraestrutura como código.

24. (FGV – Senado Federal – 2022) Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes.

Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

25. (FCC – PGE-MG– 2022) A transição de DevOps para DevSecOps requer a compreensão e utilização de técnicas e práticas específicas que podem garantir a segurança do software. Uma especialista em Engenharia de Software recomendou, dentre outras, as seguintes ferramentas e tecnologias para essa transição em uma empresa:
I. É usado para verificar o código sem realmente executá-lo. Este tipo de ferramenta ajuda a encontrar vulnerabilidades em potencial no código-fonte, evitando que ocorram várias vulnerabilidades do tipo zero-day. Common Weakness Enumeration (CWE) é uma das classificações de avisos mais comuns produzidos por estas ferramentas. CWE é uma lista oficial ou dicionário de pontos fracos de segurança comuns exploráveis por invasores para obter acesso não autorizado ao sistema.
II. Da mesma forma que as ferramentas que executam testes de caixa preta, estes analisadores dinâmicos podem identificar vulnerabilidades do programa, como injeções de SQL, estouros de buffer e similares.

III. Este tipo de ferramenta analisa o comportamento do aplicativo, implementando uma análise de segurança contínua, sendo uma das tecnologias de segurança usadas em tempo de execução.

Os itens I, II e III correspondem, correta e respectivamente, a

a) RASP (Runtime Application Self-Protection) – SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing).
b) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing).
c) SAST (Static Application Security Testing) – IAST (Interactive Application Security Testing) – DAST (Dynamic Application Security Testing).
d) IAST (Interactive Application Security Testing) – SAST (Static Application Security Testing) – RASP (Runtime Application Self-Protection).
e) SAST (Static Application Security Testing) – DAST (Dynamic Application Security Testing) – RASP (Runtime Application Self-Protection).

26. (FCC – TRT - 17ª Região (ES)– 2022) Para construir um pipeline como código no Jenkins, um analista utilizou um arquivo de texto simples conhecido como

a) Pipejenkins que especifica o encadeamento dos estágios do processo DevSecOps.
b) Scripted que especifica o enlace crítico entre as etapas primárias DevOps.
c) JenkinsFile que normalmente contém um código Groovy Domain Specific Language.
d) JenkinsGroovy que especifica o enlace crítico usando código DomainPline.
e) Declarative que especifica o enlace crítico usando código GroovyPline.

27. (IBFC – MGS– 2022) Quanto aos conceitos básicos de DevOps, segundo a Amazon, analise as afirmativas abaixo, dê valores Verdadeiro (V) ou Falso (F).

( ) Lança-se versões de software em ciclos periódicos, lineares, mas longos.
( ) As equipes de desenvolvimento ficam separadas dos operadores de software.
( ) DevOps é a combinação de filosofias culturais, práticas e ferramentas.

Assinale a alternativa que apresenta a sequência correta de cima para baixo.

a) V - F - F
b) V - V - F

c) F - V - V
d) F - F - V

28. (FUNDEP (Gestão de Concursos) – UFJF – 2022) Considere a sequência de comandos executados com sucesso em um repositório git para implementação de uma nova funcionalidade.

$ git branch cadastro-funcionario
$ git checkout cadastro-funcionario
...
$ git commit -a -m "Implementação do cadastro de funcionários"
$ git checkout master && git merge cadastro-funcionario
$ git push

Em relação à cultura DevOps e ao controle de versão, assinale a alternativa correta.

a) A funcionalidade desenvolvida foi registrada, mas não enviada ao servidor.
b) Foi utilizada a técnica de feature branch para implementação da funcionalidade.
c) O comando de merge foi aplicado no branch cadastro-funcionario.
d) O comando de commit pode não ter registrado todasasmudanças realizadasnaimplementação.
e) O comando de push foi realizado no branch cadastro-funcionario.

29. (FEPESE – FAPESC– 2022) Muitas organizações têm adotado práticas de DevOps no desenvolvimento de software.

Assinale a alternativa correta em relação ao assunto.
a) Baseado no modelo waterfall, DevOps é ideal para auxiliar equipes a manter um ritmo com modelos de desenvolvimento e entrega acelerados, tais como integração contínua e entrega para produção contínua (CI/CD).
b) Integração Contínua (Continuous integration - CI) é uma prática de desenvolvimento de software na qual desenvolvedores regularmente atualizam o seu código (realiza um commit) em um repositório compartilhado.
c) O uso de DevOps tem como objetivo criar uma cultura de colaboração entre as equipes de desenvolvimento e de operações que permite aumentar o fluxo de trabalho completado, reduzindo a quantidade de deploys, ao mesmo tempo aumentando a estabilidade e robustez do ambiente de produção.

d) A integração contínua (Continuous Integration - CI) foca na disponibilização de blocos de código completos para um repositório em intervalos regulares de tempo. Estes blocos de Código devem sempre estar em condições de serem executados para serem testados ou colocados em produção.
e) No uso de microservices, como uma arquitetura monolítica, CD permite aos desenvolvedores serem responsáveis por partes maiores e gerenciáveis do código que implementam funcionalidades individuais e trabalhar nestas em paralelo.

30. (IBADE – SEA-SC – 2022) Em um modelo DevOps existe um método para entregar aplicações com frequência aos clientes, visando integração, entrega e implantação contínuas. Chamamos esse método de:

a) git
b) pipeline
c) flowchart
d) reentrante
e) CI/CD

31. (UFAC – UFAC– 2022) O administrador DevOps acessou um servidor Linux da empresa, ao utilizar o comando "df" no "i" apontou 100% de uso. Julgue a informação recebida pelo console:

a) A memória swap atingiu sua capacidade máxima
b) A memória RAM não possui espaço disponível
c) O número de conexões máximas foi atingido
d) O sistema de arquivos raiz está cheio
e) Nenhuma das alternativas está correta.

GABARITO

1. Errado
2. Letra A
3. Letra A
4. Letra C
5. Letra C
6. Letra C
7. Letra B
8. Errado
9. Errado
10. Correto
11. Letra A
12. Correto
13. Errado
14. Correto
15. Letra D
16. Letra A
17. Letra D
18. Correto
19. Correto
20. Correto
21. Letra B
22. Letra A
23. Letra E
24. Letra D
25. Letra E
26. Letra C
27. Letra D
28. Letra B
29. Letra B
30. Letra E
31. Letra D

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.