



v1.0

Treinamento Oficial EXIN DevOps Master





Henrique Bueno
henrique@estabil.is





Standard
Consulting
Partner

Channel Partner





A Certificação Exin DevOps Master



O foco desta certificação é aumentar o conhecimento, juntamente com habilidades práticas, permitindo que você atue como um facilitador de práticas de sucesso para o DevOps em uma equipe e consiga promover seus princípios na organização.



A certificação EXIN DevOps Master destina-se a quem trabalha em uma equipe DevOps ou em uma organização que considera a possibilidade de passar pela transição para implementar e trabalhar com DevOps.

O público alvo inclui:

- Desenvolvedores e profissionais técnicos que atuam com desenvolvimento de software
- Product Owners
- Scrum Master,
- Gerentes de Projetos
- Engenheiros e Gerentes de Testes
- Gerentes de Serviços de TI
- Gerentes de Processo
- Praticantes de Lean IT

Tipo de exame: Pergunta de múltipla escolha

Número de perguntas: 50

Nota de aprovação: 65 (33 questões)

Tempo designado para o exame: 120 minutos

Idioma: Português

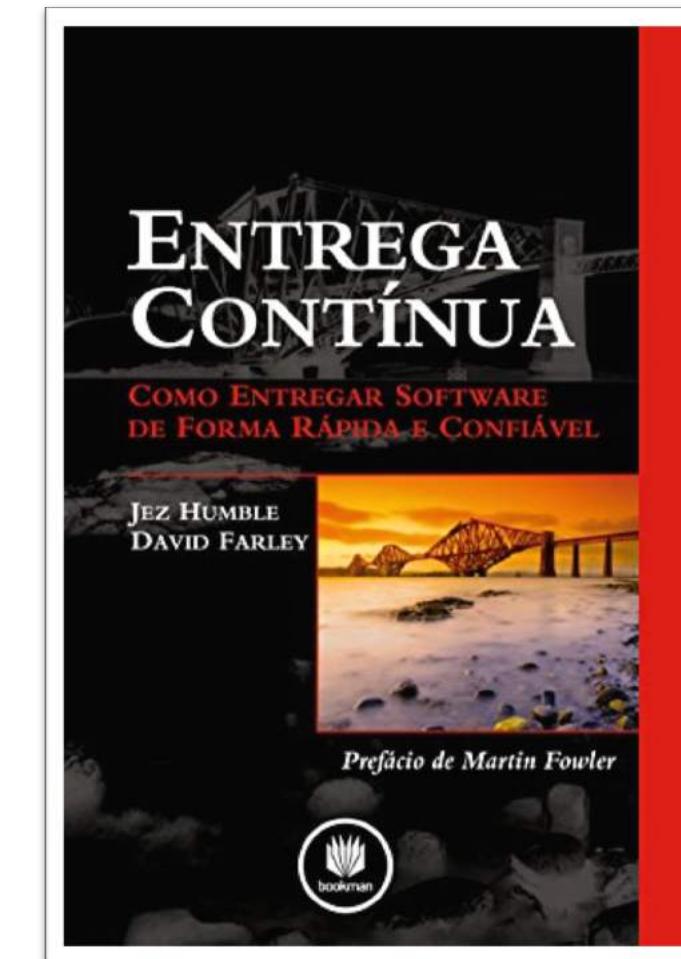
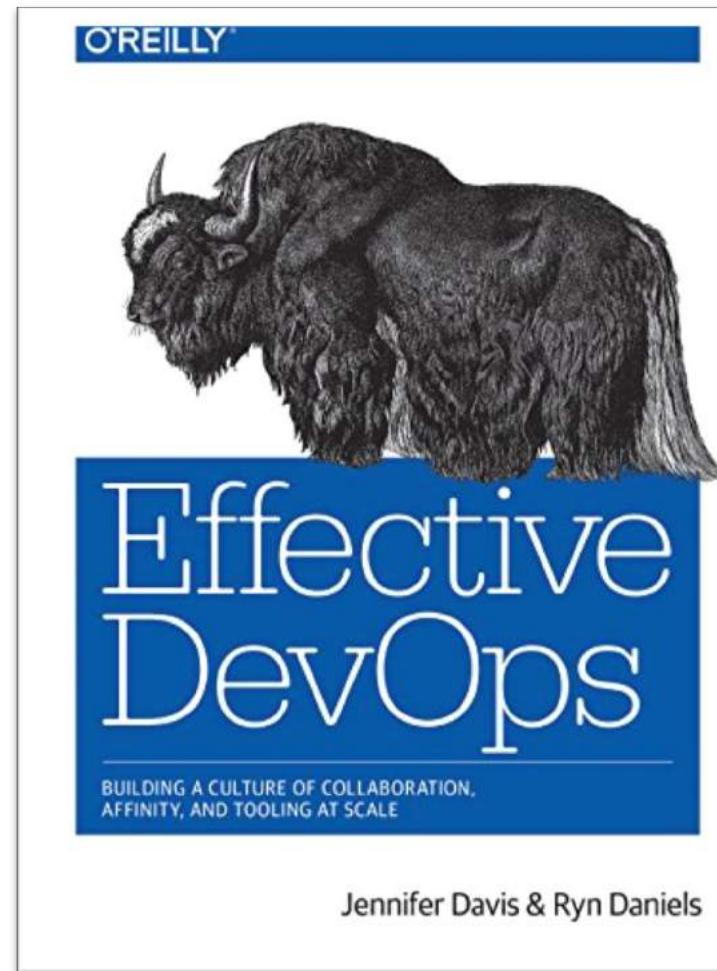
Aplicação do exame: Ao final do treinamento

Requisitos para a certificação

1. Treinamento DevOps Master credenciado, incluindo a conclusão bem-sucedida das Atribuições práticas
2. Realização bem-sucedida do exame DevOps Master

O treinamento é uma parte obrigatória da certificação. Os candidatos deverão ter conhecimentos básicos de princípios DevOps e de conceitos Lean e Agile. Este conhecimento pode ser adquirido:

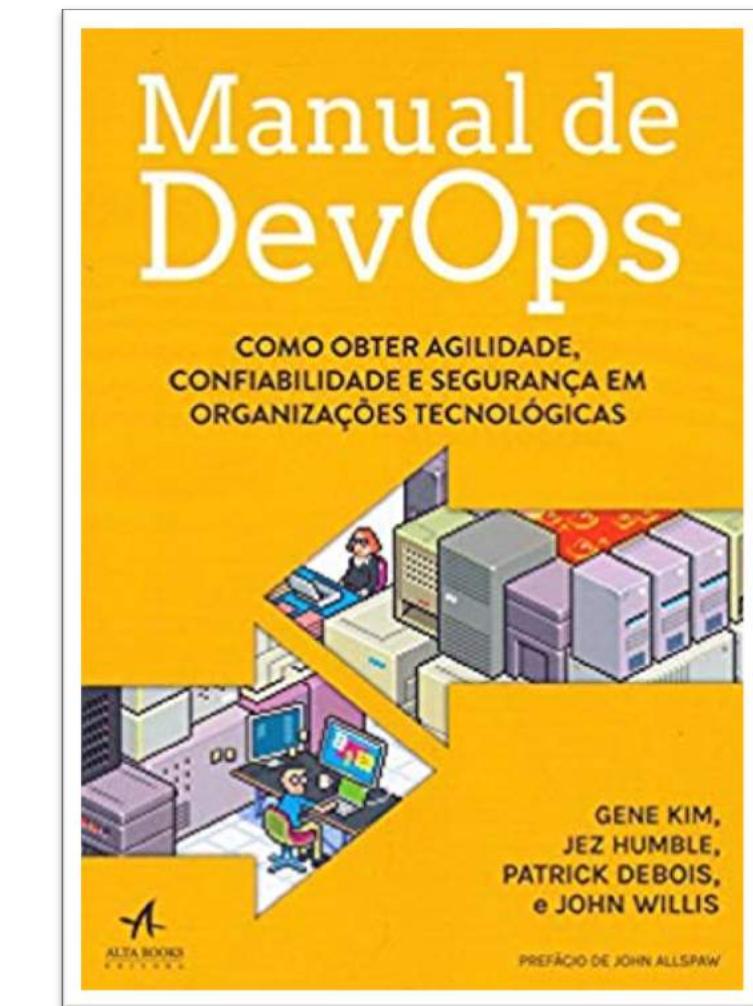
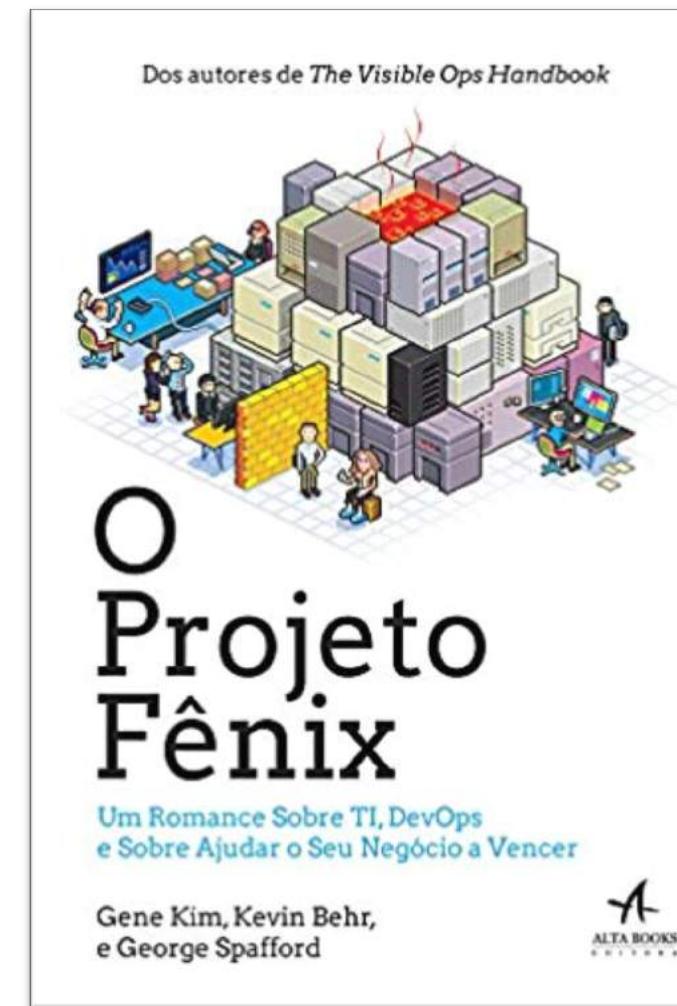
- por meio de aprendizagem
- por meio de um dia de treinamento adicional “Introdução ao DevOps”;
Ou...
- lendo The Phoenix Project (consulte a lista de literatura).



Literatura adicional recomendada

EXIN

DevOps MASTER™



Requisitos do Exame



DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
1 Adoção do DevOps		28%	14	
1.1 Mentalidade (Mindset) DevOps e seus Benefícios				
1.1.1 Analisar anti-padrões DevOps em um cenário				
1.1.2 Explicar os benefícios do DevOps				
1.1.3 Explicar por que o DevOps se adequa tão bem ao atual processo de desenvolvimento de software				
1.1.4 Explicar por que DevOps precisa de um mindset específico para trabalhar				
1.1.5 Explicar como o DevOps se encaixa com as práticas Lean e Agile Scrum				
1.2 Cultura Organizacional				
1.2.1 Explicar por que os 4 pilares de DevOps Eficaz (Colaboração, Afinidade, Ferramentas e Dimensionamento) são tão importantes				
1.2.2 Analisar um cenário com partes faltantes da mentalidade DevOps				
1.2.3 Explicar como criar uma equipe com um grupo de pessoas, focando fomentar a colaboração, a mentalidade DevOps, a empatia e a confiança				
1.2.4 Analisar uma situação que tenha um equívoco sobre a colaboração e identificar o método correto de solução do problema				
1.2.5 Analisar uma situação em que há necessidade de gestão de conflitos e identificar a melhor solução				
1.2.6 Explicar como o gerenciamento de recursos humanos pode fomentar a diversidade e quais benefícios isso traz para a organização				
1.3 Princípios e Conceitos DevOps				
1.3.1 Explicar o uso e a utilidade de diferentes metodologias de desenvolvimento de software (Cascata (Waterfall), Agile, Scrum) e seus princípios básicos				
1.3.2 Explicar o uso e utilidade de diferentes metodologias de operações (Gerenciamento de Serviços de TI)				
1.3.3 Explicar o uso e a utilidade da metodologia de sistemas Lean				

Requisitos do Exame

EXIN

DevOps MASTER™

		% Peso	Questões	Referência de Literatura	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2 Planejamento, Requisitos e Desenho		18%	9				
2.1 Gerenciamento do Ciclo de Vida de Aplicativos ou Serviços							
2.1.1 Explicar como o DevOps agrega valor ao Gerenciamento do Ciclo de Vida do Aplicativo moderno		4%	2				7
2.1.2 Explicar por que a DevOps melhora a experiência do cliente quando usada para o Gerenciamento do Ciclo de Vida do Serviço							
2.2 Termo de Abertura do Projeto (Definição de escopo) e Controle Visual							
2.2.1 Explicar como o escopo do projeto DevOps deve ser determinado		4%	2				5,7
2.2.2 Explicar por que o Controle Visual em um projeto DevOps facilita as práticas DevOps							
2.3 Desenho da Infraestrutura e Arquitetura							
2.3.1 Explicar como o DevOps muda ou influencia o design de infraestrutura e arquitetura de TI		4%	2	3,4	11	5,7	
2.3.2 Explicar por que a Computação em nuvem e as técnicas de virtualização tornam a integração de Dev e Ops mais fácil							
2.4 Requisitos e acordos de nível de serviço							
2.4.1 Explicar como o DevOps altera os requisitos e os acordos de nível de serviço		2%	1				7
2.5 Implementando uma Estratégia de Testes							
2.5.1 Explicar por que e como a Estratégia de Teste precisa ser alterada ao fazer a transição para o DevOps		4%	2				4
2.5.2 Analisar Histórias de Usuário, Histórias de Teste e Histórias de Operações para completude							

Requisitos do Exame

EXIN

DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
3 Desenvolvimento e implantação		30%	15	
3.1 Entrega Contínua e Integração Contínua				
3.1.1	Explicar por que a Entrega Contínua é essencial para DevOps eficaz			
3.1.2	Analisar como integrar a Entrega Contínua em um cenário			
3.1.3	Analisar como resolver problemas com a Entrega Contínua em um cenário	12%	6	16
3.1.4	Explicar por que a Integração Contínua é essencial para DevOps eficaz			3,15
3.1.5	Analisar como alcançar a Integração Contínua em um cenário com uma equipe distribuída ou um sistema de controle de versão distribuído			4
3.1.6	Analisar como resolver problemas com Integração Contínua em um cenário			
3.2 Pipeline de implantação				
3.2.1	Explicar a lógica da anatomia de um pipeline de implantação DevOps	4%	2	5,6
3.2.2	Explicar como usar scripts de criação e implantação			5
3.3 Implantação contínua				
3.3.1	Explicar por que o plano de iteração e o plano de liberação devem ser alterados para um DevOps eficaz	4%	2	10
3.3.2	Analisar como implementar a implantação contínua em um cenário			8
3.4 Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)				
3.4.1	Explicar os conceitos Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)	4%	2	4,7
3.4.2	Analisar um cenário para um problema com Ji-kotei-Kanketsu, Ritmo, Trabalho em andamento ou Fluxo Único e encontrar uma solução adequada			
3.5 Automação, Ferramentas e Testes				
3.5.1	Explicar por que a automação é importante para o DevOps eficaz			
3.5.2	Explicar como usar ferramentas para facilitar DevOps em geral			
3.5.3	Explicar como usar ferramentas para dar suporte à mentalidade e cultura do DevOps	6%	3	4,11,12,13
3.5.4	Explicar por que é importante que o teste de DevOps seja automatizado			3,4,5,6,7,8,9
3.5.5	Analisar um cenário e escolher a maneira correta de automatizar um teste de aceitação			

Requisitos do Exame

EXIN

DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
4 Operação e Dimensionamento				
4.1 Gerenciamento de Dados; Infraestrutura e Ambientes; e Componentes e Dependências				
4.1.1 Explicar quais problemas podem ser encontrados ao gerenciar dados em bancos de dados dentro do DevOps				
4.1.2 Analisar um cenário onde um banco de dados é usado em DevOps e fornecer a melhor solução para um problema				
4.1.3 Analisar um cenário e identificar a melhor maneira de preparar um ambiente de infraestrutura para implantação ou gerenciá-lo após a implantação	10%	5		11,12,13
4.1.4 Analisar um cenário e sugerir uma estratégia comumente usada para gerenciar componentes				
4.1.5 Explicar como gerenciar dependências				
4.2 Gerenciamento de Configuração e Controle de Versão				
4.2.1 Explicar por que o controle de versão é uma chave para o DevOps eficaz				
4.2.2 Explicar como manter o controle de versão sobre dados, infraestrutura e componentes	4%	2		2,14
4.2.3 Analisar um cenário e sugerir a melhor estratégia para gerenciar um problema de gerenciamento de configuração				
4.3 Infraestrutura em Nuvens e Imutável				
4.3.1 Explicar quando é e quando não é necessário mover para a infraestrutura baseada em nuvem para ter um DevOps eficaz	2%	1	4,5,14,16	11
4.3.2 Explicar como a infraestrutura baseada em nuvem deve ser gerenciada dentro do DevOps				
4.4 Continuidade do Negócio				
4.4.1 Explicar como o DevOps pode facilitar práticas de continuidade de negócios	2%	1		2,4
4.5 Dimensionamento				
4.5.1 Analisar um cenário, explicar se e por que é importante dimensionar para cima ou para baixo nessa situação, e identificar a melhor maneira de fazer isso				
4.5.2 Analisar um cenário que deu errada a dimensionamento, e identificar uma boa maneira de resolver o problema	4%	2	14,15,16,17	
4.5.3 Explicar como a política social e práticas de contratação suportam escalonamento DevOps				

Requisitos do Exame



DevOps MASTER™

					Referência de Literatura	
		% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
5 Fim da vida		2%	1			
5.1 Condições de Fim de Vida de um produto ou serviço						
5.1.1 Expliquem que condições devem ser cumpridas antes de terminar um serviço ou produto		2%	1			7

Módulo 1

Adoção DevOps

Requisitos do Exame



DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
1 Adoção do DevOps		28%	14	
1.1 Mentalidade (Mindset) DevOps e seus Benefícios				
1.1.1 Analisar anti-padrões DevOps em um cenário				
1.1.2 Explicar os benefícios do DevOps				
1.1.3 Explicar por que o DevOps se adequa tão bem ao atual processo de desenvolvimento de software				
1.1.4 Explicar por que DevOps precisa de um mindset específico para trabalhar				
1.1.5 Explicar como o DevOps se encaixa com as práticas Lean e Agile Scrum				
1.2 Cultura Organizacional				
1.2.1 Explicar por que os 4 pilares de DevOps Eficaz (Colaboração, Afinidade, Ferramentas e Dimensionamento) são tão importantes				
1.2.2 Analisar um cenário com partes faltantes da mentalidade DevOps				
1.2.3 Explicar como criar uma equipe com um grupo de pessoas, focando fomentar a colaboração, a mentalidade DevOps, a empatia e a confiança				
1.2.4 Analisar uma situação que tenha um equívoco sobre a colaboração e identificar o método correto de solução do problema				
1.2.5 Analisar uma situação em que há necessidade de gestão de conflitos e identificar a melhor solução				
1.2.6 Explicar como o gerenciamento de recursos humanos pode fomentar a diversidade e quais benefícios isso traz para a organização				
1.3 Princípios e Conceitos DevOps				
1.3.1 Explicar o uso e a utilidade de diferentes metodologias de desenvolvimento de software (Cascata (Waterfall), Agile, Scrum) e seus princípios básicos				
1.3.2 Explicar o uso e utilidade de diferentes metodologias de operações (Gerenciamento de Serviços de TI)				
1.3.3 Explicar o uso e a utilidade da metodologia de sistemas Lean				

1.1 – Mentalidade DevOps e seus benefícios

	% Peso	Questões	Referência de Literatura		
			Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
1.1 Mentalidade (Mindset) DevOps e seus Benefícios 1.1.1 Analisar anti-padrões DevOps em um cenário 1.1.2 Explicar os benefícios do DevOps 1.1.3 Explicar por que o DevOps se adequa tão bem ao atual processo de desenvolvimento de software 1.1.4 Explicar por que DevOps precisa de um mindset específico para trabalhar 1.1.5 Explicar como o DevOps se encaixa com as práticas Lean e Agile Scrum	10%	5	1, 2, 3	1	1, 2, 3

DevOps é um conjunto de práticas que enfatiza a automação, comunicação e colaboração durante todo o ciclo de vida do produto com o objetivo de viabilizar a entrega contínua de valor para o cliente e, ao mesmo tempo, manter a excelência operacional.



DevOps envolve apenas Dev e Ops

Conceitos e ideias do DevOps englobam todas as funções de uma organização. Ideias e práticas que auxiliam Dev e Ops a se comunicarem melhor e trabalharem com mais eficiência podem ser aplicadas em toda a empresa. Qualquer dois ou mais times podem se beneficiar dos princípios do DevOps.

Por exemplo, processos DevOps efetivos entre Vendas e Jurídico permitem a criação automática de contratos baseado em um consistente catálogo de contratos.

DevOps é um time

Criar um time chamado DevOps ou renomear o atual time não será suficiente para se criar uma cultura DevOps.

Se sua empresa já possui problemas de comunicação entre Dev e Ops, criar um novo departamento só vai agravar este problema.

DevOps é cargo

Devops é o núcleo de um movimento cultural, e suas idéias e princípios precisam ser usados em toda a organização para serem eficazes. Por isso não faz sentido ter pessoas com o cargo de DevOps.

DevOps só é relevante para Startups

Empresas de todos os tamanhos, incluindo as Enterprises, podem se beneficiar de princípios e práticas DevOps.

DevOps significa fazer todo o trabalho com a metade de pessoas

Algumas pessoas pensam que DevOps é uma forma de a mesma pessoa desenvolver e administrar a infraestrutura. Este conceito equivocado é perigoso porque leva pessoas a trabalharem muito mais que o ideal e deixam de ter um equilíbrio entre vida pessoal e trabalho.

DevOps não ajuda sua empresa a economizar dinheiro por reduzir o número de pessoas. Ao invés disso, DevOps possibilita que sua empresa melhore a eficiência e qualidade do seu trabalho, reduzindo o número e duração de indisponibilidades, diminuindo tempo de desenvolvimento e melhorando a efetividade individual e do time.

Só existe um jeito certo de se fazer DevOps

Devops incentiva o pensamento crítico sobre processos, ferramentas e práticas. Ser uma organização de aprendizagem requer questionar e repetir processos, não aceitar as coisas como o “caminho único” ou a maneira como as coisas sempre foram feitas.

Se não existe o jeito certo, também não existe o jeito errado de se fazer DevOps.

Devops intencionalmente não é rigidamente definido como Scrum ou ITIL. As empresas que estão realizando o devops com mais sucesso estão aprendendo com tranquilidade e iterando para descobrir quais ferramentas e processos são mais eficazes para elas.

É possível prever o tempo necessário para implementar DevOps

DevOps não possui um estado definido ou mensurável. É um processo contínuo, é uma jornada, não um destino.

Visto que muito do DevOps envolve cultura, é difícil prever quanto tempo será necessário para que as mudanças ocorram e tenham efeito.

DevOps é somente sobre ferramentas

Devops é um movimento cultural. Na sua empresa, as ferramentas que você usa hoje fazem parte de sua cultura.

Antes de decidir sobre uma mudança, cabe a você reconhecer as ferramentas no ambiente que fizeram parte da cultura existente, compreender as experiências dos indivíduos com essas ferramentas e observar o que é semelhante e diferente entre as experiências de outras pessoas. Este exame e avaliação ajudam a esclarecer quais mudanças precisam ser feitas.

A tecnologia impacta a velocidade e as estruturas organizacionais em sua organização. Fazer mudanças drásticas nas ferramentas, embora possam ter valor para um indivíduo ou equipe, pode ter um custo que desacelera a organização como um todo.

Cultura da culpa

A cultura da culpa tende a culpar e punir as pessoas quando erros são cometidos, seja em nível individual ou organizacional. Como consequência, quem errou vai tentar transferir a culpa e omitir informações. Este comportamento não funciona em uma cultura de abertura e colaboração.

Uma outra forma dessa cultura é quando avaliações de desempenho contribuem para uma atmosfera de hostilidade entre colegas de trabalho.

Quando as pessoas estão muito concentradas em simplesmente evitar ter um dedo apontado para elas, elas são menos focadas no aprendizado e na colaboração.

Silos

Descreve a mentalidade de equipes que não compartilham seus conhecimentos com outras equipes da mesma empresa. Em vez de ter metas ou responsabilidades comuns, as equipes isoladas têm papéis muito distintos e segregados.

Combinado com uma cultura de culpa, isso pode levar à retenção de informações como uma forma de segurança no trabalho.

Muitas vezes, em um ambiente em silos, você encontrará equipes diferentes usando ferramentas ou processos completamente diferentes para concluir tarefas semelhantes, pessoas que precisam escalar gestores para obter recursos ou informações de pessoas de outra equipe.

Análise de causa raiz

Muitas vezes, a análise de causa raiz está associada à determinação de uma causa raiz única. Concentra a atenção nas causas diretas, e não nos elementos adicionais que podem ser fatores contribuintes.

Há uma suposição implícita na análise de causa raiz de que os sistemas falham de maneira linear, o que não é o caso de nenhum sistema complexo.

Erro humano

A ideia de que uma pessoa cometeu um erro que causou diretamente uma falha, é frequentemente citado como a causa raiz de uma análise de causa raiz. Com isso muitas vezes vem a implicação de que uma pessoa diferente não teria cometido tal erro, o que é comum em uma cultura de culpa quando alguém tem que ser repreendido por seu papel em um incidente.

Essa é uma visão excessivamente simplista e é usada prematuramente como o ponto de parada para uma investigação. Isso tende a presumir que os erros humanos são cometidos devido a negligência, fadiga ou incompetência, e deixa de investigar outras centenas de fatores que contribuíram para que a pessoa tomasse a decisão ou executasse a ação que eles fizeram.

Implantar software manualmente

Entrega manual é o processo de implantação em que cada passo é visto como separado e atômico, executado individualmente ou por uma equipe.

É necessário fazer julgamentos a cada passo do processo, o que o torna mais sujeitos a erro humano. Mesmo que esse não seja o caso, diferenças na ordenação e no tempo de execução dos passos podem levar a resultados diferentes. Essas diferenças raramente são boas.

Implantar software manualmente

Os sinais desse anti-padrão são:

- Produção de documentação extensa e detalhada que descreve os passos a serem executados e como eles podem falhar.
- Dependência de testes manuais para confirmar que a aplicação está funcionando.
- Chamadas frequentes aos desenvolvedores para que esses expliquem algo que está dando errado no dia da entrega de uma versão.
- Ambientes em um cluster que têm configurações diferentes, por exemplo, dois ou mais servidores de aplicação com configurações diferentes de bancos de dados, sistemas de arquivos com interface diferente e assim por diante.
- Entregas de versão que levam mais do que alguns minutos para executar.
- Entregas de versão imprevisíveis, ocasionando rollback por motivos desconhecidos.
- Noites em claro antes do dia da entrega de uma versão, tentando entender como fazê-la funcionar.

Testar em ambiente similar ao de produção tarde demais

Nesse antipadrão, a primeira vez em que um software é implantado em um ambiente similar ao de produção é quando todo o desenvolvimento está concluído – ou pelo menos o que foi definido como concluído pelo time de desenvolvimento.

Testar em ambiente similar ao de produção tarde demais

Esse antipadrão se parece com o seguinte:

- Testes feitos apenas na própria máquina.
- A geração de uma versão para um ambiente de homologação é a primeira vez em que o pessoal de operação se envolve com uma versão do software.
- Há pouca ou nenhuma colaboração entre o time de desenvolvimento e as pessoas que realmente fazem a implantação.

Não trabalhar com infra ágil

Muitas organizações gerenciam a configuração dos seus ambientes de produção por meio de uma equipe de operação. Se uma mudança é necessária como, por exemplo, a mudança de uma configuração do banco de dados ou o aumento do número de threads em um thread pool no servidor de aplicação –, ela é feita manualmente nos servidores de produção. Se for feito algum registro de tal mudança, é provável que seja um registro em um banco de dados de mudanças de configuração.

Não trabalhar com infra ágil

Sinais desse antipadrão são:

- Depois de o software ter sido implantado com sucesso várias vezes em um ambiente de homologação, a implantação em produção falha.
- A equipe de operações demora muito para preparar um ambiente.
- Falta de padronização no ambiente de produção.
- Você não pode voltar para uma configuração anterior de configurações de infraestrutura.
- Servidores em clusters possuem, de forma não intencional, versões diferentes de sistema operacionais, infraestrutura de terceiros, bibliotecas ou patches.
- A configuração de um sistema é feita por meio da modificação direta da configuração nos sistemas de produção.

Cria um processo de liberação (release) confiável, previsível e passível de repetição, que, por sua vez, gera grandes reduções no tempo de ciclo e entrega novas funcionalidades e correções aos usuários rapidamente.

O relatório State of DevOps de 2015, publicado pela Puppet, descobriu que as empresas que estão fazendo devops estão superando aquelas que não estão, mostrando numericamente que a ênfase em ter equipes e indivíduos trabalhando juntos efetivamente é melhor para negócios do que silos cheios de pessoas que não trabalham bem em equipe.

Organizações DevOps de alto desempenho implantam o código com mais frequência, têm menos falhas, recuperam-se dessas falhas mais rapidamente e têm funcionários mais felizes.

Frequência de Implantação	30x
Lead Time de Implantação	200x
Tempo médio de recuperação (MTTR)	168x
Taxa de sucesso de mudança	60x

Muda o foco do que para o porquê

Concentrar-se na cultura e nos processos incentiva a interação e a melhoria de como e por que fazemos as coisas.

Quando mudamos nosso enfoque do **que** para o **porquê**, nos é dada a liberdade e a confiança para estabelecer significado e propósito para o nosso trabalho, que é um elemento-chave da satisfação no trabalho.

Reação mais rápida à necessidade de negócios

Nossa meta como profissionais de software é fornecer softwares de valor aos usuários o mais rápido possível. A velocidade é essencial porque existe um custo de oportunidade associado ao não fornecimento de software. Você só pode começar a obter um retorno do seu investimento quando o software for liberado.

Os princípios de DevOps também podem ser aplicados ao negócio

DevOps não é apenas suporte de TI. Devops também pode ser usado para apoiar a estratégia de negócios e melhorar os processos de negócios.

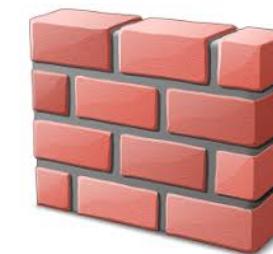
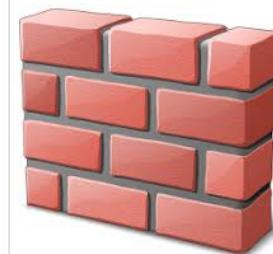
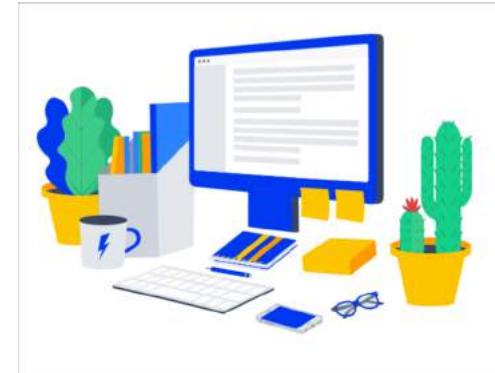
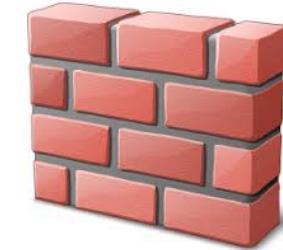
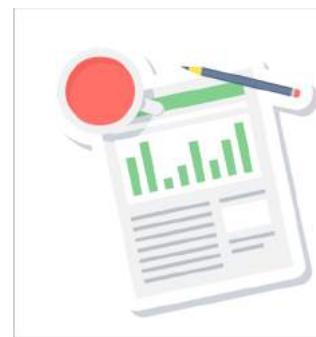
Um negócio sustentável e bem-sucedido é mais do que as equipes de desenvolvimento e operações. Limitar nosso pensamento apenas àquelas equipes que escrevem software ou o implantam em produção faz com que todo o negócio seja um desserviço.

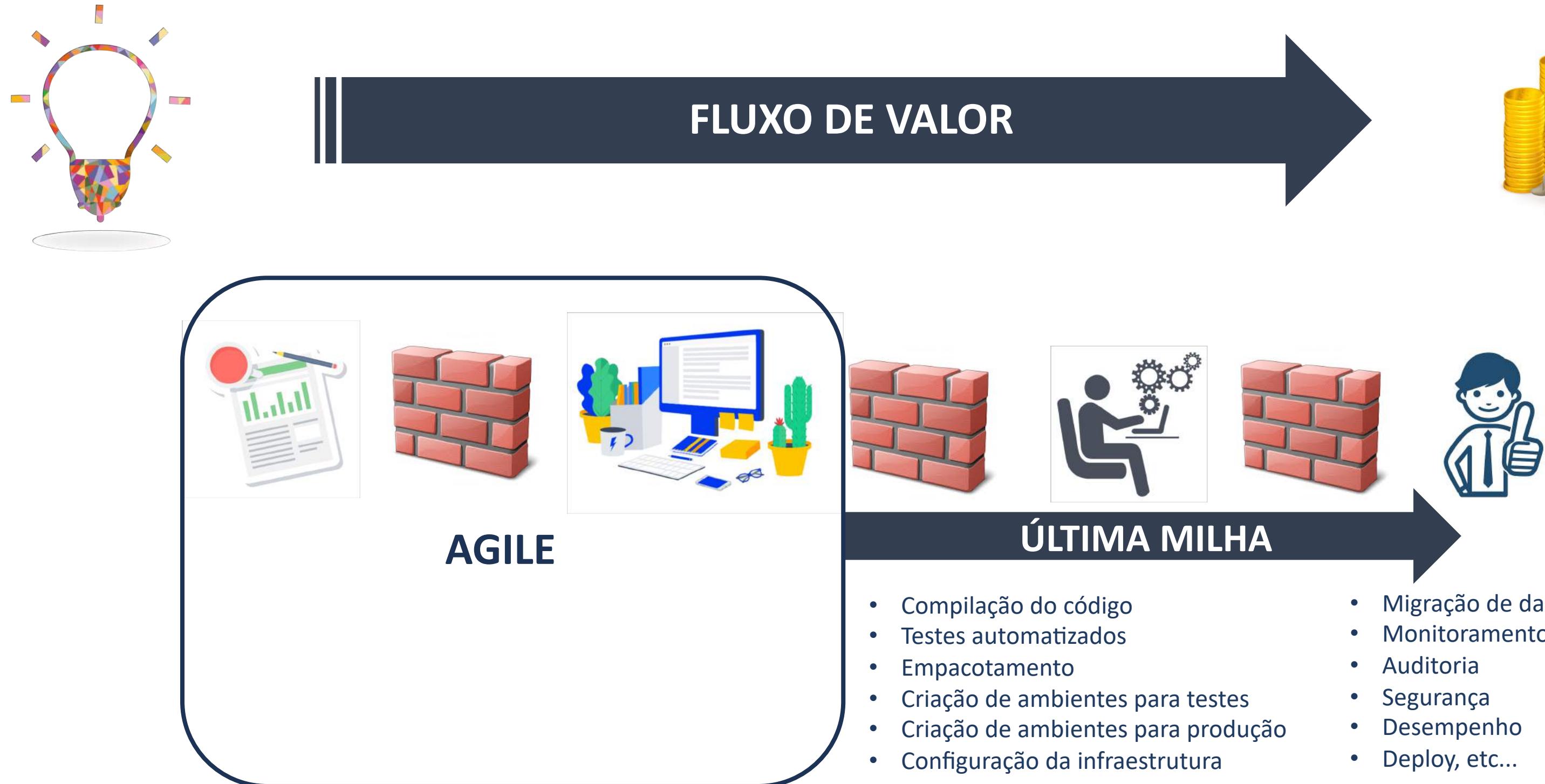
Por que o DevOps se adequa tão bem ao atual processo de desenvolvimento de software?

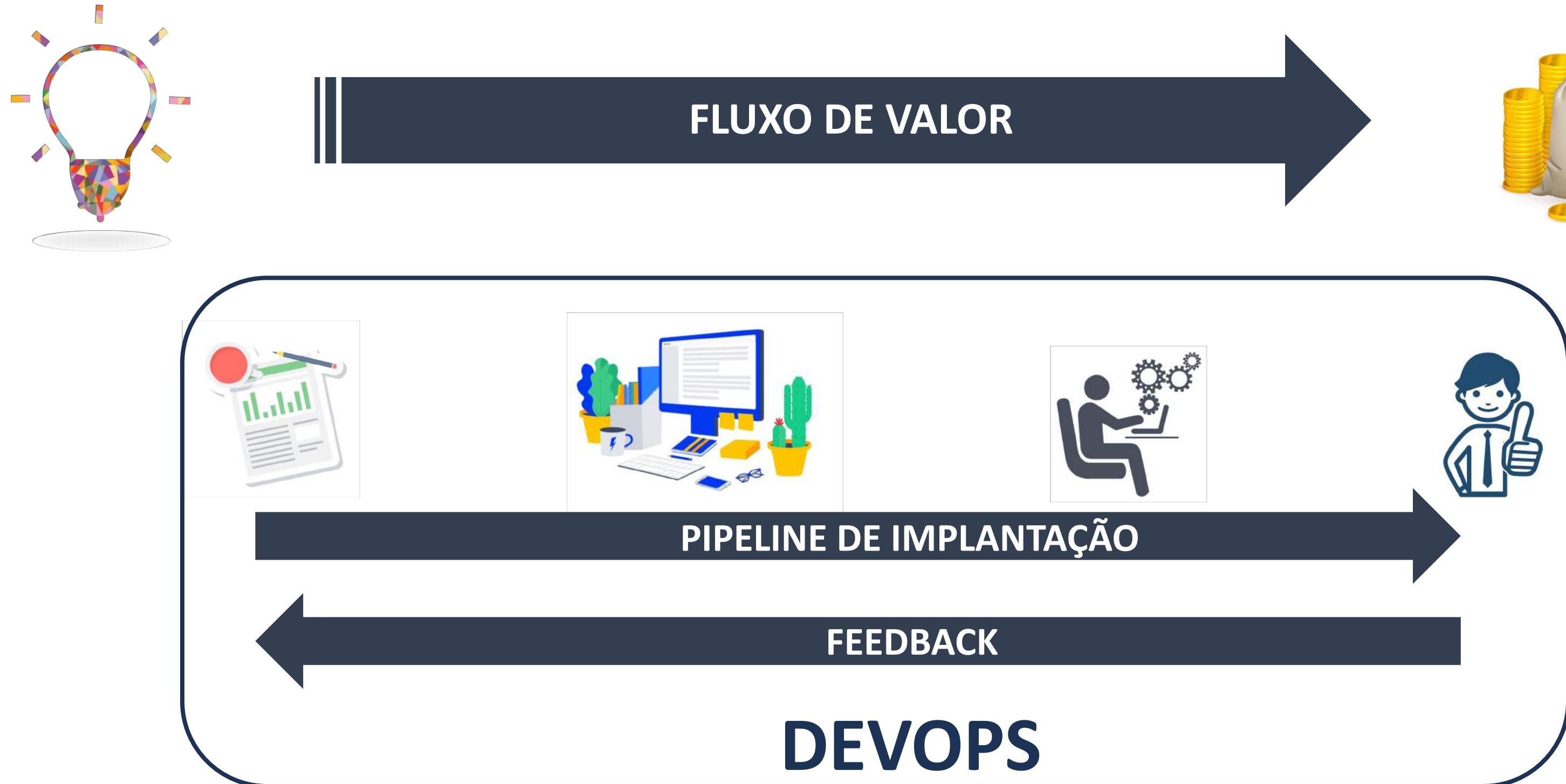
- O software não existe separadamente dos usuários e desenvolvedores
- Os métodos de desenvolvimento de software se concentram principalmente no desenvolvimento com pouca vinculação à implantação e às operações
- Devops é sobre adaptar e inovar a estrutura social, cultura e tecnologia para trabalhar de forma mais eficaz



FLUXO DE VALOR





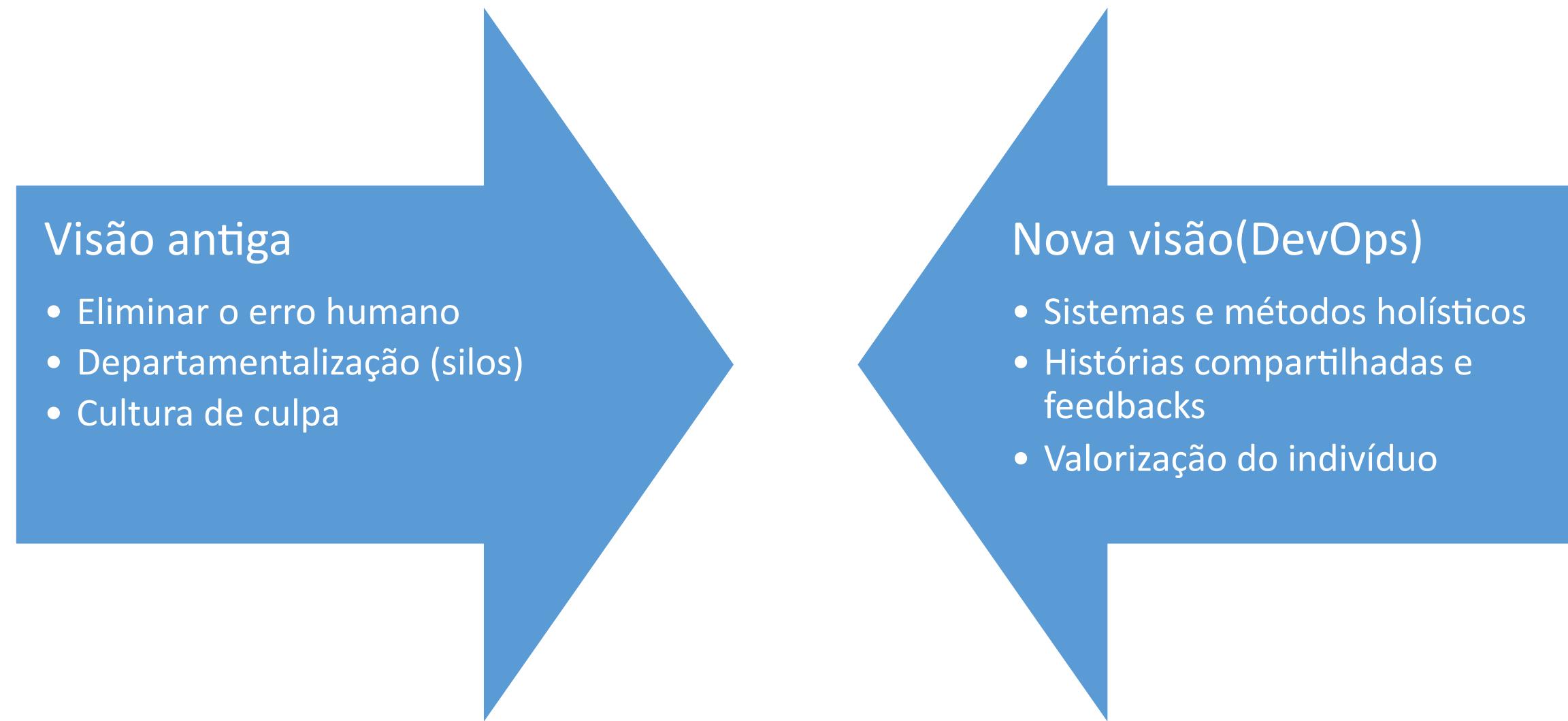


Por que precisamos de um Mindset específico para o DevOps?

O DevOps é um movimento cultural que muda a forma como os indivíduos pensam sobre seu trabalho, valoriza a diversidade do trabalho realizado, apoiam processos que aceleram a entrega de valor e medem o efeito da mudança social e técnica.

É uma maneira de pensar e de trabalhar que permite que indivíduos e organizações desenvolvam e mantenham práticas de trabalho sustentáveis.

É uma estrutura cultural para compartilhar histórias e desenvolver empatia, permitindo que pessoas e equipes pratiquem seus ofícios de maneira eficaz e duradoura.



A visão antiga encara o “erro humano como a causa do problema”, é uma mentalidade em que o foco está na eliminação do erro humano. Erros são cometidos por “maçãs podres” que precisam ser jogadas fora.

Essa visão é encontrada em culturas de culpa, pois pressupõe que os erros são frequentemente causados por malícia ou incompetência. Indivíduos responsáveis por falhas devem ser culpados e envergonhados (ou simplesmente demitidos).

Essa nova visão encara o “erro humano como um sintoma de problemas mais profundos no sistema”. É uma mentalidade que vê os erros humanos como estruturais e não pessoais.

As pessoas fazem escolhas e tomam as ações com base em seu contexto e o que faz mais sentido para elas, não com malícia ou incompetência intencional.

As organizações devem avaliar os sistemas de forma holística ao procurar minimizar ou responder a problemas.

Entender e abraçar a nova visão é fundamental para entender o movimento DevOps.

Essa nova visão nos encoraja a compartilhar histórias, pois tudo é uma oportunidade de aprendizado.

Histórias compartilhadas:

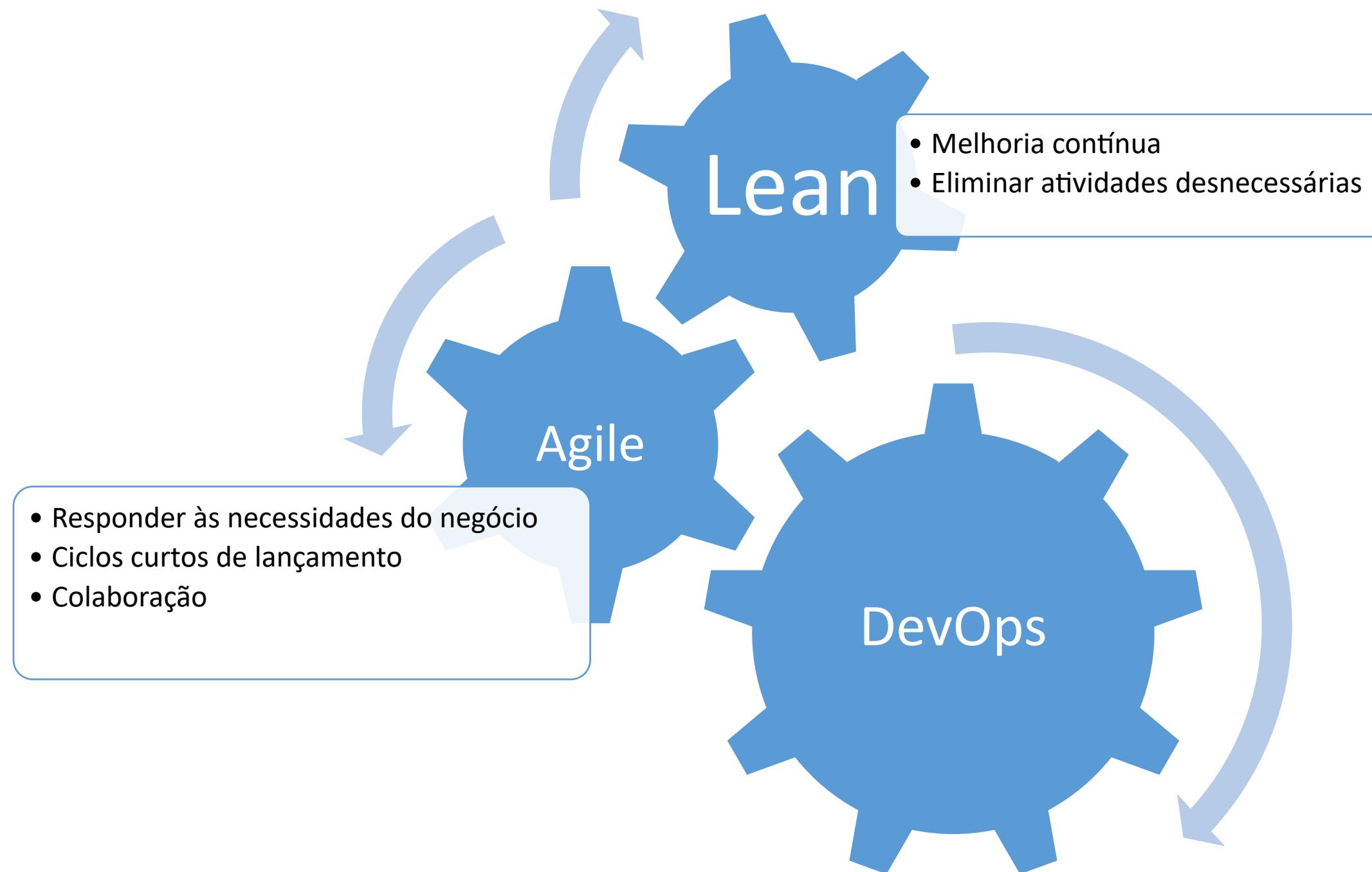
- Levam a uma maior transparência e confiança dentro de uma equipe
- Instruem nossos colegas sobre como evitar um erro dispendioso sem ter que vivenciá-lo diretamente
- Aumentam o tempo gasto na solução de novos problemas, permitindo mais inovação.

Quando essas histórias são compartilhadas em todo a empresa, impactamos a empresa como um todo, criando novas oportunidades, conhecimento e compreensão compartilhada.

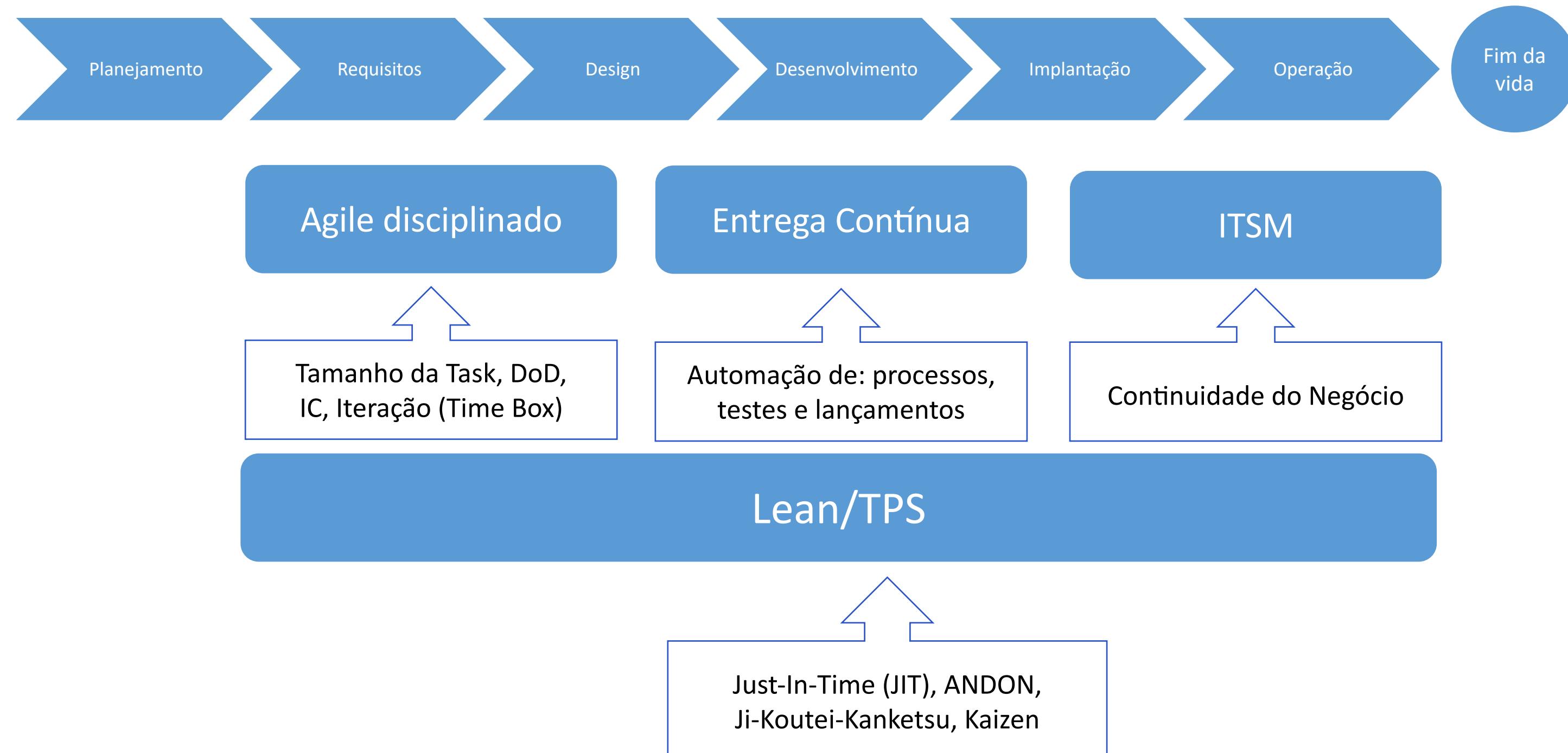
Como o DevOps se encaixa com as práticas Lean e Agile Scrum

EXIN

DevOps MASTER™



Como o DevOps se encaixa com as práticas Lean e Agile Scrum



- Defina hipóteses por problemas encontrados
- Então pense em ações de contra-medida para verificar sua hipótese.
- As ações de contra-medida devem ser definidas como atividades básicas diárias
- Definir KPIs por semana porque as pessoas podem sentir uma sensação de realização

KAI ZEN
改 善
MUDAR MELHOR



Essa nova visão nos encoraja a compartilhar histórias, pois tudo é uma oportunidade de aprendizado.

Histórias compartilhadas:

- Levam a uma maior transparência e confiança dentro de uma equipe
- Instruem nossos colegas sobre como evitar um erro dispendioso sem ter que vivenciá-lo diretamente
- Aumentam o tempo gasto na solução de novos problemas, permitindo mais inovação.

Quando essas histórias são compartilhadas em todo a empresa, impactamos a empresa como um todo, criando novas oportunidades, conhecimento e compreensão compartilhada.

Conceito TPS (Lean) como base

A criação de uma cadeia de suprimentos de serviços de TI com fluxo contínuo é difícil porque há muitos itens e é necessário mudar sua mentalidade do já conhecido ciclo de desenvolvimento e suas metodologias.

- **Just-In-Time** – Fluxo de peça única
- **ANDON** – Interromper todo o processo quando ocorre um defeito
- **Ji-Koutei-Kanketsu (JKK)** – Concluir 100% de um item contribui para a qualidade
- **Kaizen** – melhoria contínua

Para um pipeline de implantação eficiente é necessário construir um fluxo de peça única e automatizar o máximo possível de atividades.

Ágil Disciplinado

Uma equipe de desenvolvimento ágil disciplinada é essencial para o sucesso de uma implementação de DevOps.

Ágil Disciplinado significa:

1. Velocidade constante
2. Adaptabilidade para mudança
3. Sempre lance código de alta qualidade e livre de bugs

Entrega Contínua

A entrega contínua é a implementação automatizada dos processos de criação, implantação, teste e lançamento de aplicações.

Um foco importante está em testes, como testes de aceitação e testes de desempenho.

Cada organização terá diferenças na implementação de seu pipeline de implantação, dependendo de seu fluxo de valor para o lançamento de software.

Um fator-chave de sucesso é estabelecer apenas um único pipeline de implantação para serviços de TI.

Gerenciamento de serviços de TI (ITSM)

Como a tecnologia é um componente essencial da maioria dos processos de negócios, a disponibilidade contínua ou alta de serviços de TI é essencial para a sobrevivência do negócio como um todo. Isto é conseguido através da introdução de medidas de redução de risco e opções de recuperação.

É necessário realinhar o ITSM para DevOps, criando um ITSM leve que é **estritamente focado na continuidade dos negócios** com um conjunto de informações mínimas necessárias.



Sua empresa está se preparando para um grande Go-Live. Após quatro meses de desenvolvimento, os Devs solicitam ao time de Ops que implementem o software em um ambiente de homologação.

Vários problemas críticos são identificados logo no início dos testes, ficando claro que a liberação não está pronta para produção.

Qual anti-padrão DevOps é demonstrado nesse cenário?

- A) Implantação em um ambiente semelhante à produção apenas depois do Desenvolvimento
- B) Implantação em um ambiente semelhante à produção sem uma Solicitação de Mudança adequada
- C) Implantação na pré-produção com documentação de aplicativos insuficiente ou incompleta
- D) implantação em pré-produção com falta de apoio adequado do time de Desenvolvimento



Você, como especialista em DevOps, foi contratado por uma empresa para auxiliar times Scrum a trabalharem no conceito DevOps.

Seu gestor solicitou que você explique à gerência a importância de se trabalhar com DevOps.

Qual o principal argumento que você apresentaria?

- A) Isso os ajudará a obter uma certificação.
- B) Isso não fornecerá resultados melhores do que o Scrum.
- C) Isso apoiará diretamente os resultados do negócio.
- D) Isso apoiará o setor de Desenvolvimento a entregar software mais rapidamente.



Você faz parte do time de Operações e vocês desejam implementar uma nova prática.

Você precisa convencer o gerente do time dos benefícios dessa nova prática.

O que não seria um bom motivo?

- A) Explicar o objetivo da prática e o valor para o negócio.
- B) Explicar as vantagens e desvantagens da prática atual.
- C) Explica o Retorno sobre o Investimento da implementação dessa prática.
- D) Explica como a nova prática funciona em outra empresa.



Qual grupo de conhecimento melhor reflete os princípios DevOps?

- A) Agile (Ágil), Integração Contínua, Entrega Contínua, Cloud
- B) Integração Contínua, Entrega Contínua, ITIL
- C) Agile (Ágil), Entrega Contínua, Gerenciamento de serviços de TI, Lean/TPS
- D) PMBoK, Arquitetura, Entrega Contínua, ITIL



O que é ITSM leve?

- A) Uma ITSM centrada na continuidade de negócios
- B) Uma nova versão do ITIL® proposta como padrão
- C) Uma má implementação de processos ITIL®
- D) Uma ITSM centrada no gerenciamento de liberações



Qual é a etapa importante para tornar o processo de DevOps mais maduro?

- A) Contratação ou treinamento de funcionários para diversificar o time
- B) Implementação do Kaizen utilizando o PDCA e visualização do processo
- C) Fornecimento de orientação pela administração para amadurecer ainda mais
- D) Seleção de ferramentas que se ajustam bem a seu processo



Qual não é o efeito da construção de um processo contínuo do Desenvolvimento para Operações?

- A) Automação de processos, que reduz a carga de trabalho humano.
- B) Fluxo entre Desenvolvimento e Operações por meio do trabalho padrão e listas de verificação.
- C) Otimização a partir da perspectiva do cliente em termos de Valor.
- D) Sincronização da operação e ciclo de gestão entre Desenvolvimento e Operações.



O principal benefício de uma boa prática de DevOps é que ela cria um processo repetitivo, confiável e previsível.

Que processo é esse?

- A) Configuração
- B) Desenvolvimento
- C) Liberação (release)
- D) Serviço



Uma CIO designa seu funcionário mais confiável, Michael, um Scrum Master, a um projeto. O time de Desenvolvimento prepara-se para construir um pipeline de implantação. Michael confia nas boas intenções e na espontaneidade da equipe de desenvolvimento, mas gostaria que eles fossem mais disciplinados. Além disso, deve haver maior frequência de liberações. Michael deseja que o time de Desenvolvimento implemente liberações mais frequentes.

Um membro do time diz: “O mais importante a fazer sobre esse novo Pipeline de Implantação é automatizá-lo. Devemos primeiro automatizar o Pipeline de Implantação”. Essa declaração está correta?

- A) Sim, está correto. Automatizar o Pipeline de Implantação é o fator mais importante para aumentar a eficiência.
- B) Sim, está correto. Ao concentrar-se na criação de um Pipeline de Implantação automatizado, você supera problemas potenciais que podem ocorrer mais tarde.
- C) Não, não está correto. Alcançar um Fluxo Único (Fluxo Contínuo) e um processo de implantação sólido deve ser a primeira prioridade. A automação do processo pode vir mais tarde.
- D) Não, não está correto. Em vez de automatizar o Pipeline de Implantação, o processo de testes deve ser automatizado primeiro.



A empresa MyApp implementa mudanças em seu processo de desenvolvimento e liberação. Eles ouviram falar sobre DevOps e após uma rápida análise de suas práticas de trabalho, eles afirmaram que executam princípios e práticas DevOps eficazes.

Analizando um de seus muitos aplicativos, os quais utilizam todas as suas melhores práticas e metodologias, é possível observar os seguintes comportamentos para suas etapas de desenvolvimento e liberação:

- Uma nova versão do aplicativo é liberada para Produção a cada três meses
- Três Sprints ociosas são concluídas por Liberação, para fins de componentes de testes e amostragem de controle de qualidade.
- A priorização e a liberação de recursos são feitos em um plano de projeto com fases rígidas que entrega por módulo, começando com as infraestruturas de TI e, em seguida, módulos administrativos e funcionalidade de negócios.

Essas práticas refletem uma implementação de DevOps eficaz?

- A) Sim. Eles têm atualizações regulares e a priorização correta dos componentes de valor agregado.
- B) Sim. Eles usam práticas de Scrum para trabalhar no desenvolvimento de modo rápido e eficiente.
- C) Não. Eles não estão nem fazendo Scrum corretamente, muito menos seguindo as práticas de DevOps.

1.2 – Cultura organizacional

	% Peso	Questões	Referência de Literatura		
			Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
1.2 Cultura Organizacional <ul style="list-style-type: none">1.2.1 Explicar por que os 4 pilares de DevOps Eficaz (Colaboração, Afinidade, Ferramentas e Dimensionamento) são tão importantes1.2.2 Analisar um cenário com partes faltantes da mentalidade DevOps1.2.3 Explicar como criar uma equipe com um grupo de pessoas, focando fomentar a colaboração, a mentalidade DevOps, a empatia e a confiança1.2.4 Analisar uma situação que tenha um equívoco sobre a colaboração e identificar o método correto de solução do problema1.2.5 Analisar uma situação em que há necessidade de gestão de conflitos e identificar a melhor solução1.2.6 Explicar como o gerenciamento de recursos humanos pode fomentar a diversidade e quais benefícios isso traz para a organização	12%	6	6, 7, 8, 9, 10, 11, 12	5, 6	

Colaboração

O processo de desenvolvimento em direção a um resultado específico por meio do apoio a interações e comentários de múltiplos indivíduos.

Afinidade

Objetivos organizacionais compartilhados, empatia e aprendizado entre diferentes grupos de pessoas

Ferramentas

Aceleram e pouparam custos, mas devem se encaixar nos métodos de trabalho

Escalar

DevOps pode ser aplicado em diferentes organizações à medida que crescem, amadurecem ou até mesmo tornam-se mais enxutas

A combinação desses quatro pilares permitirá que você aborde os aspectos culturais e técnicos de sua organização.

Faz sentido que sua organização se concentre em um ou dois pilares de cada vez, enquanto tenta fazer mudanças, mas, no final, é a combinação de todos os quatro trabalhando juntos que possibilitará uma mudança duradoura e efetiva.

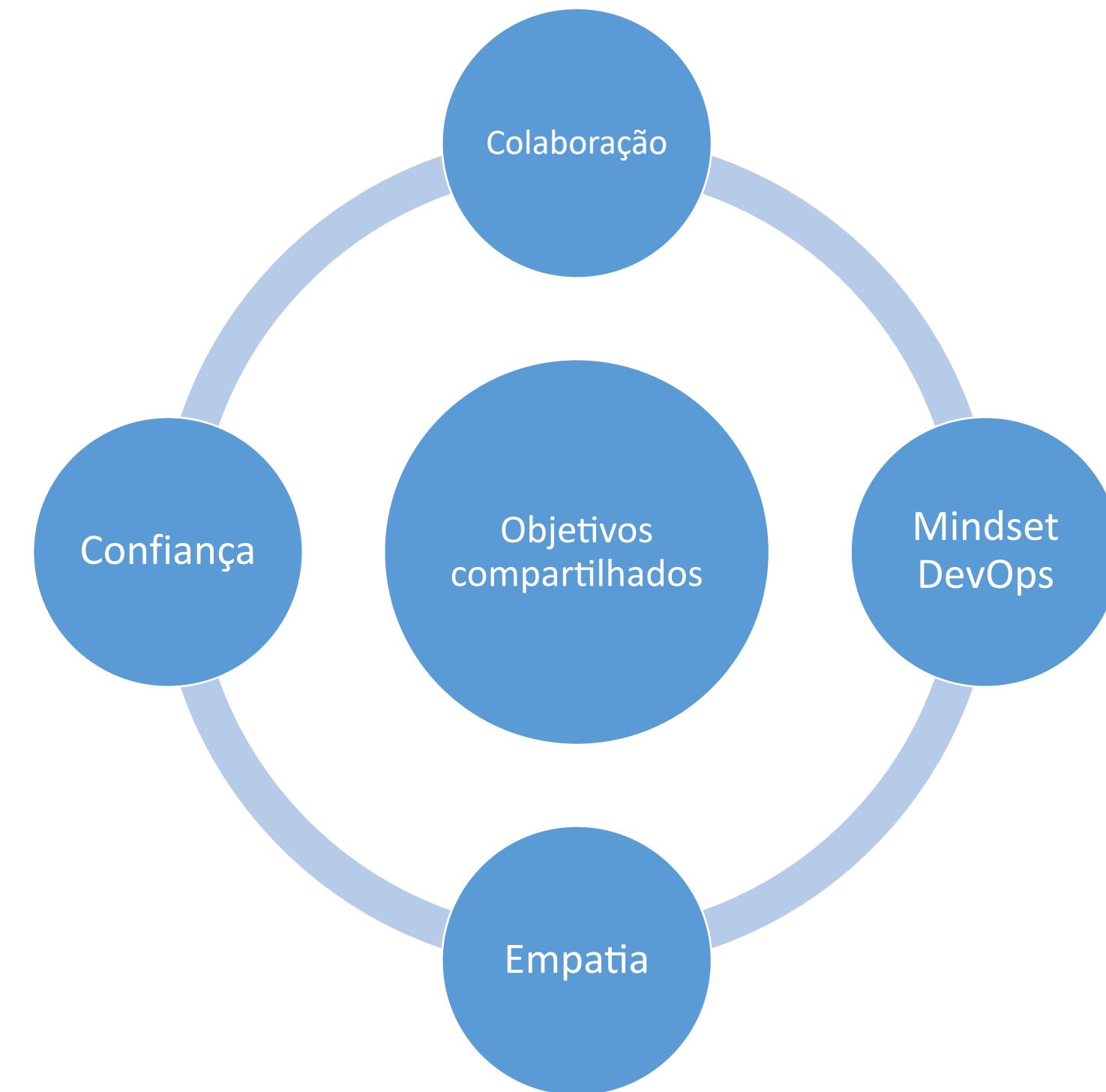
Ao implementar o DevOps, ferramentas e práticas devem mudar. No entanto, o mais importante é mudar primeiro a cultura organizacional.

É importante não ignorar os dois primeiros pilares (Colaboração e Afinidade), que cobrem as normas e valores de nossas culturas e interações interpessoais.

O uso efetivo da ferramenta é necessário para uma transformação DevOps bem-sucedida, mas não é suficiente.

Resolver os conflitos interpessoais e conflitos entre times, que surgem dentro das organizações é fundamental para promover os relacionamentos duradouros que, em última instância, criam um ambiente de DevOps.

Como construir um time



Para que você possa ter uma cultura DevOps eficaz, você deve focar em implementar as seguintes características:

- Comunicação eficaz
- Empatia e confiança
- Pessoal e recursos humanos

O que é um Time?

Um Time é um grupo de pessoas trabalhando em direção a um objetivo comum, interdependente entre si, com alguma familiaridade entre os membros.

As equipes possuem um conjunto de convicções expressas pela maioria dos participantes. Além disso, eles têm forças internas e externas agindo sobre eles que estabelecem e mantêm a identidade do grupo.

Um sinal claro de que você tem um time e não um grupo é quando o time ***mantém um ritmo de trabalho constante em prol de um objetivo comum***. Você só consegue ter objetivos comuns, empatia e uma cultura colaborativa através de uma ***comunicação eficaz***.

Uma parcela importante do que faz uma iniciativa bem-sucedida é fazer com que as pessoas trabalhem juntas mais efetivamente do que antes. Existem várias maneiras diferentes pelas quais as pessoas podem trabalhar, especialmente quando seus estilos de trabalho e prioridades são diferentes, mas chegar a um ponto em que as pessoas estão colaborando é fundamental para obter o máximo delas.

Muito do que está envolvido em alcançar os objetivos compartilhados e a empatia de uma cultura colaborativa é a **comunicação eficaz**, uma habilidade muito necessária para qualquer organização que queira elevar seu desempenho.

As relações entre os membros individuais são uma grande parte do que une essas comunidades e as torna tão benéficas, e esses relacionamentos exigem **confiança, empatia e reciprocidade**.

A colaboração é o processo de construir um resultado específico através do apoio e interações de várias pessoas.

ToM (Theory of Mind) é a capacidade de reconhecer a própria perspectiva, e que os outros têm uma perspectiva distinta e diferente nascida de seu próprio contexto.

Examinar como os indivíduos são diferentes e explorar como essas diferenças impactam a perspectiva em potencial ajudam a expandir nosso próprio ToM, constroem o entendimento mútuo e ajudam a resolver conflitos cruciais para o DevOps efetivo, com efeito nivelando nossa capacidade como companheiros de equipe mais inteligentes.

Diferenças individuais e background

Background profissional

- Experiência em Startups ou Enterprises
- Fluência técnica
- Hierarquia de papéis
- Formações (graduações) diferentes
- Anos de experiência

Background pessoal

- Nacionalidade
- Raça
- Gênero
- Orientação sexual
- Nível de educação

Metas

- Pessoais
- Profissionais

Estilo cognitivo

- Introvertido vs extrovertido
- Questionador vs adivinhador
- Iniciador vs finalizador
- Pensador analítico, crítico e lateral
- Purista vs pragmático

Competição: alguém se concentra unicamente em perseguir suas próprias necessidades às custas dos outros. Isso pode parecer um membro da equipe que escolhe projetos “bons” para si, deixando um trabalho menos importante para os outros membros da equipe.

Fuga: ambas as partes tentam evitar o conflito direto, muitas vezes com um aumento de comportamentos passivos-agressivos, conflitos indiretos e tensão. Em casos como este, você pode ver longos segmentos de e-mail onde as pessoas tentam mudar o trabalho ou culpar sem ser muito direto, ou pode haver muita reclamação interna de uma equipe sobre outra equipe, sem nunca falar diretamente com as pessoas que estão reclamando.

Meio-termo: todos os lados tentam encontrar uma solução mutuamente aceitável que envolva todos desistindo um pouco das suas próprias necessidades, a fim de tentar alcançar um resultado “justo”.

Colaboração: é semelhante a adaptação, na medida em que é considerada ganha-ganha, mas envolve muito mais criação de entendimento mútuo, aprendizado e garantia de que todos os lados realmente atendam às suas necessidades.

Mentalidade são nossas crenças pessoais sobre nós mesmos e como encaramos nossas possibilidades.

Jason Moser, um professor assistente de psicologia na Michigan State University, examinou os mecanismos neurais de diferentes mentalidades.

Ele observou que durante o julgamento de concluir tarefas e cometer erros, aqueles com uma mentalidade fixa mostravam menos atividade cerebral do que aqueles com uma mentalidade de crescimento, o que é consistente com as descobertas de que mentalidades de crescimento estão associadas a respostas adaptativas a erros.

Mudar a forma como pensamos literalmente muda a forma como o cérebro funciona e responde aos erros.

Cultivando a mentalidade correta

A pesquisa mostrou que a mentalidade das pessoas sobre suas próprias habilidades e, mais especificamente, de onde essas habilidades vêm, pode ter um impacto significativo sobre como elas aprendem e crescem.

Uma mentalidade de crescimento pode permitir que pessoas e organizações aprendam e se adaptem mais rapidamente em seus ambientes.

Na prática, isso significa que os indivíduos e as equipes podem reagir mais rapidamente a eventos com impacto na produção ou responder e mudar de direção mais rapidamente durante o ciclo de vida de um projeto, o que traz benefícios para toda a empresa.

Mentalidade fixa

Uma mentalidade fixa - acreditando que habilidades e traços são inatos e estáticos - faz com que as pessoas sintam que precisam provar constantemente a si mesmas aos outros.

Se alguém acredita que as características são inatas e as pessoas são espertas ou não inteligentes, elas obviamente querem provar para si mesmas e para aqueles ao seu redor que elas caem na categoria inteligente.

Alguém com uma mentalidade fixa vê falhas de qualquer tipo como prova de que o indivíduo não é inherentemente inteligente, não é talentoso ou não é suficientemente valorizado.

Mentalidade fixa

Para evitar falhas e sentimentos de inadequação, as pessoas que operam com uma mentalidade fixa podem ficar longe de situações em que possam falhar.

Eles são menos propensos a trabalhar em projetos onde eles teriam que aprender novas habilidades.

As pessoas evitam a incerteza como forma de evitar o fracasso e a desaprovação. Isso significa que pessoas com mentalidade fixa têm menos probabilidade de adquirir novas habilidades no trabalho.

Eles também tendem a se concentrar muito em se comparar com seus pares - uma mentalidade muito competitiva - para confirmar suas crenças sobre seus traços.

Mentalidade de crescimento

A pessoa que possui a mentalidade de crescimento, por outro lado, se dá muito bem em ambientes que incentivam o aprendizado individual e coletivo.

Quem possui a mentalidade de crescimento acredita que suas habilidades e conhecimentos mudam com o tempo. Se atualmente não possui conhecimento sobre uma área em particular, ele acreditam que com tempo, esforço, ensino e prática suficientes, ele pode dominar o tema.

Mentalidade de crescimento

Os desafios são vistos como oportunidades de aprendizado, maneiras de adquirir novas habilidades e conhecimentos e melhorar o que já conhece.

Sem o medo do fracasso que muitas vezes caracteriza uma mentalidade fixa, aqueles com uma mentalidade de crescimento podem assumir mais riscos e crescer mais.

O fracasso é visto não como um sinal de uma falha pessoal inerente, mas simplesmente como uma parte natural do processo de aprendizagem.

Como adotar a mentalidade de crescimento:

- 1. Aprenda os fundamentos** – Dedique tempo para entender quem faz o quê, como e por quê?
- 2. Desenvolva seu nicho** – Avalie quais habilidades você precisa aprimorar e tire tempo para estudar e ensinar outros.
- 3. Reconheça suas forças e progressos** – Busque feedback de outras pessoas para avaliar suas habilidades. Todos nós temos pontos cegos.
- 4. Assegure uma prática deliberada e de qualidade** – Coloque em prática as novas habilidades que aprendeu
- 5. Desenvolva seu estilo de trabalho** – Avalie o que está indo bem e o que pode ser aprimorado
- 6. Eleve o estilo do time** – Descubra as melhores práticas para trabalhar dentro do time.

Ao dar feedback enfatize os esforços, ações, trabalho e a mentalidade da pessoa, se concentrando- no que ela pode fazer e não no que ela é, orientando para uma mentalidade de crescimento.

Uma abordagem de mentalidade fixa

Alice é claramente muito inteligente - ela entende intuitivamente a maneira como os sistemas distribuídos se comportam e interagem. Ela não é muito boa com as pessoas, e não é o tipo de pessoa a quem as pessoas vão quando precisam de ajuda.

Uma abordagem de mentalidade de crescimento

Alice se esforçou muito em compreender os sistemas distribuídos com os quais ela trabalha, e esse esforço mostra em seu profundo conhecimento de como esses sistemas se comportam e interagem. Eu adoraria que ela encontrasse maneiras de compartilhar seu conhecimento mais efetivamente em apresentações formais e interações individuais menos formais.

Empatia é a capacidade de entender e compartilhar os sentimentos de outras pessoas. É uma habilidade que pode – e deve – ser aprendida e desenvolvida.

Como desenvolver empatia:

- Ouvindo
- Fazendo perguntas
- Imaginando de outra perspectiva
- Apreciando diferenças pessoais
- Rotação de funções (rotation)
- Ponto de contato designado (designated ops)
- Bootcamps

Aprenda a ouvir

Ouvir é importante para construir empatia em geral, mas pode ser ainda mais benéfico durante desentendimentos ou outras discussões acaloradas. Muitas vezes, quando estamos discordando de alguém, estamos apenas esperando para falar e planejar o que vamos dizer, em vez de realmente ouvir e tentar entender de onde a outra pessoa está vindo.

Em vez disso, tente desacelerar e forçar-se a ouvir e, em vez de interromper, espere até que a outra pessoa termine de falar sua própria resposta.

A escuta ativa envolve refletir o que você acha que a outra pessoa acabou de dizer a eles, parafraseando ou resumindo para ter certeza do que você ouviu e entendeu o que significava ou pretendia. Isso garante que ambas as partes estejam na mesma página e estejam falando sobre a mesma coisa.

Confiança

Confiança e empatia andam de mãos dadas: Quando um cresce, o outro também cresce. Aumentar a confiança pode ajudar a aumentar a resiliência do time. Sem confiança, indivíduos podem ser muito protetores dos seus projetos ou áreas de responsabilidade, muitas vezes em detrimento da própria saúde ou da produtividade geral da equipe.

Confiança

Uma excelente forma de conquistar confiança é por meio da *percepção de justiça*.

A percepção de justiça é muito importante para a satisfação dos funcionários, o que significa que os funcionários precisam ser capazes de confiar que estão sendo tratados de forma justa.

Uma coisa que pode ajudar nesse sentido é desenvolver papéis formais, níveis de emprego e escalas salariais, além de fornecer uma quantidade razoável de transparência nessas áreas.

Disponibilidade e sustentação de ambientes 24x7

Os funcionários devem ser compensados de forma justa pelo trabalho fora do expediente que precisam fazer. Se se espera que seja uma parte regular de seu trabalho, essa expectativa deve ficar clara na descrição do trabalho para que as pessoas possam avaliar se a posição é a adequada para elas.

Permita e encoraje as pessoas a cuidarem de si mesmas e de sua saúde - por exemplo, fazendo com que elas tirem o dia depois de realizarem a manutenção de madrugada.

Assegure-se de que, se possível, as tarefas de manutenção se espalhem o suficiente ou a equipe seja grande o suficiente para que as pessoas tenham tempo suficiente para se recuperar entre esses turnos fora do expediente.

Equilíbrio entre trabalho e vida pessoal

É importante manter o equilíbrio entre vida profissional e pessoal ao planejar seu quadro de funcionários. Se as pessoas em suas equipes forem obrigadas a trabalhar regularmente à noite e aos finais de semana, isso é uma receita para a baixa qualidade do trabalho, baixa moral do time e o esgotamento.

Devops é sobre a criação de práticas de trabalho sustentáveis, e como os indivíduos devem abordar seu trabalho - o equilíbrio entre a vida e a vida é uma parte fundamental disso.

Pessoas jovens, solteiras, sem filhos acharão muito mais fácil dedicar suas horas de folga ao trabalho do que pessoas com parceiros, filhos ou outras responsabilidades familiares. Parte do crescimento e da manutenção de uma equipe diversificada e inclusiva requer a consideração desses aspectos e a consideração de como os requisitos de trabalho podem ser ajustados para serem mais inclusivos.

Considerações sobre o tamanho do time

Ter pessoas responsáveis por respostas rápidas a alertas e incidentes, seja participando de uma rotação de plantão ou tendo vários turnos de pessoas trabalhando ao longo do dia, é outra consideração.

Embora você possa achar que não tem operações suficientes para uma equipe de operações completa, evite ter apenas uma pessoa responsável pela chamada de plantão. Isso provavelmente significará compartilhar as responsabilidades de plantão entre tantas pessoas quantas forem necessárias para dar aos indivíduos a chance de recuperar-se e recuperar o atraso no sono.

Atenção!

Mesmo a curto prazo, a privação do sono pode levar à dificuldade de concentração ou bom desempenho, irritabilidade ou ansiedade, e aumento do risco de pressão alta ou ataque cardíaco - e esses efeitos aumentam quando a privação do sono é prolongada.

Abrace a diversidade dentro do time

A diversidade é fundamental para a inovação: as diferentes ideias, perspectivas e pontos de vista que vêm de diferentes origens são uma parte crucial do desenvolvimento de novas ideias.

Diversas equipes poderão desenvolver produtos que trabalham para uma base de clientes mais ampla, devido às suas experiências únicas. Quanto mais os diferentes grupos ou indivíduos trabalham juntos, mais tendem a ser as pessoas mais estimuladas criativamente.

DevOps Engineer: Sua missão é melhorar e manter o processo automatizado. O engenheiro examinará todo o processo e ferramentas automatizados.

Gatekeeper (Release Coordinator): Responsável por monitorar o status operacional e o progresso da próxima versão do serviço de TI. Tomar decisões sobre a implantação de acordo com os critérios, incluindo segurança, conformidade, requisitos regulamentares, maturidade da equipe de operação e suas visões de processo.

Process Master: Lidera a equipe e atua como um facilitador, esse papel é o mesmo que “Scrum Master” no Scrum. Implementa o controle visual em todo o processo e tem um forte foco em estabelecer um processo alinhado por fluxo com fluxo de peça única.

Service Master (Scrum: product owner): Tem toda a responsabilidade de fornecer serviços de TI Just In Time (JIT). Esse papel é como o “Product Owner” no Scrum, que está gerenciando e priorizando backlogs de produtos e a nova responsabilidade adicional de planejamento de custos para o serviço de TI.

Equipe de gerenciamento de infraestrutura: Quase todas as empresas de médio e grande porte separam as atividades de desenvolvimento e gerenciamento de infraestrutura (ou operações, como é frequentemente conhecido) em diferentes grupos ou silos.

Você não pode ensinar truques novos a um velho administrador

Novas habilidades podem ser aprendidas, sejam novas técnicas ou tecnologias, ou softskills, como se tornar um mentor ou ter empatia com outras pessoas em sua organização. No entanto, é importante ter em mente que aprender novas habilidades exige tempo e esforço.

Você não pode esperar que isso aconteça de graça. Se você é um gerente que deseja crescer e desenvolver pessoas em sua equipe, forneça a eles o tempo e os recursos necessários (seja um orçamento para livros, a opção de ir a conferências ou oportunidades de treinamento interno) e não espere que isso aconteça da noite para o dia.

Novas habilidades exigem tempo e prática para todos. Não presuma que alguém mais velho que você, diferente de você ou com um histórico diferente do seu, seja incapaz de aprender.

Estou sobrecarregado, estressado e esgotado

Se você estiver com sintomas de esgotamento, incluindo ansiedade, cansaço e diminuição da satisfação não apenas com seu trabalho, mas consigo mesmo, é importante abordar os problemas o quanto antes. Isso geralmente não é um cenário que se resolve sozinho, em vez disso, você terá que agir.

Estratégias de curto prazo

A curto prazo, encontre o maior número possível de maneiras para se dar apoio e espaço para recarregar:

- Tire uma folga em um local em que você não está verificando o e-mail ou o slack
- Delegue ou diga “não” para o trabalho.
- Entre em contato com um profissional para obter ajuda - a saúde mental é tão importante quanto a saúde física.

Estou sobrecarregado, estressado e esgotado

Estratégias de longo prazo

Também pode ajudar a fazer um inventário de todas as responsabilidades que você tem:

- Para cada uma, tente avaliar se há algo em que você possa agir para diminuir o estresse dessa responsabilidade.
- Quando você está ficando esgotado, pode se manifestar como um estresse ou mal-estar mais geral, então descubra se há algo em particular que está contribuindo mais para o seu estresse e ansiedade do que você imagina.

Estou sobre carregado, estressado e esgotado

Identifique os pontos únicos de responsabilidade

Uma coisa de nota que muitas vezes provoca estresse adicional é sentir que há algo que é exclusivamente sua responsabilidade. Você pode ser a única pessoa trabalhando em um determinado projeto, um membro de uma equipe de uma pessoa ou a única pessoa que está interessada em fazer mudanças para melhorar a cultura organizacional. Qualquer que seja a situação, isso pode aumentar o estresse e o isolamento que fazem parte do sentimento de esgotamento.

Não podemos conhecer todas as situações específicas, mas podemos dizer que nenhum projeto, emprego ou empresa vale a pena sacrificar sua saúde.

As pessoas não parecem confiar umas nas outras

Demora mais tempo para reconstruir a confiança perdida do que para construí-la do zero, por isso, se a sua cultura está se curando de ser prejudicial, entenda que as coisas não vão mudar da noite para o dia.

A confiança e a comunicação precisam ser lideradas pelo exemplo, o que pode significar garantir que os gerentes de todos os níveis da empresa, bem como os líderes da equipe, tenham treinamento adequado, para que sejam capacitados na construção de confiança, na comunicação aberta e na gestão de conflitos.

Frequentes interrupções na comunicação

Em algumas organizações, é incrivelmente comum observar pessoas interrompendo umas às outras, ou falando por cima umas das outras. Esse tipo de cultura de interrupção é um exemplo de comunicação que é competitiva, em vez de colaborativa, e oferece muito menos chances de criar confiança entre indivíduos e equipes.

Quando as organizações não abordam esses comportamentos diretamente, elas estão apoiando e reforçando-as silenciosamente. Estabelecer um compromisso para eliminar esse tipo de comportamento. Assegure-se de que toda a organização coopera através da educação e crie um espaço seguro para os indivíduos comunicarem violações.

Reducir a interrupção não apenas aumenta a compreensão, mas também ajuda as pessoas a sentirem que estão sendo ouvidas, aumentando a confiança e a empatia.



Qual é a melhor descrição da abordagem de um pilar de **Colaboração** para um DevOps Eficaz?

- A) O processo de construção dessas relações entre os times e manter a essência de compartilhamento colaborativo entre as pessoas.
- B) O processo de desenvolvimento em direção a um resultado específico por meio do apoio a interações e comentários de múltiplos indivíduos.
- C) O processo de promover colaboração, empatia e aprendizagem entre diferentes grupos de pessoas.
- D) O processo de compartilhamento de informações para apoiar melhor o negócio entre Desenvolvedores e Operações.



Qual é um sinal claro de que você tem um time e não um grupo?

- A) O time segue as regras com as quais concordaram em suas reuniões de time.
- B) O time tem reuniões eficazes que eles próprios conduzem.
- C) O time mantém um ritmo de trabalho constante em prol de um objetivo comum.
- D) O time resolve problemas, questionando o membro do time responsável.



A implementação do DevOps envolve mudar a cultura da empresa. Qual é a verdadeira declaração de mudança de cultura?

- A) A implementação do DevOps está criando uma nova mentalidade e um novo comportamento para todas as partes interessadas, com foco em um fluxo contínuo.
- B) A fim de criar o fluxo, o comportamento de todos deve ser mudado em cada área, para refletir a cultura do DevOps daquele departamento.
- C) Implantar DevOps significa que o setor de Operações deve mudar para a cultura Agile (Ágil) de Desenvolvimento, para poderem sincronizar seu trabalho.
- D) Ao implementar o DevOps, ferramentas e práticas devem mudar. No entanto, o mais importante é mudar primeiro a cultura organizacional.



Você trabalha em um e-commerce e estão empenhados em atingir uma prática de DevOps eficaz. Vocês percebem que devem focar em Colaboração e Afinidade.

O que é fundamental para criar as relações duradouras que fazem parte de um ambiente eficaz de DevOps?

- A) Implementar uma reunião diária para acordar as questões de priorização.
- B) Implementar um pipeline de softwares mais eficiente para implantação contínua.
- C) Resolver conflitos interpessoais e conflitos entre times.
- D) Resolver questões organizacionais com um sistema de comunicação livre de acusações.



Você foi contratado para dar consultoria DevOps em uma empresa de SaaS. O principal problema nesta empresa é cultural. Para que você possa implementar uma cultura DevOps eficaz, quais características você deve implementar?

1. Capacitação tecnológica
 2. Comunicação eficaz
 3. Empatia e confiança
 4. Pessoal e recursos humanos
-
- A) 1 e 2
 - B) 1, 2 e 3
 - C) 2 e 3
 - D) 2, 3 e 4



Na sua equipe de TI, todos são profissionais e comprometidos com a qualidade do serviço que fornecem. Alguns destes profissionais são terceirizados.

Os resultados estão longe das expectativas e necessidades de negócios. Além disso, quando as coisas dão errado, há desconfiança e acusação entre todos, principalmente apontando o dedo para os profissionais terceirizados.

Você, como gestor, quer dar um basta nessa situação criando um novo departamento de TI que realmente atenda às necessidades de negócios. Você quer introduzir o DevOps.

É necessário ter uma mentalidade de crescimento que exija uma mudança cultural.

Qual estratégia mais ajuda a incutir a mentalidade do DevOps?

- A) Desenvolver seu nicho, melhorar o estilo do time e demitir os indivíduos que não param de culpar outras pessoas.
- B) Desenvolver seu nicho, reconhecer seus pontos fortes, progredir e planejar para acabar com a terceirização.
- C) Desenvolver seu estilo de trabalho, melhorar o estilo do time e treinar o time para desenvolver uma cultura de não colocar a culpa nos outros.
- D) Desenvolver seu estilo de trabalho, aprender os conceitos básicos e colocar as pessoas que têm um baixo desempenho sob avaliação e treinamento.



Ana não está se dando bem com Paulo e ambos acabam entrando em conflito às vezes. Isso a deixou muito desconfortável, resultando em aumento de comportamentos passivos e agressivos, bem como em conflito e tensão indiretos. Ana enviou um e-mail ao chefe dela no qual ela transferiu trabalho ou culpa, sem ser direta o suficiente quanto ao comportamento de Paulo. Ela também se queixou com outros colegas de trabalho sobre o comportamento de Paulo sem jamais falar diretamente com ele sobre a situação. Qual estilo de negociação Ana está demonstrando?

- A) Adaptação
- B) Fuga
- C) Competição
- D) Meio termo



Nossos estilos de negociação ou de resolução de conflitos são importantes para nossa forma de comunicação. O que é verdade sobre os estilos de negociação?

- A) A colaboração e o meio-termo podem resultar em situações ganha-ganha.
- B) A colaboração em DevOps significa trabalhar para atingir as metas de outra pessoa.
- C) O meio-termo resulta em uma situação ganha-ganha, e a colaboração não gera esse resultado.
- D) Meio-termo significa chegar a um acordo para evitar o conflito.



Você contratou uma nova pessoa para seu time, Ana. Após as primeiras interações com o time, você recebe reclamações de outros membros sobre o comportamento dela, como por exemplo: ela interrompe os mais velhos questionando suas propostas, é rude e egoísta, além de que parece não entender quanto esforço foi necessário para o time chegar onde está agora.

Você sabe que ela está apenas tentando ajudar e se sente pressionado a mostrar ao time que ela pode ser um membro valioso. Qual a melhor ação a ser tomada para ajudar a gerar confiança e empatia entre Ana e o time?

- A) Treinar Ana para que ela desenvolva suas habilidades de escuta, para que ela evite interromper os outros e que ouça as ideias das outras pessoas.
- B) Treinar Ana a respeitar os funcionários mais antigos e evitar questionar suas propostas ou maneiras.
- C) Parabenizar Ana por ser direta e honesta e incentivá-la a continuar a desafiar o time.
- D) Fazer uma reunião com o time para discutir as queixas abertamente, pois um bom time não deve ter segredos.



Você trabalha em um time de DevOps já há um bom tempo. Esta é uma grande mudança na maneira antiga de fazer as coisas, na qual o CEO regularmente fazia os funcionários passar vergonha publicamente por conta dos erros que cometiam. Seu time agora está muito mais feliz, mas sua comunicação ainda não é a ideal. Você já tentou usar ferramentas de bate-papo para criar uma comunicação mais eficaz em seu time. Você já tentou dizer às pessoas quando se comunicar, o que comunicar e para quem. Você tem registrado e mantido registros de debates para reflexão.

Nenhuma dessas ações teve um impacto positivo e ainda existe um problema com a comunicação de seu time. Qual é a causa mais provável da falta de comunicação e qual é a melhor solução neste caso?

- A) Os membros de seu time não são qualificados o suficiente. Aparentemente nem todos entendem o lado técnico do trabalho e precisam de mais treinamento.
- B) Os membros de seu time ainda estão deprimidos. Eles só precisam de tempo para restaurar sua comunicação natural.
- C) Os membros de seu time não confiam no ambiente de trabalho no qual não se joga a culpa em ninguém. Ensine-lhes pelo exemplo a confiar em você e uns nos outros.



Um CTO precisa organizar o time de DevOps. O time de Operações tem alguns problemas com:

- Investigar a conformidade
- Verificar se todos os status de processos estão ‘verdes’
- Monitorar o processo de Operações

Qual função de DevOps deve realizar este trabalho?

- A) Process Master
- B) O Coordenador de Liberações (Gatekeeper)
- C) O Engenheiro de Confiabilidade
- D) Service Master



O controle visual pode ser definido como: *Todos entendem facilmente a situação apenas olhando os quadros, sem a necessidade de explicações?*

Qual função de time deve implementar um controle visual em todo o processo e tem um foco sólido no estabelecimento de um processo simplificado?

- A) Desenvolvedor
- B) Engenheiro de DevOps
- C) Process Master
- D) Service Master



Você está trabalhando em um time DevOps recém-formado, que está trabalhado junto por dois meses. Você vê que os times de Desenvolvimento e de Operações ainda estão estagnados: eles têm uma maneira diferente de gerenciar seu trabalho, uma ética de trabalho diferente e vêm as coisas de uma perspectiva ligeiramente diferente. Para realmente começar a ser um time de DevOps eles precisam de mais coesão. O que é o melhor a ser feito para alcançar a coesão?

- A) Aprovação formal adicional da gerência para práticas de DevOps.
- B) Dar ao time tempo para que as mudanças impactem todos os membros do time e as interações entre eles.
- C) Treinar cada indivíduo para evitar que ajam como ilhas isoladas.
- D) Permitir que os membros do time se especializem na área na qual mostram maior interesse.



O DevOps está relacionado com a criação de práticas de trabalho sustentáveis e o equilíbrio entre vida profissional e familiar é uma parte fundamental disso.

O que é verdade sobre equilíbrio entre vida profissional e familiar?

- A) Permitir a compensação de horas extras não planejadas é suficiente para produzir o equilíbrio entre vida profissional e familiar.
- B) Concentrar-se em ativar o equilíbrio entre vida profissional e familiar por meio de um planejamento adequado promoverá um time diversificado.
- C) Horas extras e turnos de trabalho não padronizados não são compatíveis com o equilíbrio entre vida profissional e familiar para ninguém.
- D) Horas extras e turnos de trabalho devem ser evitados para manter o equilíbrio entre vida profissional e familiar em um time diversificado.



Qual é o principal benefício de aumentar a diversidade do time para incluir uma grande variedade de culturas e origens pessoais?

- A) Traz um maior número de pontos de vista e experiências.
- B) Leva à diminuição de atritos entre o time.
- C) Limita a originalidade e a habilidade de propor novos insights.

1.3 – Princípios e Conceitos DevOps

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
1.3 Princípios e Conceitos DevOps					
1.3.1 Explicar o uso e a utilidade de diferentes metodologias de desenvolvimento de software (Cascata (Waterfall), Agile, Scrum) e seus princípios básicos	6%	3	4, 5		4
1.3.2 Explicar o uso e utilidade de diferentes metodologias de operações (Gerenciamento de Serviços de TI)					
1.3.3 Explicar o uso e a utilidade da metodologia de sistemas Lean					

Waterfall

- Processo de gerenciamento de projetos
- Progressão sequencial de um estágio para o próximo
- Tempo gasto no início do ciclo de vida para reduzir erros posteriores

Agile

- Mais leve e flexível que os métodos anteriores, como a Waterfall
- Concentra-se em indivíduos, interações e colaboração
- Fornece software funcional rapidamente em iterações curtas

Scrum (an Agile method)

- Concentra-se em maximizar a capacidade de uma equipe de desenvolvimento de responder rapidamente a mudanças nos requisitos
- Stories, Sprints, Daily Meetings, Scrum Master, Product Owner, Definition of Done

Papeis

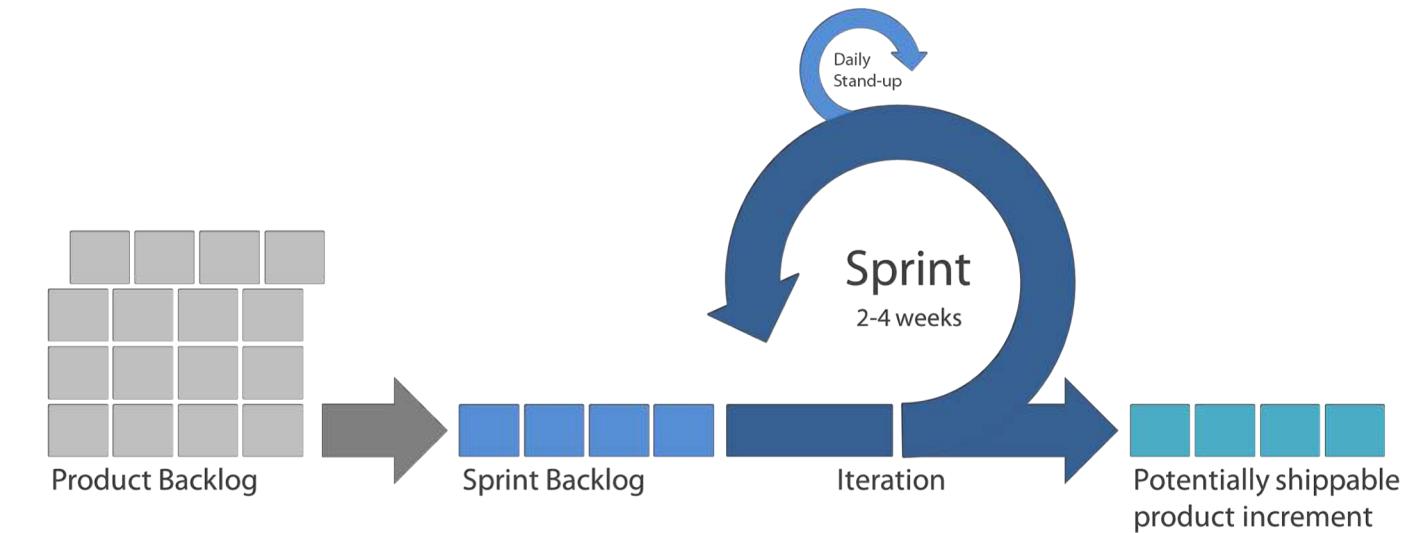
- Scrum Master
- Product Owner
- Time de Desenvolvimento

Eventos

- Daily Scrum
- Sprint
- Sprint Planning
- Sprint Review
- Sprint Retrospective

Artefatos

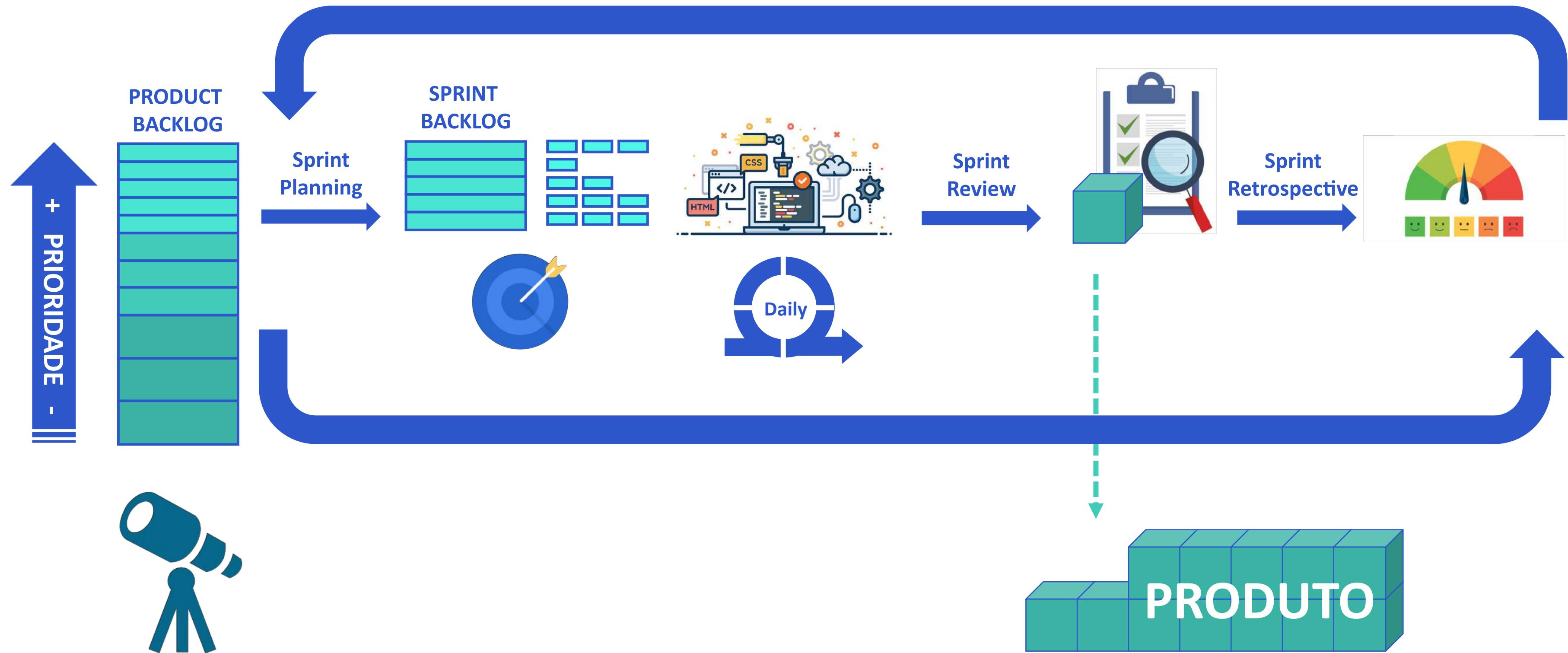
- Product Backlog
- Sprint Backlog
- Incremento
- Definição de Pronto
- Radiadores de Informação



Princípios e conceitos do DevOps

EXIN

DevOps MASTER™



É uma filosofia de gestão inspirada em práticas do sistema Toyota de gestão. O objetivo do Lean é eliminar desperdícios no processo de geração de valor.



O ITSM suporta disponibilidade e manutenção contínuas de serviços para garantir a entrega de valor e os resultados de serviços desejados

A metodologia mais utilizada é a ITIL, abrangendo estratégia de serviço, projeto de serviço, transição de serviço, operação de serviço e melhoria contínua do serviço.

Gerenciamento de Incidentes é o processo de ação para restaurar rapidamente as interrupções no serviço devido a incidentes. Os incidentes podem incluir redefinições de senha, falha da impressora ou uma mensagem de erro.

O Gerenciamento de Problemas trabalha para identificar e prevenir os problemas e incidentes de recorrência.

O gerenciamento de disponibilidade implica garantir que os serviços estejam sempre disponíveis para o cliente.

O gerenciamento de continuidade de serviços de TI envolve gerenciamento de riscos e garante a continuidade dos negócios.

O Gerenciamento do Nível de Serviço envolve planejar e definir metas de entrega de serviços organizacionais e, em seguida, medir o desempenho em relação a esses destinos. Os contratos de nível de serviço (SLAs) costumam ser usados para definir metas de nível de serviço para facilitar a medição e comparações com o desempenho real do serviço.

A Administração de Fornecedores monitora todas as relações com fornecedores, incluindo se as partes estão aderindo a contratos e acordos.

A Melhoria Contínua de Serviço da ITIL (CSI) concentra-se em encontrar oportunidades para o crescimento e aprimoramento de serviços.



O Scrum é uma prática recomendada para qualquer implementação de DevOps. Uma iteração no Scrum deve ser feita dentro de um tempo determinado. Qual palavra é usada para cada iteração no Scrum e quanto tempo deve levar no DevOps?

- A) Liberação. Deve ser o mais curto possível para as necessidades do negócio.
- B) Liberação. A duração recomendável no Guia Scrum é de três semanas.
- C) Sprint. Deve ser o mais curto possível para as necessidades do negócio.
- D) Sprint. A duração recomendável no Guia Scrum é de três semanas.



O time de Operações recebeu um pedido urgente de uma atualização de segurança de um fornecedor. Qual ação deve ser adotada pelo time de Operações?

- A) Aplicar o patch de segurança diretamente, pois é urgente. O time de Operações gerencia a infraestrutura, para que ninguém mais precise ser informado.
- B) Informar imediatamente os usuários sobre a atualização. E, então, começar a preparar-se para o patch de segurança emergencial.
- C) Emitir uma RDM (RFC) com um patch de emergência e informar a todas as partes interessadas. Em seguida, aplicar o patch de emergência de acordo com o ANS (SLA).
- D) Consulte o ANS (SLA) e, em seguida, aplique o patch de segurança e, logo após, informe as partes interessadas e os usuários com base no que foi descrito no ANS (SLA).



O time de Operações encontrou um defeito grave, o que faz com que um grande incidente ocorra. O que eles deveriam fazer?

- A) Criar uma RDM (RFC) urgente para reparar o defeito.
- B) Definir uma nova política de gerenciamento de incidentes.
- C) Fazer eles mesmos parte do trabalho de recuperação.
- D) Encontrar uma correção rápida para garantir a continuidade dos negócios.



Ao adotar uma abordagem DevOps, você deve aplicar o Lean. Isso implica detectar áreas de desperdícios a serem minimizados. Qual dos seguintes itens devem ser considerados como meta para eliminação de resíduos no desenvolvimento do software?

- A) Trabalho concluído
- B) Características extras
- C) Planejamento do projeto
- D) Habilidades do time

Módulo 2

Planejamento, Requisitos e Desenho

Requisitos do Exame

EXIN

DevOps MASTER™

		% Peso	Questões	Referência de Literatura	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2 Planejamento, Requisitos e Desenho		18%	9				
2.1 Gerenciamento do Ciclo de Vida de Aplicativos ou Serviços							
2.1.1 Explicar como o DevOps agrega valor ao Gerenciamento do Ciclo de Vida do Aplicativo moderno		4%	2				7
2.1.2 Explicar por que a DevOps melhora a experiência do cliente quando usada para o Gerenciamento do Ciclo de Vida do Serviço							
2.2 Termo de Abertura do Projeto (Definição de escopo) e Controle Visual							
2.2.1 Explicar como o escopo do projeto DevOps deve ser determinado		4%	2				5,7
2.2.2 Explicar por que o Controle Visual em um projeto DevOps facilita as práticas DevOps							
2.3 Desenho da Infraestrutura e Arquitetura							
2.3.1 Explicar como o DevOps muda ou influencia o design de infraestrutura e arquitetura de TI		4%	2	3,4	11	5,7	
2.3.2 Explicar por que a Computação em nuvem e as técnicas de virtualização tornam a integração de Dev e Ops mais fácil							
2.4 Requisitos e acordos de nível de serviço							
2.4.1 Explicar como o DevOps altera os requisitos e os acordos de nível de serviço		2%	1				7
2.5 Implementando uma Estratégia de Testes							
2.5.1 Explicar por que e como a Estratégia de Teste precisa ser alterada ao fazer a transição para o DevOps		4%	2				4
2.5.2 Analisar Histórias de Usuário, Histórias de Teste e Histórias de Operações para completude							

2.1 – Gerenciamento do Ciclo de Vida de Aplicações ou Serviços

	Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2.1 Gerenciamento do Ciclo de Vida de Aplicativos ou Serviços						
2.1.1 Explicar como o DevOps agrega valor ao Gerenciamento do Ciclo de Vida do Aplicativo moderno		4%	2			7
2.1.2 Explicar por que a DevOps melhora a experiência do cliente quando usada para o Gerenciamento do Ciclo de Vida do Serviço						

O verdadeiro benefício do DevOps

DevOps deve suportar diretamente os resultados de negócios, não apenas a colaboração com o desenvolvimento e a operação de serviços de TI, mas porque as empresas estão usando serviços de TI para dar suporte e melhorar seus negócios.

O uso do DevOps deve ser avaliado pelo resultado do negócio, não pelo escopo de um projeto de TI e pelos resultados de TI.

O objetivo do DevOps é estabelecer processos de negócios just-in-time (JIT). O DevOps visa maximizar os resultados de negócios, como aumentar as vendas e a lucratividade, aumentar a velocidade dos negócios ou minimizar o custo operacional, alinhando os processos de negócios just-in-time (JIT).

DevOps significa estabelecer a cadeia de fornecimento de serviços de TI nos negócios da mesma forma que a cadeia de fornecimento de outros produtos está incorporada nos negócios. É uma grande mudança de paradigma da entrega de software para o fornecimento de serviços de TI.

Do ponto de vista da arquitetura, o DevOps precisa estabelecer um sistema automatizado de implantação rápida.

2.2 – Termo de Abertura do Projeto (Definição de escopo) e Controle Visual

			Referência de Literatura
	% Peso	Questões	Effective DevOps Entrega Contínua Success with Enterprise DevOps
2.2 Termo de Abertura do Projeto (Definição de escopo) e Controle Visual			
2.2.1 Explicar como o escopo do projeto DevOps deve ser determinado	4%	2	5,7
2.2.2 Explicar por que o Controle Visual em um projeto DevOps facilita as práticas DevOps			

Um projeto nunca deve começar sem o Termo de Abertura do Projeto

É o primeiro documento de um projeto e estabelece as bases para o projeto. Explica o projeto em um nível estratégico e apresenta aos interessados a abordagem em relação ao projeto.

O termo de abertura do projeto é de propriedade do patrocinador do projeto e deve ser aprovada pelas partes interessadas do projeto. Quando o termo de abertura do projeto for aprovado, ele não poderá ser alterado durante o ciclo de vida do projeto.

O Termo de Abertura do Projeto contém três elementos principais:

Visão: A visão define o “Porquê” do projeto. Este é o propósito maior ou o motivo da existência do projeto.

Missão: Este é o “O quê” do projeto e afirma o que será feito no projeto para alcançar seu propósito maior.

Critérios de Sucesso: Os critérios de sucesso são testes de gerenciamento que descrevem efeitos fora da própria solução.

2.3 - Desenho da Infraestrutura e Arquitetura

			Referência de Literatura				
			% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2.3 Desenho da Infraestrutura e Arquitetura							
2.3.1 Explicar como o DevOps muda ou influencia o design de infraestrutura e arquitetura de TI		4%	2	3,4	11	5,7	
2.3.2 Explicar por que a Computação em nuvem e as técnicas de virtualização tornam a integração de Dev e Ops mais fácil							

O objetivo de DevOps é que todos os ambientes de teste devem ser semelhantes a produção, particularmente na maneira como são gerenciados.

O termo “ambiente” contempla todos os recursos que sua aplicação precisa para funcionar, bem como suas configurações.

Isso envolve:

- A configuração de hardware dos servidores que formam o ambiente e a infraestrutura de rede que os conecta.
- A configuração do sistema operacional, serviços e bancos de dados necessários para suportar as aplicações.

Recomendações:

- As equipes de desenvolvimento e gerenciamento de infraestrutura devem colaborar em todos os aspectos do gerenciamento e implantação do ambiente desde o início do projeto.
- Utilize técnicas ágeis para gerenciar infraestrutura, por exemplo provisionamento automatizado e manutenção autônoma.
- Teste em ambientes semelhantes ao de produção para detectar problemas com antecedência.
- Planeje a continuidade do serviço (Plano de Continuidade do Negócio)
- Gerenciamento de mudanças para a infraestrutura

A combinação de provisionamento automatizado e manutenção autônoma garante que a infraestrutura possa ser reconstruída em um período de tempo previsível em caso de falha.

Como Virtualização e Cloud Computing contribuem para uma infraestrutura ágil

Virtualização

- Provisionamento rápido de ambientes
- Reduzir o tempo para implantar o software
- Mais fácil oferecer infra-estrutura "como serviço"
- Padronização de hardware

Computação em nuvem

- Acesso rápido, facilmente escalável
- Implantar em uma stack completamente padronizada

2.4 - Requisitos e acordos de nível de serviço

	Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2.4 Requisitos e acordos de nível de serviço		2%	1			
2.4.1 Explicar como o DevOps altera os requisitos e os acordos de nível de serviço						7

As histórias são descobertas durante a fase de levantamento de requisitos e design do produto.

Tipos de histórias:

História do usuário: funcionalidades, regras de negócio, valor para o negócio, valor comercial e condições de operação.

História de teste: casos de teste de aceitação e critérios de aceitação de serviço.

História de operação: São definidas prioridades de serviços de TI e condições de operação para continuidade de negócios. São definidos contratos de nível de serviço e nível operacional.

2.5 – Implementando uma Estratégia de Testes

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
2.5 Implementando uma Estratégia de Testes	4%	2			4
2.5.1 Explicar por que e como a Estratégia de Teste precisa ser alterada ao fazer a transição para o DevOps					
2.5.2 Analisar Histórias de Usuário, Histórias de Teste e Histórias de Operações para completude					

Testar é uma atividade multifuncional que envolve todo o time e deve ser executada continuamente desde o início do projeto.

Devemos escrever testes em múltiplos níveis (unitários, de componentes e de aceitação) e executá-los como parte do pipeline de implantação.

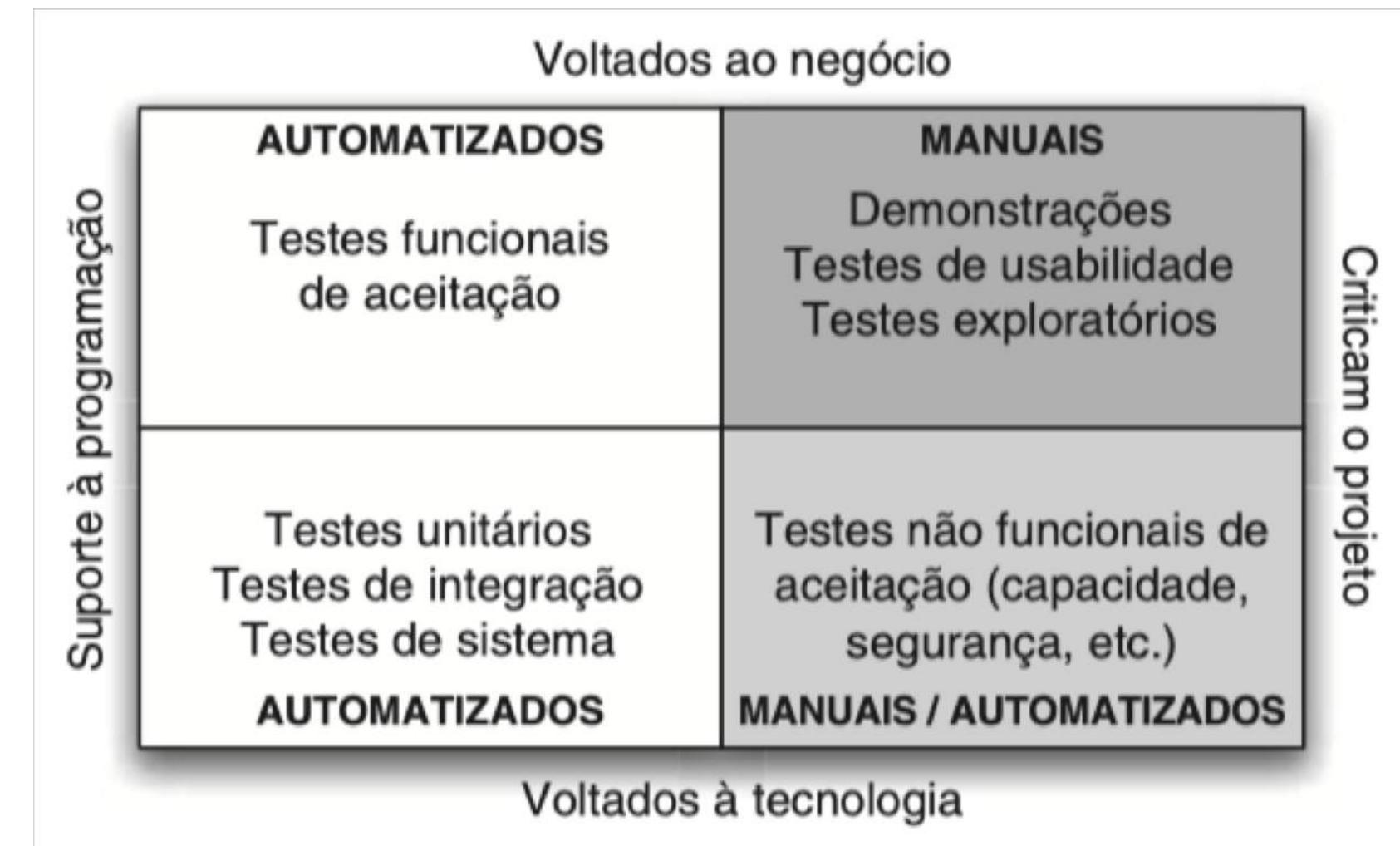
Testes manuais também são essenciais para adicionar qualidade: demonstrações, testes de usabilidade e testes exploratórios devem ser realizados continuamente ao longo do projeto.

De forma ideal, testadores colaboram com os desenvolvedores e usuários para a escrita de testes automatizados desde o início do projeto.

Estes testes são escritos antes que os desenvolvedores iniciem seus trabalhos nessas funcionalidades.

Testes de aceitação devem minimamente cobrir o caminho esperado (caminho feliz) de uma história.

O conjunto desses testes forma uma especificação executável do comportamento do sistema e, quando eles tiverem bons resultados, demonstrarão que a funcionalidade solicitada pelo cliente foi completamente implementada e de forma correta.



Testes estão fundamentalmente interconectados com nossa definição de “pronto”, e a estratégia de testes deve se concentrar em ser capaz de entregar esse entendimento a cada funcionalidade e garantir que testes estejam presentes ao longo de todo o processo.

Os critérios de aceitação devem ser escritos seguindo os princípios INVEST:

- **Independente** de outras histórias para facilitar a negociação, a priorização e a implementação;
- **Negociável** com o cliente, para que o time possa manter um ritmo sustentável para a entrega contínua de valor;
- **Valorosa**, assim cada entrega agregará grande valor ao negócio do cliente;
- **Estimável**, desse modo o time poderá estabelecer quais histórias serão implementadas de acordo com sua velocidade;
- **Small** (pequena) o suficiente para ser implementada dentro de uma iteração, sem correr grandes riscos de não a completar;
- **Testável**, apenas assim poderá ter qualidade na entrega e certeza de que o cliente aceitará a história como pronta.

Tenha em mente o valor para o usuário, questionando qual o objetivo do usuário com determinada funcionalidade.

No mínimo, nós iremos pedir aos clientes que escrevam os testes de aceitação o mais simples possível que cobrem os caminhos esperados desses cenários.

Testadores e desenvolvedores devem juntar-se o mais cedo possível para discutir os testes de aceitação antes do início do desenvolvimento.

Tenha em mente o valor para o usuário, questionando qual o objetivo do usuário com determinada funcionalidade.



Em comparação com um projeto de forma convencional, o que deve ser mudado para um projeto ser bem-sucedido em DevOps?

- A) Deve ser desenvolvida uma cadeia de fornecimento de serviços de TI, utilizando um sistema Pull e um fluxo contínuo.
- B) Os desenvolvedores devem se unir ao time de Operações para a manutenção rápida dos serviços.
- C) O time de Operações deve trabalhar para o time de Desenvolvimento É por isso que é chamado de DevOps.
- D) Os membros do time de Operações devem se unir ao time de Desenvolvimento.



O departamento de Vendas da empresa MyApp solicitou que o serviço que você está desenvolvendo possa incluir o compromisso de aumentar o número de visitantes exclusivos ao site e melhorar a taxa de cliques. Com uma pretensão assim, a posição relativa de mercado da Empresa MyApp é fortalecida, em comparação com seus concorrentes. Para compreender as necessidades do departamento de Vendas, você precisa de um foco diferente dos métodos convencionais.

Qual deve focar principalmente nas fases de planejamento, requisitos e projeto para lidar melhor com esse compromisso?

- A) Uma análise das atividades nas fases de planejamento e de projeto para garantir que seja possível construir um sistema Pull.
- B) Projeto de arquitetura microsserviço. DevOps não pode funcionar sem uma boa arquitetura.
- C) Infraestrutura em nuvem, para que todos os serviços em DevOps sejam implementados.
- D) Os Requisitos de Nível de Serviço (SLR) e o Acordo de Nível de Serviço (SLA), pois estes são sempre mais importantes.



De acordo com DevOps, o que você deve fazer no kick-off de um projeto?

- A) O DevOps reduz o ciclo de desenvolvimento. Portanto, você deve começar a discutir o Plano de Projeto ao lançar um projeto.
- B) As partes interessadas querem realizar serviços estratégicos; sendo assim, você deve começar a discutir por que deseja as funções do serviço de TI que você fornecerá.
- C) Você deve discutir o objetivo do projeto com as partes interessadas para garantir a velocidade do negócio e, assim, poder alinhar o JIT e evitar a estagnação.
- D) Você deve reunir todas as partes interessadas para discutir a continuidade do negócio e a operação, pois você deseja operar seu negócio com segurança.



Em caso de falha, suas características de infraestrutura deverão permitir que os sistemas sejam reconstruídos em um tempo previsível.

Quais são as duas características que melhor permitem essa recuperação?

- A) Detecção automatizada de erros e serviços em nuvem
- B) Implantação automatizada e virtualização do servidor
- C) Provisionamento automatizado e manutenção autônoma
- D) Restauração automática e configuração do servidor



Em DevOps, Acordos de nível de serviços (ANS's) são muito importantes para o cliente. Por quê os ANS's são importantes também para o time DevOps?

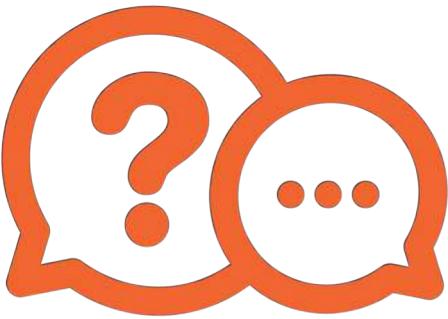
- A) Eles contem uma versão macro do plano de liberações.
- B) Eles contem itens do backlog de produto.
- C) Eles garantem que o cliente pague no prazo.
- D) Eles ajudam a focar no valor agregado ao negócio.



O time de Desenvolvimento na empresa MyApp tem enfrentado inúmeros desafios com suas práticas de testes atuais. Atualmente, eles utilizam um processo de testes de Aceitação manuais. Os desenvolvedores acreditam que o conjunto de testes da unidade que criaram é suficientemente meticuloso para proteger contra regressões.

A liderança sênior exigiu que o time de Desenvolvimento implementasse testes automatizados de aceitação para reduzir os custos totais de testes e minimizar o número de defeitos de código e regressões introduzidos no ambiente de produção. Quais princípios devem ser seguidos ao definir os critérios de aceitação para sua aplicação tendo a automação em mente?

- A) Princípios Agile (Ágil)
- B) Princípios ATAM
- C) Princípios DIVEST
- D) Princípios INVEST



Considere a seguinte história de um usuário: Como Gerente de Vendas, quero analisar a previsão de meu time cada semana por vendedor e por produto.

Para que essa história esteja em conformidade com os princípios INVEST, qual pergunta é a mais relevante para ser feita ao usuário?

- A) Você quer fazer isso?
- B) Qual é seu objetivo com isso?
- C) Qual é sua prioridade com isso?
- D) Quando você precisa disso?

Módulo 3

Desenvolvimento e Implantação

Requisitos do Exame

EXIN

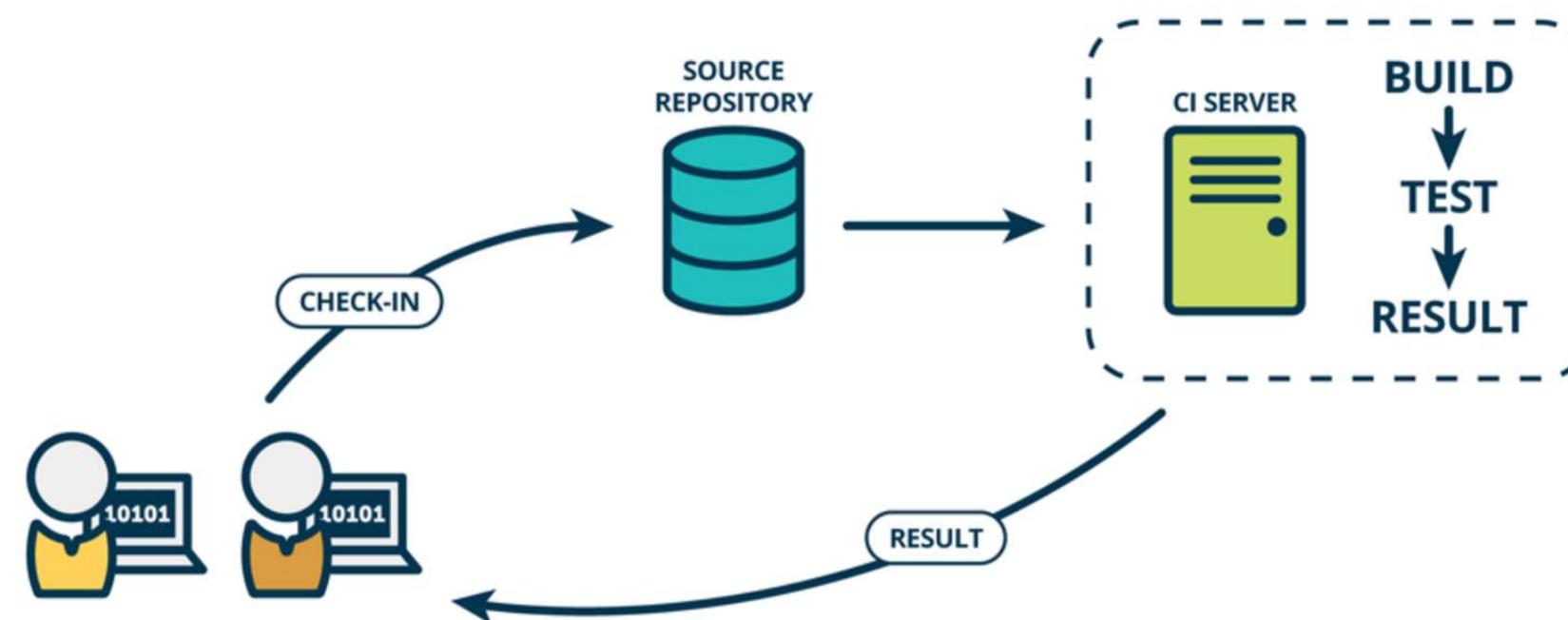
DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
3 Desenvolvimento e implantação		30%	15	
3.1 Entrega Contínua e Integração Contínua				
3.1.1	Explicar por que a Entrega Contínua é essencial para DevOps eficaz			
3.1.2	Analisar como integrar a Entrega Contínua em um cenário			
3.1.3	Analisar como resolver problemas com a Entrega Contínua em um cenário	12%	6	16
3.1.4	Explicar por que a Integração Contínua é essencial para DevOps eficaz			3,15
3.1.5	Analisar como alcançar a Integração Contínua em um cenário com uma equipe distribuída ou um sistema de controle de versão distribuído			4
3.1.6	Analisar como resolver problemas com Integração Contínua em um cenário			
3.2 Pipeline de implantação				
3.2.1	Explicar a lógica da anatomia de um pipeline de implantação DevOps	4%	2	5,6
3.2.2	Explicar como usar scripts de criação e implantação			5
3.3 Implantação contínua				
3.3.1	Explicar por que o plano de iteração e o plano de liberação devem ser alterados para um DevOps eficaz	4%	2	10
3.3.2	Analisar como implementar a implantação contínua em um cenário			8
3.4 Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)				
3.4.1	Explicar os conceitos Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)	4%	2	4,7
3.4.2	Analisar um cenário para um problema com Ji-kotei-Kanketsu, Ritmo, Trabalho em andamento ou Fluxo Único e encontrar uma solução adequada			
3.5 Automação, Ferramentas e Testes				
3.5.1	Explicar por que a automação é importante para o DevOps eficaz			
3.5.2	Explicar como usar ferramentas para facilitar DevOps em geral			
3.5.3	Explicar como usar ferramentas para dar suporte à mentalidade e cultura do DevOps	6%	3	4,11,12,13
3.5.4	Explicar por que é importante que o teste de DevOps seja automatizado			3,4,5,6,7,8,9
3.5.5	Analisar um cenário e escolher a maneira correta de automatizar um teste de aceitação			

3.1 – Integração Contínua e Entrega Contínua

	% Peso	Questões	Referência de Literatura		
			Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
3.1 Entrega Contínua e Integração Contínua 3.1.1 Explicar por que a Entrega Contínua é essencial para DevOps eficaz 3.1.2 Analisar como integrar a Entrega Contínua em um cenário 3.1.3 Analisar como resolver problemas com a Entrega Contínua em um cenário 3.1.4 Explicar por que a Integração Contínua é essencial para DevOps eficaz 3.1.5 Analisar como alcançar a Integração Contínua em um cenário com uma equipe distribuída ou um sistema de controle de versão distribuído 3.1.6 Analisar como resolver problemas com Integração Contínua em um cenário	12%	6	16	3,15	4

A integração contínua é uma prática, não uma ferramenta. Isso requer um grau de comprometimento e disciplina de sua equipe de desenvolvimento. Você precisa que todos façam pequenas alterações incrementais com frequência para integrar e concordem que a tarefa de maior prioridade no projeto é corrigir qualquer alteração que interrompa o aplicativo. Se as pessoas não adotarem a disciplina necessária para o trabalho, suas tentativas de a integração não levará à melhoria da qualidade que você espera.



Integração contínua exige que a aplicação seja compilada novamente a cada mudança feita e que um conjunto abrangente de testes automatizados seja executado.

É fundamental que, se o processo de compilação falhar, o time de desenvolvimento interrompa o que está fazendo e conserte o problema imediatamente.

O objetivo da integração é manter o software em um estado funcional o tempo todo.

Integração contínua representa uma mudança de paradigma. Sem ela, seu software será considerado como não funcional até que se prove que ele funciona, o que normalmente acontece durante o estágio de testes ou de integração.

Com integração contínua, assume-se que o software está funcionando (desde que haja um conjunto considerável de testes automatizados) com cada mudança – você sabe quando para de funcionar e pode consertá-lo imediatamente.

Equipes que trabalham com integração contínua de modo eficaz são capazes de entregar software muito mais rápido, e com menos defeitos.

Defeitos são descobertos mais cedo no processo de entrega, e é mais barato consertá-los, o que representa ganhos de custo e tempo. Dessa forma, consideramos a prática essencial para times profissionais, talvez tão importante quanto o uso de controle de versão.

De que você precisa antes de começar

1. Controle de versão
2. Processo automatizado de compilação
3. Aceitação da equipe

Quando rodar a ferramenta de IC pela primeira vez, você provavelmente vai descobrir que a máquina em que ela está sendo executada não possui o software ou as configurações necessárias para o processo de compilação.

Essa é uma oportunidade única – anote tudo o que você precisa para fazer a aplicação funcionar e coloque na página do projeto.

Você deve, então, colocar todas essas aplicações e configuração necessárias no sistema de controle de versão e automatizar o provisionamento de uma nova máquina.

Quando você estiver pronto para fazer o check-in de uma mudança:

1. Verifique se o processo de compilação está rodando. Se está, espere ele terminar. Se falhar, você precisa trabalhar com o restante da equipe para fazê-lo ter sucesso antes de enviar suas mudanças para o controle de versão.
2. Quando o processo terminar, e os testes forem bem-sucedidos, atualize o código em sua versão de desenvolvimento de controle de versão.
3. Execute os scripts de compilação e de testes na máquina de desenvolvimento, e garanta que tudo está funcionando corretamente em sua máquina, ou use a funcionalidade de compilação pessoal da ferramenta de IC.
4. Se a compilação local funcionar, faça o check-in.
5. Espere que a ferramenta de IC execute com suas mudanças.
6. Se o processo falhar, pare o que estiver fazendo e conserte o problema imediatamente em sua máquina local, voltando ao passo 3.
7. Se o processo for bem-sucedido, pule de alegria e passe para a próxima ferramenta.

Pré-requisitos:

1. Check-ins regulares
2. Crie um conjunto de testes automatizados abrangente
 1. Testes Unitários
 2. Testes de componentes
 3. Testes de aceitação
3. Mantenha o processo de compilação e de testes curto

Práticas essenciais:

1. Não faça check-ins se o processo de compilação estiver quebrado
2. Sempre rode os testes de commit localmente antes de um check-in, ou use o servidor de IC para isso
3. Espere que os testes obtenham sucesso antes de continuar
4. Nunca vá para casa com um processo de compilação quebrado
5. Esteja sempre preparado para voltar à revisão anterior
6. Limite o tempo antes de reverter
7. Não comente testes que estão falhando
8. ***Assuma a responsabilidade pelas quebras causadas por suas mudanças***
9. Desenvolvimento guiado por testes

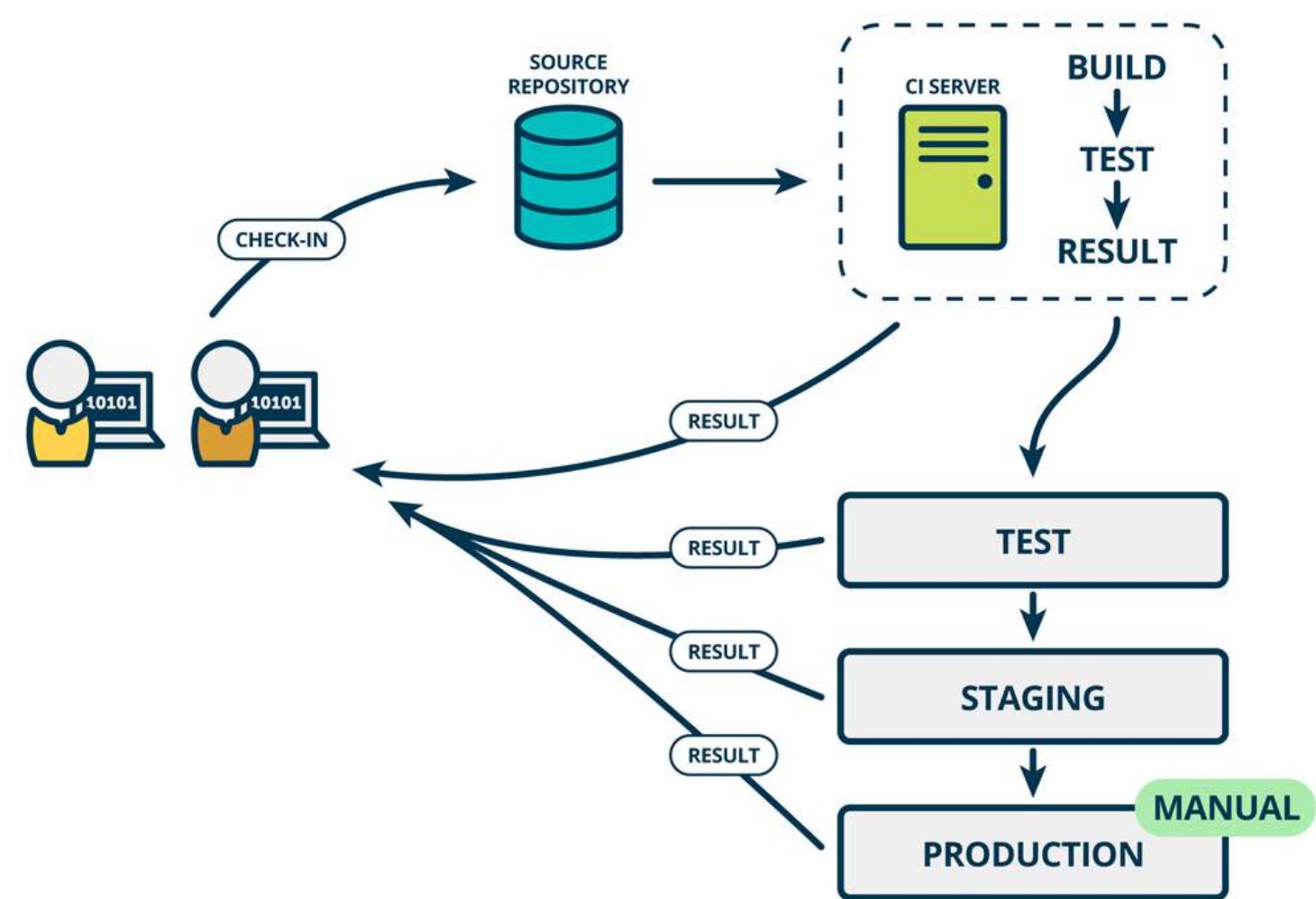
Ferramentas apoiarão equipes distribuídas

- Sistema de controle de versão distribuída
- Construção automatizada - combina código de várias fontes
- Teste automatizado - assegure-se de que a construção não esteja quebrada; se for, quem quebrou deve consertar ou reverter para que outros possam continuar trabalhando
- Liberação e implantação automatizadas (mas teste localmente antes de confirmar o código para lançamento)
- Ferramentas de comunicação eletrônica (IM e VoIP) são essenciais para a comunicação constante

A entrega contínua é a implementação automatizada dos processos de criação, implementação, teste e liberação de aplicativos.

É mais do que uma metodologia de entrega: é um paradigma completamente novo para gerir um negócio que depende de software.

Depende da colaboração efetiva entre todos os envolvidos na entrega, de suporte executivo e da vontade de fazer mudanças.



Em qualquer empresa existe o seguinte conflito:

De um lado, o negócio quer que software novo e de valor esteja à disposição dos usuários o mais rápido possível para aumentar a receita.

De outro, os responsáveis pela governança corporativa querem garantir que a organização entenda os riscos que podem levar à perda de receita ou mesmo ao fim do negócio, como a violação de regulamentação aplicável e quais processos existem para gerir esse risco.

A entrega contínua garante desempenho (entrega rápida) e conformidade (aos requisitos), gerando valor para seus usuários.

Como implementar a entrega contínua

- Revise a situação atual
- Escolha a área de foco que é imatura com base nos benefícios.
- Concorde com critérios de aceitação
- Planeje e implemente. Considere a prova de conceito em uma pequena área
- Aprimorar
- Repita para outras áreas imaturas

Entrega Contínua

Prática	Gestão de compilação e integração contínua	Ambientes e implantação	Gestão de entrega de versão e observância	Testes	Gestão de dados	Gerência de configuração
Nível 3 – Otimização Foco em melhoria do processo	Equipes se reúnem regularmente para discutir problemas de integração e como resolvê-los com automação, feedback mais rápido e melhor visibilidade.	Todos os ambientes são geridos efetivamente. O provisionamento é completamente automatizado. Virtualização é usada se aplicável.	As equipes de operações e desenvolvimento colaboram regularmente para gerir riscos e reduzir tempo de ciclo.	Rollbacks são raros. Defeitos são encontrados e resolvidos imediatamente.	Existe um ciclo de feedback de entrega e entrega quanto ao desempenho do banco de dados e processos de implantação.	Há validação regular de que a política de gerência de configuração apoia colaboração eficaz, m processo rápido de desenvolvimento, um processo de gestão e mudança auditável.
Nível 2 – Gerido quantitativamente Processos medidos e controlados	Métricas de compilação são coletadas, disponibilizadas de forma visível e mudanças são feitas com base nelas. Compilações com erro são corrigidas imediatamente.	Implantações gerenciadas e orquestradas. Processos de entrega e de reversão de problemas são testados.	A saúde de ambientes e aplicações é monitorada e gerida proativamente. O tempo de ciclo é monitorado.	Métricas e tendências de qualidade são rastreadas. Requisitos não funcionais são definidos e medidos.	Atualizações de bancos de dados e rollbacks são testadas a cada implantação. O desempenho do banco de dados é monitorado e otimizado.	Os desenvolvedores fazem check-in pelo menos uma vez por dia. Branches são usadas somente para entregas.
Nível 1 – Consistente Processos automatizados aplicados ao longo de todo o ciclo de vida da aplicação.	Compilação e testes automáticos são executados para cada check-in. Dependências são gerenciadas. Há reúso de scripts e ferramentas.	Processos completamente automatizados para implantação apertando-se um botão. O mesmo processo é usado para implantar em todos os ambientes.	Gestão de mudança e processos de aprovação são definidos e seguidos. Condições regulatórias e de observância são obtidas.	Existem testes unitários e de aceitação; estes últimos são escritos por testadores. Teste é parte do processo de desenvolvimento.	Mudanças no banco de dados são executadas automaticamente como parte do processo de implantação.	Bibliotecas e dependências são gerenciadas. Políticas de controle de versão são determinadas pelo processo de gestão de mudança.
Nível 0 – Repetível Processo documentado e parcialmente automatizado	Compilação e testes automatizados são regulares. Qualquer binário pode ser recriado do código-fonte usando um processo automatizado.	Implantação automatizada para alguns ambientes. A criação de novos ambientes é barata. Toda a configuração foi externalizada e versionada.	Entregas infrequentes e árduas, mas confiáveis. Rastreabilidade limitada de requerimentos por entregas.	Testes automatizados escritos como parte do desenvolvimento de uma história.	Mudanças para o banco de dados são feitas com scripts automatizados versionados por aplicação.	Controle de versão é usado para tudo que é exigido para recriar o software: código-fonte, configuração, scripts de compilação e implantação, migração de dados.
Nível -1 – Regressivo Processos que não podem ser repetidos, pouco controlados e reativos	Processos manuais para compilação de software. Nenhum gerenciamento de artefato e relatórios.	Processos manuais de implantação. Binários específicos por ambiente. Ambientes provisionados manualmente.	Entregas infrequentes e sem confiabilidade.	Testes manuais após a implantação.	Migrações de dados sem versionamento e feitas manualmente.	Controle de versão pouco ou não usado.

Problema : Implantações infrequentes e com erros.

Sintomas:

- Os testadores demoram muito para encerrar relatórios de defeitos. Note que esse sintoma não é exclusivamente causado por implantações infrequentes.
- Os clientes demoram muito para testar ou confirmar que histórias estão completas.
- Os testadores encontram problemas que os desenvolvedores corrigiram há muito tempo.
- Ninguém confia nos ambientes de UAT, de desempenho ou de IC, e as pessoas demonstram ceticismo em relação a quando uma nova entrega estará disponível.
- Demonstrações acontecem raramente.
- A aplicação raramente é demonstrada de maneira funcional.
- A velocidade da equipe (taxa de progresso) é menor que a esperada.

Problema : Implantações infrequentes e com erros.

Causas possíveis

Há muitas razões possíveis. Eis algumas das mais comuns:

- O processo de implantação não é automatizado.
- Não há hardware suficiente disponível.
- A configuração de hardware e do sistema operacional não é gerenciada corretamente.
- O processo de implantação depende de sistemas que estão fora do controle da equipe.
- Não há pessoas suficientes que entendem dos processos de compilação e implantação.
- Testadores, desenvolvedores, analistas e equipe de operações não estão colaborando o suficiente durante o desenvolvimento.
- Os desenvolvedores não estão sendo disciplinados o suficiente em manter a aplicação funcional por meio de pequenas mudanças incrementais e frequentemente causam problemas em funcionalidades existentes.

Problema : As equipes de entrega não conseguem implementar uma estratégia eficaz de testes.

Sintomas:

- Defeitos de regressão aparecem o tempo todo.
- O número de defeitos continua aumentando mesmo quando a equipe gasta a maior parte do tempo corrigindo-os (obviamente, esse sintoma só se manifestará se você tiver um processo de testes).
- Os clientes reclamam de um produto de baixa qualidade.
- Os desenvolvedores suspiram ou fazem cara de pânico quando um novo pedido de funcionalidade chega.
- Os desenvolvedores reclamam da capacidade de manutenção do código, mas nada melhora.
- O tempo de implementar uma funcionalidade qualquer continua aumentando, e a equipe está sempre atrasada.

Problema : As equipes de entrega não conseguem implementar uma estratégia eficaz de testes.

Causas possíveis

Há essencialmente duas fontes principais desse problema: colaboração ineficiente entre testadores e o restante da equipe de entrega e/ou testes implementados de maneira inadequada ou pouco automatizados.

Problema : O processo de compilação não é gerenciado de forma adequada.

Sintomas:

- Os desenvolvedores não fazem check-ins frequentes (pelo menos uma vez por dia).
- O estágio de commit não está funcionando.
- Há uma grande quantidade de defeitos.
- Há uma longa fase de integração antes de qualquer entrega.

Problema : O processo de compilação não é gerenciado de forma apropriada.

Causas possíveis

- Os testes automatizados demoram muito para rodar.
- O estágio de commit demora muito para rodar (menos do que cinco minutos é ideal, mais do que dez minutos é inaceitável).
- Os testes automatizados falham de maneira intermitente, com falsos positivos.
- Ninguém se sente à vontade para reverter check-ins.
- Quase ninguém entende ou consegue fazer mudanças no processo de IC.

Problema : Ambientes não podem ser comissionados e nem aplicações podem ser instaladas de maneira confiável usando um processo automatizado.

Sintomas:

- Falhas misteriosas acontecem em ambientes de produção.
- Novas implantações são eventos assustadores e carregados de tensão.
- Equipes substanciais se dedicam à configuração e gestão de ambientes.
- Implantações em produção precisam passar por rollbacks ou patches frequentes.
- Tempo de queda inaceitável de ambientes de produção.

Problema : Ambientes não podem ser comissionados e nem aplicações podem ser instaladas de maneira confiável usando um processo automatizado.

Causas possíveis

- Diferenças nos ambientes de aceitação e produção.
- Um processo de gestão de mudanças ruim ou que não é seguido nos ambientes de produção e homologação.
- Colaboração insuficiente entre as equipes de operação, gestão de dados e entrega.
- Monitoramento ineficiente dos ambientes de produção e homologação, levando a problemas na detecção de incidentes.
- Instrumentação e logging insuficientes nas aplicações.
- Testes não funcionais insuficientes.



Sua empresa está mudando sua forma de trabalhar e começando a usar DevOps. Sua equipe abraçou essa mudança. Você está discutindo as práticas recomendáveis para a fase de confirmação do código. Seu colega de trabalho Sidney diz: “*Você deve efetuar check in, mesmo quando a compilação estiver com defeito. Talvez sua alteração faça a compilação voltar a funcionar*”. Essa prática parece saudável?

- A) Não. Você nunca deve efetuar check in em uma compilação com defeito. Seus colegas de trabalho devem corrigir a compilação antes de você efetuar login.
- B) Não. Primeiro, você deve testar se sua alteração corrige a compilação. Se corrigir, você deve inserir seu código.
- C) Sim. Ao inserir seu código, você permite que a compilação faça mais testes. Isso ajuda a identificar o problema.
- D) Sim. Seus colegas de trabalho já estão trabalhando para corrigir a compilação; assim, você pode facilmente inserir seu código.



A Integração Contínua é uma prática essencial para o DevOps. É tão importante quanto utilizar um sistema de controle de versão para o desenvolvimento.

Qual é o principal benefício da Integração Contínua?

- A) Ela permite que o time priorize melhor o desenvolvimento e o teste.
- B) Ela permite que o usuário esteja envolvido com os testes continuamente.
- C) Ela permite testar a integração adequada para a produção de serviços de TI.
- D) Ela permite que você entregue o software mais rápido, com menos erros e com menor custo.



Para um novo produto, sua equipe precisa desenvolver um Pipeline de Implantação. Como parte da Integração Contínua, você precisa definir o estágio de Confirmação do pipeline. Você pode discutir esta fase com os membros de sua equipe.

O Process Master afirma: "É suficiente compilar o teste no estágio de Confirmação (Commit), para que o Pipeline de Implantação mantenha a velocidade." Isso é verdade?

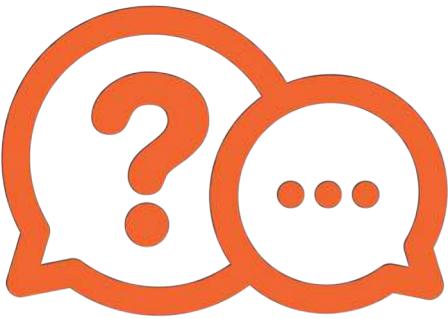
- A) Sim. O foco do estágio de Confirmação é confirmar o código. Não há necessidade de ter pressa.
- B) Sim. Não há problema, pois há um estágio de testes depois do estágio de Confirmação.
- C) Não. O processo de confirmação deve ser tão lento quanto possível para que o fluxo seja contínuo.
- D) Não. O controle de versão e o teste básico devem ser feitos no estágio de Confirmação.



Um time de Desenvolvimento está interessado no DevOps. Eles estão principalmente interessados na Integração Contínua (IC). Eles, atualmente, desenvolvem e mantêm 3 soluções grandes e 4 menores. Eles utilizam práticas Scrum. Cada sprint leva 4 semanas, criando uma média de 1 liberação confirmada para o ambiente de Teste a cada 10 ou 15 dias e 1 liberação para produção por mês. Eles querem criar um caso de negócio qualitativo para a gestão deles apoiar seu investimento e esforço a fim de criar uma prática de IC.

Quais benefícios tangíveis de IC ajudam mais nesse caso de negócio?

- A) A implantação no ambiente de teste uma vez por dia poderia aumentar os benefícios para o negócio e diminuir consideravelmente os custos de desenvolvimento.
- B) Isso ajuda o espírito de equipe. Como eles já estão utilizando o Scrum, a IC não gerará benefícios mensuráveis para o negócio.
- C) Isso aumenta a estabilidade do negócio com os melhores testes de Integração, mantendo a velocidade de liberação para evitar custos adicionais.
- D) A liberação para a produção uma vez por dia poderia aumentar os benefícios para o negócio e diminuir consideravelmente os custos de desenvolvimento.



A Entrega Contínua é um novo paradigma para a execução de um negócio que depende de softwares. O Pipeline de Implantação é o cerne de qualquer implementação de DevOps eficaz, pois ajuda o objetivo final da empresa e sua governança corporativa.

Como um Pipeline de Implantação contribui para a Governança Corporativa?

- A) Ele ajuda a alcançar ao mesmo tempo desempenho e conformidade, ao fornecer alto valor.
- B) Ele ajuda a gerar apenas a conformidade, pois um bom Pipeline de Implantação aumenta a transparência.
- C) Ele ajuda a aumentar apenas a conformidade, pois um bom Pipeline de Implantação ajuda a entregar de modo rápido.
- D) Ele ajuda a priorizar o desempenho ou a conformidade, pois ambos não podem ser alcançados ao mesmo tempo.



Sua empresa está mudando sua forma de trabalhar e começando a usar DevOps. Sua equipe abraçou essa mudança. Você está discutindo as práticas recomendáveis para a fase de confirmação (commit) do código.

Sua colega de trabalho Sun diz: “Quando a compilação é quebrada, e ninguém assume a responsabilidade, deveríamos descobrir quem fez isso e contar o que houve, para que a pessoa possa corrigir a compilação”.

Essa é uma boa ideia?

- A) Sim. Somente a pessoa que quebrou a compilação pode corrigi-la; você deve identificá-las, mesmo que seja desconfortável.
- B) Sim. Você deve sempre assumir a responsabilidade por quebrar uma compilação. Se não fizer isso, seus colegas de trabalho podem impor essa regra.
- C) Não. DevOps é um ambiente livre de culpa. Se um colega de trabalho não assumir a responsabilidade, não o force.
- D) Não. Primeiro, você deve retomar o desenvolvimento. Então, arranje tempo para identificar a pessoa responsável e puni-la por isso.



Você está avaliando os Compiladores da Empresa, uma organização de médio a grande porte que adotou as práticas do DevOps alguns anos atrás. Eles contrataram você para determinar seu atual estado de maturidade. Ao terminar, você deve dar sugestões de aprimoramento. Eles querem saber em qual área devem se concentrar para alcançar o nível de maturidade seguinte: Nível 2 – Gerenciado Quantitativamente.

Você constata que a maioria das áreas é de Nível 1 – Consistente com duas exceções:

- Ambientes e Implantação. Essa área gerencia liberações orquestradas e testou os processos de Liberação e de reversão.
- Gerenciamento de Compilação e Integração Contínua. Nesta área, você constata compilações e testes automatizados regulares e qualquer compilação podem ser recriados a partir do controle de origem utilizando um processo automatizado.

Determine o nível de maturidade nessas duas áreas, com base nas informações fornecidas. Então, forneça sua recomendação para o foco de aprimoramento.

Em qual dessas duas áreas os Compiladores da Empresa devem trabalhar, antes de avançar para o Nível 2?



- A) Ambientes e Implantação e Gerenciamento de Compilação e Integração Contínua estão ambos no Nível 0. O trabalho deve ser finalizado em ambos os ambientes ao mesmo tempo.
- B) Ambientes e Implantação e Gerenciamento de Compilação e Integração Contínua estão no Nível 1 ou acima. O trabalho deve ser finalizado em outras áreas para progredir.
- C) Ambientes e Implantação estão no Nível 0. Desenvolver Gerenciamento e Integração Contínua está no Nível 1. O foco deve estar primeiramente em Ambientes e Implantação.
- D) Ambientes e Implantação estão no Nível 2. Desenvolver Gerenciamento e Integração Contínua está no Nível 0. O foco deve estar somente em Gerenciamento de Compilação e em Integração Contínua.

3.2 – Pipeline de Implantação

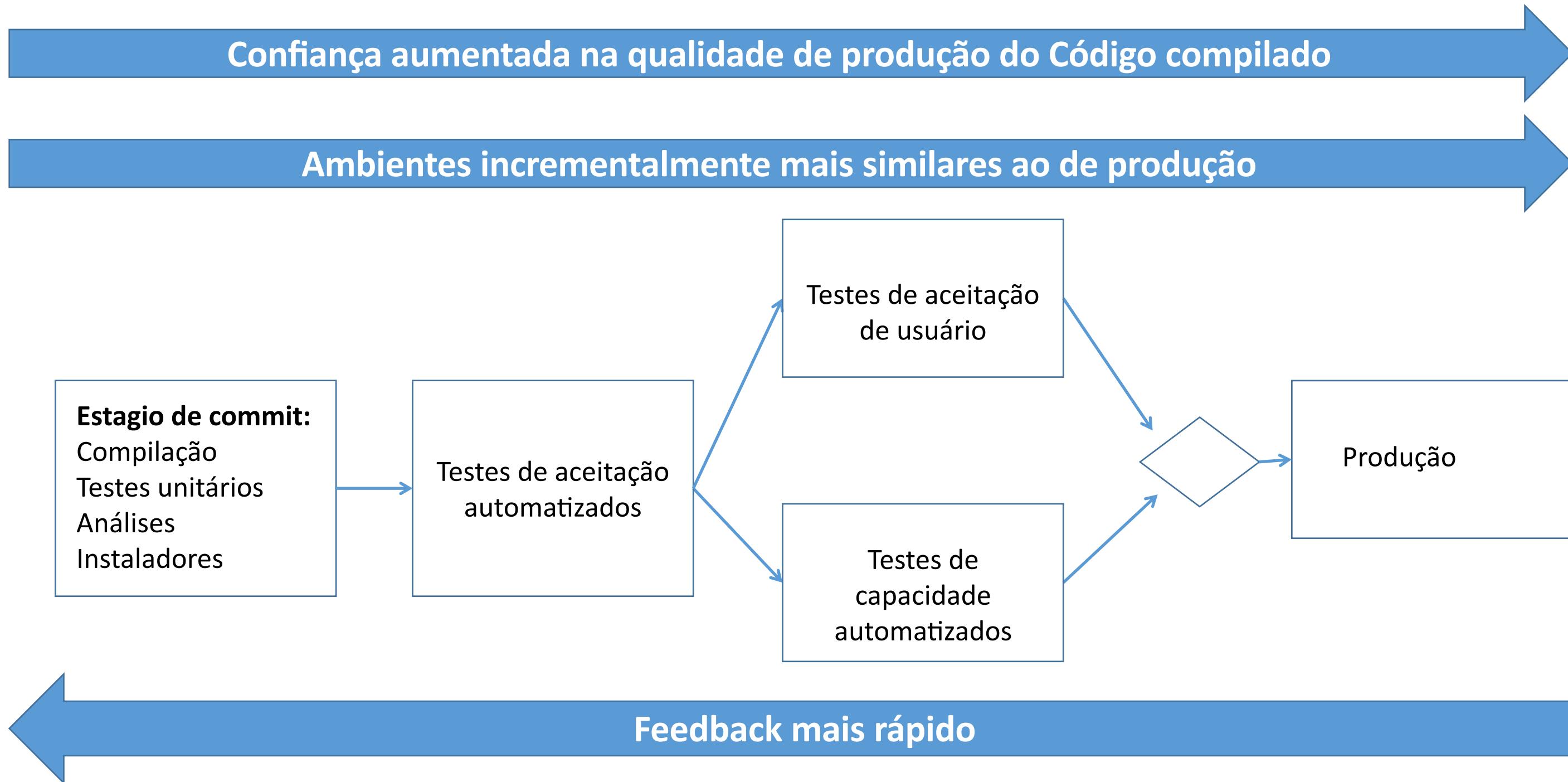
			Referência de Literatura			
			% Peso	Questões	Effective DevOps	Entrega Contínua
3.2 Pipeline de implantação						
3.2.1	Explicar a lógica da anatomia de um pipeline de implantação DevOps		4%	2		5,6
3.2.2	Explicar como usar scripts de criação e implantação					5

Pipelines de implantação criam um processo confiável, automatizado e passível de repetição, para que mudanças cheguem à produção o mais rápido possível.

Em um nível abstrato, o pipeline de implantação é uma manifestação automatizada do processo de levar o software do controle de versão até os usuários.

Contribuem para criar software da mais alta qualidade usando um processo de alta qualidade, reduzindo massivamente o risco de entregas.

Anatomia de um pipeline de implantação



O estágio de commit garante que o sistema funcione em um nível técnico. Ele compila, passa com sucesso por um conjunto de testes automatizados (rápidos) e é feita a análise de código.

Passos do estágio de commit:

- Compilar o código (se necessário).
- Executar um conjunto de testes (rápidos).
- Criar binários para uso nos demais estágios.
- Realizar análise de código para verificar sua qualidade.
- Preparar artefatos, como bancos de dados, para uso nos demais estágios.

Métricas úteis do estágio de commit:

- Cobertura de testes (80% é um bom número)
- Quantidade de código duplicado
- Acoplamento do código
- Número de avisos
- Estilo de código

O estágio de testes de aceitação automatizados garante que o sistema funcione dos pontos de vista funcional e não funcional; que seu comportamento atenda às necessidades dos usuários e às especificações do cliente.

O estágio de testes manuais garante que o sistema seja usável e atenda seus requisitos, detectando defeitos que não foram encontrados pelos testes automatizados, e que ele tenha valor para seus usuários. Esse estágio normalmente inclui ambientes de testes exploratórios, ambientes de integração e UAT (user acceptance testing, ou testes de aceitação de usuário).

O estágio de entrega coloca o sistema nas mãos dos usuários, seja com um software empacotado ou por sua implantação em um ambiente apropriado de produção ou homologação (um ambiente de homologação é um ambiente de testes idêntico ao ambiente de produção).

Independentemente de estar começando um novo projeto do zero ou tentando criar um pipeline automatizado para um projeto existente, você deve usar uma abordagem incremental para a implementação de um pipeline de implantação.

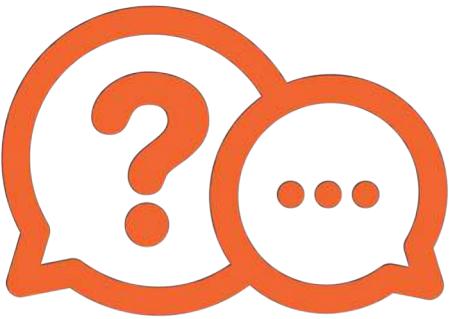
Geralmente, os passos são os seguintes:

- Modelar o fluxo de valor e criar um esqueleto do processo.
- Automatizar os processos de compilação e implantação.
- Automatizar os testes unitários e análise de código.
- Automatizar os testes de aceitação.
- Automatizar a entrega como um todo.

- Crie um script para cada estágio em sua implantação
- Utilize a tecnologia apropriada para implantar sua aplicação
- Utilize os mesmos scripts para implantar em todos os ambientes
- Separe as informações de configuração do script e as armazene no controle de versão
- Utilize as ferramentas de gerenciamento de pacotes de seu sistema operacional
- Garanta que o processo de implantação seja idempotente
- Evolua seu sistema incrementalmente

Ferramentas de compilação orientadas à tarefa, como o Ant, NAnt e MsBuild, descrevem a rede de dependências como um conjunto de tarefas.

Uma ferramenta orientada ao produto, como Make, descreve as coisas em termos do produto gerado, como um executável.



Considere a anatomia de um pipeline básico de implantação.
Que fase afirma que o sistema funciona no nível funcional e não funcional?

- A) Teste automatizado de aceitação
- B) Teste de compilação e de unidade
- C) Teste manual de aceitação
- D) Controle de versão



É uma prática recomendada do DevOps utilizar o mesmo processo para implantar em todos os ambientes em que seu aplicativo é executado. Isso garante que a compilação seja testada de maneira eficaz. Você está utilizando scripts para automatizar o processo de compilação e de implantação.

Qual é a melhor maneira de fazer isso?

- A) Utilize um script para cada ambiente e mantenha-os como parte do sistema de controle de versão.
- B) Utilize um script específico para cada ambiente para abordar as diferenças entre ambientes.
- C) Utilize os mesmos scripts para cada ambiente, tendo parâmetros manuais para configurações específicas.
- D) Utilize os mesmos scripts para implantar em cada ambiente e gerenciar informações de configuração separadamente.



Um Pipeline de Implantação é definido para alcançar liberações rápidas, replicáveis e confiáveis.
Se alguma parte do Pipeline falhar, o que deve ser feito?

- A) Continuar com a inserção. Resolver o problema quando ele for retirado da lista de backlog.
- B) Continuar com a próxima inserção, mas somente até a fase de controle de versão.
- C) Parar os check-ins e esperar até que o grupo ou a pessoa responsável corrija o problema.
- D) Parar a linha e colocar todo o time de DevOps no trabalho a fim de corrigir o problema.



Qual dos seguintes itens não é considerado um princípio e prática geral para um script de desenvolvimento e de implantação?

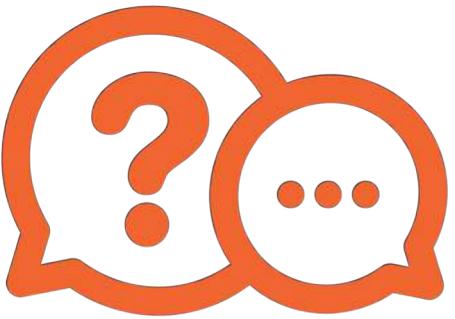
- A) Criar um script para cada estágio de seu Pipeline de Implantação.
- B) Desenvolver incrementalmente seu sistema de implantação.
- C) Usar uma tecnologia apropriada para implantar seu aplicativo.
- D) Usar scripts diferentes a serem implantados em todos os ambientes.



Para um novo produto, sua equipe precisa desenvolver um Pipeline de Implantação. Como parte da Integração Contínua, você precisa definir o estágio de Confirmação do pipeline. Você pode discutir esta fase com os membros de sua equipe.

O Process Master afirma: "É suficiente apenas compilar o código no estágio de Commit para que o Pipeline de Implantação mantenha a velocidade." Isso é verdade?

- A) Sim. O foco do estágio de Confirmação é confirmar o código. Não há necessidade de ter pressa.
- B) Sim. Não há problema, pois há um estágio de testes depois do estágio de Commit.
- C) Não. O processo de confirmação deve ser tão lento quanto possível para que o fluxo seja contínuo.
- D) Não. Testes rápidos devem ser feitos no estágio de Commit.



Você cria arquivos executáveis usando uma ferramenta de compilação. Que tipo de ferramenta de compilação é essa?

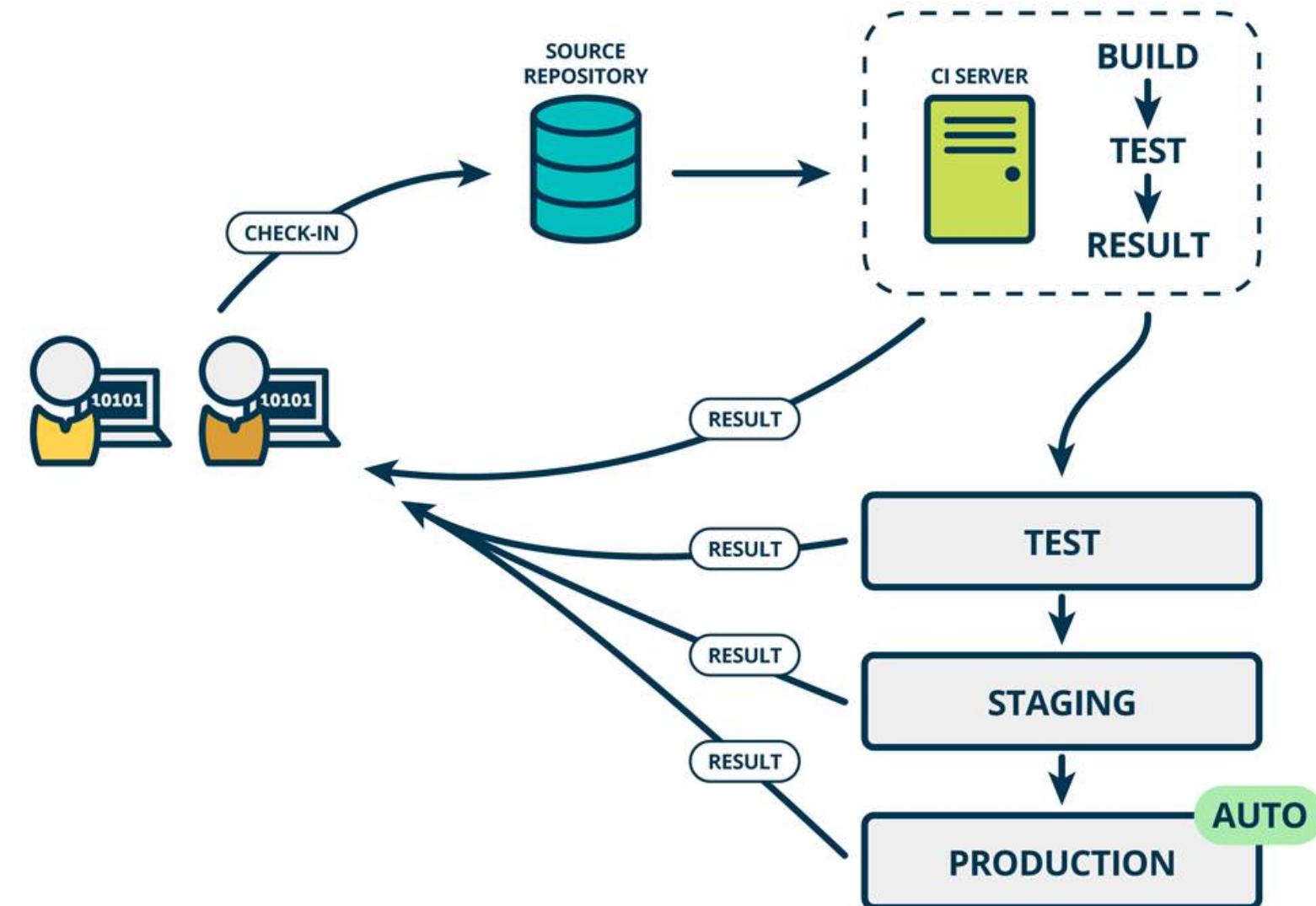
- A) Orientado a dados
- B) Orientado a dependências
- C) Orientado a produtos
- D) Orientado a tarefas

3.3 – Implantação Contínua

			Referência de Literatura			
			% Peso	Questões	Effective DevOps	Entrega Contínua
3.3 Implantação contínua						
3.3.1	Explicar por que o plano de iteração e o plano de liberação devem ser alterados para um DevOps eficaz		4%	2		10
3.3.2	Analisar como implementar a implantação contínua em um cenário					8

Na implantação contínua todas as alterações que passam seus testes automatizados são implementadas automaticamente em produção.

A diferença principal entre uma implantação e uma entrega de versão é a capacidade de rollback.



Boas práticas

- As pessoas que fazem a implantação deveriam estar envolvidas na criação do processo
- Mantenha um log de atividades de implantação
- Não remova arquivos antigos: mova-os
- Implantações são responsabilidade de toda a equipe
- Tenha um período de transição para novas implantações
- Falhe rapidamente
- Não faça mudanças diretamente no ambiente de produção

A parte mais crucial do planejamento de entrega é reunir os representantes de cada parte da organização envolvida na entrega: infraestrutura, operações, desenvolvimento, testes, DBAs e suporte. Essas pessoas devem se reunir regularmente durante o ciclo de vida do projeto e trabalhar continuamente para tornar o processo de entrega mais eficiente.

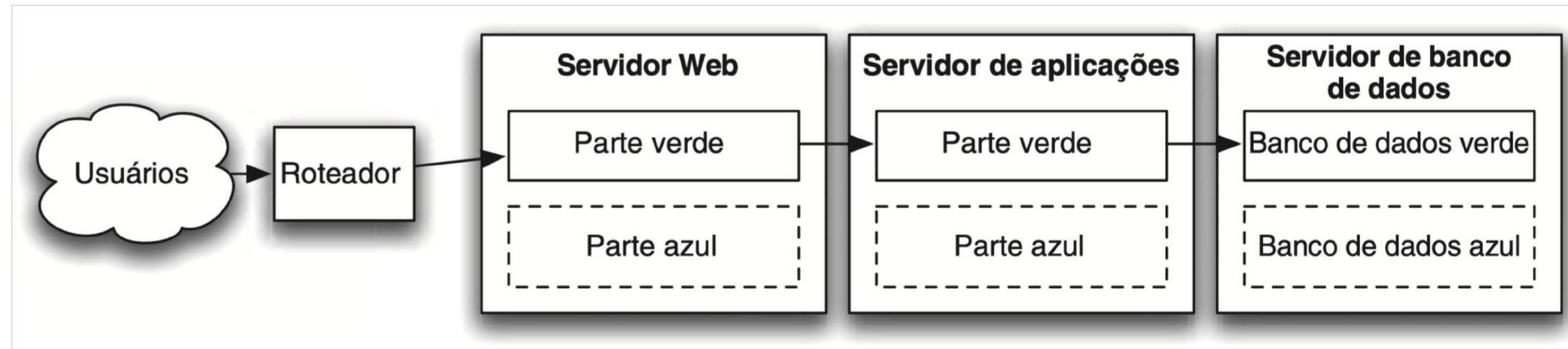
Como planejar a estratégia de implantação

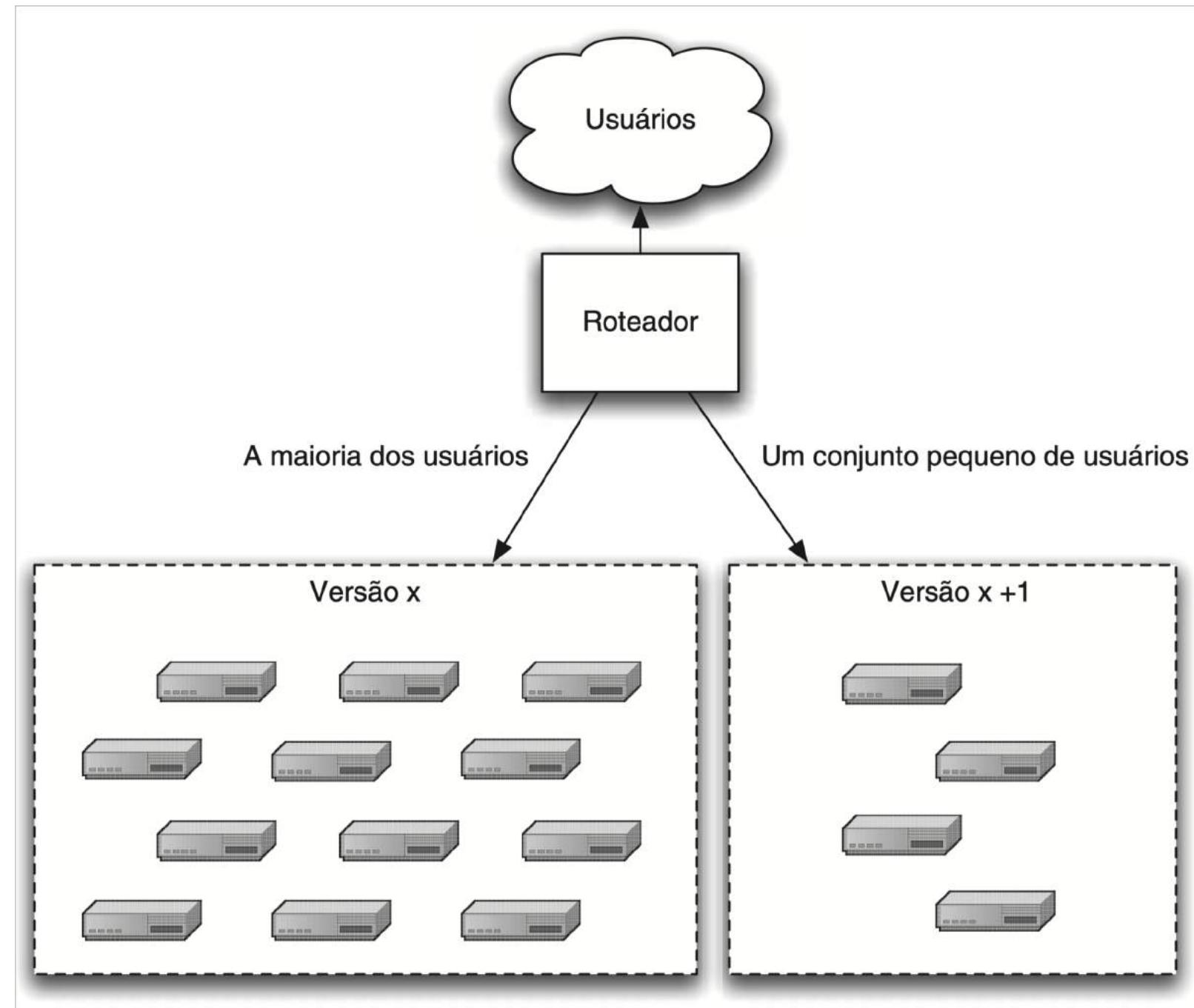
Deve ser parte do seu planejamento e, pelo menos em parte, influenciar suas decisões de desenvolvimento desde o início do projeto. A estratégia de implantação irá, e deve, evoluir com o tempo, tornando-se mais correta e mais detalhada à medida que a primeira entrega se aproxima.

A parte mais crucial do planejamento de entrega é reunir os representantes de cada parte da organização envolvida na entrega: infraestrutura, operações, desenvolvimento, testes, DBAs e suporte. Essas pessoas devem se reunir regularmente durante o ciclo de vida do projeto e trabalhar continuamente para tornar o processo de entrega mais eficiente.

Uma implantação sem parada é aquela em que o processo real de mudar os usuários de uma versão para outra acontece de maneira quase instantânea.

É fundamental que seja possível voltar para a versão anterior quase que de imediato se algo der errado.







A empresa MyApp tem uma forte reputação na construção de aplicativos robustos e poderosos de compras pela web. Eles fizeram isso para muitas grandes organizações de varejo. Eles usam parte de um modelo SDLC (Software Development Lifecycle) tradicional para assegurar que o rigor adequado, o controle e a QA (Quality Assurance) sejam seguidos antes de liberar o aplicativo de compras pela web para o cliente. No entanto, recentemente surgiu uma competição agressiva. Seus concorrentes lançam aplicativos de compras pela web em um ritmo muito mais rápido e os disponibiliza em dispositivos móveis.

A MyApp precisa repensar como implantar e lançar aplicativos. Ainda existem alguns desafios:

- O Desenvolvimento muitas vezes desconhece os requisitos operacionais do aplicativo
- Eles experimentam falhas em scripts de implantação
- As atualizações falham ao implantar novos aprimoramentos a clientes existentes
- O time de Operações está constantemente apagando incêndios para manter os aplicativos existentes de compras em execução.

Qual é a melhor etapa para a MyApp adotar a mudança para um modelo de Implantação Contínua?



- A) O time de Desenvolvimento envolve Operações no início de um projeto e cria scripts de implantação no final de um projeto.
- B) O time de Desenvolvimento deve envolver Operações em todo o projeto e colaborar e criar scripts de implantação juntos.
- C) O time de Operações solicita que o de Desenvolvimento, no final do projeto de desenvolvimento, planeje e desenvolva scripts de implantação para o time de Operações.
- D) O time de Operações deve envolver-se na liberação e criar scripts de implantação, quando o aplicativo estiver totalmente desenvolvido e testado pelo time de Desenvolvimento.



Tradicionalmente, a empresa MyApp tem utilizado o modelo SDLC comum para fornecer softwares em produção. Devido ao aumento da concorrência no mercado, eles precisam encontrar uma forma mais eficaz de implantar aplicativos em produção de maneira muito mais rápida. No momento, um novo aprimoramento a um aplicativo existente leva de 2 a 3 meses para ser liberado em produção. Eles precisam reduzir o tempo de ciclo para 3 dias.

Qual prática principal de Implantação Contínua abaixo não será benéfica para a MyApp reduzir seu tempo de ciclo de liberação conforme necessário?

- A) Automatizar e validar todos os testes de unidade, testes de componentes e aceitação com 100% de cobertura.
- B) Automatizar todos os processos de compilação, implantação e liberação.
- C) Implantação automática das mudanças do aplicativo de forma contínua em produção.
- D) Certificar-se de que todas as mudanças de aprimoramento passem pela Garantia de Qualidade para extensa análise antes de passar para a produção.

3.4 – Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
3.4 Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)					
3.4.1 Explicar os conceitos Ji-Kotei-Kanketsu, Ritmo, Trabalho em Andamento e Fluxo Único (Fluxo Contínuo)	4%	2			4,7
3.4.2 Analisar um cenário para um problema com Ji-kotei-Kanketsu, Ritmo, Trabalho em andamento ou Fluxo Único e encontrar uma solução adequada					

Para criar processos alinhados, o JKK é o método mais eficaz para guiar o comportamento da equipe de DevOps.

O JKK é uma forma de trabalhar com qualidade, o que significa um entendimento claro dos objetivos, a compreensão do caminho certo para o trabalho, a obtenção do trabalho certo para 100% de conclusão e a manutenção da qualidade exigida sem inspeções.

Ji-Kotei-Kanketsu

Compreensão clara das metas, garantia de alta qualidade de trabalho, defeitos que nunca são passados para o próximo processo, definição de pronto é vital.

Com este método de abordagem, o que significa que “para garantir que nenhum defeito ocorra nas linhas e que itens defeituosos nunca sejam passados para o próximo processo”, as ações necessárias são tomadas para atingir a meta de “zero defeito” em todos processos de produção.

Obeya

"grande sala" ou "sala de guerra", refere-se a uma forma de gerenciamento de projetos usada em empresas asiáticas (incluindo a Toyota) e é um componente da manufatura enxuta e, em particular, do Sistema Toyota de Produção.

Ritmo

o ritmo que a equipe de DevOps trabalha e implanta na produção

Work-in-progress (WIP)

Reducir o WIP para reduzir os gargalos e melhorar o tempo de ciclo.

Limitar a quantidade de trabalho que você iniciou, mas ainda não concluiu é uma excelente maneira de aumentar o rendimento em seu pipeline de desenvolvimento de software.

Fluxo de uma peça

Trabalhando em um item de cada vez para completar como um indivíduo ou uma equipe, ritmo acelerado, fluxo suave.



O DevOps tem conceitos muito importantes do Agile (Ágil), derivado do Sistema Toyota de Produção.

Por que é importante para o DevOps adotar o Ji-Koutei-Kanketsu (JKK)?

- A) Cria uma Definição de pronto com base na conclusão de 100% de um item de alta qualidade.
- B) Ajuda o Product Owner a gerenciar o backlog de produtos para a conclusão de 100%.
- C) É um fator-chave de sucesso estabelecer um pipeline de implantação único para os serviços de TI.
- D) Fornece uma definição de qualidade baseada em Kaizen ou melhoria contínua.



Qual é o principal benefício da utilização do sistema Obeya?

- A) Facilita as queixas do cliente, para garantir que o time receba bastante feedback a fim de melhorar continuamente
- B) Lida com a pressão dentro dos times, para que os membros do time possam manter um ritmo sustentável
- C) Melhora os relatórios de erros diários, o que garante menos retrabalho e reduz a transferência de erros para outras estações de trabalho
- D) Rápida tomada de decisão, com base na situação atual, por meio da rapidez de coleta e compartilhamento de informações



O CTO acredita que seria mais eficaz aplicar determinados conceitos Lean ao implementar o DevOps.

Quais princípios ou práticas Lean serão mais eficazes ao introduzir o DevOps?

- A) Kaizen e 5S. Como o Agile (Ágil) e o DevOps baseiam-se no cerne dos conceitos Lean, e Kaizen e 5S são a base do Lean, eles serão mais eficazes ao introduzir o DevOps.
- B) Kaizen antecipado. O DevOps requer feedback do time de Operações para o time de Desenvolvimento. O Kaizen cria antecipadamente um ciclo de feedback de baixo para cima, ajudando a aplicar este princípio no DevOps.
- C) Sistema Obeya. O DevOps integra diferentes processos de estilo de gestão. O sistema Obeya ajuda a visualizar todo o processo, permitindo uma introdução bem-sucedida do DevOps.
- D) Fluxo de peça única e JKK. Os benefícios do DevOps vão da criação de processos de baixo para cima a um fluxo de valor único. Um Fluxo Único/Fluxo Contínuo permite isso, e o JKK ajuda a simplificar e implementar o fluxo.



A MyApp é uma empresa de tecnologia que fornece soluções personalizadas de tecnologia com base na nuvem para uma carteira diversificada de clientes. A fim de aumentar a velocidade de seus negócios e conseguir expandir sua carteira de clientes, eles adotaram uma abordagem de DevOps para sua área de entrega ao cliente.

Após a adoção dessa abordagem, eles notaram uma melhoria na qualidade dos seus produtos de software, com menos impacto e detecção de erros mais rápida dentro do processo de entrega. Eles não perceberam uma grande alteração em sua velocidade para o mercado; portanto, o objetivo do negócio não está sendo atingido. Eles ainda precisam aumentar a velocidade da entrega.

Qual prática ajuda a aumentar ainda mais a velocidade, sem comprometer a qualidade?

- A) Extreme programming
- B) Ji-Kotei-Kanketsu
- C) Fluxo Único/Fluxo Contínuo

3.5 – Automação, Ferramentas e Testes

		% Peso	Questões	Referência de Literatura		
				Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
3.5 Automação, Ferramentas e Testes						
3.5.1	Explicar por que a automação é importante para o DevOps eficaz					
3.5.2	Explicar como usar ferramentas para facilitar DevOps em geral					
3.5.3	Explicar como usar ferramentas para dar suporte à mentalidade e cultura do DevOps					
3.5.4	Explicar por que é importante que o teste de DevOps seja automatizado					
3.5.5	Analizar um cenário e escolher a maneira correta de automatizar um teste de aceitação	6%	3	4,11,12,13	3,4,5,6,7,8,9	

Automação suporta os objetivos do DevOps

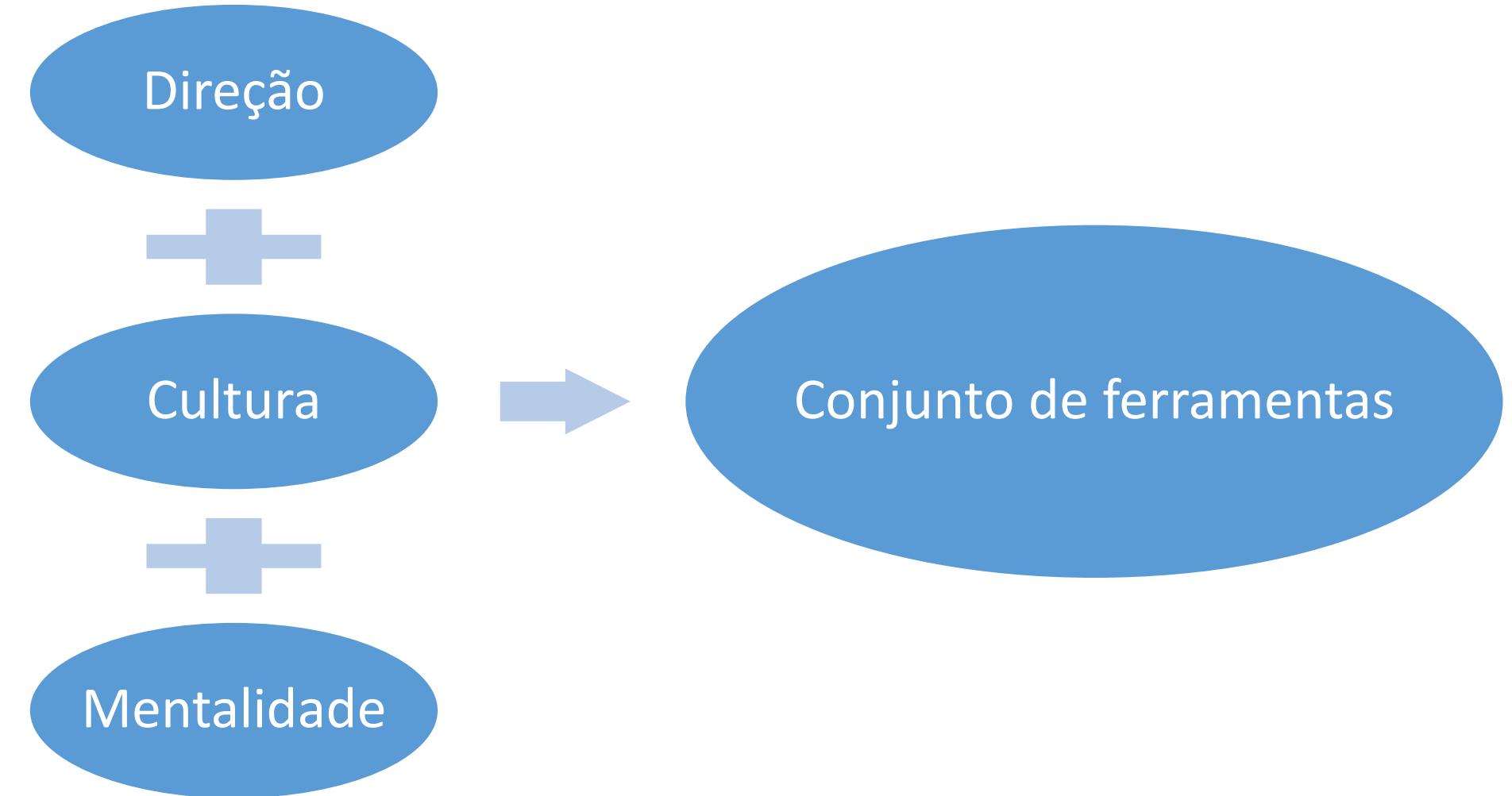
simplificando e removendo as tarefas repetitivas que podem estar sujeitas a erros humanos

e reduzindo o trabalho, energia e / ou materiais

para melhorar a qualidade e precisão

As equipes de DevOps precisam das ferramentas para realizar seus trabalhos de forma eficaz

- Desenvolvimento de software
- Ambiente de desenvolvimento local
- Controle de versão
- Gerenciamento de artefatos (gerenciamento de repositório)
- Containerização
- Testando
- Gestão de infra-estrutura
 - Gerenciamento de operações de TI (painéis de controle)
 - Monitoramento de aplicativo, rede e sistema
 - Gerenciamento de serviços de TI
 - balcão de atendimento e gerenciamento de incidentes
 - Gerenciamento de nível de serviço



As ferramentas são aceleradoras, aumentando nossa velocidade ao impulsionar a mudança com base na cultura e na direção atuais de uma organização.

A forma como as ferramentas são usadas e a facilidade com que podem ser usadas impactam na aceitação e proliferação de aspectos específicos da cultura.

As ferramentas são usadas pelas pessoas, para ajudá-las a trabalhar com outras pessoas, para criar soluções para as pessoas, e não podemos remover esse lado humano da equação de ferramentas. As ferramentas podem impactar e ser impactadas pela forma como trabalhamos e interagimos, e todos esses fatores e interações devem ser considerados para trazer uma mudança significativa e duradoura

Testes de aceitação são um estágio fundamental no pipeline de implantação: eles são responsáveis por conduzir a equipe a um estágio além da integração contínua.

Uma vez que tiver testes de aceitação em seu pipeline, você está testando os critérios de aceitação do negócio de sua aplicação, isto é, validando que o código entregue funcionalidade de valor aos usuários.

Um teste de aceitação individual tem o objetivo de verificar se um critério de aceitação de uma história ou um requisito foi atendido.

O conjunto de testes de aceitação de uma aplicação tanto garante que ela entrega o valor de negócio esperado como a protege de regressões e defeitos que quebram funções preexistentes da aplicação.

O objetivo dos testes de aceitação é provar que a aplicação faz o que o cliente espera, e não que funciona da maneira como os programadores acham que funciona.

Eles testam histórias completas em uma versão da aplicação em um ambiente como o de produção.

Testes de aceitação automatizados:

- Reduzem os custos de longo prazo
- Identificam defeitos mais rapidamente
- Melhoram a colaboração entre testadores, desenvolvedores e clientes

Ferramentas como Cucumber, JBehave, Twist e FitNesse permitem que você escreva testes de aceitação como estes em arquivos de texto, mantendo-os sincronizados com a aplicação.

Funcionalidade: Realizar uma compra

Cenário: Uma compra deve debitar em conta corretamente

Dado um título chamado de debênture

E um usuário chamado Dave com 50 reais em sua conta

Quando eu entro no sistema como Dave

E selecione o título debênture

E faço uma compra de 4 títulos a 10 reais cada

E a compra termina com sucesso

Então restam 10 reais na conta



Você quer alcançar os seguintes resultados:

- Reduzir os custos de longo prazo
- Identificar os defeitos mais rapidamente
- Ter melhor colaboração entre testadores, desenvolvedores e clientes

Qual abordagem de teste mais ajuda você fazer isso?

- A) Um teste de aceitação automatizada, pois ela reduz o ciclo de comentários, gerando esses efeitos.
- B) Testes unitários automatizados, pois economiza tempo, deixando tempo para os testadores colaborarem mais.
- C) Testes de aceitação manual para cada lançamento, pois isso garante que você encontre todos os defeitos juntos.
- D) Testes unitários e de componentes, pois os testadores podem facilmente apontar erros uns dos outros.

Módulo 4

Operação e Dimensionamento

Requisitos do Exame

EXIN

DevOps MASTER™

		% Peso	Questões	Referência de Literatura
				Effective DevOps Entrega Contínua Success with Enterprise DevOps
4 Operação e Dimensionamento				
4.1 Gerenciamento de Dados; Infraestrutura e Ambientes; e Componentes e Dependências				
4.1.1 Explicar quais problemas podem ser encontrados ao gerenciar dados em bancos de dados dentro do DevOps				
4.1.2 Analisar um cenário onde um banco de dados é usado em DevOps e fornecer a melhor solução para um problema				
4.1.3 Analisar um cenário e identificar a melhor maneira de preparar um ambiente de infraestrutura para implantação ou gerenciá-lo após a implantação	10%	5		11,12,13
4.1.4 Analisar um cenário e sugerir uma estratégia comumente usada para gerenciar componentes				
4.1.5 Explicar como gerenciar dependências				
4.2 Gerenciamento de Configuração e Controle de Versão				
4.2.1 Explicar por que o controle de versão é uma chave para o DevOps eficaz				
4.2.2 Explicar como manter o controle de versão sobre dados, infraestrutura e componentes	4%	2		2,14
4.2.3 Analisar um cenário e sugerir a melhor estratégia para gerenciar um problema de gerenciamento de configuração				
4.3 Infraestrutura em Nuvens e Imutável				
4.3.1 Explicar quando é e quando não é necessário mover para a infraestrutura baseada em nuvem para ter um DevOps eficaz	2%	1	4,5,14,16	11
4.3.2 Explicar como a infraestrutura baseada em nuvem deve ser gerenciada dentro do DevOps				
4.4 Continuidade do Negócio				
4.4.1 Explicar como o DevOps pode facilitar práticas de continuidade de negócios	2%	1		2,4
4.5 Dimensionamento				
4.5.1 Analisar um cenário, explicar se e por que é importante dimensionar para cima ou para baixo nessa situação, e identificar a melhor maneira de fazer isso				
4.5.2 Analisar um cenário que deu errada a dimensionamento, e identificar uma boa maneira de resolver o problema	4%	2	14,15,16,17	
4.5.3 Explicar como a política social e práticas de contratação suportam escalonamento DevOps				

4.1 – Gerenciamento de Dados; Infraestrutura e Ambientes; e Componentes e Dependências

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
4.1 Gerenciamento de Dados; Infraestrutura e Ambientes; e Componentes e Dependências	10%	5			11,12,13
4.1.1 Explicar quais problemas podem ser encontrados ao gerenciar dados em bancos de dados dentro do DevOps					
4.1.2 Analisar um cenário onde um banco de dados é usado em DevOps e fornecer a melhor solução para um problema					
4.1.3 Analisar um cenário e identificar a melhor maneira de preparar um ambiente de infraestrutura para implantação ou gerenciá-lo após a implantação					
4.1.4 Analisar um cenário e sugerir uma estratégia comumente usada para gerenciar componentes					
4.1.5 Explicar como gerenciar dependências					

O gerenciamento e a organização de dados impõem um conjunto específico de problemas para os processos de teste e implantação:

- Enorme volume de informações
- Dados devem ser persistentes
- Dificuldade em automatizar o processo de migração do banco de dados
- Pode ser bem complexo gerenciar cargas de dados para testes

Princípios e práticas importantes:

- Versione seu banco de dados e use ferramentas como o DbDeploy para gerenciar migrações automaticamente.
- Use Schemas para manter a compatibilidade com versões anteriores e posteriores
- Para seus testes, procure não usar dumps dos dados de produção. Ao invés disso, crie conjuntos de dados personalizados, selecionando cuidadosamente um subconjunto menor de dados de produção.

Um aspecto extremamente importante é a habilidade de reproduzir um ambiente, juntamente à aplicação que roda nele, de forma automatizada através de scripts.

Seu script deve primeiramente criar a estrutura do banco de dados, instâncias, esquemas, e assim por diante, e então preencher as tabelas do banco de dados com qualquer dado de referência necessário para que sua aplicação seja inicializada.

Em sua forma mais simples, o processo de implantação de um novo banco de dados é:

- Apague o que estava lá antes.
- Crie a estrutura do banco de dados, instâncias, esquemas, etc.
- Carregue o banco de dados com dados.

Precisamos ter um cuidado especial em ambientes em que várias aplicações utilizam um mesmo banco de dados, pois uma mudança pode impactar outras aplicações.

Neste cenário precisamos fazer o seguinte:

1. Testar as mudanças em um ambiente orquestrado, no qual eles montam um banco de dados de testes que se assemelhe às características de um banco de dados de produção (ambiente de staging).
2. Manter um registro de quais aplicações usam quais objetos do banco de dados, para que você consiga saber quais mudanças irão afetar quais aplicações.
3. Trabalhar com os times de manutenção de outras aplicações, para aceitar e coordenar quais mudanças podem ser feitas aos objetos do banco de dados e quais aplicações ele impacta.
4. Fazer com que as aplicações funcionem com diversas versões do banco de dados; assim, o banco de dados pode ser migrado independentemente das aplicações que dependem dele.

Rollback sem perda de dados

Existem circunstâncias em que não é possível apenas rodar os scripts de rollback, por exemplo, quando o processo de rollback envolve reedição de dados de tabelas temporárias. Nesse caso, restrições de integridade podem ser violadas pelos novos registros que foram adicionados desde a atualização.

Neste caso, seu script deverá fazer o seguinte:

- Copiar os dados e a chave primária da tabela principal na tabela temporária.
- Em seguida, recriar a estrutura da tabela principal com o esquema original.
- Por fim, copie os dados da tabela temporária para a tabela principal e restabeleça quaisquer restrições referenciais.

Abordagem holística

É importante ter uma abordagem holística em relação ao gerenciamento de toda a infraestrutura, se baseando nos seguintes princípios:

- O estado desejado da infraestrutura deve ser especificado por meio de configuração no controle de versão.
- A infraestrutura deve ser autônoma, isto é, deve corrigir a si mesma até chegar ao estado desejado de forma automática.
- Você deve sempre ser capaz de saber o estado atual da infraestrutura por meio de instrumentação e monitoramento.

Melhores práticas para criar e gerenciar a infraestrutura necessária para os deploys

Há quatro áreas a se considerar na estratégia de monitoramento:

1. Instrumentar aplicativos e infraestrutura para coleta de dados.
2. Armazenar os dados para que possam ser facilmente recuperados para análise.
3. Criar dashboards que agregam as informações monitoradas e as apresentam em um formato adequado para operações e para o negócio.
4. Configurar notificações para que as pessoas possam saber dos eventos que as interessam.

Gerência de serviços de infraestrutura

Sabemos que problemas ocorrerão em produção, portanto é importante implementarmos alguns artefatos para nos ajudarem a lidar quando problemas ocorrerem:

1. Implementar controle de versão para todas as configurações de infraestrutura.
2. Instalar uma ferramenta de monitoramento de rede, garantindo que saiba quando problemas de conectividade ocorrerem.
3. Ativar log nas aplicações para registrarem WARN, INFO e DEBUG de eventos inesperados.
4. Tornar o ambiente de testes de integração o mais semelhante possível ao de produção.
5. Garanta que os *smoke tests* verifiquem todas as conexões durante a implantação para identificar problemas de roteamento ou conectividade.
6. Ter ferramentas forenses para analisar despejos de memória quando a aplicação falha ou tem grandes problemas de degradação.

O que é um componente?

Um componente é uma estrutura de código em uma aplicação, com uma API bem definida, que poderia ser trocada por outra implementação.

A principal característica de um sistema baseado em componentes é que a base de código é dividida em partes pequenas que fornecem comportamento através de interações limitadas bem definidas com outros componentes.

Há vários motivos pelos quais os componentes tornam o processo de desenvolvimento de software mais eficiente:

1. Dividir o desenvolvimento em blocos menores e mais valiosos
2. Projetar e realizar a manutenção de software com uma clara delimitação das responsabilidades
3. Limitar o impacto das mudanças
4. Facilitar o entendimento e a alteração da base do código.
5. Oferecer liberdade de otimização do processo de compilação e de implantação

Razões para separar um componente de sua base de código:

1. Parte da base de código precisa ser implantada independentemente (por exemplo, um servidor ou um cliente).
2. Você quer converter uma base de código monolítica em um núcleo e um conjunto de funções, talvez para substituir parte de um sistema com implementações alternativas ou para criar mecanismos de extensibilidade para os usuários.
3. O componente é uma interface para outro serviço (por exemplo, um framework ou um serviço com uma API)
4. A compilação e montagem tomam muito tempo.
5. Demora muito para que o projeto abra no ambiente de desenvolvimento.
6. Sua base de código é muito grande para uma só equipe.

Bibliotecas se referem a pacotes de software que sua equipe não controla, exceto na escolha de uso. Bibliotecas raramente são atualizadas.

Componentes são elementos de código dos quais sua aplicação depende, mas que foram desenvolvidos pela própria equipe ou por outras equipes na empresa. Componentes são, na maioria das vezes, atualizados com frequência.

Dependências em tempo de compilação precisam estar presentes quando o código é compilado ou passa pelo processo de vinculação (se necessário).

Dependências de tempo de execução devem existir quando a aplicação está rodando, executando suas funções normais.

Estratégia para uso de componentes:

1. Tenha aplicações bem projetadas e que permitam build por componentes.
2. Definir um componente
3. Organize as equipes por área funcional / grupo de histórias em vez de componentes
4. Crie componentes em pipelines e, em seguida, move para um pipeline de integração
5. Considere as relações de dependências
6. Use um repositório de artefatos

Há duas formas razoáveis de gerenciar bibliotecas:

1. Armazená-las no sistema de controle de versão.
2. Declará-las e usar uma ferramenta como Maven ou Ivy para baixá-las de repositórios na Internet ou (preferencialmente) do repositório de artefatos de sua organização.

Armazená-las no sistema de controle de versão é a solução mais simples e funciona bem para projetos pequenos.

Utilize também uma convenção para os nomes das bibliotecas que inclua seu número de versão.

Outra prática importante ao gerenciar dependências de ferramentas é gerenciar seu próprio repositório de artefatos. Dois exemplos de repositórios são Artifactory e Nexus.

Isso ajuda a garantir que compilações possam ser repetidas exatamente e evita um inferno de dependências controlando quais versões de cada biblioteca estão disponíveis para os projetos dentro da organização. Essa prática também torna mais fácil fazer a auditoria de bibliotecas e evita violações de restrições legais, como usar bibliotecas sob licenças GPL ou BSD.

Componentes no pipeline

Procure ter apenas um pipeline para toda a aplicação, englobando todos os componentes. Desta forma, toda vez que uma mudança é feita, tudo é compilado e testado.

Entretanto, em algumas circunstâncias há benefícios em separar o sistema em vários pipelines, como por exemplo:

- Partes da aplicação têm um ciclo de vida diferente.
- Equipes diferentes trabalham em funcionalidades diferentes.
- Componentes usam tecnologias ou processos de compilação diferentes.
- Quando existem componentes compartilhados com outros projetos.
- Quando temos componentes estáveis que já não mudam com tanta frequência.
- A aplicação demora a ser compilada e a criação de processos separados para cada componente torna o processo todo mais rápido.

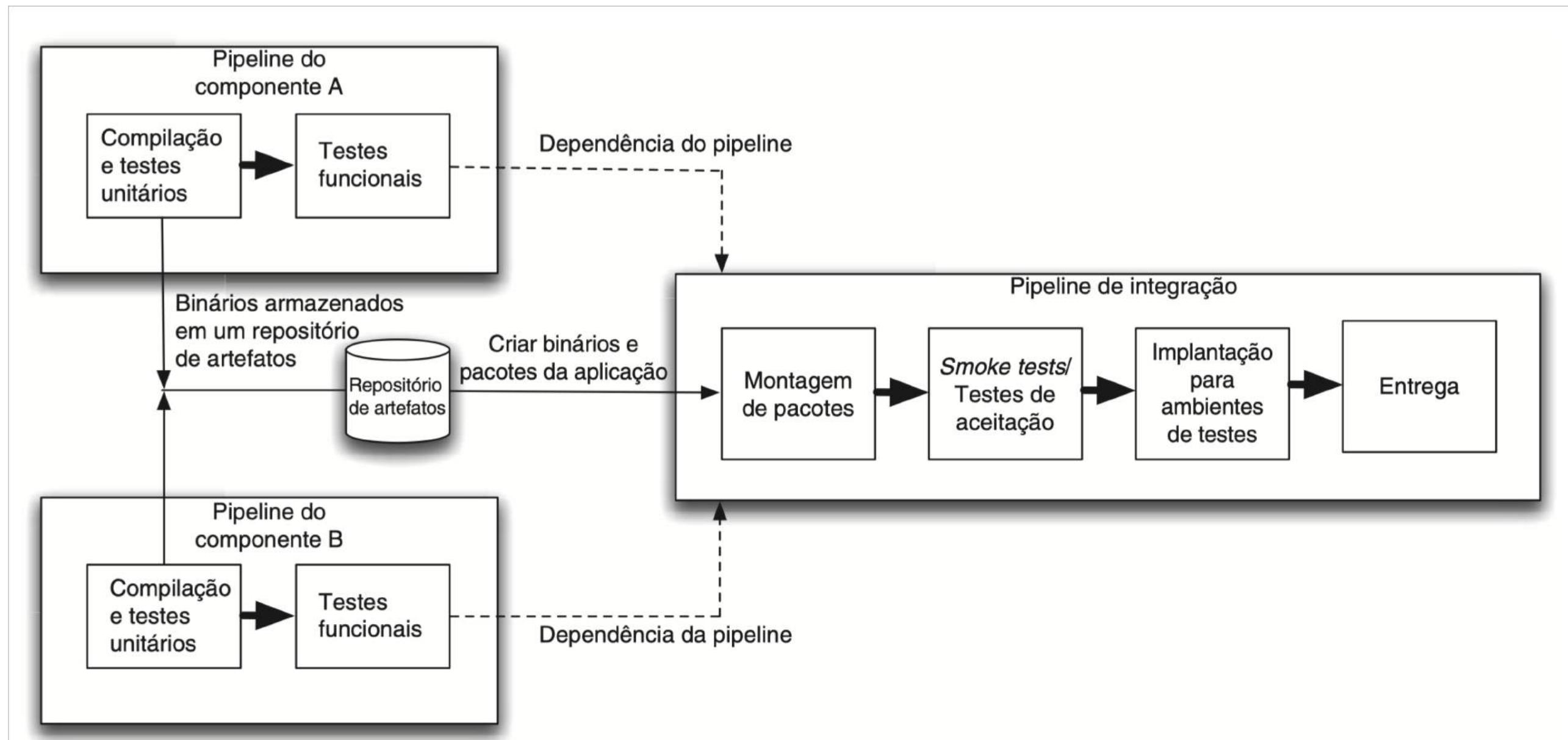
Componentes no pipeline

De maneira geral, o princípio pelo qual você deve se guiar deve ser o de minimizar o número de pipelines que opera.

Um pipeline é melhor do que dois, dois são melhores do que três, e assim por diante.

Preocupe-se em otimizar o pipeline e torná-lo o mais eficiente possível antes de criar outro.

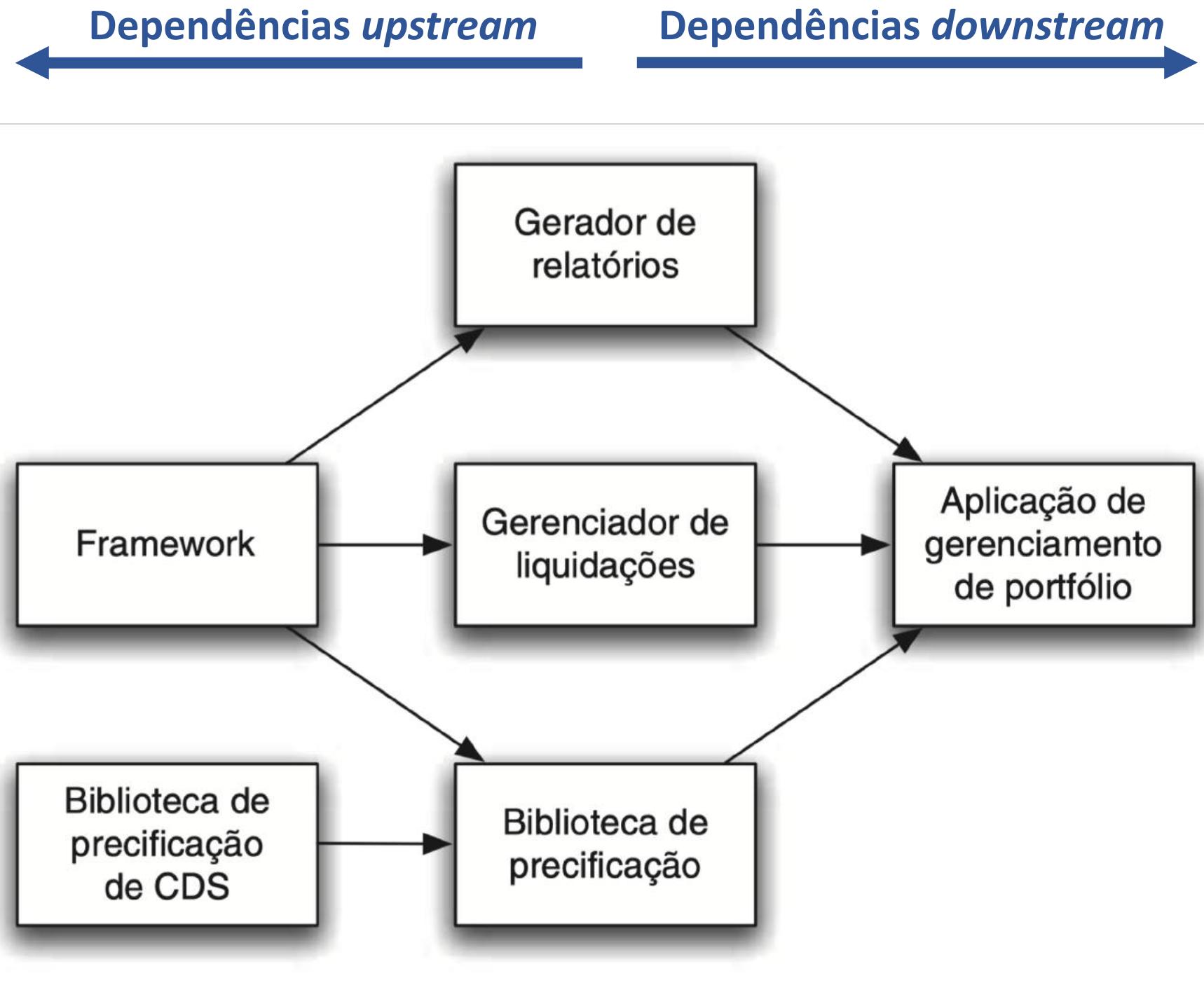
Exemplo de pipeline de integração de componentes



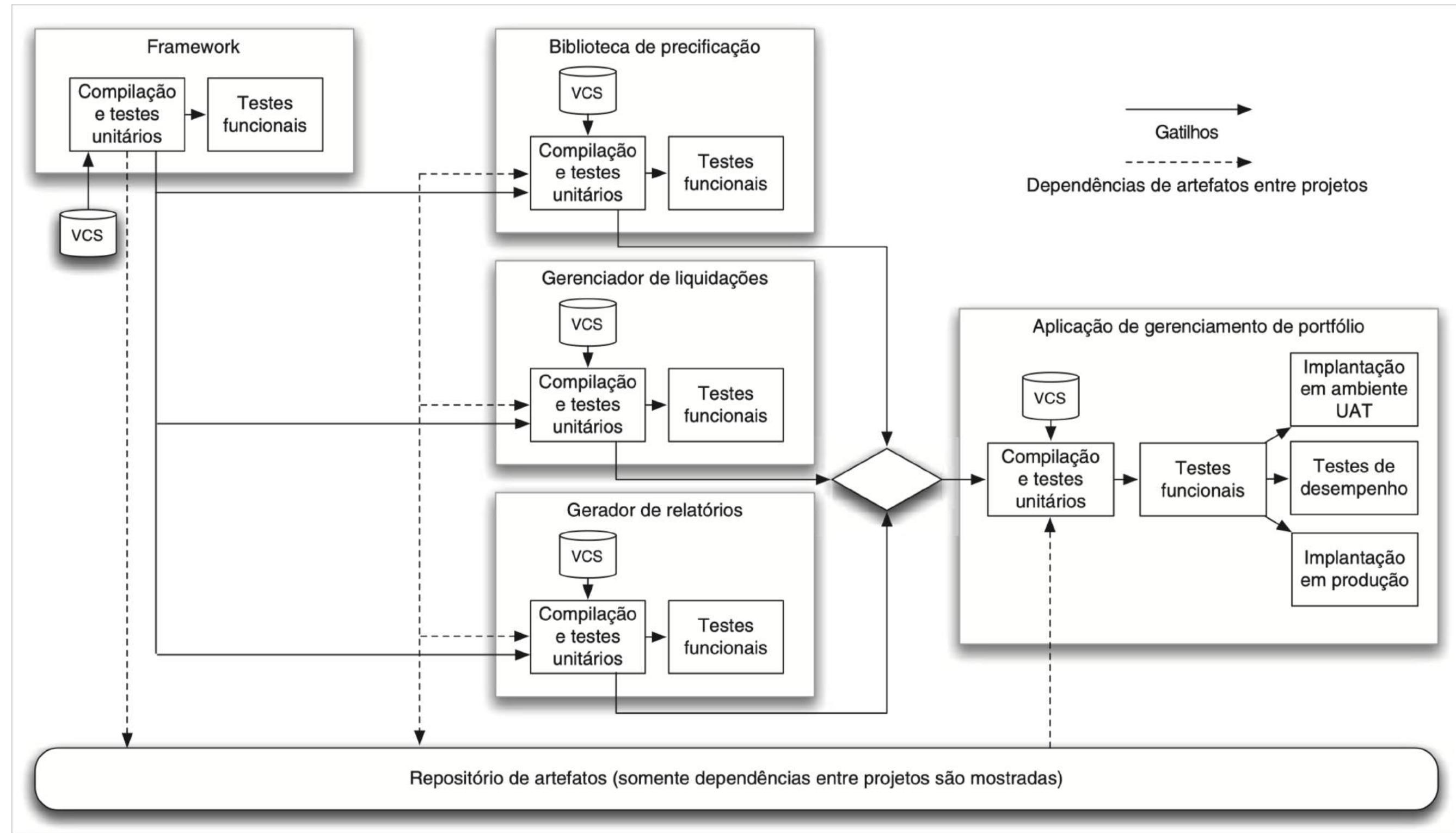
Dependências em tempo de compilação precisam estar presentes quando o código é compilado ou passa pelo processo de vinculação (se necessário).

Dependências de tempo de execução devem existir quando a aplicação está rodando, executando suas funções normais.

Grafos de dependências



Pipeline de componentes

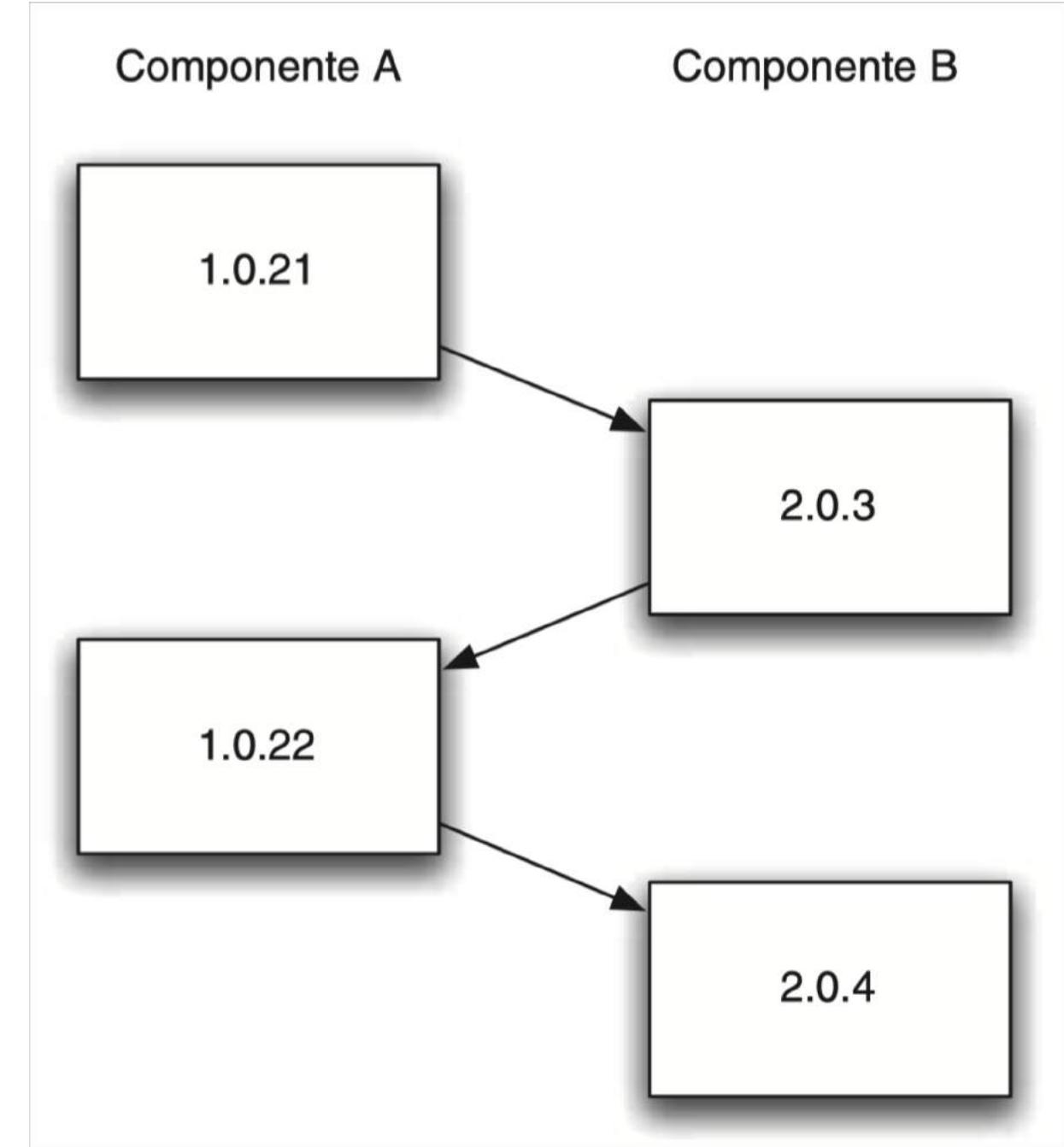


O problema mais desagradável de dependências provavelmente é o de dependências circulares. Isso ocorre quando o grafo contém ciclos.

Exemplo:

O componente **A** depende do componente **B**. Porém o componente **B** também dependente do componente **A**.

Ou seja: para compilar **A**, precisamos compilar **B**, mas para compilar **B**, precisamos de **A**.

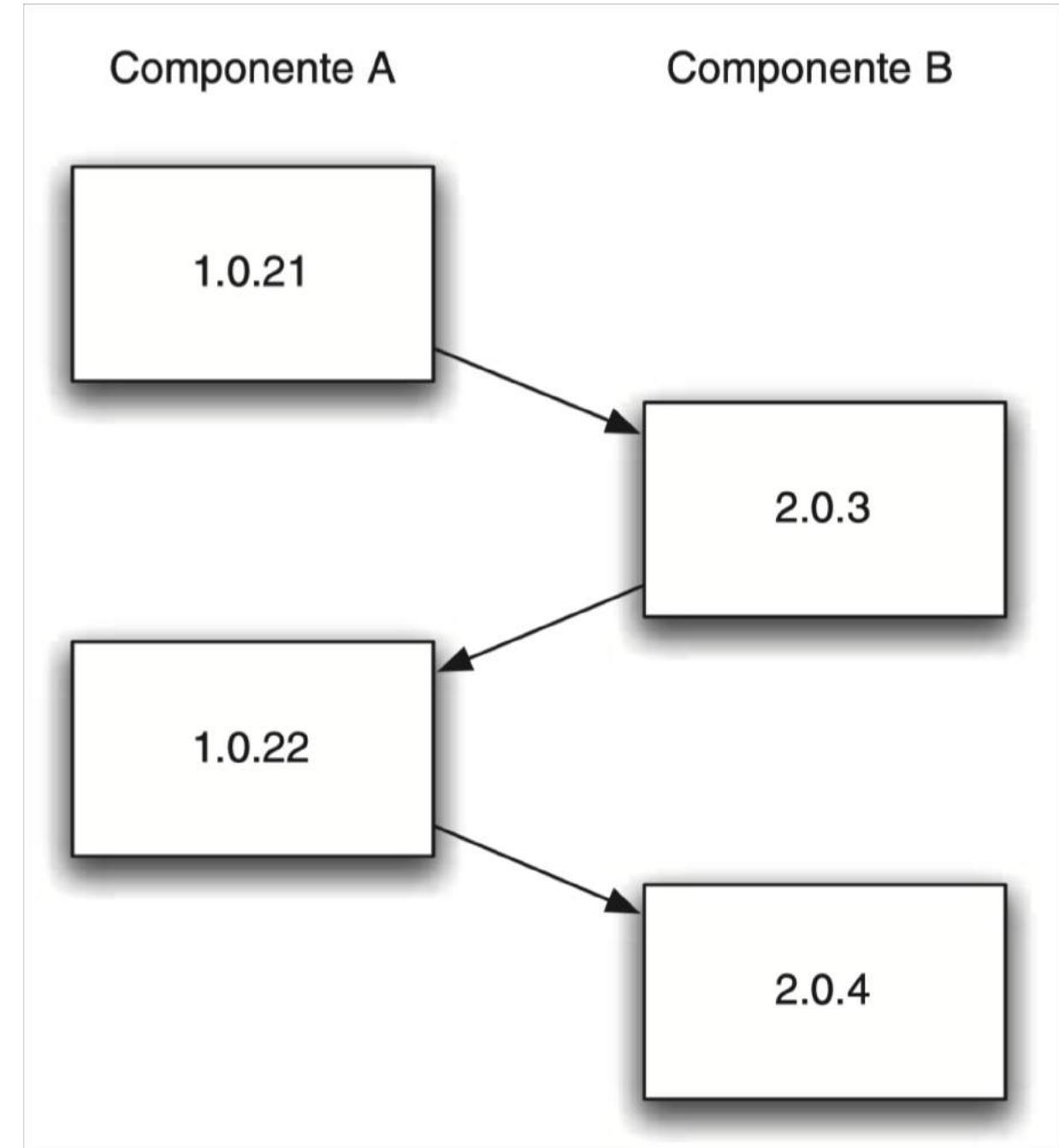


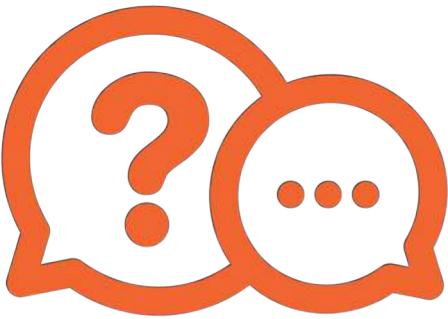
Como lidar com dependências circulares?

Evite isso ao máximo e, quando ocorrer, trate como um débito técnico que deve ser corrigido.

Enquanto não é corrigido, tenha sempre uma versão do componente A que possa ser usada para compilar o componente B, que pode ser usado para compilar o novo componente A e, por fim, compilar o novo componente B.

Este processo se chama *Escada de compilação de dependências circulares*.





O que os testadores devem evitar fazer como ponto de partida para seu banco de dados de teste?

- A) Criar uma cópia do conjunto de dados de produção atual por meio de um despejo (dump) para fins de teste.
- B) Criar um conjunto de dados reduzido que use o mínimo de dados de teste para testar o comportamento esperado.
- C) Criar e gerenciar conjuntos de dados menores para fins de testes específicos.
- D) Criar conjuntos de dados personalizados, selecionando cuidadosamente um subconjunto menor de produção de dados.



A empresa MyApp está interessada em integrar todos os aplicativos por meio de um único banco de dados, pois seria caro demais a MyApp alterar sua arquitetura atual do aplicativo. O desafio que eles enfrentam é que as mudanças feitas nos objetos de um banco de dados específico, do qual diversos aplicativos dependem, atrapalham a funcionalidade de alguns aplicativos, embora funcione para aplicativos que precisam das mudanças atualizadas em objetos do banco de dados.

1. Testar as mudanças em um ambiente orquestrado, no qual eles montam um banco de dados de testes que se assemelhe às características de um banco de dados de produção.
2. Manter um registro de quais aplicativos usam quais objetos do banco de dados.
3. Ativar aplicativos para funcionarem com diversas versões do banco de dados central, para que o banco de dados possa migrar independentemente dos aplicativos dos quais depende.
4. Trabalhar com os times de manutenção de outros aplicativos, para aceitar e coordenar quais mudanças podem ser feitas aos objetos do banco de dados e quais aplicativos ele impacta.

Quais ações a MyApp deve adotar?

- A) 1, 2 e 3. Coordenação com outros times levará tempo demais.
- B) 1, 2 e 4. Várias versões do banco de dados central põe em perigo a integridade dos dados.
- C) 1, 2, 3 e 4. Todas as ações descritas beneficiariam a MyApp, dentro do orçamento.
- D) 2, 3 e 4. Os testes em um ambiente orquestrado não são os mesmos da Produção.



Você deseja automatizar a implantação de um novo banco de dados como parte da automação de testes de aceitação.

Quais passos devem ser incluídos no script de automação antes de alimentar dados no banco de dados?

- A) Exclua registros de banco de dados. Crie novas tabelas de banco de dados.
- B) Exclua o banco de dados anteriormente existente. Crie a estrutura de banco de dados, instâncias de bancos de dados e esquemas.
- C) Exclua o banco de dados anteriormente existente. Crie novas tabelas de banco de dados.



A empresa MyApp está interessada em criar scripts de reversão que alteram as estruturas existentes em seu banco de dados de desenvolvimento de aplicativo, mas, ao mesmo tempo, não quer perder os dados na tabela de banco de dados atual. A MyApp usa um banco de dados temporário como um meio de manter os dados seguros e não permitir alterações neles.

O time de Desenvolvimento da MyApp quer garantir a execução bem-sucedida dos scripts de reversão. Ao mesmo tempo, a fim de manter a integridade referencial, os principais identificadores exclusivos originais devem ser mantidos.

Qual é a melhor maneira para o time de Desenvolvimento implementar isto?



A)

- Popular os dados da tabela principal na tabela temporária.
- Em seguida, recriar a estrutura da tabela principal com o esquema original.
- Por fim, copie os dados da tabela temporária de volta para o esquema da tabela recém criada.

B)

- Copiar os dados e a chave primária da tabela principal na tabela temporária.
- Em seguida, recriar a estrutura da tabela principal com o esquema original.
- Por fim, copie os dados da tabela temporária para a tabela principal e restabeleça quaisquer restrições referenciais.

C)

- Manter todas as restrições referenciais da tabela principal.
- Em seguida, preencha os dados da tabela principal na tabela temporária. Recrie a nova estrutura da tabela com o esquema atualizado.
- Por fim, copie os dados da tabela temporária de volta para o esquema da tabela recém criada.

D)

- Remover todas as restrições referenciais da tabela principal.
- Em seguida, preencha os dados da tabela principal na tabela temporária. Recrie a nova estrutura da tabela com o esquema atualizado.
- Por fim, copie os dados da tabela temporária para a tabela principal e restabeleça quaisquer restrições referenciais.



A empresa MyApp está tendo problemas com a identificação e solução de problemas de aplicativos implantados em sua infraestrutura.

Eles estão em constante modo de combate a incêndios e há cada vez mais incidentes graves e falhas de energia reportados pela Central de Serviços. O impacto das falhas de energia os coloca em risco de perder alguns clientes-chave. Eles não conseguiram utilizar seu aplicativo de construção comercial crítico para fornecer cotações e estimativas em tempo hábil, conforme prometido nos ANS's (SLAs) com os quais eles se comprometeram.

Uma ampla análise foi realizada em todos os principais serviços de infraestrutura, tais como roteadores, DNS, serviços de diretório e foram todos devidamente configurados e otimizados de maneira eficaz para hospedar aplicativos da MyApp.

A MyApp apresenta quatro ações possíveis para solucionar problemas de aplicativos, para que possam identificar causas e eliminar as falhas do aplicativo de maneira eficaz:



1. Estabelecer o controle de versão para cada parte da configuração de infraestrutura de redes da qual os aplicativos dependem para funcionar: Arquivos da zona DHCP para DHCP para configurações do firewall e do roteador para SMTP e outros serviços.
2. Ativar o registro (loggin) de aplicativos para que os aplicativos registrem no nível de ADVERTÊNCIA, quando um erro ocorre, uma possível conexão de rede expira ou feche inesperadamente. Além disso, para que registrem no nível de INFORMAÇÕES e de DEPURAÇÃO sempre que o aplicativo abrir ou fechar uma conexão.
3. Tornar o ambiente de testes de integração o mais semelhante possível ao de produção, inclusive utilizando as mesmas peças de acessórios com as mesmas conexões físicas entre eles.
4. Ter e utilizar as ferramentas forenses e analisar despejos de memória do aplicativo quando um aplicativo falha ou tem grandes problemas de degradação para ajudar a fazer uma depuração detalhada do aplicativo.

Quais ações eles devem adotar?

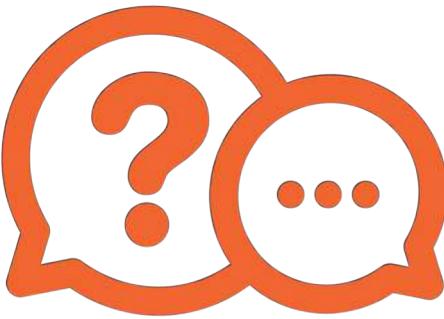
- | | |
|----------------|-------------|
| A) 1, 2 e 4 | C) 2 e 4 |
| B) 1, 2, 3 & 4 | D) 2, 3 e 4 |



Para evitar confusão por parte do time de Operações, deveria haver regras claramente definidas para gerenciamento da infraestrutura.

Qual não seria o princípio de gerenciamento da infraestrutura?

- A) Automatizar as ferramentas de configurações necessárias.
- B) Monitorar o estado da infraestrutura por meio de dashboards.
- C) Especificar regras de controle de versão na configuração.
- D) Escrever um novo script universal de fundamentos de gestão.



A empresa MyApp está enfrentando grande dificuldade com a identificação de problemas e com a solução de problemas de aplicativos implantados em sua infraestrutura. Eles estão sempre apagando incêndios e tem havido um aumento no número de interrupções e incidentes graves relatados à Central de Serviços.

Eles estão interessados em criar uma estratégia de monitoramento robusta que os ajude a monitorar mais eficazmente seu ambiente de produção para que possam resolver mais eficazmente grandes incidentes e também identificar os problemas mais cedo para evitar incidentes graves.

Ao criar uma estratégia de monitoramento, a primeira área principal a ser considerada deve ser:
Instrumentar seus aplicativos e sua infraestrutura para conseguir coletar os dados de que a MyApp precisa.

Quais as três ações adicionais que terão melhor resultado ao criar uma estratégia de monitoramento?



A)

1. Implementação de um novo sistema de instrumentação.
2. Criação de dashboards que reúnem os dados e os apresentam em um formato adequado para operações e para o negócio.
3. Calcular as correlações do evento manualmente depois que um número predefinido de eventos acontecer.

B)

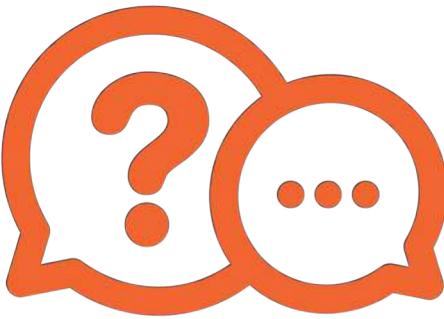
1. Implementação de um novo sistema de instrumentação.
2. Criação de dashboards que reúnem os dados e os apresentam em um formato adequado para operações e para o negócio.
3. Configurar o envio de notificações à Central de Serviços quando algum evento acontecer.

C)

1. Armazenar os dados para que possam ser facilmente recuperados para análise.
2. Criação de dashboards que reúnem os dados e os apresentam em um formato adequado para operações e para o negócio.
3. Usar um software pago de uma empresa mundialmente conhecida que tenha um mecanismo de correlação de eventos.

D)

1. Armazenar os dados para que possam ser facilmente recuperados para análise.
2. Criação de dashboards que reúnem os dados e os apresentam em um formato adequado para operações e para o negócio.
3. Configurar o envio de notificações para que as pessoas sejam avisadas sobre os eventos relevantes.



Você deseja adotar uma abordagem abrangente para gerenciar toda a sua infraestrutura de TI.

Quais dois princípios o ajudarão a fazer isso?

A)

1. O estado desejado da infraestrutura deve ser especificado por meio da configuração controlada por Mudanças.
2. Conhecer sempre o real estado de sua infraestrutura por meio do monitoramento e do gerenciamento de eventos.

B)

1. O estado desejado da infraestrutura deve ser especificado por meio da configuração controlada por Mudanças.
2. Conhecer sempre o real estado de sua infraestrutura por meio da instrumentação e do gerenciamento de incidentes.

C)

1. O estado desejado da infraestrutura deve ser especificado por meio da configuração controlada por versões.
2. Conhecer sempre o real estado de sua infraestrutura por meio do gerenciamento de incidentes e eventos atuais.

D)

1. O estado desejado da infraestrutura deve ser especificado por meio da configuração controlada por versões.
2. Conhecer sempre o real estado de sua infraestrutura por meio da instrumentação e do monitoramento.



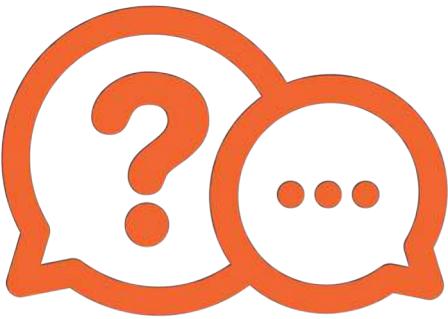
A empresa MyApp está interessada em identificar qual é o melhor ponto de partida estratégico para tornar seu aplicativo mais intercambiável e reutilizável.

A MyApp deseja alterar a estratégia para:

- Dividir o desenvolvimento em blocos menores e mais valiosos
- Projetar e realizar a manutenção de software com uma clara delimitação das responsabilidades
- Limitar o impacto das mudanças
- Facilitar o entendimento e a alteração da base do código.
- Oferecer liberdade de otimização do processo de compilação e de implantação

Qual estratégia melhor apoiará essas metas?

- A) Esconder a nova funcionalidade do cliente até que ela esteja completamente terminada.
- B) Fazer as alterações gradativamente como uma série de pequenas mudanças individuais de liberação.
- C) Usar a filial por abstração para realizar alterações em larga escala no código-base.
- D) Usar componentes para dissociar partes de aplicativos que mudam a diferentes taxas.



Você é um Process Master. O time de Desenvolvimento tem múltiplos Pipelines de Implantação que estão funcionando bem, dos quais eles têm muito orgulho. Você tem dúvidas sobre os múltiplos Pipelines de Implantação.

Qual ação você deve adotar?

- A) Escutar o motivo para o time de Desenvolvimento ter múltiplos pipelines. Pode haver múltiplos pipelines, quando houver uma boa razão para isso.
- B) Analisar as normas operacionais de cada pipeline individual com o time de Desenvolvimento. Pipelines que têm normas aceitáveis podem ser mantidos.
- C) Veja o que acontece. Quando o time de Desenvolvimento desenvolve produtos satisfatórios, não há necessidade de aumentar a pressão sobre eles e reduzir a velocidade.
- D) Trabalhar com o time para entender os múltiplos problemas em potencial que os pipelines podem gerar e concentrar-se na necessidade de um único pipeline de implantação.



Considere estas duas afirmações sobre as dependências do tempo de compilação:

1. As dependências do tempo de compilação devem estar presentes quando o aplicativo for executado, realizando sua função habitual.
2. As dependências do tempo de compilação devem estar presentes quando o aplicativo é compilado e vinculado (linked).

E estas duas afirmações para as dependências do tempo de execução:

- A. As dependências do tempo de execução devem estar presentes quando o aplicativo for executado, realizando sua função habitual.
- B. As dependências do tempo de execução devem estar presentes quando o aplicativo é compilado e vinculado (linked).

Quais são as duas opções corretas?

- A) 1 e A
- B) 1 e B
- C) 2 e A
- D) 2 e B



A empresa MyApp já passou por alguns problemas de inicialização fatais com seus aplicativos. Ao analisar a questão mais a fundo, determinou-se que os componentes do aplicativo tinham dependências incomuns, o que quer dizer que para criar o componente A, o desenvolvedor precisa compilar o componente B, mas, para tal, o desenvolvedor precisava do componente A.

A fim de resolver essa situação, o desenvolvedor pode criar uma versão do componente A que pode ser usada para criar o componente B. O desenvolvedor pode usar uma nova versão do componente B para compilar a nova versão A.

Que nome damos a esse processo?

- A) Compilação de componente circular
- B) Escada de compilação de dependência circular
- C) Escada de compilação de dependência de componente
- D) Escada de compilação de componente modular



Qual é a solução mais simples para o gerenciamento de bibliotecas para projetos pequenos de software?

- A) Verificar as bibliotecas no Controle de versão e usar convenções de nomenclatura para as bibliotecas que incluem seu número de versão.
- B) Verificar as bibliotecas fora do Controle de versão e usar convenções de nomenclatura para as bibliotecas que incluem seu número de versão.
- C) Declarar as bibliotecas e usar uma ferramenta como Maven ou Ivy para baixar a partir de repositórios da Internet ou do repositório de artefatos que você tem na organização.
- D) Inicializar as bibliotecas e usar uma ferramenta como Maven ou Ivy para baixar a partir de repositórios da Internet ou do repositório de artefatos que você tem na organização.

4.2 – Gerenciamento de Configuração e Controle de Versão

		% Peso	Questões	Referência de Literatura		
				Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
4.2 Gerenciamento de Configuração e Controle de Versão						
4.2.1	Explicar por que o controle de versão é uma chave para o DevOps eficaz					
4.2.2	Explicar como manter o controle de versão sobre dados, infraestrutura e componentes					
4.2.3	Analizar um cenário e sugerir a melhor estratégia para gerenciar um problema de gerenciamento de configuração	4%	2			2,14

Gerência de configuração se refere ao processo pelo qual todos os artefatos relevantes ao seu projeto, e as relações entre eles, são armazenados, recuperados, modificados e identificados de maneira única.

Embora sistemas de controle de versão sejam uma ferramenta óbvia em gerência de configuração, a decisão de usá-los é apenas o primeiro passo no desenvolvimento de uma estratégia de gerência de configuração.

Você tem uma boa estratégia para gerência de configuração se conseguir dar uma resposta afirmativa a todas as questões abaixo:

- Eu posso facilmente realizar mudanças incrementais nesses itens individuais e implantá-las em quaisquer dos meus ambientes ou em todos eles?
- Eu posso satisfazer todos os requisitos de conformidade a regulamentações aos quais estou sujeito?

A chave para gerenciar ambientes é tornar sua criação um processo completamente automatizado. Deveria ser sempre mais barato criar um novo ambiente do que corrigir um antigo. Ser capaz de reproduzir seus ambientes é essencial por vários motivos:

- Elimina o problema de ter pedaços aleatórios da infraestrutura cuja configuração é compreendida apenas por alguém que não está mais na organização e não pode ser contatado. Esse é um risco grande e desnecessário.
- Consertar um ambiente pode demorar horas. É sempre melhor ser capaz de reproduzir o ambiente em um tempo previsível.
- É essencial ser capaz de criar cópias do ambiente de produção para fins de teste. Em termos de configuração de software, ambientes de testes devem ser réplicas exatas dos ambientes de produção, de modo que problemas de configuração sejam descobertos o mais cedo possível.

Há dois princípios que formam a base de uma gerência de configuração eficiente:

- Mantenha os binários *independentes* da informação de configuração
- Mantenha toda a informação de configuração em um só lugar

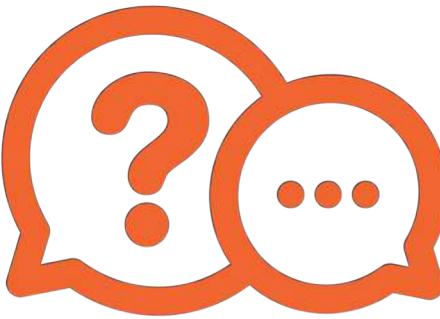
A aplicação desses princípios em todas as partes de seu sistema irá abrir o caminho até o ponto em que a criação de novos ambientes, a atualização de partes do sistema e aplicação de novas configurações sem indisponibilizar o sistema se torna um processo simples e automatizado.

Todo sistema de controle de versão permite que se acrescente uma descrição a um check-in. O motivo mais importante para usar mensagens descritivas em commits é que, se uma compilação falhar, você saberá quem causou a falha e por quê.

Como escrever uma boa mensagem de commit: Uma mensagem em vários parágrafos em que o primeiro parágrafo é um resumo e os demais acrescentam mais detalhes.

O primeiro parágrafo é o que aparece em listagens de commits – pense nele como o título de uma reportagem, dando ao leitor informação suficiente para que ele decida se está interessado em continuar lendo ou não.

Você também deve incluir um link para um identificador em sua ferramenta de gerenciamento de projeto para a funcionalidade ou o defeito em que está trabalhando.



O Controle de versão é a ferramenta mais óbvia do gerenciamento de configurações e é apenas o primeiro passo para apoiar a estratégia de gerenciamento de configuração de uma organização. Quais são os outros dois principais aspectos de uma boa estratégia de gerenciamento de configurações?

A)

1. Pode realizar uma alteração incremental em qualquer um dos itens individuais e implementar a alteração em qualquer e em todos os meus ambientes.
2. Pode satisfazer a requisitos específicos de auditoria interna.

B)

1. Pode realizar uma alteração incremental em qualquer um dos itens individuais e implementar a alteração em qualquer e em todos os meus ambientes.
2. Pode satisfazer a todos os regulamentos de conformidade sujeitos à minha organização.

C)

1. Pode realizar apenas alterações programadas semanais em qualquer um dos itens individuais e implementar a alteração em qualquer e em todos os meus ambientes.
2. Pode satisfazer a requisitos específicos de auditoria interna.

D)

1. Pode realizar apenas alterações programadas semanais em qualquer um dos itens individuais e implementar a alteração em qualquer e em todos os meus ambientes.
2. Pode satisfazer a todos os regulamentos de conformidade que a minha organização está sujeita.



A empresa MyApp tem administrado as configurações do aplicativo de uma forma muito eventual para seus aplicativos críticos. O time de Desenvolvimento instala partes do aplicativo manualmente e edita, também manualmente, os arquivos de configuração relevantes para fazê-los funcionar.

Frequentemente, depois de fazer as edições manuais nos arquivos de configuração, o aplicativo falha. O time de Desenvolvimento tem dificuldades de voltar para um bom estado conhecido com alguma confiança, pois não há nenhum registro da última configuração conhecida.

Isso gera atrasos significativos na obtenção dos aprimoramentos de aplicativos implementados. Pequenas alterações atrapalham o aplicativo, tornando difícil encontrar e resolver os problemas. Isso também complica a criação de ambientes para testes manuais.

A MyApp deseja ter um processo eficaz de gerenciamento de configuração para resolver seus atuais pontos problemáticos.

O que não é considerado uma estratégia eficaz para alcançar isso?



- A) Manter sempre juntos arquivos binários e informações de configuração.
- B) Automatizar totalmente o gerenciamento dos arquivos de configuração de aplicativos.
- C) Manter todas as informações de configuração em um local único e centralizado.
- D) Manter arquivos binários independentes das informações de configuração.



Como principal desenvolvedor para a empresa MyApp, você encontra algumas repetidas falhas no processamento de ordens de serviço para o desenvolvimento do aplicativo crítico.

Ao analisar a questão mais a fundo, você identifica um erro no código do aplicativo crítico que foi atualizado ontem como uma melhoria adicional para o processamento de ordens de serviço. Você pode usar seu sistema de controle de versão para obter mais informações sobre essa linha de código específica. Aparece uma mensagem de confirmação informando “melhoria adicionada”.

Você tenta fazer uma correção no código, mas, como consequência, alguma coisa dá errado. Então, você demora quase 8 horas para fazer o aplicativo voltar a funcionar.

Qual é o estilo recomendado desta mensagem de confirmação?



- A) Escrever uma mensagem de confirmação com vários parágrafos fornecendo uma seção de resumo e todos os detalhes de suporte sobre o que exatamente foi atualizado.
- B) Escrever uma mensagem de confirmação com vários parágrafos fornecendo detalhes sobre quando, por que, como e por quem o código foi atualizado e o que exatamente foi atualizado.
- C) Escrever uma curta mensagem de confirmação fornecendo detalhes sobre quando, por que, como e por quem o código foi atualizado e o que exatamente foi atualizado.
- D) Escrever uma mensagem curta de confirmação informando o que foi atualizado no código e quando ele foi atualizado.

4.3 – Infraestrutura em Nuvens e Imutável

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
4.3 Infraestrutura em Nuvens e Imutável					
4.3.1 Expliar quando é e quando não é necessário mover para a infraestrutura baseada em nuvem para ter um DevOps eficaz	2%	1	4,5,14,16		11
4.3.2 Expliar como a infraestrutura baseada em nuvem deve ser gerenciada dentro do DevOps					

Razões para usar a nuvem:

- Diminui esforços com aquisição, instalação e manutenção de equipamentos.
- Flexibilidade para expandir e contratar novos recursos
- Rápida adoção
- Possibilidade de possuir soluções padronizadas
- Se adapta com muitos dos princípios do DevOps
- Padronização da stack
- Possibilidade de ter uma **biblioteca** de templates de máquinas virtuais

A maior utilidade da virtualização para pipelines de entrega está na facilidade de manter e provisionar novos ambientes.

Razões para não usar a nuvem:

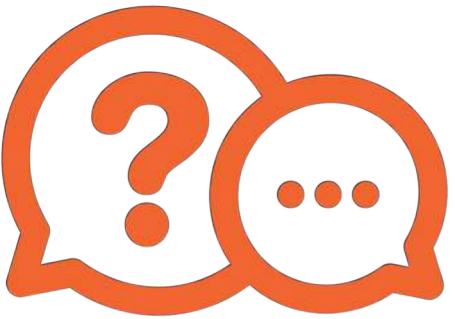
- Migrar para a nuvem achando que apenas isso é suficiente para adotar DevOps
- Se você já fez muitos investimentos em infraestrutura local
- Preocupações com segurança e níveis de serviço de terceiros
- Preocupações com *vendor and platform lock-in*
- Questões relacionadas à desempenho, onde servidores locais possuem melhor desempenho que servidores virtuais.



Seu time de DevOps está trabalhando no aplicativo MyAppGo. A exigência mais importante de seu aplicativo é que ele deve ser portátil. Você argumenta que o MyAppGo não deve ser hospedado na nuvem.

Com base neste caso, em qual característica da nuvem você se baseia para afirmar isso?

- A) Custos e economias do modelo da nuvem
- B) Provisionamento e manutenção dinâmicos
- C) Segurança e níveis de serviço
- D) Fidelidade do fornecedor e portabilidade do aplicativo (Lock-In)



Qual recurso de virtualização melhor ajuda a garantir que a configuração do ambiente será a mesma durante as fases do Pipeline de Implantação?

- A) Instantâneo da imagem do disco virtual
- B) Instantâneo da imagem da máquina virtual
- C) Biblioteca de templates da máquina virtual



Você trabalha em um time de DevOps que está em processo de efetivar a implantação automatizada.

Qual característica da plataforma na nuvem melhor permite a implantação automatizada?

- A) Disponibilidade e capacidade dinâmica
- B) Flexibilidade de provisionamento
- C) Dimensionamento e disponibilidade
- D) Stack padronizado

4.4 – Continuidade do Negócio

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
4.4 Continuidade do Negócio	2%	1			
4.4.1 Explicar como o DevOps pode facilitar práticas de continuidade de negócios					2,4

O DevOps não é apenas um aprimoramento do desenvolvimento ágil e da entrega contínua, mas também um gerenciamento de aplicações de TI para permitir o crescimento e manter a continuidade dos negócios.

Práticas DevOps que contribuem para a continuidade do negócio:

- A infraestrutura deve ser autônoma, isto é, deve corrigir a si mesma até chegar ao estado desejado de forma automática.
- Em caso de falhas catastróficas, a infraestrutura deve ser simples de recriar através de um processo automatizado.
- Você deve sempre ser capaz de saber o estado atual da infraestrutura por meio de instrumentação e monitoramento.
- Tudo o que é necessário para recriar o ambiente deve estar no controle de versionamento.
- O estreito relacionamento entre a equipe de desenvolvimento da aplicação e a equipe de gerenciamento de infraestrutura.

Empresas possuem tanto Sistema de Engajamento (SoE) quanto Sistema de Registro (SoR).

O SoE é focado em velocidade. O SoR está focado na continuidade dos negócios.

O problema é como o SoR pode se adaptar rapidamente a mudanças no SoE para manter a continuidade dos negócios.

O SoR na maioria das empresas está lutando com o uso de aplicações ou sistemas legados e pode ser alterado com o DevOps construindo processos alinhados com os conceitos just-in-time (JIT).

Existem 3 tipos de implementação de Enterprise DevOps que dependem do modelo de negócios da empresa.

Maneira de TOYOTA: (complexo e avançado)

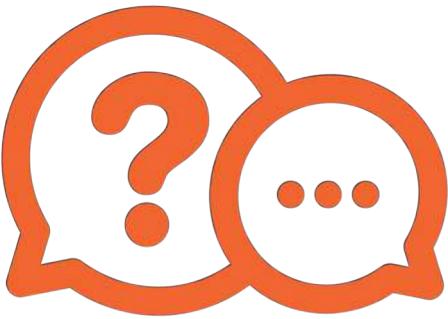
Se concentra nos serviços estratégicos de TI e oferece vantagem estratégica para os negócios.
Essa estrutura é mais adequada para provedores de serviços de TI.

Colaboração: (Standard)

Se concentra em apenas fornecer serviços de TI rápidos e frequentes e operação confiável.
É mais adequado para SoE e SoR.

Entrega Contínua: (Básico)

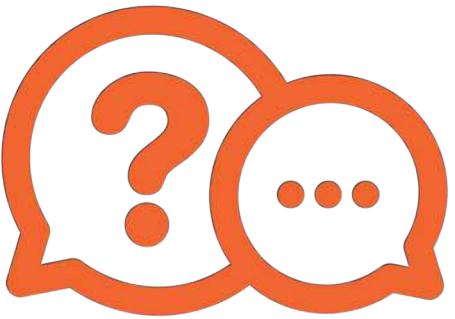
Se concentra em lançamentos rápidos e frequentes de software.
É mais adequado para fornecedores de produtos digitais.



Você trabalha para um prestador de serviços de TI. Como parte de seu plano de Continuidade de Negócios, você deseja garantir que sempre possa cumprir os níveis de serviço mínimos acordados.

Você deseja garantir a continuidade dos serviços de TI. Como o DevOps pode ajudá-lo com o Gerenciamento da Continuidade dos Serviços de TI?

- A) Os valores culturais de Afinidade e Colaboração do DevOps garantem que o Serviço seja altamente valorizado pelos membros do time de DevOps.
- B) O DevOps prepara as rotinas de desastre do time e as práticas da metodologia Obeya, introduzindo deliberadamente o caos no sistema.
- C) As medidas de redução de riscos e as opções de recuperação são provavelmente codificadas, pois Operações está trabalhando em conjunto com Desenvolvimento.
- D) O gerenciamento de nível de serviço torna-se mais importante no DevOps, pois é tarefa do Mestre do Processo monitorar isso.



Qual implementação de DevOps é mais adequada para uma empresa que usa a abordagem do Sistema de Registro (SoR)?

- A) Colaboração
- B) Entrega Contínua
- C) A Maneira Toyota

4.5 – Dimensionamento

Referência de Literatura	% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
4.5 Dimensionamento					
4.5.1 Analisar um cenário, explicar se e por que é importante dimensionar para cima ou para baixo nessa situação, e identificar a melhor maneira de fazer isso	4%	2	14,15,16,17		
4.5.2 Analisar um cenário que deu errada a dimensionamento, e identificar uma boa maneira de resolver o problema					
4.5.3 Explicar como a política social e práticas de contratação suportam escalonamento DevOps					

O dimensionamento é sobre a evolução, o crescimento e o avanço da organização como um todo durante todo o seu ciclo de vida.

As organizações de sucesso devem saber como escalar - isto é, crescer ou encolher conforme necessário. Escala pode significar coisas diferentes para pessoas diferentes, dependendo do contexto.

Por exemplo, o dimensionamento pode significar:

- Expandir a base de clientes
- Aumentar a receita
- Expandir um projeto ou equipe para atender à demanda
- Manter ou melhorar uma proporção de pessoas para sistemas ou dinheiro gasto
- Crescer mais rápido que os concorrentes
- Expandir para novas localidades

Estrutura organizacional

Times pequenos (regra das duas pizzas) e multifuncionais podem ser altamente produtivos quando trabalham em um mesmo produto.

Em contrapartida, times organizados por função também possuem seus benefícios, como redução de custos e compartilhamento de conhecimento.

Se sua empresa possui times funcionais mas não tem problemas de comunicação entre eles, talvez não seja necessário reestruturar os times para que sejam multifuncionais. A orientação é: Não reestruture os times apenas por mudar, essa reestruturação deve resolver um problema.

Avaliar o impacto dos ciclos de release

Nem todo software desenvolvido é projetado para estar imediatamente disponível 24/7/365 ou para atualizar constantemente o conteúdo. Entenda e avalie a importância e a complexidade dos projetos e seus lançamentos para descobrir qual ciclo de lançamento faz mais sentido para cada um.

Projetos diferentes podem funcionar melhor com diferentes cadências de lançamento.

Exemplo:

- Aplicação Mobile
- Software embutido (embedded)
- Bankline
- Rede Social
- Aplicação web de investimentos

Escalar o time

Ao avaliar terceirizar pessoas ou todo um time, tenha em mente que embora você possa economizar dinheiro com a folha de pagamento, é provável que haja um aumento nos custos em termos de diminuição da colaboração e da afinidade entre indivíduos e equipes.

Terceirização pode ser uma grande fonte de conflito dentro de uma organização, direta ou indiretamente.

Escalar o time

Terceirização criando silos funcionais

Um dos maiores problemas com os silos é a falta de comunicação e colaboração entre eles, onde as pessoas tendem a acumular conhecimento e informações ao mesmo tempo em que empurram a responsabilidade para outros grupos.

Quando um grupo é terceirizado, eles podem se sentir como se ninguém estivesse disposto a compartilhar informações com eles ou mantê-los informados devido ao status de “outsider”, portanto, certifique-se de que existem vias claras de comunicação entre as duas empresas. equipes terceirizadas. O uso de mídia de comunicação compartilhada (como um bate-papo em grupo ou uma lista de e-mail que abranja ambas as equipes) ou o incentivo a atualizações regulares de status podem ajudar a garantir que as informações sejam compartilhadas.

Escalar o time

Equipes terceirizadas com status “inferior”

Na hierarquia social do local de trabalho, seja formal ou informal, as equipes terceirizadas geralmente sentem que têm um status mais baixo do que as equipes internas.

A nível organizacional, pode ajudar a tentar incluir equipes terceirizadas em celebrações de equipes e afins, porque as pessoas que se sentem apreciadas e gostam de fazer parte de um grupo são frequentemente muito mais motivadas no trabalho.

Nos níveis individual e de equipe, mantenha-se atento a qualquer pessoa que trate os funcionários terceirizados de maneira desrespeitosa e que não tolere tal comportamento.

Escalar o time

Conflitos de responsabilidade entre equipes internas e terceirizadas

Se uma equipe interna está tentando acumular responsabilidade por si mesma, talvez deixando apenas trabalho insignificante ou tedioso para equipes terceirizadas, ou tentando empurrar a responsabilidade (e muitas vezes culpa) para o grupo terceirizado, as responsabilidades entre as equipes também podem ser um ponto de discórdia aqui.

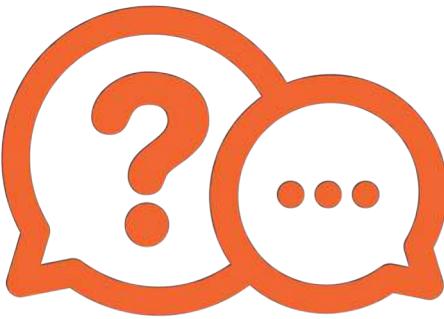
As responsabilidades definidas com regularidade e clareza podem ajudar a eliminar a confusão ou a tensão nessa área (junto com a comunicação regular) e a encontrar maneiras de as responsabilidades ou projetos serem compartilhados entre internos e terceirizados.

Manter as equipes pequenas e flexíveis

As organizações crescem, as equipes podem crescer sem um design inteligente. Grandes equipes são menos propensas a compartilhar conhecimento ou aprender livremente umas com as outras. A falta de familiaridade dentro da equipe de pontos fortes e fracos de cada membro leva a uma menor flexibilidade de carga de trabalho. As tarefas se alinham aos indivíduos, o que geralmente cria gargalos, especialmente quando os indivíduos se tornam dependências circulares em cenários complexos.

Dicas valiosas

- Equipes pequenas trabalham melhor (regra das duas pizzas)
- Promova de dentro e contrate de fora
- Ao contratar, avalie aspectos interpessoais e culturais, além de necessidades técnicas
- Invista em treinamento e suporte a pessoas em inicio de carreira
- Implemente fatores motivacionais, tais como benefícios, oportunidades e horário flexível, além de salários competitivos



Uma pequena startup está se consolidando bem. Nos últimos dois anos, eles viram sua clientela aumentar em 20% a cada trimestre. Eles gostariam de implementar o DevOps, mas não têm orçamento para contratar nenhum especialista adicional ou novas pessoas.

É possível adotar o DevOps sem contratar um novo grupo de programadores super-estrelas?

- A) Sim, desde que a organização crie funções e responsabilidades claras como suporte à cultura do DevOps. Fazer isso permitirá que os times possam aprender no trabalho.
- B) Sim, se criarem um entendimento compartilhado de metas, fornecerem e incentivarem oportunidades de aprendizagem para os times e criarem um ambiente sem culpa.
- C) Não. Seria um custo excessivo para romper a cultura organizacional. Seria melhor aguardar até que haja mais receita antes de mudar para a cultura do DevOps.
- D) Não. Você precisa de uma habilidade especializada definida para adotar o DevOps. O time atual não tem as habilidades necessárias para lidar com esse tipo de mudança organizacional.



Sua empresa está crescendo muito rápido e não consegue acompanhar as demandas de contratação. Portanto, mesmo que seja indesejado para o DevOps, parte do trabalho será terceirizado em um subcontrato. O time terceirizado e seu time de DevOps estão com problemas de compartilhamento de responsabilidades. Alguns membros de seu time culpam o time terceirizado por não ter feito o trabalho deles.

Qual é a maneira recomendada para resolver esse conflito?

- A) Defina, de modo claro, algumas tarefas e trabalhe para incutir a importância da responsabilidade compartilhada entre os dois times.
- B) Misture os membros de seu time e os membros do time terceirizado para que precisem colaborar uns com os outros.
- C) Lembre aos membros de seu time de que, em DevOps, você nunca culpa ninguém.
- D) Pare a terceirização e peça a seu time que faça horas extras até que a questão da contratação seja resolvida.



A empresa MyApp deseja otimizar sua estrutura organizacional atual para poder expandi-la e oferecer suporte para um maior compartilhamento de conhecimentos e especialização. Eles desejam certificar-se de que a estrutura ofereça um suporte eficaz a um time distribuído, ou seja, para que todos os times tenham acesso a sistemas críticos.

Eles precisam decidir se querem times multifuncionais ou times com uma única função, mas sabem que todos os membros devem ser capazes de se comunicar bem uns com os outros. Além disso, precisam decidir o tamanho ideal do time ideal para eles, pois as decisões são tomadas de forma hábil e eficaz e não são afetadas pelo raciocínio em grupo.

Você orienta a MyApp sobre o tipo de time e o tamanho do time. Qual é a melhor recomendação para esse cenário?

- A) Equipes multifuncionais de 10 a 12 membros.
- B) Equipes multifuncionais de 6 a 8 membros.
- C) Equipes com uma única função de 10 a 12 membros.
- D) Equipes com uma única função de 6 a 8 membros.



A empresa MyApp enfrenta dificuldades em manter os 10 membros em seus times de Desenvolvimento e Operações, em condições saudáveis. Os membros do time não entendem claramente os pontos fortes e fracos uns dos outros. Determinadas tarefas são especificamente atribuídas a certos indivíduos. Isso muitas vezes resulta em gargalos. Às vezes, os membros estressados de um time podem se indispor com seus colegas de trabalho.

Em qual estratégia-chave de dimensionamento e crescimento de times a MyApp deve se concentrar para abordar essa questão específica?

- A) Criar um time centralizado
- B) Estimular a colaboração
- C) Manter os times pequenos e flexíveis
- D) Gerir conflitos

Módulo 5

Fim da Vida

Requisitos do Exame



DevOps MASTER™

					Referência de Literatura	
		% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
5 Fim da vida		2%	1			
5.1 Condições de Fim de Vida de um produto ou serviço						
5.1.1 Expliquem que condições devem ser cumpridas antes de terminar um serviço ou produto		2%	1			7

5.1 – Condições de Fim de Vida de um produto ou serviço

		Referência de Literatura				
		% Peso	Questões	Effective DevOps	Entrega Contínua	Success with Enterprise DevOps
5.1	Condições de Fim de Vida de um produto ou serviço					
5.1.1	Expliquem que condições devem ser cumpridas antes de terminar um serviço ou produto	2%	1			

Algumas situações que levam ao fim da vida de um produto ou serviço:

- Um novo produto/serviço foi lançado em substituição ao atual
- Mudança de legislação
- Produto/serviço ficou obsoleto
- Substituição por um produto/serviço mais barato
- Terceirização (SaaS)

O Service Master é o responsável por decidir o fim da vida útil do serviço de TI, incluindo as condições para quando e como isso acontecerá.

Pontos a considerar

- O que fazer com os dados?
- O que fazer com o código, documentação, componentes criados, etc?
- Quando desativar e em qual sequência?
- Garantir que qualquer serviço de substituição esteja pronto antes de fechar o velho



Dentro da sua empresa, um dos serviços de TI que esteve em funcionamento por anos está sendo cada vez menos usado. Na verdade, após uma pesquisa, você descobriu que ninguém está mais utilizando. Quem decide se o tempo para esse serviço deve terminar?

- A) Todo o time de DevOps, pois essas decisões não podem ser tomadas por uma única pessoa.
- B) O Process Master, pois ele decide se os itens na lista de backlogs precisam ser feitos ou não.
- C) O Engenheiro da Confiabilidade, pois ele reúne todas as questões relacionadas a clientes e qualidade.
- D) O Service Master pois ele pode determinar em quais condições o serviço deve terminar.

