DEVOPS ESSENTIALS PROFESSIONAL CERTIFICATE (DEPC®)

PROFESSIONAL CERTIFICATE
CertiProf®
Professional Knowledge
DEVOPS ESSENTIALS
DEPC®

CertiProf®
Professional Knowledge

(DEPC®) VERSÃO 032020

www.certiprof.com

Objetivos

- Conhecer a importância da DevOps e aprender como aplicar esta metodologia em diferentes projetos.
- Certificação profissional.

Quem é a CertiProf®?

CertiProf® oferece uma ampla gama de certificados profissionais para indivíduos e empresas. Nossa missão é preparar profissionais da mais alta qualidade e reconhecidos internacionalmente.

Com uma equipe internacional que é especializada na implementação de materiais, nosso instituto é um dos principais fornecedores que não apenas oferece uma educação excepcional no mercado dos EUA, como também está expandindo suas atividades para regiões da América Latina.

Capacitamos e ajudamos pessoas a alcançarem seu máximo, fornecendo as ferramentas e o treinamento necessários para aumentar seu desempenho, habilidades e melhorar seu desenvolvimento profissional.

Quem deve participar deste workshop de certificação?

Qualquer pessoa que esteja interessada em expandir seus conhecimentos em DevOps.

Agenda

O que é o DevOps?

A palavra DevOps é uma contração de "Desenvolvimento" (Development) e "Operações" (Operations).

O DevOps é uma nova tendência na indústria de TI que visa melhorar a agilidade do Serviço de entrega de TI. O movimento enfatiza a comunicação transparente, a colaboração juntamente com a integração entre o software de Desenvolvedores e operações de TI.

DevOps reconhece que os desenvolvedores e operadores de TI não são grupos isolados que podem interagir uns com os outros, mas não realmente trabalhar juntos.

O DevOps ajuda a organização a criar serviços e software de TI rapidamente, o que resulta em uma redução no número de iterações.

As equipes de DevOps alcançam o sucesso com o uso dois componentes-chave chamados "comunicação" e "visibilidade em tempo real".

É essencial ter as ferramentas certas e combinar os serviços, o DevOps se preocupa em saber se uma ferramenta fornece a capacidade de interagir e funcionar de forma eficaz.

O DevOps é um desenvolvimento relativamente novo no setor de TI, que enfatiza a comunicação e a colaboração entre o software de desenvolvedor e outros profissionais de TI como a equipe de operadores, com o objetivo de automatizar o processo de entrega de software e as mudanças na infraestrutura.

Os objetivos básicos do DevOps são estabelecer um ambiente em que códigos, testes e desenvolvimento de software possam ser feitos com rapidez, frequência e segurança.

Não há uma única ferramenta DevOps que trabalhe na colaboração e integração entre as equipes de desenvolvedores, testadores e operações.

Uma cadeia de ferramentas de DevOps é usada, e consiste em várias ferramentas que se ajustam em diversas categorias do processo em todas as fases, desde o desenvolvimento até a implementação.

Essas ferramentas são utilizadas nos processos que envolvem os times de codificação, construção, teste, embalagem, liberação, configuração e monitoramento.

Por que DevOps?

Comunicação tradicional entre desenvolvimento e operações.

Definições de DevOps

Não há um acordo claro e universal sobre sua definição.

Existem várias opiniões sobre o que é e o que não é DevOps, geralmente é definido como uma nova forma de organização, uma cultura ou até mesmo uma nova maneira de pensar.

O que não é DevOps?

DevOps não é uma estratégia para todos.

Há uma grande diversidade de tecnologias de negócios e drivers a serem considerados para estabelecer a estratégia de adoção do DevOps.

DevOps não é automação.

DevOps implica automação. DevOps é mais que automação.

DevOps não é uma ferramenta implementada.

Embora existam ferramentas que são usadas no DevOps, não devemos limitar seu escopo a ferramentas específicas como Chefs ou Jenkins. Isso limita o amplo escopo como se uma única ferramenta de automação fosse comparável com todo o DevOps.

DevOps não é uma equipe de trabalho nova e separada das outras áreas de TI.

Ter uma equipe de DevOps separada elimina o objetivo de evitar possíveis atritos e falta de comunicação entre os desenvolvedores e os operadores de TI, pois isso cria outro silo.

Definição de DevOps Segundo seus Líderes

"Nós nos referimos a "DevOps" como o resultado da aplicação de princípios eficientes ao fluxo de valor de TI".

Livro de receitas do DevOps (DevOps Cookbook).

"Uma mistura de padrões projetados para melhorar a colaboração entre desenvolvedores e operadores. O DevOps visa compartilhar metas e incentivos, bem como tem processos e ferramentas compartilhados".

Michael Hüttermann.

"Um movimento de pessoas que se preocupam em desenvolver e operar sistemas confiáveis, seguros e de alto desempenho em escala".

Jez Humble.

"DevOps é uma cultura ou um movimento profissional".

Adam Jacob, CTO da Chef.

"DevOps é como um movimento filosófico".

Gene Kim, Fundador da TripWire, CTO e Autor.

História

2008

As sementes do movimento DevOps foram plantadas durante a conferência de Agile, realizada em Toronto em 2008, pelo desenvolvedor de software Patrick Debois, que tinha experiência em múltiplas funções em uma grande organização na indústria de TI como desenvolvedor, administrador de sistemas, especialista em rede, gerente de projetos e até mesmo testador.

Ele mostrou que poderia haver maneiras melhores de conseguir um ótimo trabalho na resolução de conflitos entre equipes de desenvolvimento e operações. Ele logo foi reconhecido como o líder da ideia por trás do conceito de DevOps e outros continuaram a resolver esses desafios.

2009

John Allspaw e Paul Hammond, dois funcionários no grupo Flickr, apresentaram uma palestra, intitulada "Dez implantações no dia: cooperação de Dev e Ops no Flickr".

Nessa palestra Allspaw e Hammond mostraram claramente como o conflito levou desenvolvedores e operadores a "apontar os dedos" para culpar um ao outro.

Eles apontaram que a única maneira viável de construir e implantar um software era tornando as operações e o desenvolvimento integrados e transparentes.

Inspirado por isso, Debois organizou sua própria conferência (DevOpsDay). O nome desse movimento foi reduzido para DevOps após essa convenção. O primeiro evento de DevOps foi organizado por Debois em Ghent, na Bélgica.

2010

O primeiro DevOpsDay organizado nos Estados Unidos tinha defensores do DevOps, como Andrew Clay Shafer, Damon Edwards e outros. Os eventos atraíram a atenção global e avançaram em direção à comunidade de DevOps.

Introdução da hashtag #DevOps que acabou sendo uma fonte rica de informações.

2011

Muitas análises previram o surgimento do DevOps globalmente até 2020.

Fora desenvolvidas ferramentas de código aberto, como o Vagrant, que funcionava com o Chef, o Puppet e ferramentas similares de gerenciamento de configuração.

2012

Os DevOpsDays foram organizados em todo o mundo e tornaram-se eventos de TI muito concorridos e interessantes sobre o pensamento inovador no domínio DevOps.

2013

Mike Loukides, uma figura proeminente no mundo DevOps, juntamente com Debois editou alguns textos fundamentais do DevOps. Ele afirmou que é fácil pensar no DevOps em termos das ferramentas que são usadas nele. Mas que, na realidade, este é um acordo íntimo entre as

equipes de desenvolvimento e operações.

Muitos livros sobre o DevOps apareceram. Alguns dos mais notáveis são "The Phoenix Project", "Implementando o Ddesenvolvimento Lean de Software" ("Implementing Lean Software Development"), "Web Operations" e "A Startup Enxuta" (The Lean Startup), etc.

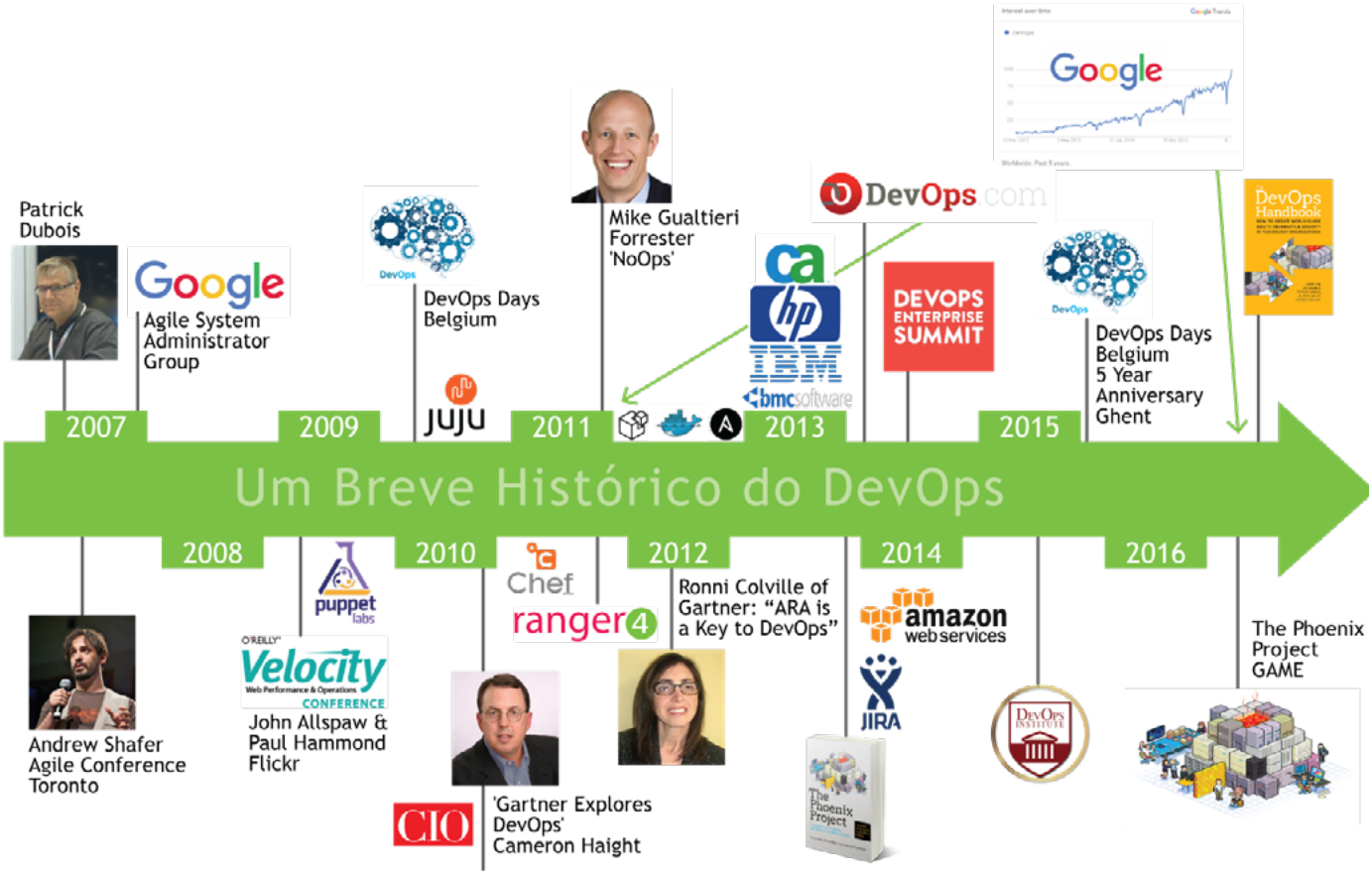
2014

O mundo tecnológico em evolução apresenta novas oportunidades para o conceito de DevOps na forma de explosão de novas aplicações, dispositivos e comunicações em ambientes móveis e computação em nuvem.

Em uma pesquisa realizada pela Puppet Labs, 16 % dos 1486 entrevistados afirmaram que fazem parte do DevOps em suas organizações.

2015 em diante

DevOps é presente e futuro...



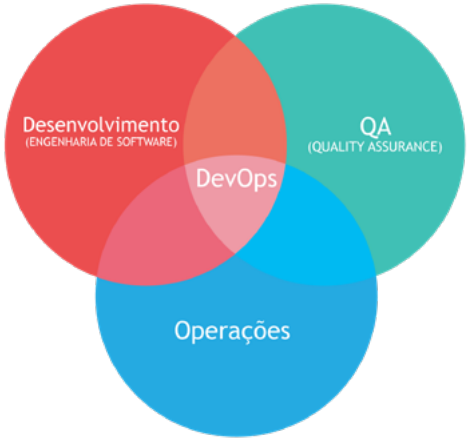
Objetivo do DevOps

O melhor objetivo oferecido pelo DevOps é fazer uma iteração mais rápida durante a fase de desenvolvimento.

Isto é conseguido evitando o atrito entre desenvolvedores e operadores, tanto quanto seja possível. Isso é alcançado garantindo transparência e integração entre a equipe de desenvolvimento e de operações. O objetivo do DevOps é estabelecer processos de negócios alinhados no fluxo "just in time" (JIT).

O DevOps busca maximizar os resultados do negócios, como aumentar as vendas e a rentabilidade, melhorar a velocidade dos negócios, ou minimizar o os custos operacionais, ao alinhar os processos de negócios "just in time" (JIT).

As empresas buscam obter novas aplicações ou serviços para finalidades específicas, mas é necessário um tempo considerável para codificar o projeto, alguns dias para garantir a qualidade (Quality Assurance), outros mais para lidar com problemas de implementação, manutenção e em muitos casos eles precisam retornar devido a diferenças entre o que era esperado e o que foi entregue.



Muitos projetos levam meses nesta inevitável rodada de eventos. Nesse contexto, o DevOps ajuda a tornar a implementação rápida e sem esforço. Portanto, o objetivo geral do DevOps é conseguir uma implantação rápida.

- O DevOps tem como objetivo estabelecer a cadeia de fornecimento de serviços de TI no negócio, da mesma forma que a cadeia de fornecimento de outros produtos é incorporada. É uma grande mudança de paradigma, desde a entrega de software até o fornecimento de serviços de TI.

- Do ponto de vista arquitetural, o DevOps precisa estabelecer um sistema de implementação automatizado rápido.

- Existem muitas metodologias e ferramentas que podem ser usadas.

- O DevOps não possui um modelo para implementação, cada organização precisa pensar e construir seu próprio processo DevOps para melhorar seus negócios.

Alguns modelos formulados:

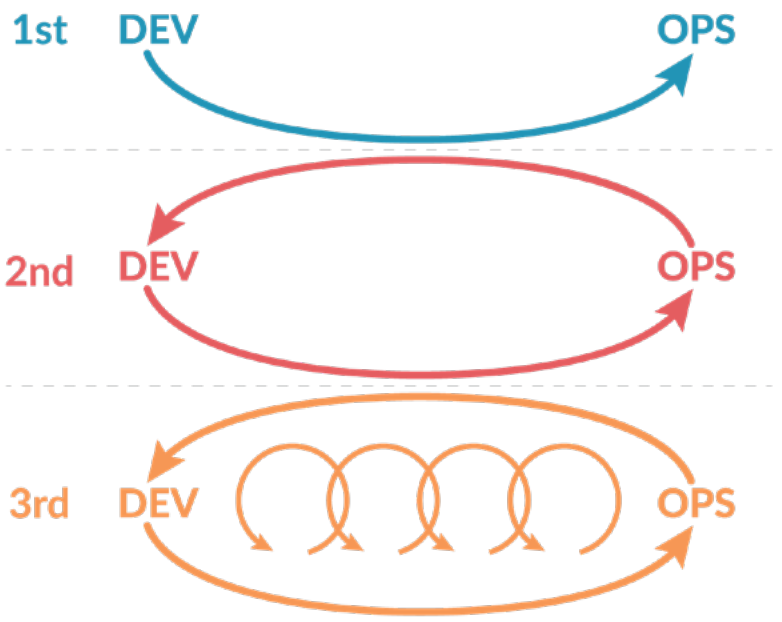
- DevOps Implementation Framework (DIF).
- DevOps Roadmap.
- DevOps Journey.
- DevOps Process.

Adoção do DevOps

O enfoque incremental concentra-se na ideia de minimizar o risco e o custo de uma adoção de DevOps, ao mesmo tempo em que desenvolve as habilidades e o impulso necessários para alcançar uma implementação bem-sucedida em toda a empresa.

Os "Três Caminhos" de Gene Kim estabelecem essencialmente diferentes formas de adoção incremental de DevOps:

- **O primeiro caminho:** Pensamento de sistemas.
- **O segunda caminho:** Amplificar loops de feedback.
- **O terceiro caminho:** Cultura da experimentação contínua e da aprendizagem.



Benefícios

O DevOps garante um tempo mais rápido de comercialização dos prazos de entrega e, portanto, melhora a rentabilidade dos investimentos (ROI).

Fundamentalmente, o DevOps é uma aplicação do conceito de desenvolvimento Agile e, portanto, o principal benefício do DevOps é o desenvolvimento mais rápido de software e a entrega frequente, melhorando o resultado final.

DevOps traz um grande benefício ao melhorar a colaboração entre o desenvolvedor e as equipes de operação. Isto é feito através da melhoria da transparência que é essencial para uma tomada de decisão eficaz.

Atualmente, as equipes de desenvolvimento precisam dividir seus silos departamentais e se comunicar e colaborar com outras equipes de TI no ambiente dinâmico atual.

O DevOps melhora a agilidade, oferecendo um ambiente de colaboração, comunicação e integração em equipes alocadas em lugares distintos em uma organização global de TI.

Outro benefício significativo do DevOps é a detecção precoce e a devida correção mais rápida dos erros, resultando na entrega dos melhores serviços aos clientes.

Um dos principais benefícios do DevOps é o lançamento, a implementação, o monitoramento e a correção contínua.

O desenvolvimento de software atual exige que as equipes se envolvam na entrega contínua sem falhas, em um curto período de tempo para os prazos de lançamento no mercado e ciclos de lançamento mais curtos.

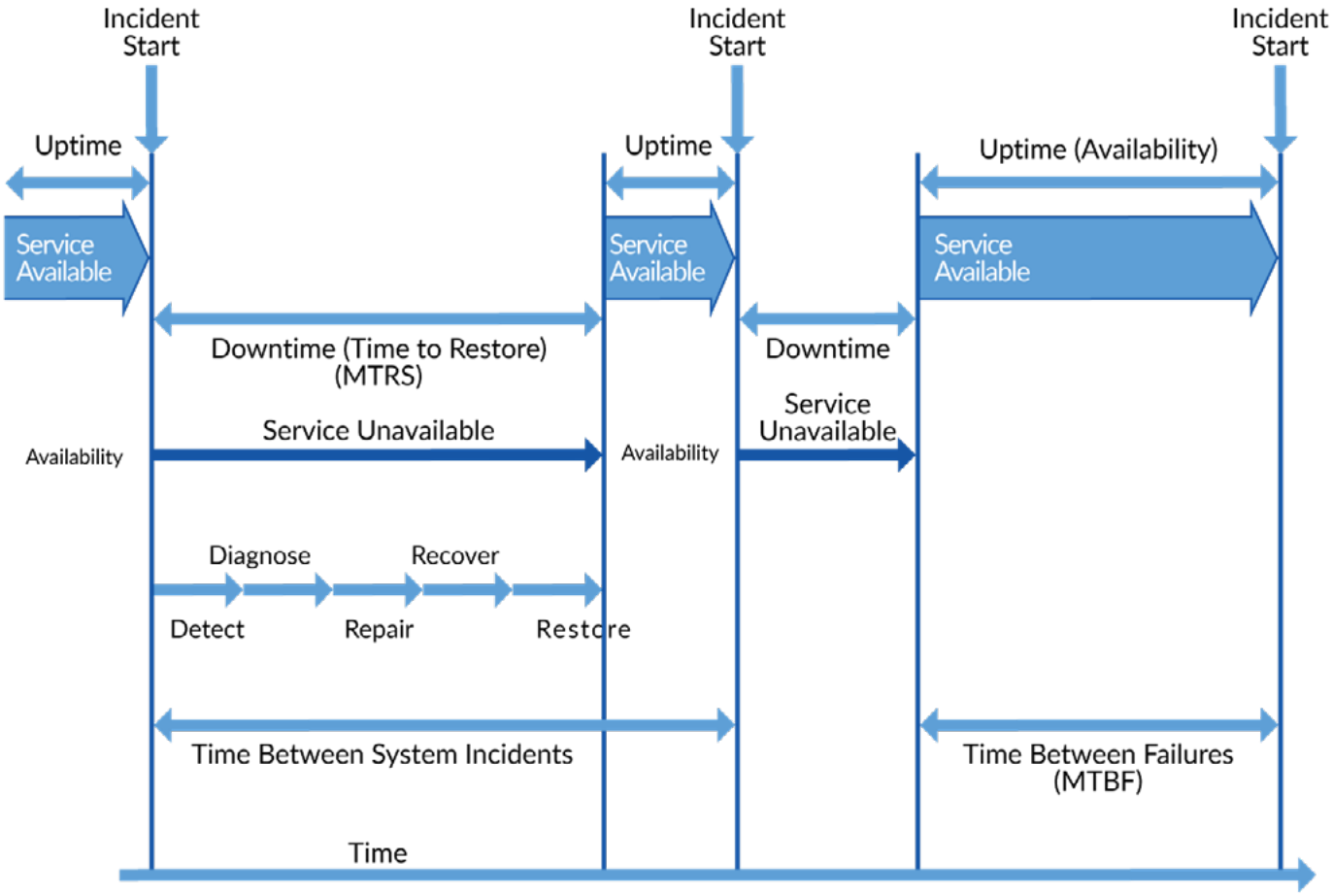
Estabilidade

Há melhorias no Mean Time To Recover (MTTR).

É o tempo médio para reparar um serviço de TI ou outro item de configuração após uma falha.

O MTTR é medido desde o momento em que o item de configuração falhou até ser reparado.

O MTTR não inclui o tempo necessário para recuperar ou restaurar. Às vezes, é usado incorretamente em vez do tempo médio para reestabelecer o serviço.



Workshop

Tempo:
30 Minutos

CertiProf®
Professional Knowledge

www.certiprof.com

Análise em Grupos

Relatório State of DevOps 201x

Pilares do DevOps

Agile

1. Velocidade.
2. Adaptabilidade a mudanças.
3. Lançamento sem erros (JKK Concept).

ITSM

1. Valor do Conceito (ITIL®).
   - 1.1 Utilidade.
   - 1.2 Garantia.

Entrega Contínua

Just-in-time (JIT)

A produção just-in-time (JIT), também conhecida como sistema de produção Toyota (TPS), é uma metodologia voltada principalmente para reduzir os tempos de fluxo dentro da produção, bem como os tempos de resposta fornecedores e clientes.

Teve origem e desenvolvimento no Japão, principalmente nos anos 60 e 70 e particularmente na Toyota. O JIT permite reduzir custos, especialmente para armazenamento de materiais, peças para embalagem e produtos finais.

A essência do JIT é que os insumos chegam na fábrica, ou os produtos ao cliente, "just in time", que são pouco antes de serem utilizados e somente nas quantidades necessárias. Isso reduz ou até mesmo elimina a necessidade de armazenar e mover as entradas do depósito para a linha de produção (no caso de uma fábrica).

Sistema de Produção Toyota (SPT)

O Sistema de Produção Toyota (SPT) (Toyota Production System ou TPS em inglês) é um sistema integral de produção "Integral Production System" e gestão, que surgiu na empresa automotiva japonesa de mesmo nome.

Originalmente, o sistema foi projetado para fábricas de automóveis e suas relações com fornecedores e consumidores, no entanto, os conceitos foram implantados em diversas outras áreas. Este sistema é um ótimo precursor para o genérico Lean Manufacturing.

O desenvolvimento do sistema é atribuído principalmente a três pessoas: o fundador da Toyota, Sakichi Toyoda, seu filho Kiichiro e o engenheiro Taiichi Ohno, que criaram este sistema entre 1946 e 1975. Originalmente chamado de "Produção Just-in-Time". Os diretores da SPT são mencionados no livro "O Modelo Toyota".

Kaizen

Kaizen, japonês para “melhoria”.

Quando usado no sentido comercial e aplicado ao local de trabalho, kaizen se refere a atividades que aprimoram continuamente todas as funções e envolvem todos os funcionários, desde o CEO até os trabalhadores da linha de montagem.

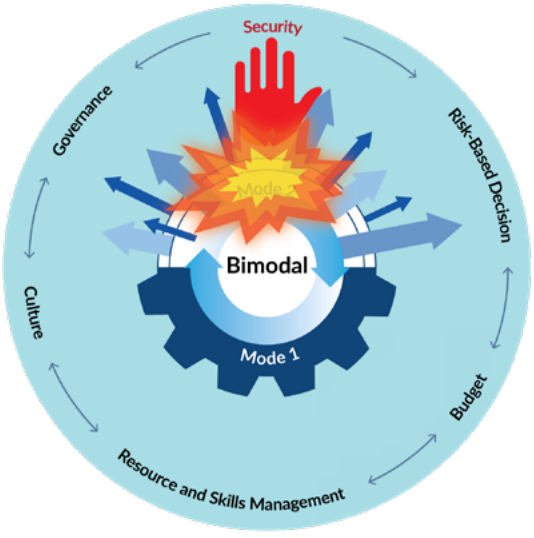
Também se aplica a processos, como compras e logística, que cruzam os limites da organização na cadeia de suprimentos. Tem sido aplicado em cuidados de saúde, psicoterapia, treinamento de vida, governo, bancos e outras indústrias.

BIMODAL - Gartner

Bimodel está em ação.

Se você tentar aplicar as convenções de segurança do modo 1 em iniciativas do modo 2, será ultrapassado e a empresa estará exposta a riscos insustentáveis.

Até 2019, 30 % dos CISCOs adaptarão as práticas de gerenciamento de risco, apoiarão a TI bimodal e melhorarão as taxas de sucesso do modo 2, enquanto reduzem os custos.



http://www.gartner.com/it-glossary/?s=Bimodal

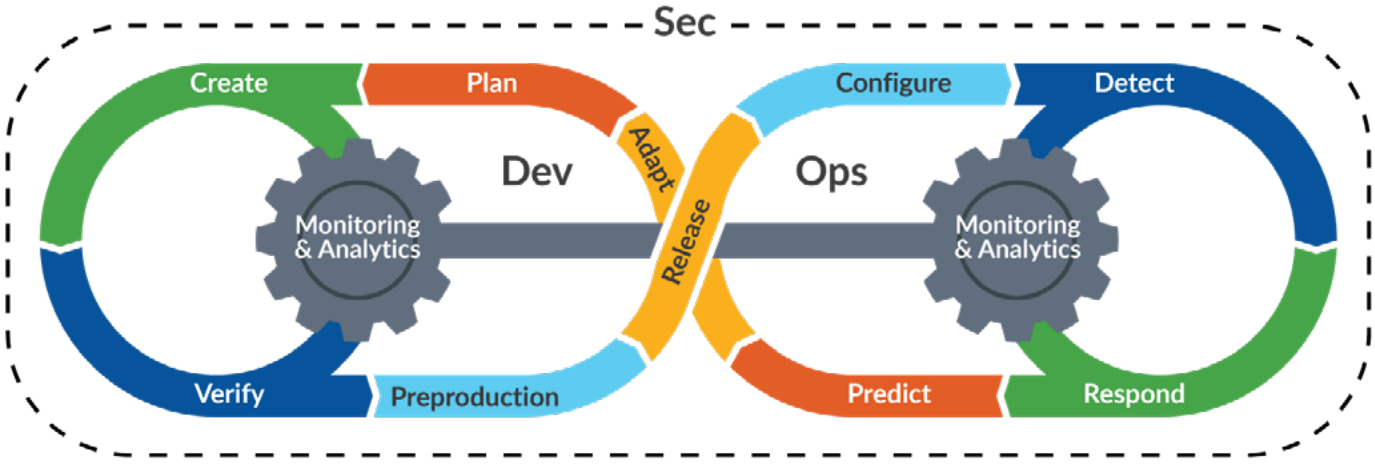
TI Bimodal: Duas abordagens diferentes, mas coerentes, bem diferentes, ambas essenciais.

	Modo 1	Modo 2
Meta	Confiabilidade.	Agilidade.
Valor	Preço por rendimento.	Acesso, marca, experiencia do cliente.
Enfoque	Lineal, cascata, High-ceremony da aplicação de desenvolvimento Agile.	Iterativo, low-ceremony, não linear, lean startup, kanban, aplicação de desenvolvimento Agile.
Governança	Planejado, base de aprovação.	Empírica, contínua, implícita no enfoque.
Abastecimiento	Suprimentos de negócios, ofertas de longo prazo.	Pequenos, novos vendedores, acordos de curto prazo.
Talento	Bom em processos convencionais, projetos.	Bom em novas abordagens e lidando com incertezas.
Cultura	TI central, arm´s-length para clientes.	Focado no negócio, fechado aos clientes.
Tempos de Ciclo	Longo (meses).	Curto (dias, semanas).

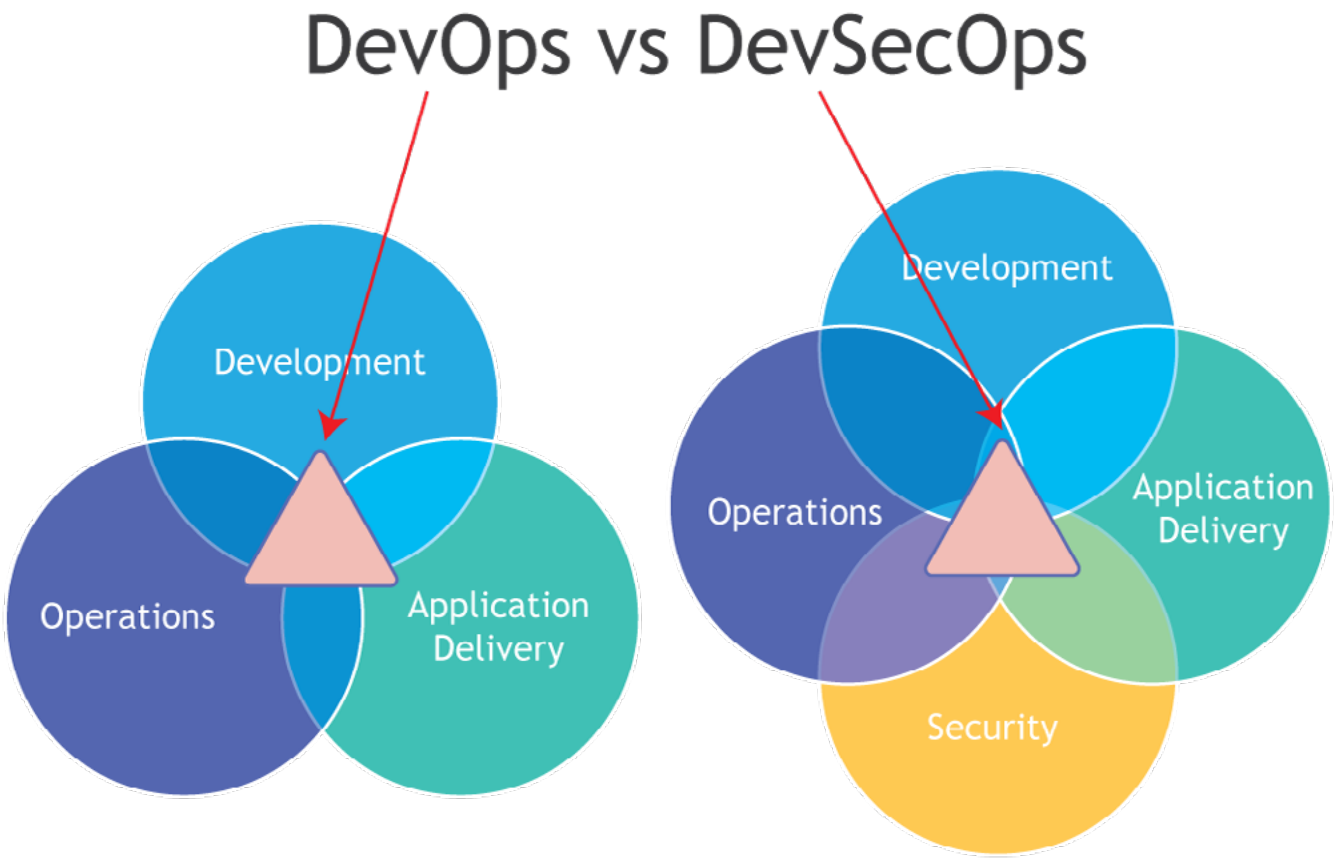
http://www.gartner.com/it-glossary/?s=Bimodal

DevSecOps

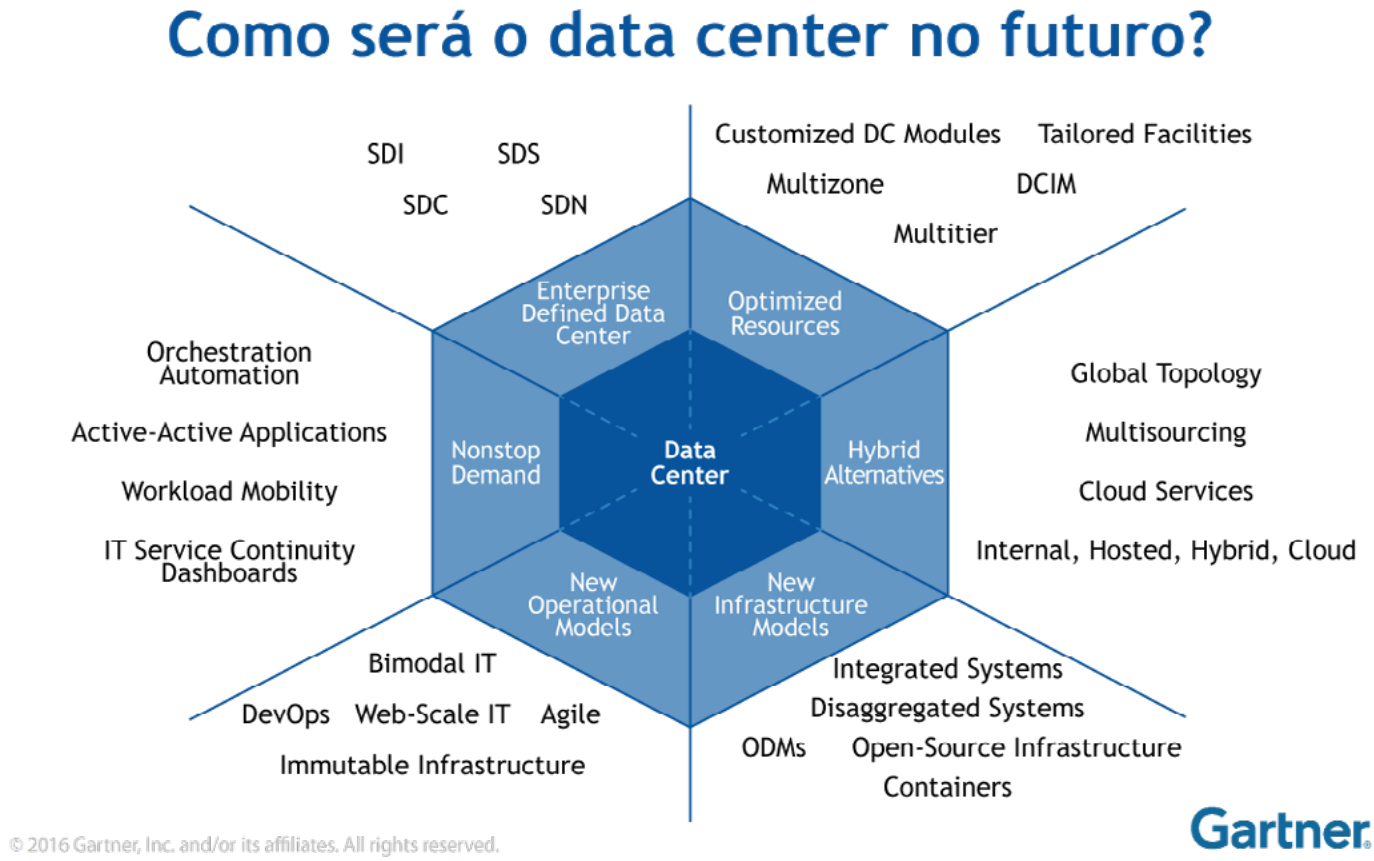
DevOps tem um relacionamento nada fácil com segurança.



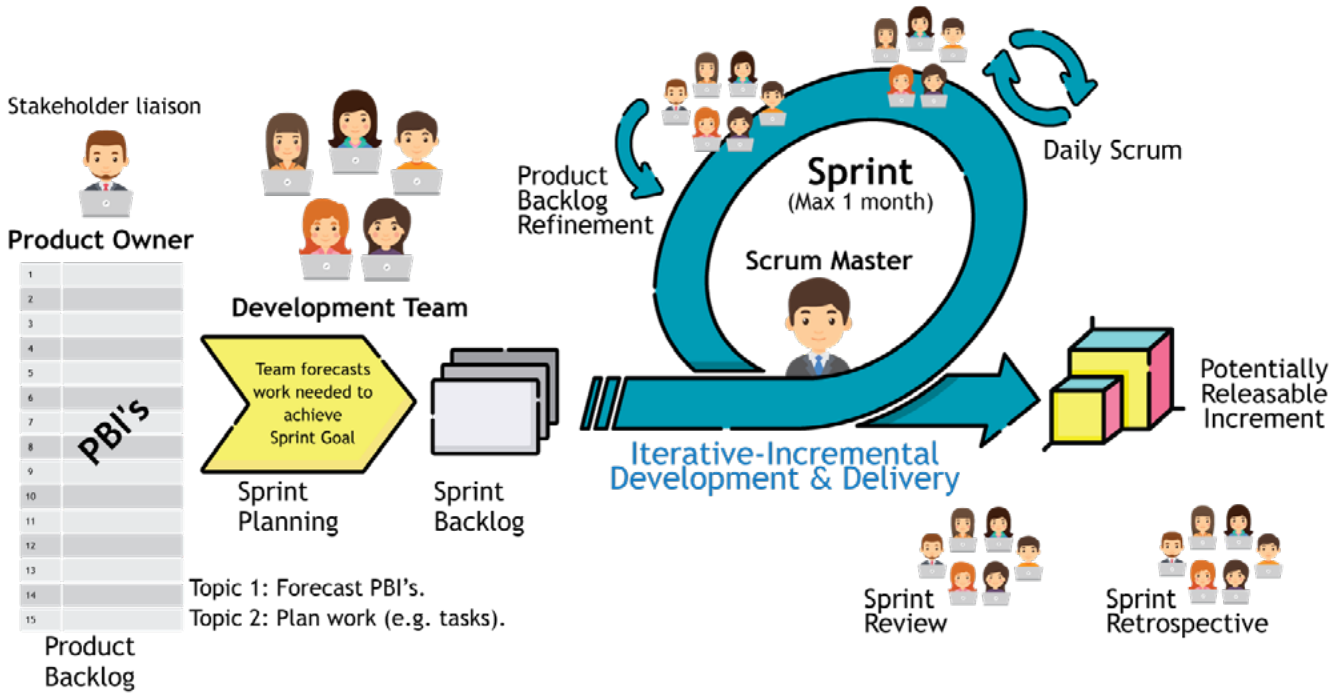
Se o DevOps não prestar atenção à segurança, pode facilitar a rápida introdução de vulnerabilidades.



Novo Modelo Operacional



Scrum



O Scrum é um framework para desenvolver e manter produtos complexos. É formado por papéis, eventos, artefatos e as regras que os ligam.

Ken Schwaber e Jeff Sutherland desenvolveram o Scrum.

Fonte: www.scrumguides.org/docs/scrumguide

Work in Progress, WIP (Trabalho em andamento)

Um limite de WIP (work in progress) é uma estratégia para evitar gargalos no desenvolvimento de software.

Ele é acordado antes do início do projeto pela equipe de desenvolvimento e é executado pelo facilitador da equipe.

Por exemplo, uma equipe pode dividir as tarefas que devem ser executadas para uma característica no design, código, teste e implantação. Quando um limite de WIP é atingido para uma tarefa específica, a equipe inteira para e trabalha em conjunto para eliminar o gargalo. O objetivo de trabalhar dessa maneira é garantir que toda a equipe assuma o controle do projeto e produza códigos de alta qualidade.

Acordo de Nível de Serviço (SLA) e OLAS

Um Acordo de Nível de Serviço (Service-level agreement ou SLA) é definido como um compromisso oficial que prevalece entre um provedor de serviços e o cliente.

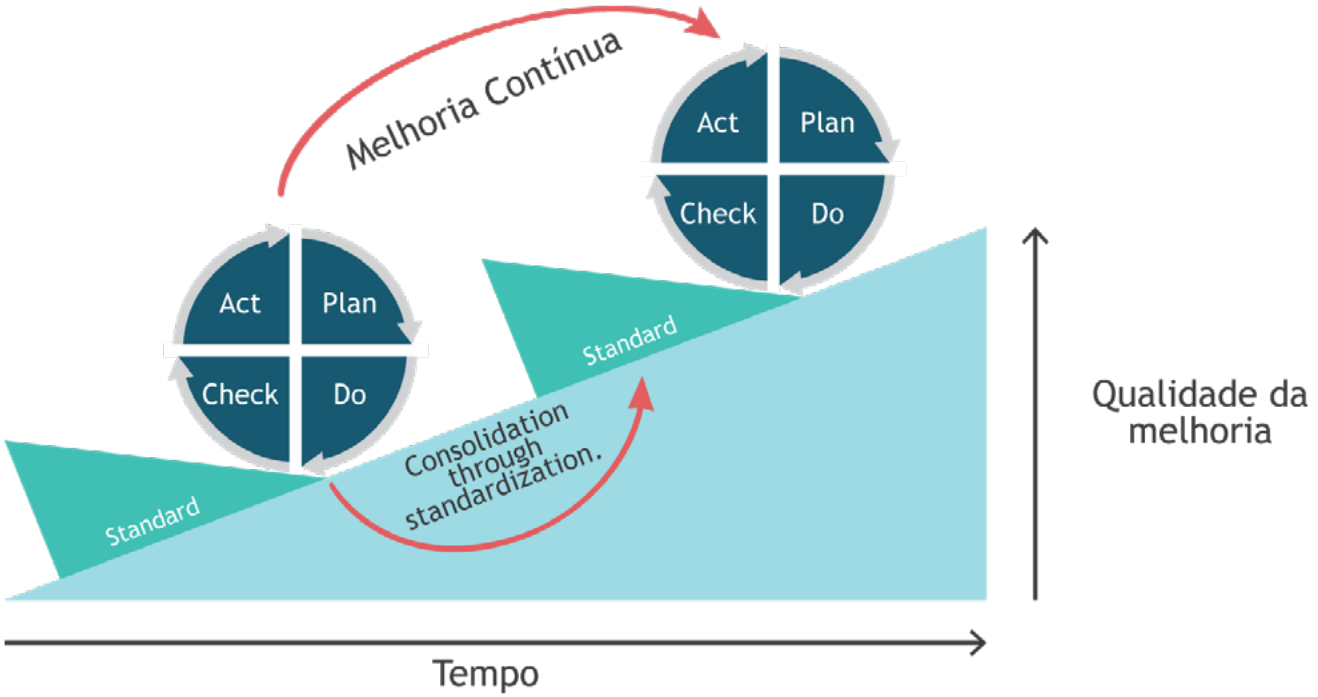
Aspectos particulares do serviço (qualidade, disponibilidade, responsabilidades) são acordados entre o provedor de serviços e o usuário do serviço.

Acordos de Nível Operacional (Operational Level Agreement ou OLAs) podem ser usados por grupos internos para dar suporte a SLAs.

Ciclo PDCA (Plan-Do-Check-Act cycle – PDCA cycle)

O PDCA (planejar-fazer-verificar-agir ou planejar-fazer-verificar-ajustar) é um método iterativo de gestão de quatro passos, utilizado nos negócios para o controle e melhoria contínua de processos e produtos.

Também é conhecido como círculo/ciclo/roda de Deming, ciclo de Shewhart, círculo/ciclo de controle, ou PDSA (plan-do-study-act).



Outra versão deste ciclo PDCA é o OPDCA. O "O" adicionado significa “observação” ou como algumas versões dizem “Observe a situação atual". Essa ênfase na observação e na condição atual está relacionada à manufatura enxuta ou à literatura do sistema de produção da Toyota.

Definição de “Pronto” (no Agile/Scrum)

Definição de "Pronto" Quando um item do Backlog do Produto ou um incremento é descrito como “Pronto”, todos devem entender o que o “Pronto” significa.

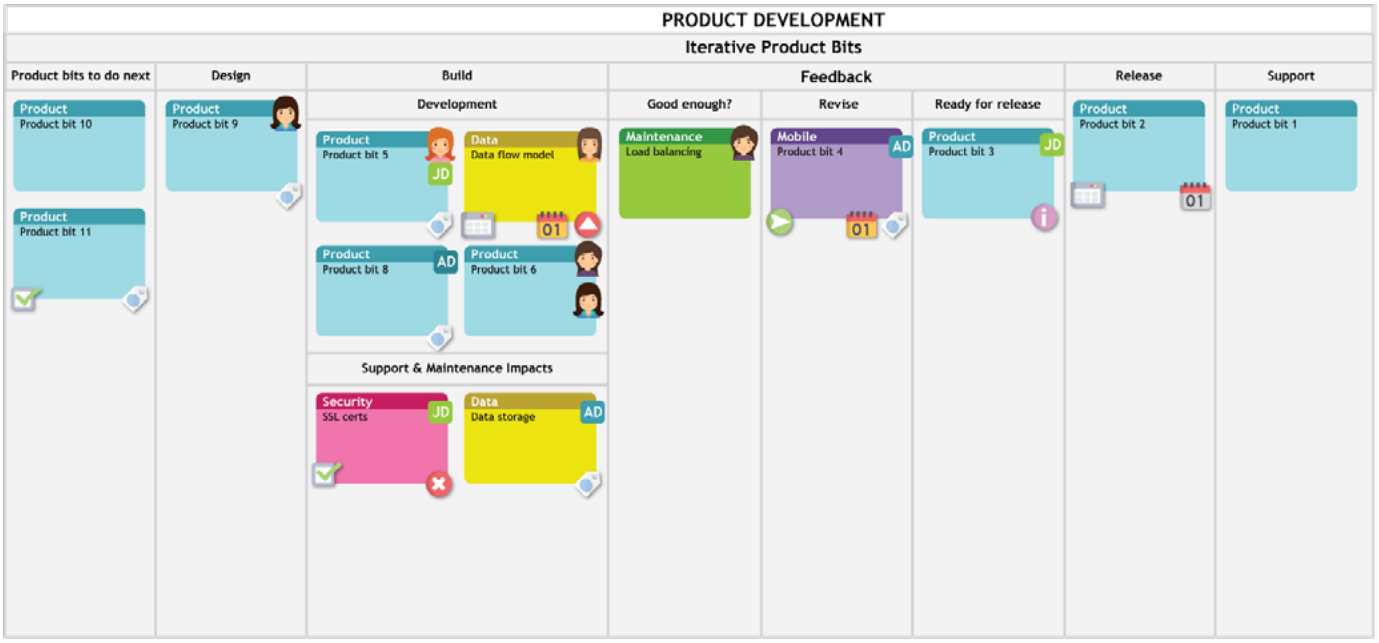
Embora isso possa variar em cada Time Scrum, os integrantes devem ter um entendimento compartilhado do que significa o trabalho estar completo, assegurando a transparência.

Esta é a Definição de “Pronto” para o Time Scrum e é usada para assegurar quando o trabalho esta completado no incremento do produto. A mesma definição orienta o Time de Desenvolvimento no conhecimento de quantos itens do Backlog do Produto podem ser selecionados durante o Planejamento da Sprint.

O propósito de cada Sprint é entregar incrementos de funcionalidades potencialmente liberáveis que aderem à definição atual de “Pronto” do Time Scrum.

Kanban para Equipes DevOps

O método Kanban pode ajudar as equipes de DevOps a colocar alguma ordem em seu trabalho diário, e a teoria lean pode definitivamente ser aproveitada para melhorar o fluxo através das equipes de DevOps.



Ciclo de vida do Desenvolvimento de Software

Ciclo de vida do software significa desenvolver o software, desde o estágio inicial até o estágio final.

O objetivo principal é:

Definir todas as fases intermediárias necessárias para validar o desenvolvimento da aplicação e garantir que o software esteja em conformidade com os requisitos para a implementação e verificação dos procedimentos de desenvolvimento, garantindo que métodos apropriados tenham sido utilizados.

No final de cada estágio, um teste é realizado para que as revisões possam ser corrigidas.

Desenvolvimento Ágil de Software

Não há uma abordagem universal para gerenciar com sucesso qualquer projeto de desenvolvimento de software. Toda a metodologia deve ser adaptada ao contexto do projeto (recursos técnicos e humanos, tempo de desenvolvimento, tipo de sistema, etc.)

Historicamente, os métodos tradicionais tentaram abordar o maior número de situações de projeto, o que requer um esforço considerável para se adaptar, especialmente nos requisitos pequenos e muito variáveis do projeto.

As metodologias Agile oferecem uma solução para quase todos os projetos que possuem essas características.

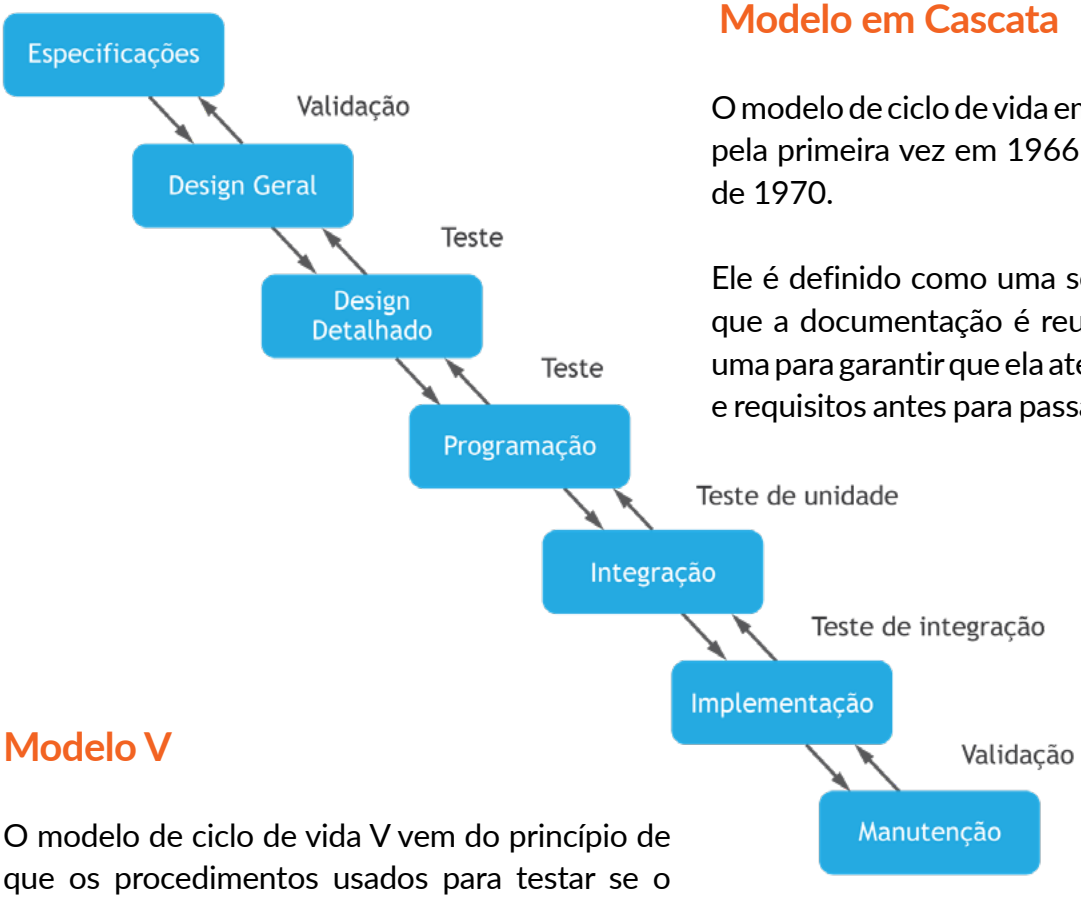
Uma das qualidades mais notáveis de uma metodologia Ágil é sua simplicidade, que reduz os custos de implementação em uma equipe de desenvolvimento.

A metodologia Ágil dá maior valor ao indivíduo, a colaboração com os clientes e ao desenvolvimento de software de forma incremental através de iterações curtas.

Essa abordagem foi considerada eficaz em projetos com requisitos dinâmicos e que tenham a necessidade de reduzir drasticamente o tempo de desenvolvimento, mantendo a alta qualidade.

Abordagens de desenvolvimento Ágil revolucionaram as formas de produzir software. Isso gerou um debate entre seus seguidores e céticos como uma abordagem alternativa às metodologias tradicionais.

Modelo em Cascata

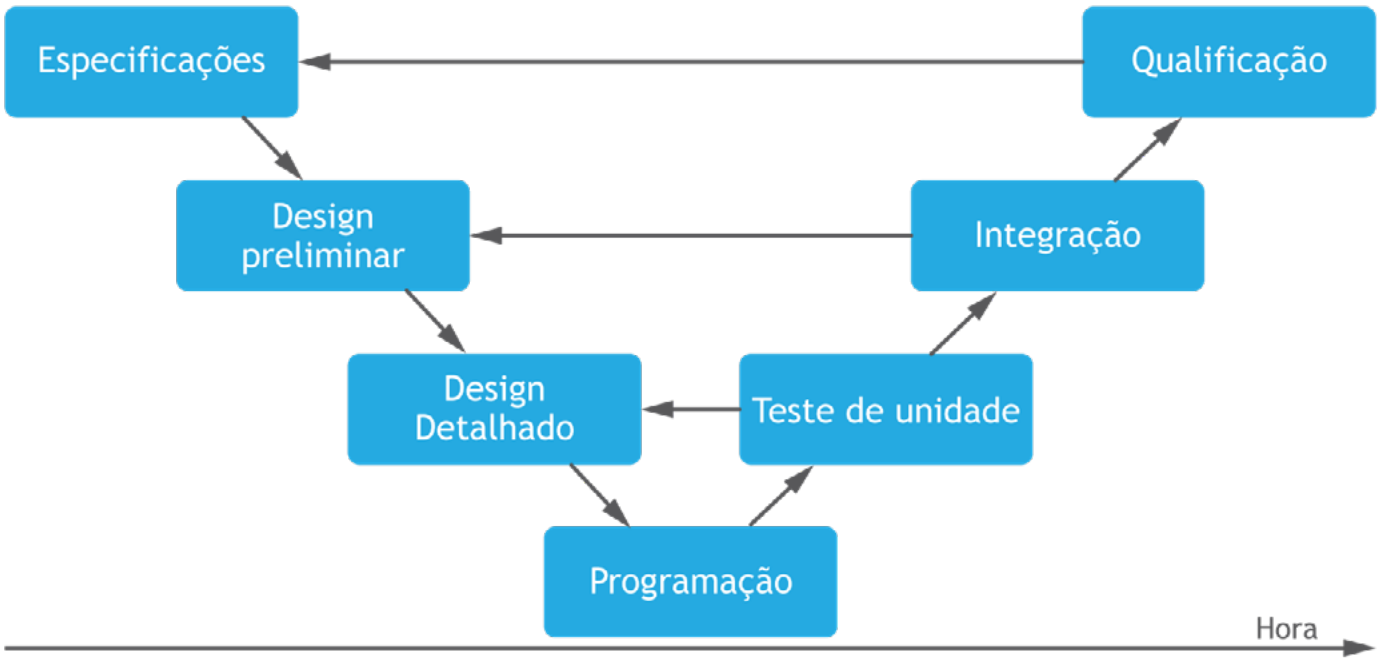


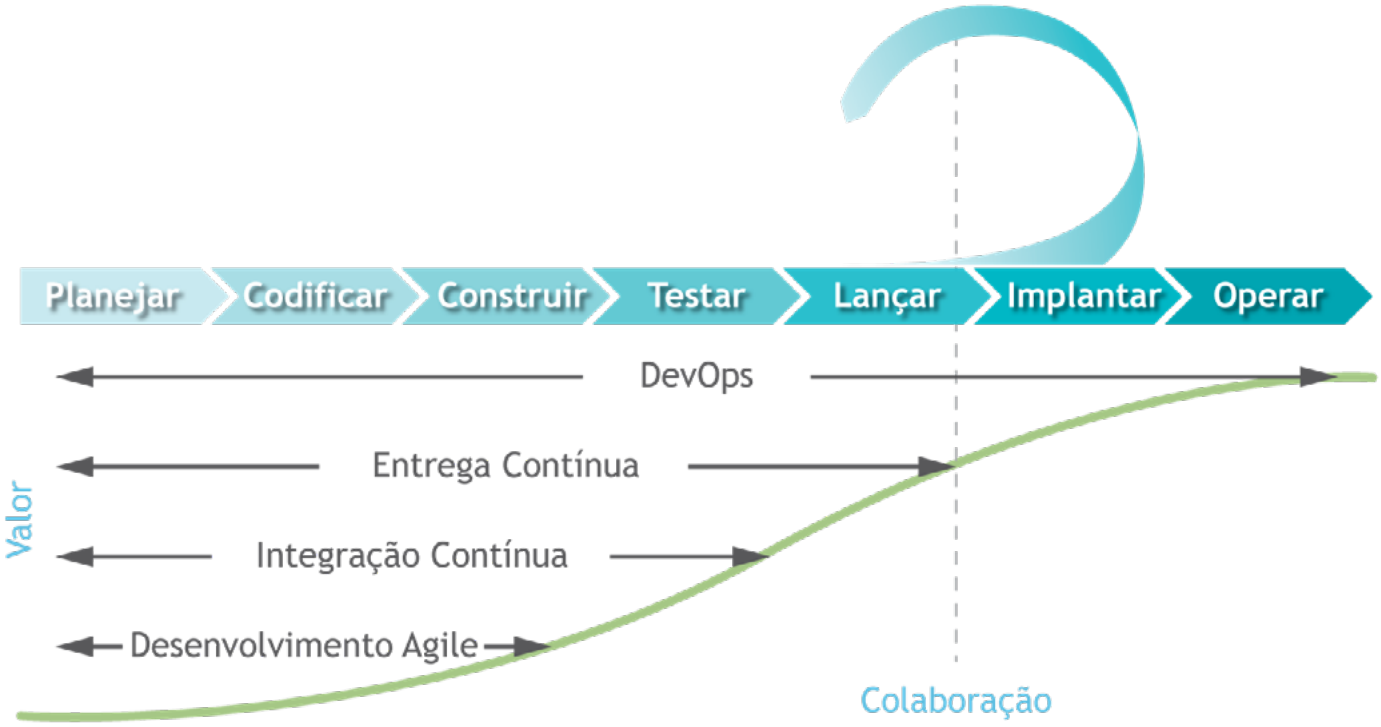
O modelo de ciclo de vida em cascata foi projetado pela primeira vez em 1966 e concluído por volta de 1970.

Ele é definido como uma sequência de fases em que a documentação é reunida no final de cada uma para garantir que ela atenda às especificações e requisitos antes para passar para a próxima fase.

Modelo V

O modelo de ciclo de vida V vem do princípio de que os procedimentos usados para testar se o aplicativo atende às especificações já devem ter sido criados na fase de design.





Integração Contínua

Cada pedaço de código é integrado ao sistema assim que o código estiver pronto.

Os sistemas podem ser integrados e construídos várias vezes em um dia.

Todos os testes são feitos e devem ser aprovados para que o novo código seja incorporado definitivamente.

A integração contínua geralmente reduz a fragmentação dos esforços dos desenvolvedores.

O time de desenvolvimento está mais preparado para modificar o código conforme necessário, pois lhe são fornecidas identificação e correção para os erros de integração.

Pequenas entregas:

A ideia é produzir rapidamente versões do sistema que operantes, embora obviamente não tenham todas as funcionalidades esperadas para o sistema, mas que sejam um resultado de valor para o negócio.

A entrega não deve demorar mais de 3 meses.

Monitoração:

O monitoramento de carga fornece feedback para a equipe no processo do XP. Sua responsabilidade é verificar o grau de sucesso entre o tempo estimado e o tempo real gasto, comunicando os resultados para melhorar as estimativas futuras. Ele também rastreia o processo de cada iteração e avalia se as metas são alcançáveis dentro das restrições de tempo e recursos apresentados e também determina quando fazer alterações para atingir os objetivos de cada iteração.

Integração contínua ou CI são práticas originadas no mundo do desenvolvimento de software, mas também podem ser muito úteis para a equipe de operações.

Um bom exemplo é uma equipe de arquitetos trabalhando na planta de uma casa. Todos os dias eles trabalham para criar o design dela de forma independente, e toda sexta-feira eles tentam construir a estrutura uma vez que todos os planos foram combinados.

Esse tipo de trabalho em equipe é bom, mas o que acontece se dois arquitetos trabalharem na mesma parte da estrutura e elaborarem dois designs diferentes? Certamente, um problema ocorrerá. No entanto, isso poderia ser evitado, como? Através da integração contínua.

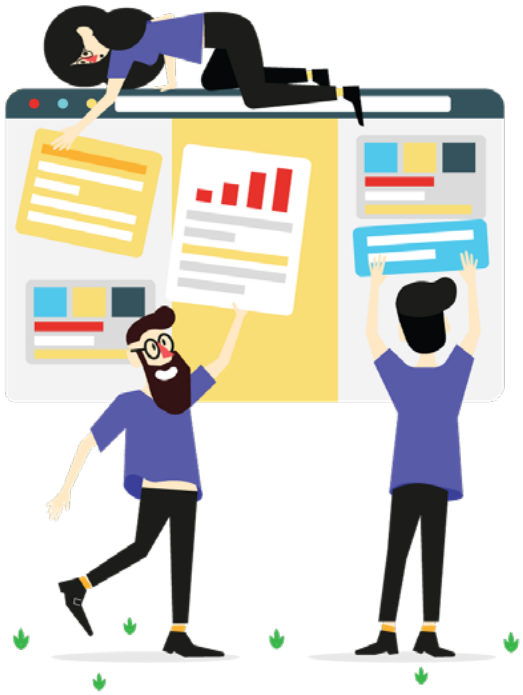
Ao implementar o CI, todos os membros da equipe devem entregar todas as alterações feitas todos os dias.

Cada vez que algo novo é adicionado ao plano, você será solicitado a escrever um teste que verifique se o que você adicionou está correto.

Quaisquer alterações futuras que ajustem o design devem ser notificadas imediatamente.

Benefícios da Integração Contínua

- Manter o repositório de uma única fonte.
- Automatizar.
- Fazer com que cada estrutura seja avaliada automaticamente.
- Testes em produção paralela.
- Automatizar a implementação.



O principal objetivo da integração contínua é que os membros da equipe conheçam a necessidade de um trabalho integrado.

Testes automatizados podem detectar erros sempre que um membro da equipe tentar fazer uma alteração.

A CI exige que os desenvolvedores trabalhem códigos integrados em um repositório compartilhado várias vezes ao dia.

Cada verificação será feita por meio de uma compilação automatizada.

A CI permite que as equipes detectem problemas rapidamente assim que eles aparecem.

Com a CI, os erros são detectados cedo.

Algumas empresas ou equipes acreditam que é possível construir e entregar sem CI, mas hoje pode ser um requisito.

Muitos acreditam que é possível desenvolver mais rapidamente sem a implementação da CI.

Com projetos em ascensão e crescimento, nem a empresa nem a equipe se tornarão mais eficientes sem a CI. Com os erros de CI detectados rapidamente, a confiança aumenta e isso leva a uma maior eficiência na entrega de software.

Com a integração contínua, haverá menos Back-tracking para descobrir onde um erro se originou. Isso permite que mais tempo seja usado na construção dos recursos.

A integração contínua é rentável, ou seja, é econômica.

Evitar a integração contínua é caro.

Não seguir a abordagem contínua significa períodos mais longos entre as integrações, por isso é exponencialmente mais difícil encontrar os problemas e resolvê-los.

Entrega Contínua

Isso permite que os clientes usem o software e voltem com comentários.

O feedback permite que o desenvolvedor faça melhorias com o software em desenvolvimento. Existem basicamente três pontos de melhoria:

- O primeiro a ser melhorado é, obviamente, a entrega do software.
- Em seguida, o ambiente para o qual o software é entregue também pode ser aprimorado para possibilitar a melhoria da eficiência e do desempenho.
- O terceiro fator a melhorar após a entrega do feedback é o processo para o qual o software está sendo entregue.

Essas melhorias permitem que o desenvolvedor de software seja mais eficiente, mais capaz e mais rápido no que eles estão entregando e, esperamos, a um custo menor.

A entrega de software não é um processo simples de entrega para produção. Há uma série completa de ciclos de entrega de software em vários ambientes pelos quais eles precisam passar.

Chama-se linha de entrega. Estes ambientes são:

DEV: Começa com o ambiente de desenvolvimento.

BUILD: O processo de construção de software.

QA: Existe também o ambiente de controle de qualidade e geralmente há mais de um em tais ambientes, cada um suportando cada tipo de ambiente usado para o software que é entregue.

Outros ambientes de pré-produção ou não-produção podem incluir:

- **SIT** (Teste de Integração do Sistema).
- **UAT** (Teste de aceitação do usuário) Pré-prod e vários outros que podem ser úteis em sua entrega, dependendo de como a organização está estruturada.

O ambiente final no ciclo de vida é o ambiente de produção, onde o software é executado, o cliente o utiliza e onde o software realmente precisa chegar.

A implementação automatizada é a possibilidade de o software ser implantado em qualquer ambiente em qualquer momento.

A entrega contínua representa a capacidade de implantar o software em qualquer ambiente específico em um horário específico.

Aprendizagem Contínua

"O aprendizado contínuo no nível de equipe é o aprofundamento e a ampliação das capacidades de (re) estruturação do grupo para atender às mudanças nas condições, adicionando novas habilidades e conhecimento e (re) criando em um sistema cada vez mais sofisticado através da reflexão sobre ações e consequências".

Valerie I. Sessa em seu livro: De aprendizagem contínua, perspectivas individuais, de grupo e organizacionais.

Cultura	• Mudar a maneira como pensamos e nos comportamos na organização. • Tornar-se um só. • Grassroots. • Cooperação.
Automação	• Configurar Itens. • Infraestrutura como código.
Lean	• Foco no valor e no cliente. • Reduzindo o tempo gasto em atividades sem valor.
Avaliação	• Avaliar a todo momento. • Apresentar melhorias.
Compartilhamento (Sharing)	• Compartilhar. • Colaborar. • Transparência.

DevOps e Agile

- DevOps com Agile é uma combinação interessante. Sempre começa com um usuário, o cliente, então algum conceito para ser levado ao mercado. Isto é conhecido como ciclo conceito-a-efetivo.

- Para obter isso do usuário ao desenvolvimento, as pessoas desenvolvem produtos, geralmente que respondem aos requisitos do cliente de acordo com suas necessidades.

- E, por meio das operações, o Agile te leva do usuário para o desenvolvimento, e o DevOps leva você do desenvolvimento até as operações nas quais você terá algo que pode realmente fornecer aos seus clientes.

O DevOps possui vários componentes, alguns dos quais são:

- Forte Controle da Fonte.
- Automação (automação do software).
- Testes iniciais e frequentes.
- Pequenos incrementos na entrega.
- Melhorias contínuas.

Equipes coesas: Significa trabalhar em estreita colaboração para produzir valor, para colocar produtos no mercado.

Esses componentes podem parecer familiares se você for um praticante ágil.

Por exemplo, no XP/Agile Engineering Practices, esses componentes já existem:

- **Test driven development:** Semelhante aos testes iniciais e frequentes.
- **Pequenas entregas:** Semelhantes à entrega de pequenos incrementos.
- **Integração contínua:** Semelhante à automação.
- **Equipe completa:** Assim como a equipe coesa ou o trabalho em equipe que acontece entre os desenvolvedores e os clientes.
- **Padrões de codificação:** Semelhantes ao controle de fonte forte.

Você pode usar as práticas de engenharia ágil junto com a capacidade de operar e pode terminar com o DevOps.

Esta é a capacidade de desenvolver e operar produtos e softwares rapidamente e depois trazê-los ao mercado. Em suma, começa com um conceito e o objetivo de convertê-lo em realidade. Você tem a ponte entre usuários e desenvolvedores, Agile e DevOps envolve desenvolvedores e operadores.

DevOps e Scrum

O Scrum foi originalmente formulado para projetos de desenvolvimento de software. É uma estrutura ágil que permite que projetos complexos sejam concluídos mais rapidamente.

No entanto, com o Scrum, as possibilidades são infinitas. Pode ser usado para qualquer campo inovador e projeto/tarefa complexos. Esse framework é muito simples, mas você definitivamente deveria estar trabalhando na criação da cultura DevOps.

Ao aprender a implementar as práticas de DevOps junto com as práticas do Scrum, é necessário decidir quanto tempo cada iteração levará: elas são chamadas de sprints no Scrum.

Cada sprint é uma representação do tempo necessário para a equipe desenvolver e testar o código. A equipe deve comprometer-se a ter um aplicativo executável em cada conclusão do sprint. O sprint de duas semanas é o mais comum para algumas equipes Scrum.

DevOps e ITSM (ITIL)

DevOps e ITIL precisam um do outro. Por quê? Porque eles têm funções que beneficiam ambos.

DevOps pode proporcionar:

- Trabalho colaborativo.
- Tempos de implantação rápidos e contínuos.
- Entrega mais rápida de funções.
- Foco no trabalho importante.
- Estabilidade no ambiente.

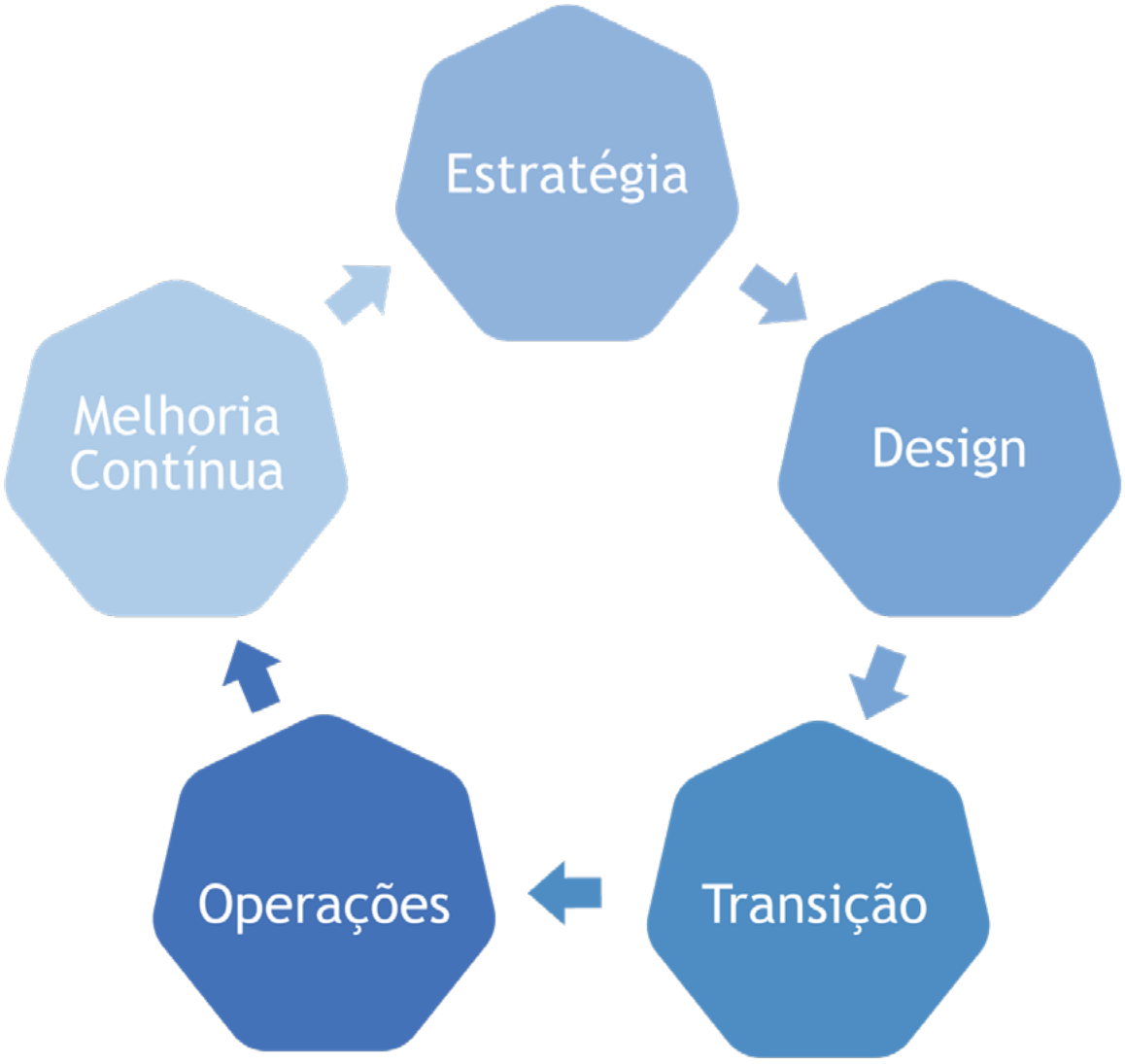
ITIL, por outro lado, pode proporcionar:

- Estrutura com seu ciclo de vida.
- Relações de negócios.
- Melhor qualidade e confiabilidade dos serviços.

Embora o DevOps, por si só, seja um processo muito útil. Pode ser melhorado quando une forças com o ITIL.

Com ITIL e DevOps, uma organização desfrutará de mais benefícios, como um escopo mais vigoroso de serviços, uma melhor perspectiva das estratégias, maiores perspectivas sobre as melhorias, melhores perspectivas sobre a atividade de transição e os rigores dos processos de design de serviços.

Se o tempo permitir, recomenda-se na preparação da certificação DevOps Essentials fazer uma análise rápida do documento AXELOS no DevOps e no ITIL, onde uma integração das práticas do DevOps com as fases do ITIL é vista. Disponível para baixar o portal AXELOS. www.Axelos.com

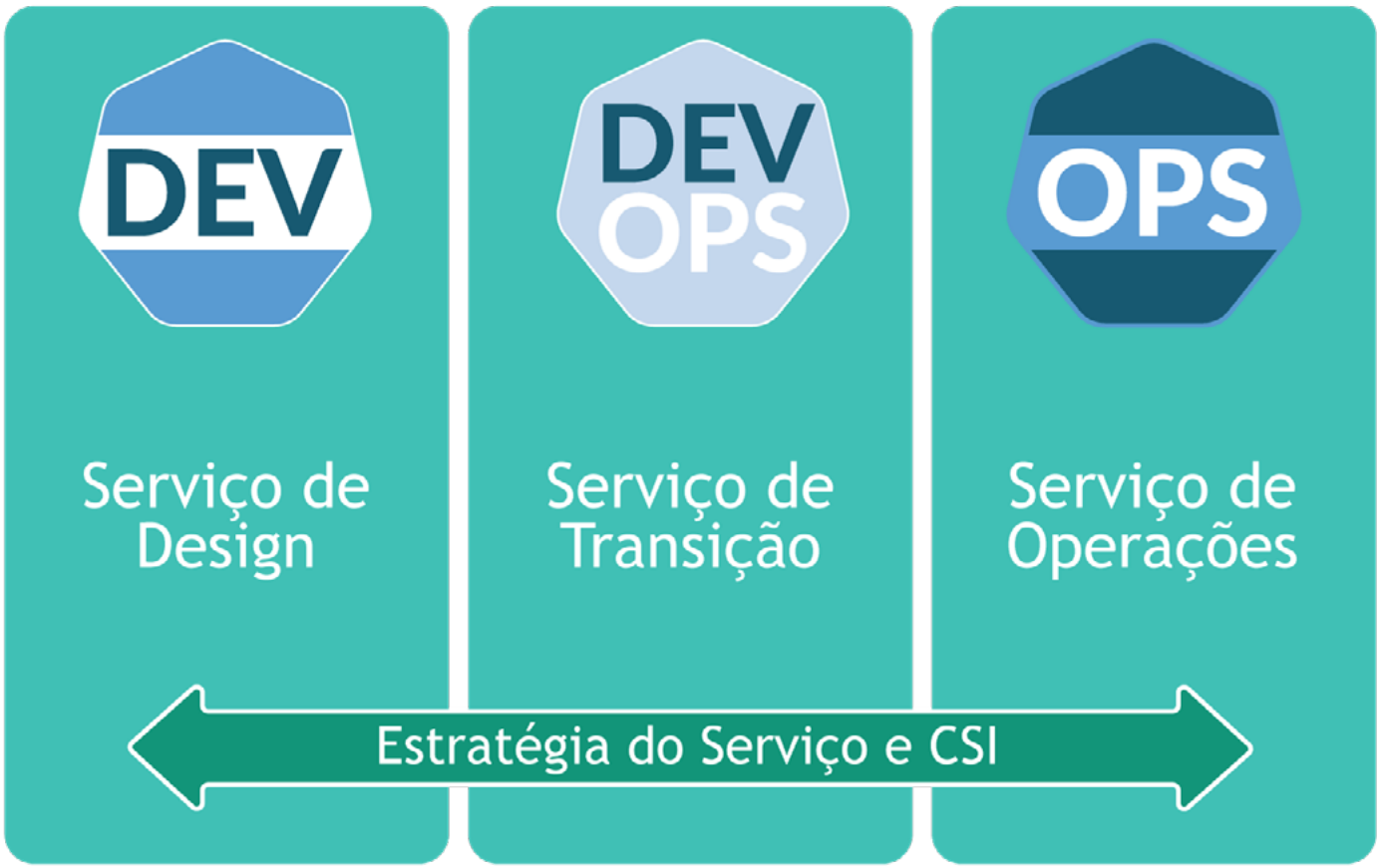


Outra percepção é que o ITIL e DevOps não podem trabalhar juntos porque não são compatíveis.

Sempre foi considerado que a organização deve escolher um e permanecer nele. Não é assim que deve ser.

Na verdade, há mais sinergias entre esses dois do que diferenças. No entanto, muitas organizações não percebem isso.

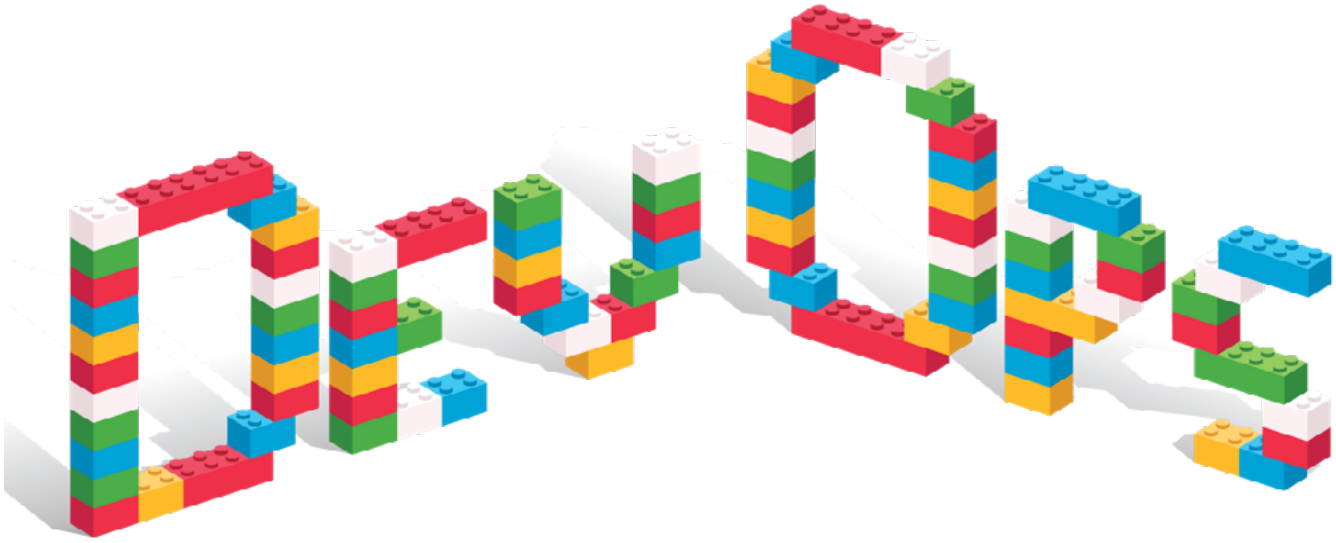
Portanto, eles estão perdendo muito em melhorias de serviço, que eles poderiam introduzir e desenvolver apenas observando como eles podem aproveitar e equilibrar esses frameworks.



DevOps industry architecture, figure 9.
"ITIL® is a (registered) Trade Mark of AXELOS Limited. All rights reserved
©Copyright AXELOS

Abaixo está um exemplo de facilitação baseada no livro de **Dana Pylayeva** com duração máxima de 3 horas.

Recomenda-se o uso dos slides fornecidos pelo autor para contextualizar a simulação de ambientes **Pre-DevOps.**



Em que consiste o jogo?

- Empresa fictícia que desenvolve e produz software.
- Software funcional: Animal de LEGO.
- Chocolate: Documentação anexada.
- O jogo se chama Legos e Chocolates (se realiza com doces).

Equipes:

- Scrum.
- Grupo de IT (Administradores de sistemas, Engenheiros de segurança e Engenheiros de lançamento).
- Representantes da empresa.

Cultura DevOps

O mundo da tecnologia é um ambiente em constante mudança e, portanto, o desenvolvimento de software é exatamente assim. Entre essas mudanças está a cultura DevOps.

Não se pode negar que, com o tempo, o impacto do DevOps cresceu significativamente. Especificamente no atual ambiente de TI competitivo e acelerado.

A ampla disponibilidade de ferramentas de programação, linguagens e serviços de software possibilitou a criação de opções quase ilimitadas para desenvolvedores de software na criação de aplicativos inovadores. Ser o desenvolvedor e o criador, permite que essa pessoa se mantenha ágil.

- Hoje em dia, há uma linha tênue entre os papéis que os desenvolvedores desempenham e já não é suficiente ter uma experiência sólida.
- Pequenas, médias e grandes empresas estão agora adotando essa nova cultura de DevOps para impulsionar seus aplicativos e programas e poder responder rapidamente às mudanças.
- Na construção de DevOps, existem fatores-chave a serem considerados. Em primeiro lugar, é importante ser generalista. Ser um especialista em um campo (tecnologia ou software) simplesmente não funciona mais.
- Produtos e empresas estão mudando ao longo do tempo, portanto, é necessário ter a capacidade de saber como trabalhar em múltiplas áreas e criar um melhor valor.

A integração contínua também é essencial. Isso significa poder enviar código diretamente para os usuários. Este é um processo de testar o código enquanto ele está no estágio de desenvolvimento.

Isso permite que os desenvolvedores ajam imediatamente, à medida que os comentários são retornados. Os códigos testados continuamente são menos propensos a erros e são mais fáceis de serem liberados também.

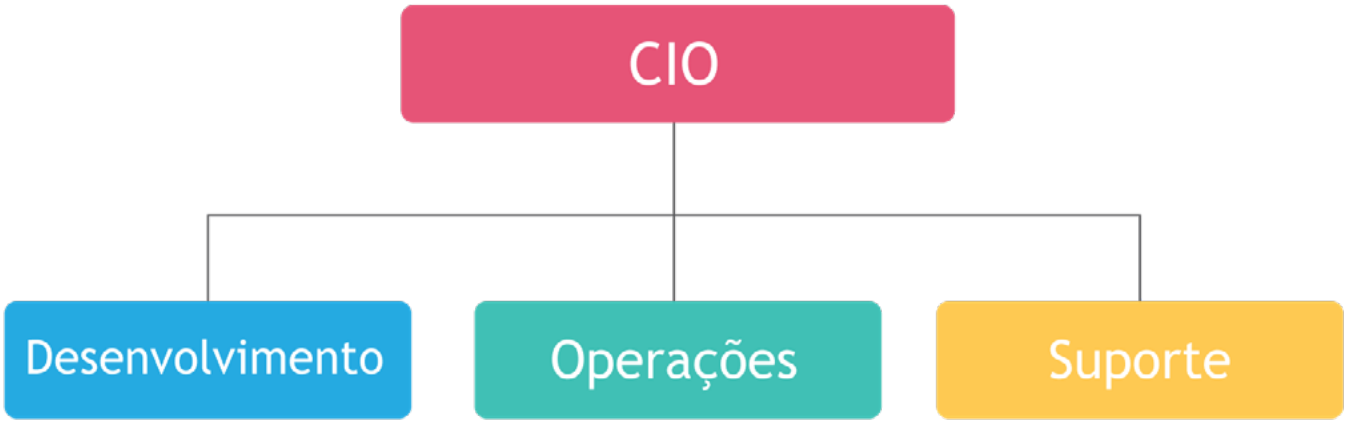
Outros fatores incluem monitoramento contínuo, implantação contínua (obtendo o novo código no processo de produção o mais rápido possível) e resiliência (criando aplicativos resilientes para que você possa responder aos erros de forma cuidadosa e detalhada).

Cultura Orientada ao DevOps

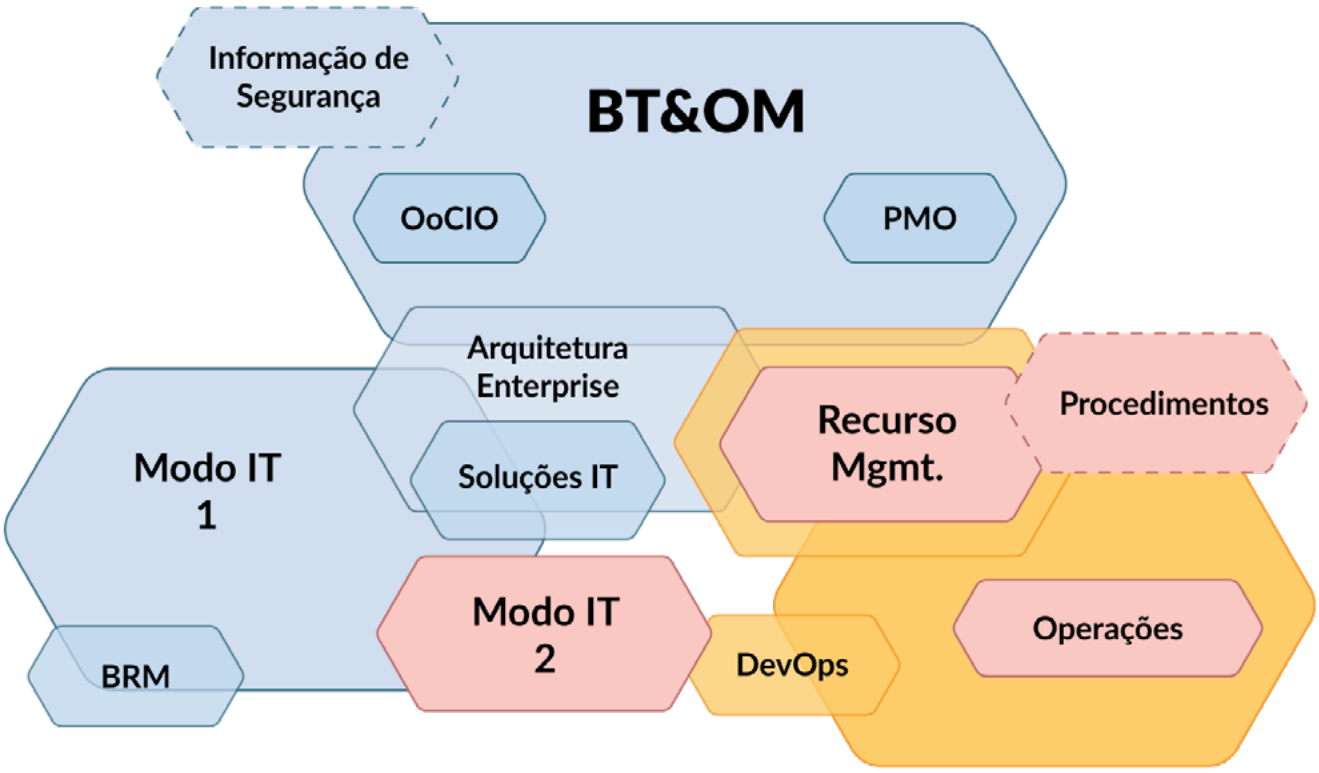
- Alta cooperação (apenas uma equipe).
- Responsabilidades e riscos compartilhados.
- Treinamento contínuo.
- Comunicadores premiados.
- Falha = Descoberta e Melhoria.
- A inovação é contínua.
- Comprometimento.

Organização-Legado

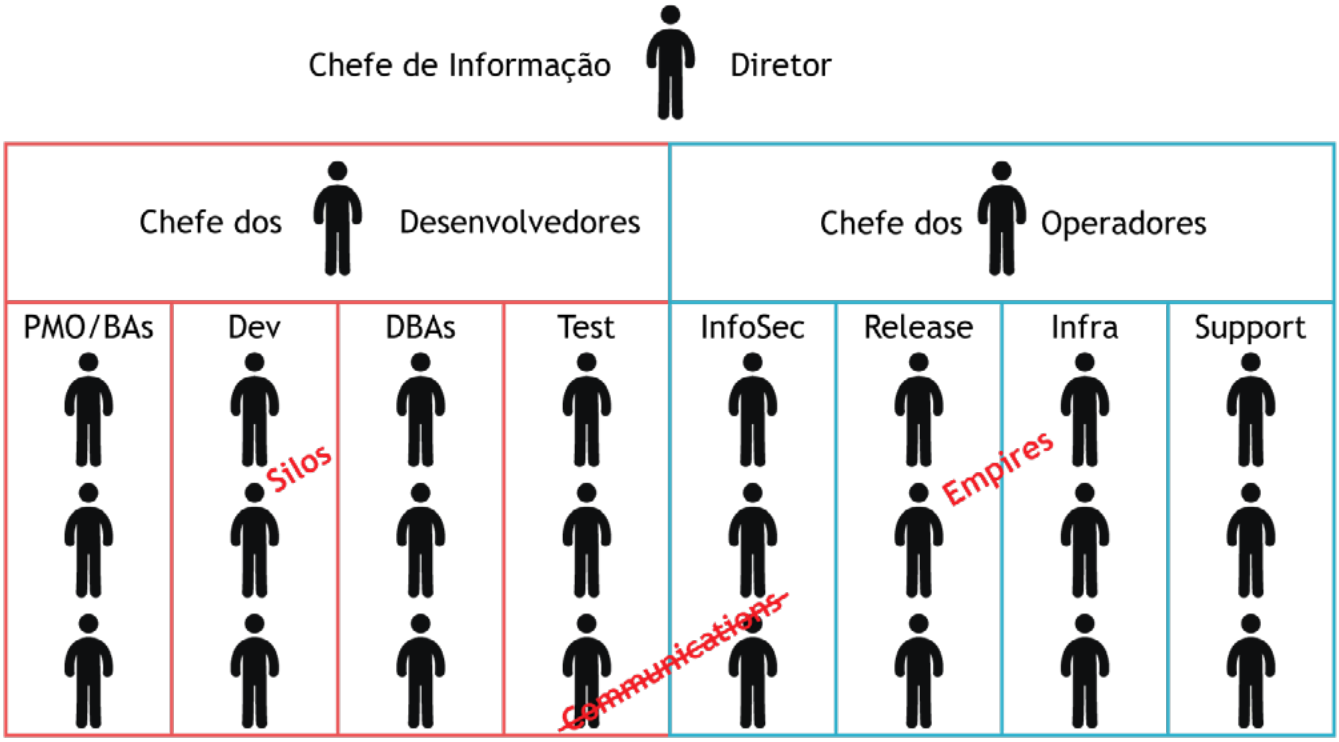
Desempenho, otimização de recursos, estrutura e linhas de relatórios, comando e controle.



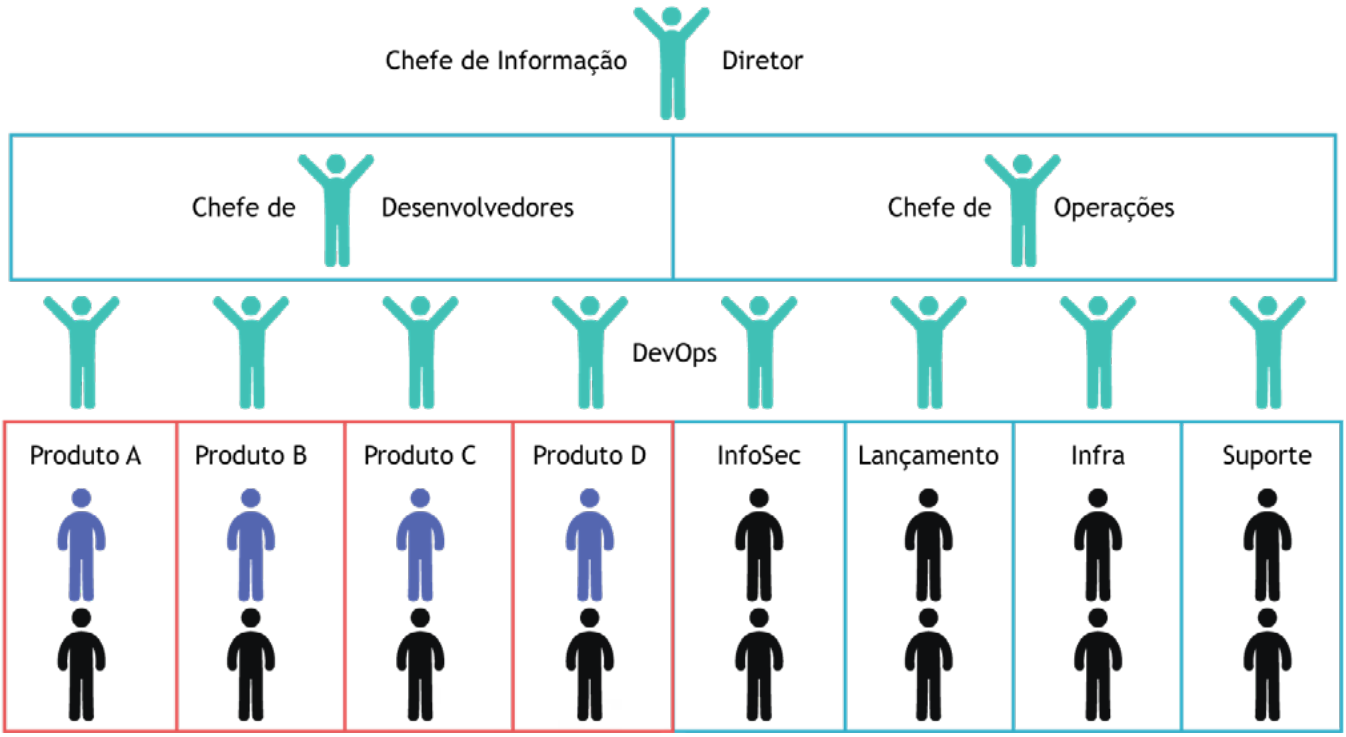
Alinhamento, integração, coordenação de processos e regras governamentais, liderança e colaboração.



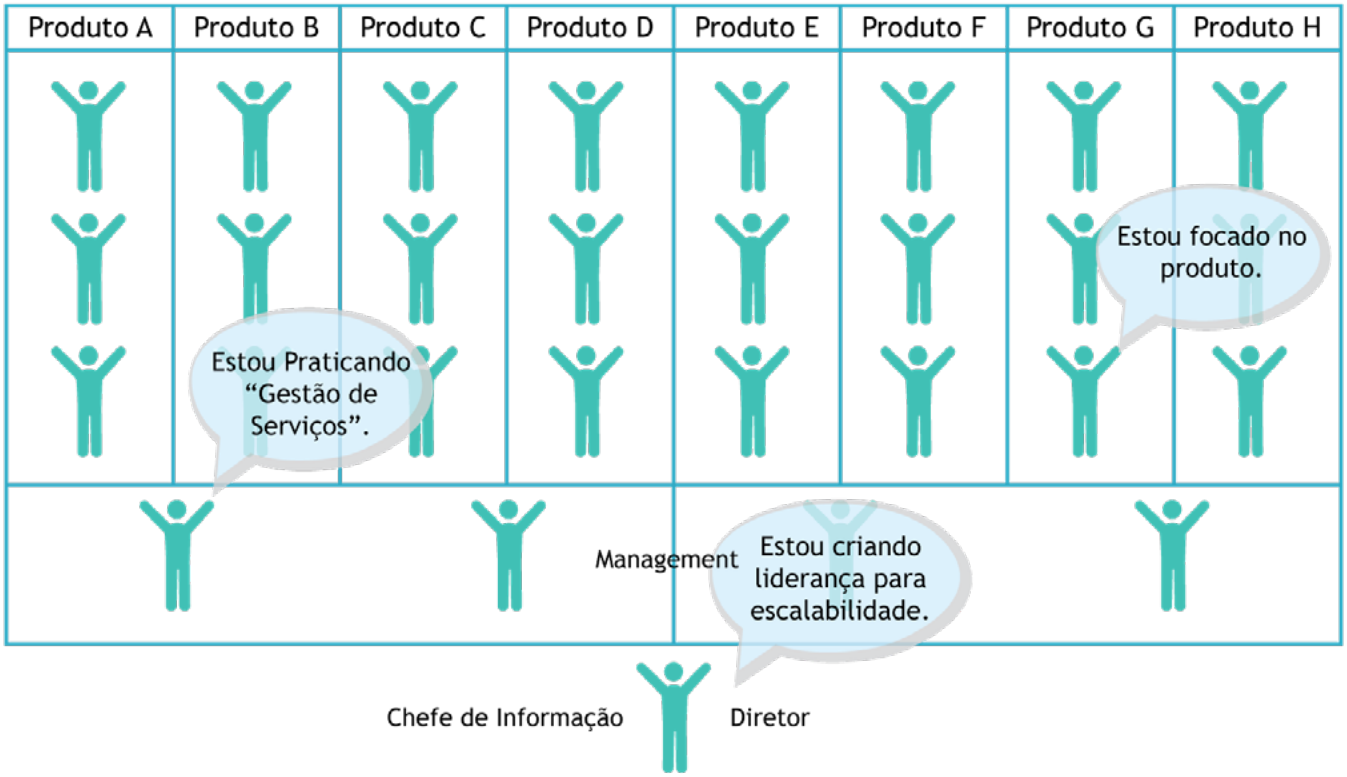
Operadores e Desenvolvedores Tradicionais



Se Criam Equipes Dedicadas de DevOps



DevOps Total



Papéis das Equipes

- Mestre de Processos.
- Mestre de Serviços.
- Engenheiro DevOps.
- Coordenador de lançamento.
- Engenheiro de Confiabilidade.
- Equipe de desenvolvimento.
- Equipe de operação.

Fonte: "Regra das Duas Pizzas" da Amazon

Strategic Management Office (SMO)

- Organização plana para pequenas organizações.
- Organização de matrizes para organizações grandes e complexas.

Adoção Incremental

Uma abordagem alternativa para a adoção agressiva do DevOps é uma adoção incremental; Embora isso possa levar mais tempo, representará um risco significativamente menor e garantirá uma implementação bem-sucedida.

A abordagem começa com o estabelecimento de pequenos objetivos, como a implementação do DevOps em aplicativos individuais mantidos por uma pequena equipe de desenvolvedores. Uma vez que isso tenha sido feito com sucesso, as lições aprendidas podem ser aplicadas para objetivos cada vez maiores, até que finalmente haja um equilíbrio saudável entre as melhorias do processo e a execução do processo.

O principal desafio continua sendo encontrar um equilíbrio entre os complexos ambientes de TI da organização, a velocidade de implementação e o risco, que devem ser feitos para trabalhar juntos.

Portanto, DevOps deve ser planejado para melhoria contínua e não fazer a implementação de um evento único.

Uma estratégia incremental deve ser preferencialmente feita em uma abordagem cíclica, em camadas, que minimize os riscos e os custos da adoção do DevOps, e construa os conjuntos de habilidades internas necessárias e o ímpeto para implementar ciclos maiores e mais rápidos.

Cada passo será construído na etapa anterior, em termos de investimento e preparação ao longo do ciclo global. As horas dedicadas ao desenvolvimento serão realistas para a maioria das empresas e variarão amplamente de acordo com a cultura e a política de uma organização.

É provável que as reuniões do grupo consumam a maior parte das horas, com estimativas baseadas em pequenas equipes de implementação de 1 a 3 pessoas que interagirão continuamente com o pessoal-chave de 2 a 4 pessoas em um aplicativo.

Pensamento de Sistemas

O mais importante é que as organizações de DevOps sempre pensem em termos de sistemas. Sistemas significa conjuntos de processos integrados que trabalham para objetivos comuns.

Para organizações de DevOps, Sistema significa o negócio como um todo, não seus departamentos ou equipes específicas.

Organizar atividades de confiança no desenvolvimento de software envolve procedimentos e processos variados.

As equipes de desenvolvedores escrevem o código e, em seguida, testadores testam e os operadores fornecem a infraestrutura para executá-los e, finalmente, os clientes os consomem.

Pode haver muitas pessoas e procedimentos envolvidos nesse processo.

Pensamento de sistemas significa que cada equipe deve estar ciente das ações de todas as outras equipes que trabalham nas linhas de desenvolvimento e somente após esse entendimento as entregas são feitas para o cliente.

O DevOps define como as ações podem afetar não apenas a equipe, mas todo o sistema. Isso significa que os desenvolvedores podem ter maior visibilidade em todo o ciclo de vida dos códigos que escrevem.

Isso também significa estar ciente das possíveis ramificações de uma mudança, em vez de esquecer as modificações após a conclusão dos códigos.

Isso também significa que os administradores de sistemas devem entender completamente o impacto do desempenho nos aplicativos, uma vez liberados para os clientes.

O pensamento sistemático também é uma excelente abordagem para pensar sobre a culpa.

Tradicionalmente, quando alguém forçava partes do código em produção, o que causaram grandes interrupções no serviço, elas seriam crucificadas pelos colegas e pela gerencia.

Eles seriam marcados, repreendidos e até mesmo demitidos.

Um pensamento sistemático foi capaz de eliminar essas inclinações iniciais para a culpa individual.

É sobre não culpar um indivíduo que teve a infelicidade de pressionar o botão, mas o sistema que permitiu que isso acontecesse.

O pensamento sistemático tratará incidentes como esses como falhas no sistema.

A equipe de DevOps investigará imediatamente as causas; não por pessoas que fizeram isso, mas por causas de como isso pôde acontecer. É possível investigar as causas de por que os testes automatizados não conseguiram detectar a falha, o que leva ao princípio da aprendizagem e experimentação.

Experimentação e Aprendizagem

As organizações de DevOps sempre experimentam e aprendem com erros anteriores.

Em um caso em que um desenvolvedor interrompe o sistema, o incidente seria completamente investigado reunindo pessoas apropriadas e incorporando providências para evitar recorrências futuras.

O incidente seria corrigido de forma que mesmo que alguém fizesse a mesma coisa novamente, o sistema evitaria a falha.

As pessoas aprendem com seus erros, e a organização permite que todos experimentem, em vez de forçá-los a se concentrar em um conjunto restrito de tarefas.

As organizações de DevOps estão sempre incentivando a criatividade e o pensamento fora da caixa, em vez de preservar as abordagens da mentalidade das formas antigas.

Isso, naturalmente, encorajaria todos a serem humildes, nenhuma ideia perfeita, todas podem ser desafiadas.

O DevOps foi capaz de estabelecer precedentes na experimentação sendo aceitável, mesmo se eles terminarem em fracassos, uma vez que normalmente as falhas são as melhores maneiras de aprender.

Feedback

Sem nenhum feedback, o aprendizado seria ineficiente, portanto, a implementação do DevOps sempre estimula diferentes tipos de feedback.

O feedback ocorre entre pares nas mesmas equipes e entre empresas na forma de feedback do cliente.

O DevOps mantém os canais de comunicação abertos para facilitar o feedback e permitir a infiltração do cliente nos níveis de desenvolvimento e operação.

O DevOps incentiva o desenvolvimento rápido e seus ciclos de feedback não são diferentes.

Para fornecer os recursos ao cliente no menor tempo possível, é essencial que o feedback seja recebido o mais rápido possível.

As abordagens de DevOps reverteram as abordagens tradicionais e os clientes controlam o desenvolvimento por meio de canais abertos de feedback.

Os clientes podem fornecer feedback imediato que é exibido e priorizado e convertido em novos códigos que são aplicados à infraestrutura para ir aos clientes novamente por meio de um loop de feedback.

É igualmente importante para a participação e interação das pessoas que oferecem o aplicativo, como analistas de negócios, engenheiros de rede, desenvolvedores, testadores, etc.

Os riscos de produção são mitigados testando conceitos em múltiplas fases e superando a resistência à mudança. Embora uma abordagem de processo incremental leve anos, os resultados são sempre transformadores, o ponto final é que o DevOps é dinâmico.

Ferramentas para o Suporte da Cultura DevOps

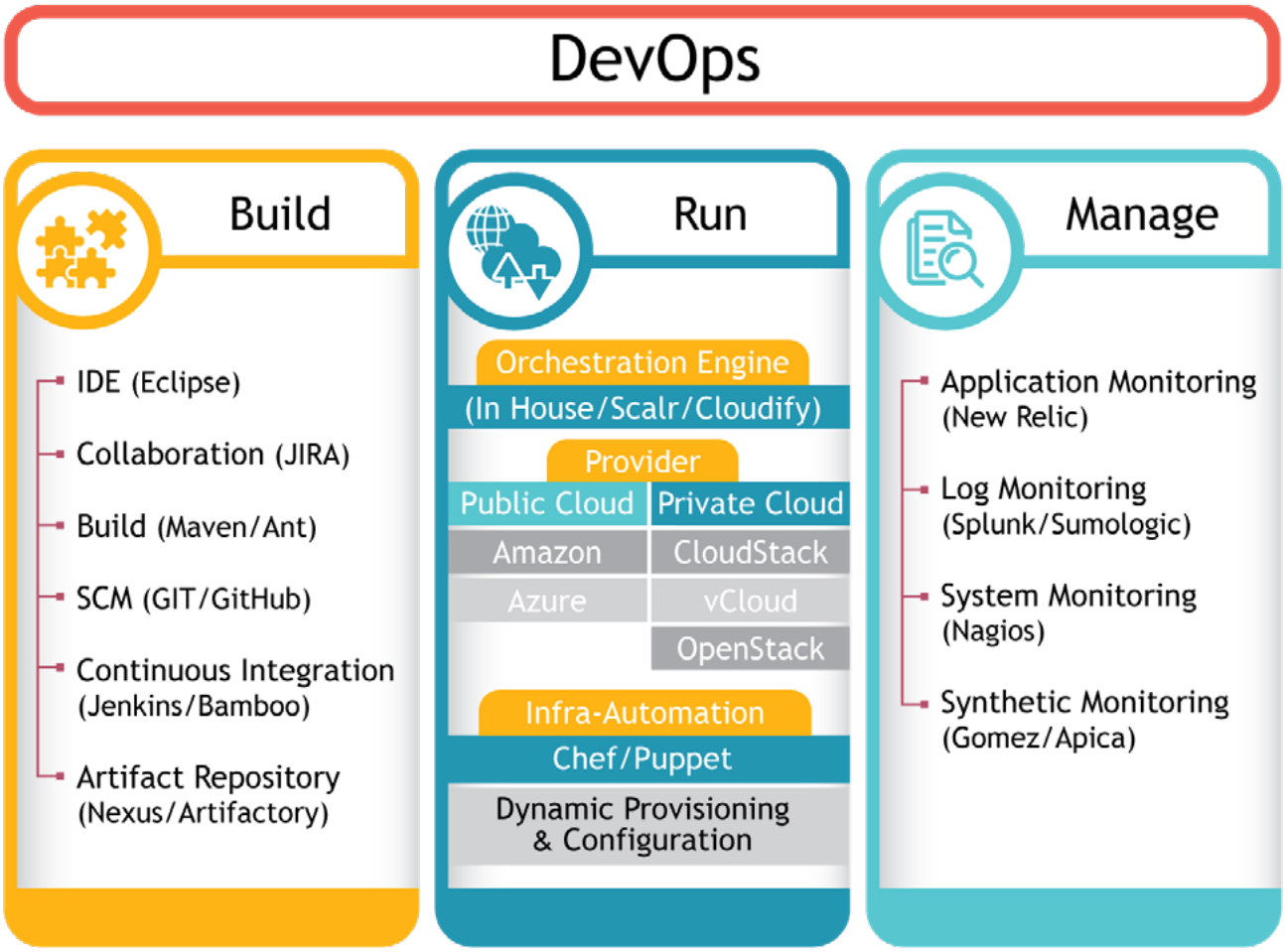
Utilizar as ferramentas não é o mesmo que fazer DevOps.

Fazer o DevOps sem o uso de ferramentas é complexo se entendermos que o DevOps tem componentes de automação. DevOps é e será um tema de Cultura e não será apenas o conjunto de ferramentas.

Vejamos um exemplo de que as ferramentas não fazem cultura. Pense em uma empresa que deseja que suas reuniões internas comecem no horário marcado. Se a empresa comprar software de gerenciamento de reuniões, as reuniões começarão no horário marcado?

Se a cultura interna é iniciar reuniões atrasadas, devemos trabalhar para mudar a cultura interna primeiro. A ferramenta por melhor que seja, não conseguirá mudar a cultura.

Quais ferramentas para DevOps ou chamadas DevOps que são conhecidas pelos candidatos à certificação DevOps Essentials? O que essas ferramentas fazem?



Sem as ferramentas adequadas, a fusão de papéis e responsabilidades do DevOps pode criar caos e resultados indesejáveis, que incluem problemas relacionados à escalabilidade, confiabilidade e gerenciamento de carga.

Ferramentas DevOps-GIT

O GIT foi produzido seguindo os requisitos da comunidade Linux para SCM, ou seja, o software Source-Control-Management que poderia suportar sistemas distribuídos. Essa é a ferramenta mais comumente usada para gerenciar fontes que está disponível atualmente.

Sua vantagem é ter ótimos recursos adicionais em aplicações de bifurcação e tração; por disso, o GitHub também fornece plug-ins capazes de conectar o Jenkins e facilitar a integração e a implementação.

Os testes de velocidade que são executados em uma operação de rede GIT geram resultados impressionantes, pois os protocolos GIT para transferência de dados são altamente otimizados.

No entanto, com o passar do tempo, as implementações internas do GIT revelam limitações que podem afetar a velocidade e a eficiência dos canais de entrega.

Plataforma Cloud

A automação na nuvem é composta de objetos de automação discretos ou tarefas que realizam trabalhos como arquitetura de servidores da VM, instalação de imagens do sistema operacional e implantação de aplicativos e funções de rede.

As tarefas na automação da nuvem são executadas por muitas ferramentas de automação, como scripts e ferramentas de gerenciamento local para gerenciamento de configuração.

A automação DevOps usa funções de automação discretas como ferramentas de integração, compilação e gerenciamento contínuos para configurações de software.

As orquestras do DevOps são uma coordenação automatizada por processos personalizados de DevOps nas ferramentas e tarefas de organização e automação que estão sendo realizadas no processo.

A infraestrutura em nuvem é executada em conjunto com as ferramentas DevOps para a automação e orquestração de implantações de aplicativos e coordena o suporte para os processos desejados.

A orquestração vai além da simples tração da infraestrutura de nuvem para abranger as tarefas de automação de DevOps em coordenação com a infraestrutura para obter a orquestração de DevOps.

Os desenvolvedores que usam o Docker criam ambientes Docker no laptop para desenvolver e testar aplicativos localmente em containers. Nesse ambiente, os testes são realizados em pilhas de serviços compostos com vários containers do Docker.

Vários containers do Docker convergem como baterias de serviço único, como baterias LAMP, e levam segundos para criar uma instância.

Docker

O Docker traz portabilidade para aplicativos por meio da tecnologia de container, onde os aplicativos podem ser executados em unidades autônomas que se movem pelas plataformas.

Ele consiste em um mecanismo Docker (uma ferramenta leve de empacotamento e tempo de execução e um Docker Hub) que são serviços de nuvem para compartilhamento de fluxo de trabalho e automação.

O Docker tem sido uma parte vital da próxima geração de testes de Yelp e infraestrutura para gerenciamento de serviços.

O isolamento dependente e o rápido arranque de containers permitem um ciclo de desenvolvimento mais curto e aumentam as velocidades de teste em mais de 400 %.

Alguns ambientes podem executar o host do Docker dentro de outro host do Docker (Docker-in-Docker) em ambientes de compilação.

O Docker pode aumentar a velocidade de um pipe IC usando Union-FileSystems e Copy-on-Write (COW).

Para obter velocidades maiores para fornecer software de Entrega Contínua (CD), muitas técnicas do Docker podem ser usadas.

JS

O JS permite a criação fácil de aplicativos de rede rápidos e escalonáveis.

As plataformas JS possuem bibliotecas que permitem que os aplicativos sirvam como servidores Web sem a necessidade de softwares como o Apache-HTTP-Server ou o Microsoft-IIS.

Sem precisar fazer com que os servidores da Web encurtem as listas de tarefas do DevOps e otimizem o fluxo de código do desenvolvimento para a implantação. Além disso, o JavaScript pode ser usado tanto no cliente quanto no servidor.

Os desenvolvedores encontram uma vantagem de eficiência devido à reutilização do código. Os aplicativos NODE.JS podem ser executados no tempo de execução NODE.JS no Microsoft Windows, OSX, Linux e FreeBSD.

Ele foi projetado para operação rápida com ótima experiência do usuário. Um modelo de E/S sem bloqueio controlado por eventos no NODE.JS permite aplicativos leves que são altamente eficientes.

Muitas das principais organizações de pensamento utilizam o NODE.JS, como SAP, Groupon, LinkedIn, PayPal e Wal-Mart.

Chef

As ferramentas Chef DevOps fornecem estruturas para automatizar e gerenciar a infraestrutura por meio de DSL simples.

O ativo real é um código que traz os servidores e serviços que fornecem vida.

O Chef coloca uma confiança em suas definições reutilizáveis, chamadas de livros de receitas e receitas escritas em Ruby.

Esse nível de abstração aumenta a produtividade e permite que os desenvolvedores criem facilmente configurações personalizadas para aplicativos. As receitas são executadas independentemente usando o CHEF SOLO ou um servidor que tenha o CHEF SERVER que atua como hubs para os dados de configuração.

Os servidores armazenam livros de receitas ou políticas aplicadas aos nós e aos metadados que descrevem o nó registrado gerenciado pelo chef-client.

O Chef é uma ferramenta de gerenciamento multiplataforma para Windows, Linux, Mac OS, etc., e pode ser integrado com os principais provedores de cloud.

Jenkins

Esse é um mecanismo para integração extensível e contínua que é usado como uma das principais ferramentas dos engenheiros de DevOps que desejam monitorar execuções de trabalho repetidas.

Com o Jenkins, o engenheiro de DevOps acha mais fácil integrar as mudanças nos projetos e pode acessar facilmente as saídas quando descobrem que algo está errado.

As principais características são seus links permanentes, a integração de RSS / email / mensagens instantâneas, rotulagem pós-evento, seu relatório de teste JUnit / TestNG e as compilações distribuídas.

Puppet

O Puppet permite lidar com o gerenciamento da configuração e do software durante a realização de mudanças rápidas e repetitivas.

O Puppet impõe automaticamente consistência em ambientes e pode trabalhar em máquinas físicas e virtuais.

Ele possui cadeias de ferramentas comuns e suporta as principais práticas recomendadas no DevOps que incluem entrega contínua.

O Puppet Enterprise oferece a orquestração de data centers, automatizando a configuração e o gerenciamento de máquinas e software.

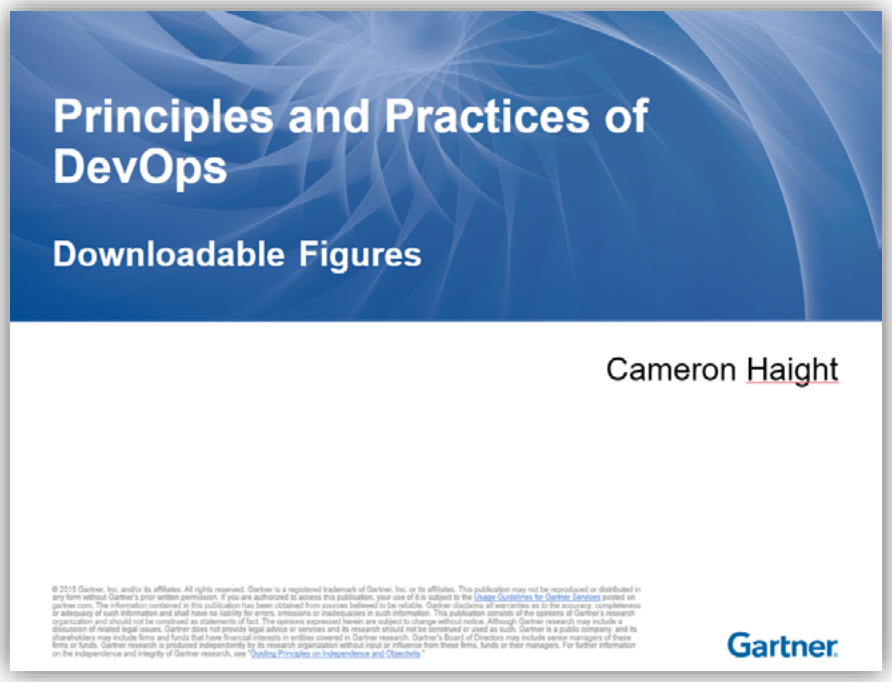
Há também versões de software livre do Puppet que foram usadas pela Universidade de Stanford para preencher lacunas no desenvolvimento de software para o serviço de biblioteca digital e administração do sistema, necessárias para manter os serviços funcionando com segurança.

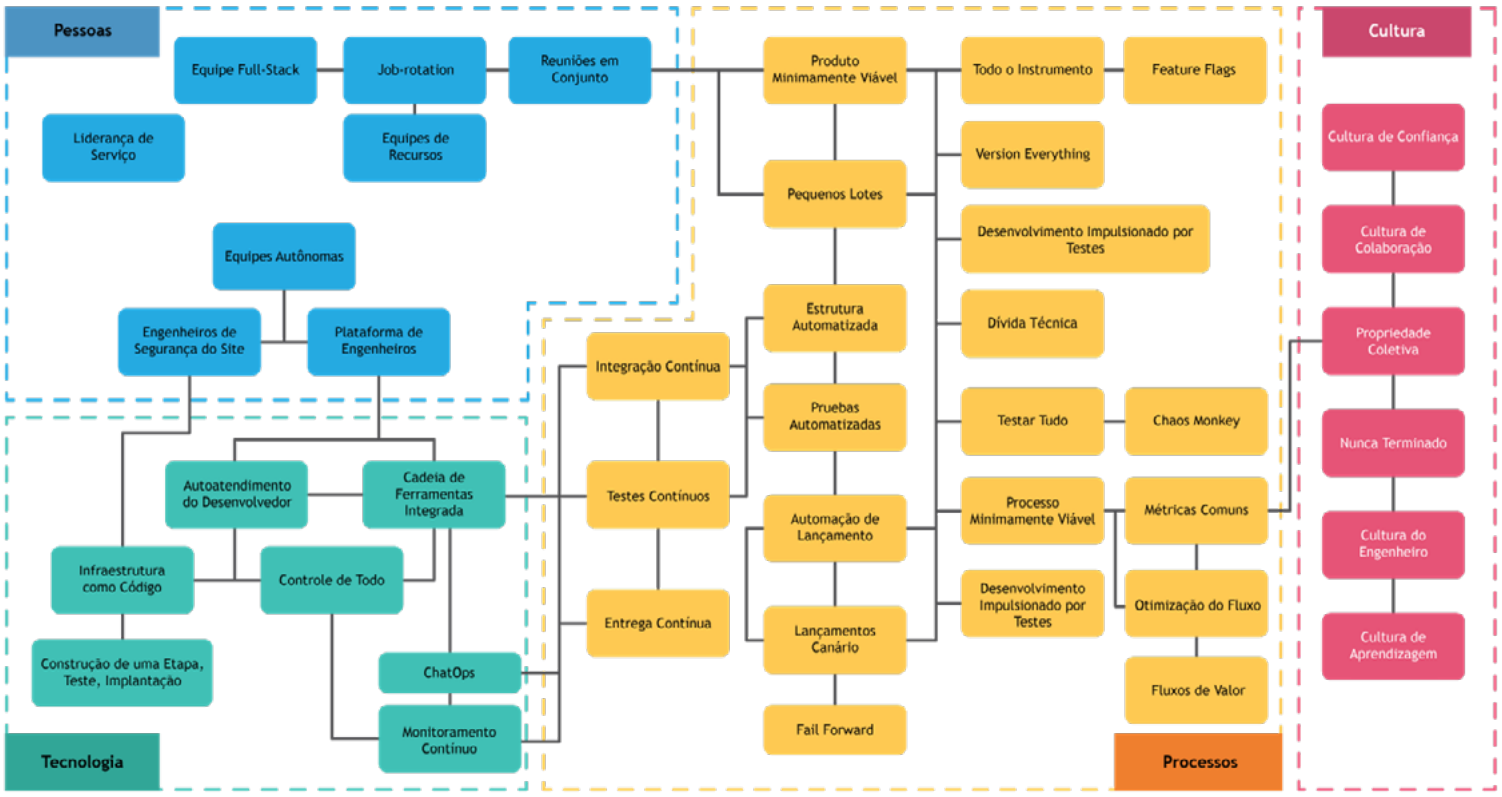
P.P.T e Cultura

O próximo slide apresenta o modelo de DevOps proposto pelo Gartner Group.

Os candidatos ao Certificado Profissional de DevOps Essentials (DEPC) devem entender os componentes do modelo.

www.gartner.com





Modelo de Maturidade DevOps

Esse modelo examina o DevOps de três pontos de vista: processos, automação e colaboração. E abrange uma série de estados claramente definidos no caminho para um ambiente de DevOps otimizado.

A exploração de maturidade do DevOps dá a você uma compreensão do nível de maturidade de sua organização em termos de padronização de processos, ferramentas de automação e abordagens colaborativas, juntamente com insights sobre suas oportunidades de melhoria.

Checklist - DevOps

Os checklists DevOps não são estáticos ou exclusivos e não existem manifestos descrevendo o DevOps, mas devem ser adaptadas às necessidades da organização, interações humanas e outros critérios de natureza específica.

Os checklists poderiam ajudar um procedimento com a criação de culturas de DevOps, mas eles não são considerados maneiras exclusivas de proceder com uma transformação organizacional.

A configuração da infraestrutura de DevOps para as várias necessidades de negócios de uma organização é focada em processos de continuidade de negócios e recuperação de desastres que podem afetar significativamente as empresas pelos custos do tempo de inatividade. Também é necessário entender essas necessidades de negócios para criar uma estratégia de DevOps que possa ter impactos significativos em uma organização.

Autoavaliação

Nas empresas, existe uma necessidade de avaliação contínua que é a base para o desenvolvimento escalável da empresa DevOps.

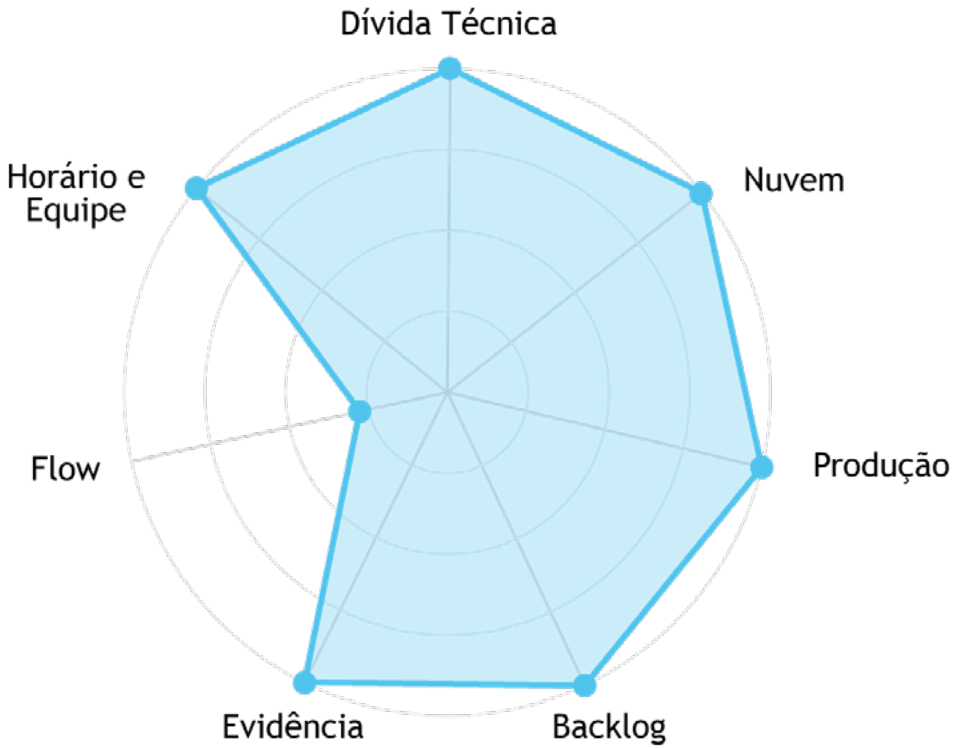
- A autoavaliação treinará as equipes de desenvolvimento e de operações para acelerar a entrega rápida e as iterações com base no feedback preciso e acionável dos produtos.

- A avaliação de resultados e impactos é uma disciplina importante e crítica que as equipes de TI devem adotar durante a implementação do DevOps. Esse processo deve ser abrangente, automático e contínuo.

- Sem avaliação contínua em um kit de ferramentas de DevOps, existe o risco de ficar cego e perder grandes oportunidades que podem estar facilmente ao alcance.

- Essas informações serão cruciais quando os líderes de negócios e de TI priorizarem seus investimentos futuros e fizerem concessões que possam surgir.

O resultado dessa autoavaliação deve ser usado para ajudar a otimizar suas práticas e ferramentas de DevOps. Ao decidir o próximo passo para a sua jornada através do DevOps, você deve considerar o mercado competitivo, a cultura organizacional, os processos internos e as ferramentas atuais para fortalecer cada uma das áreas de atuação do DevOps.



Fonte:http://aka.ms/DevOpsMaturity
http://devopsassessment.azurewebsites.net/es-ES/

Inovações rápidas no negócio podem atuar como vantagens competitivas.

A autoavaliação ajuda a determinar e medir quais recursos do DevOps são muito valiosos e devem ser melhorados na cultura de negócios.

É caro manter e dificulta a inovação, o que desacelera a equipo de desenvolvimento, assim sendo obrigatório um processo de auto-avaliação contínuo.

DevOps Report 2017+

É importante saber como o DevOps está se comportando na indústria americana e em outros países. O DevOps Report é produzido anualmente pela Puppet e deve ser conhecido e analisado pelos interessados no Certificado Profissional DevOps Essentials da CertiProf®.

O relatório deste ano explora novas áreas:

- A influência da liderança nas transformações de DevOps.
- Como as equipes de alto e baixo desempenho automatizam de maneira diferente.
- O impacto da arquitetura e estrutura da equipe no desempenho de TI.

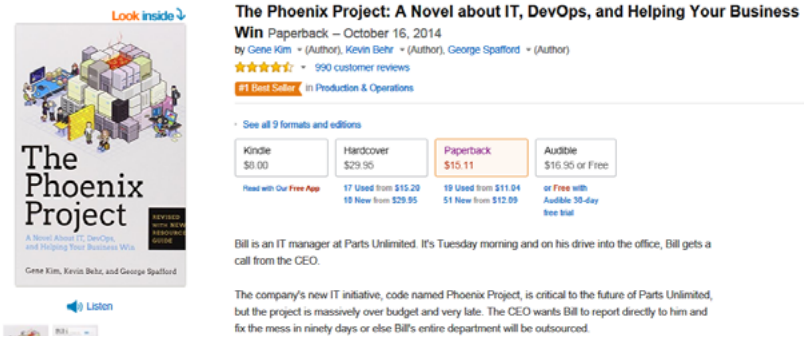
www.puppet.com

Análise do estado de DevOps da Puppet

Literatura

The Phoenix Project

Gene Kim, Kevin Behr, George Spafford.
ISBN-10: 0988262576.
ISBN-13: 978-0988262577.
IT Revolution Press (10 de Enero, 2013).



Referências

- Jens Smeds, K. N. (2015). IEEE 11th International Conference on Global Software Engineering (ICGSE). Lecture Notes in Business Information Processing Agile Processes in Software Engineering and Extreme Programming , 166-177.
- Kort, W. D. (2016). Dashboards and Reporting. DevOps on the Microsoft Stack , 77-96.
- Kort, W. d. (2016). Integrating Testers into DevOps. DevOps on the Microsoft Stack , 205-229.
- Noureddin Kerzazi, B. A. (2016). Who needs release and devops engineers, and why? Proceedings of the International Workshop on Continuous Software Evolution and Delivery - CSED '16.
- Stephen Jones, J. N. (2016). Management challenges for DevOps adoption within UK SMEs. Proceedings of the 2nd International Workshop on Quality-Aware DevOps - QUDOS.

DEVOPS ESSENTIALS PROFESSIONAL CERTIFICATE (DEPC)

CertiProf®

Professional Knowledge

certiprof.com @Certiprof @CertiProf CertiProf Certiprof_llc

www.certiprof.com