Fundamentos do DevOps: Transformando a Tecnologia da Informação

Abril, 2024

Formação acadêmica

Universidade de Fortaleza - UNIFOR
MESTRADO EM INFORMÁTICA APLICADA, Engenharia de Sistemas / Blockchain
2015 – 2017

Centro Universitário Farias Brito
MBA EM GERÊNCIA DE PROJETOS DE TI, Gestão de Projetos
2012 – 2014

Centro Universitário Farias Brito
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, Ciência da Computação
2006 – 2010

Contatos:
WHATSAPP: 85988724906
LINKEDIN
https://www.linkedin.com/in/samantha-almeida-713b639/

Empresas de Atuação:

PagueMenos, 3corações, IVIA, VTI, DATAPREV, M. Dias Branco Tradição & Qualidade, BRQ Digital Solutions, Atlântico, cast, indra, UNIFOR, TOTVS

Atividades Avaliativas

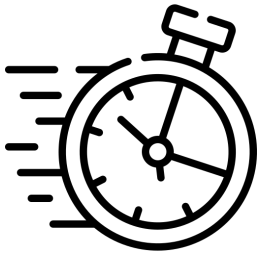
- Encontro 1:
  - Atividade 1: Resolução de questões para fixação teórica. (Em sala)
  - Atividade 2: Diagnóstico DevOps

- Encontro 2:
  - Atividade 4: Exercício de Métricas DevOps. (Em sala)
  - Atividade 5: Prática Devops.

O que é DevOps?

Literatura de Referência

Engenharia de Software



\+ Rápido

\+ Barato

Melhor

\+ Seguro

Referência de Artigo:
https://jrsinclair.com/articles/2017/faster-better-cheaper-art-of-making-software/

Apagão na TI

1. No Brasil, **53 mil** profissionais irão se graduar anualmente até 2025, enquanto a demanda projetada é de **800 mil** novos talentos, apontam estudos da Brasscom. O resultado é um **déficit de 530 mil** profissionais, evidenciando ainda mais o desequilíbrio entre oferta de trabalho e talentos disponíveis.

2. Mercado cresce mais de 20% ao ano segundo a Brasscom.

3. Faltam profissionais qualificados. (Automação e Inteligência Artificial)

O que é DevOps?

Definição de DevOps

Representa a união entre as equipes de desenvolvimento e operações, visando aprimorar eficiência, qualidade e velocidade na entrega de software.

Filosofia e Objetivos

A quebra de barreiras tradicionais entre desenvolvimento e operações, enfatizando a comunicação, colaboração, integração e automação.

Mudança Cultural

Promove a partilha de responsabilidades e a colaboração como práticas padrão.

O que é DevOps?

1. DevOps é uma abordagem de desenvolvimento de software que combina desenvolvimento e operações para melhorar a colaboração e a eficiência entre as equipes.

2. O principal objetivo do DevOps é automatizar e integrar o processo de desenvolvimento e implantação de software, permitindo entregas mais rápidas e confiáveis de aplicativos.

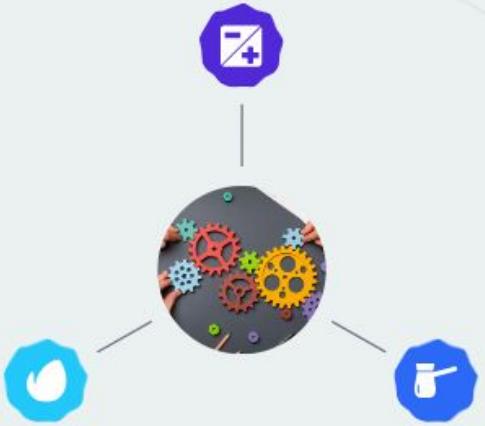
3. Os benefícios do DevOps incluem ciclos de desenvolvimento mais curtos, maior confiabilidade e estabilidade do software, resolução mais rápida de problemas e melhor colaboração entre equipes.

Benefícios do DevOps: Ciclos de desenvolvimento mais curtos e colaboração

A adoção do DevOps traz uma série de benefícios, incluindo ciclos de desenvolvimento mais curtos e colaboração efetiva entre equipes.

Entrega Rápida

Permite a entrega contínua de software, reduzindo o tempo de lançamento de novas funcionalidades.



Maior Qualidade

Contribui para a melhoria da qualidade do software por meio de automação e feedback contínuo.

Maior Colaboração

Facilita a colaboração entre equipes, promovendo um ambiente de trabalho integrado e eficiente.

Pilares Principais do DevOps

Integração Contínua (CI): Automatiza a compilação e teste do código cada vez que um membro da equipe faz alterações, garantindo que o código novo seja compatível com o código existente.

Entrega Contínua (CD): Permite a entrega automática do código testado para ambientes de produção ou teste, facilitando lançamentos rápidos e seguros.

Automação: Abrange desde testes até implantações, passando por monitoramento das aplicações em produção. A automação reduz os riscos associados ao erro humano e aumenta a eficiência dos processos.

Cultura Colaborativa: Encoraja uma cultura organizacional onde as barreiras entre as equipes são minimizadas. Isso inclui compartilhar responsabilidades e comunicar-se efetivamente durante todo o ciclo de vida do desenvolvimento.

Princípios Fundamentais do DevOps

Colaboração e Comunicação

Este princípio enfatiza a eliminação dos silos entre as equipes, promovendo transparência e responsabilidade compartilhada.

Automação

A automação é essencial, do desenvolvimento à produção, para acelerar processos e reduzir erros, resultando em entregas mais rápidas e confiáveis.

Feedback Contínuo

O ciclo de feedback rápido permite correções ágeis e adaptações às necessidades dos usuários, melhorando continuamente a qualidade e satisfação.

Melhoria Contínua

Buscar constantemente a otimização promove uma cultura de inovação e melhoria, essencial para se manter competitivo em um mercado em evolução.

Colaboração e Comunicação

Importância da Colaboração e Comunicação

- A colaboração entre as equipes de desenvolvimento e operações, incentivada pelo DevOps, melhora a eficiência dos processos.
- A comunicação transparente e a responsabilidade compartilhada resolvem conflitos e promovem uma entrega de software mais rápida e estável.
- A quebra de barreiras organizacionais permite inovações e ajustes ágeis, alinhando os produtos às necessidades do mercado.

Automação no DevOps

Pilar da Automação no DevOps

- A automação abrange testes, integração e implantação, reduzindo erros manuais e melhorando a eficiência dos processos.
- Ferramentas como Jenkins e Ansible são essenciais para criar pipelines de CI/CD, garantindo entregas frequentes e estáveis.
- A automação libera as equipes para focarem em tarefas mais estratégicas, aumentando a satisfação do cliente.
- Automating infrastructure with tools like Docker through IaC enhances scalability and consistency.

O Ciclo de Feedback Contínuo

Importância do Feedback Contínuo

- O feedback constante entre as equipes no processo de desenvolvimento e operações permite a identificação ágil de problemas em ambientes de produção.
- Rápida resposta a feedbacks resulta em melhorias contínuas nos processos e produtos, alinhando-os mais estreitamente às necessidades dos usuários finais.
- A iteração baseada em feedback fortalece a qualidade do software, promovendo uma cultura de excelência e inovação no ambiente de TI.

A Importância da Melhoria Contínua

Princípio da Melhoria Contínua

- O objetivo é buscar constantemente formas de otimizar processos, ferramentas e produtos, aprendendo com os erros passados e implementando lições aprendidas.
- Promove uma cultura de inovação dentro da organização, permitindo competir em um mercado em rápida evolução.
- A melhoria contínua é essencial para garantir que a empresa permaneça ágil e capaz de atender às demandas e expectativas do mercado e dos clientes.

Vantagens Competitivas do DevOps

- O DevOps permite lançamentos mais rápidos e frequentes, colocando inovações no mercado em um ritmo acelerado.
- A redução de erros, facilitada pela automação e práticas DevOps, aumenta a qualidade do software e a satisfação do cliente.
- As melhorias contínuas, baseadas no feedback e na iteração rápida, alinham os produtos às necessidades dos usuários finais, impulsionando a competitividade.
- A cultura de colaboração e a mudança para processos ágeis promovidas pelo DevOps estimulam a inovação e a eficiência, conferindo às empresas uma vantagem no mercado.

A importância do DevOps na Tecnologia da Informação

Transformação da TI

O DevOps é um divisor de águas para empresas que buscam prosperar em um mercado competitivo e acelerado, promovendo uma cultura de colaboração e automação.

Impacto na Satisfação e Retenção de Talentos

O ambiente colaborativo e focado na melhoria contínua, incentivado pelo DevOps, contribui para a satisfação e retenção de profissionais, fortalecendo a cultura organizacional.

Entrega Contínua de Produtos Funcionais

O DevOps possibilita a entrega contínua de produtos funcionais e estáveis, melhorando significativamente o tempo de colocação no mercado e a satisfação do cliente.

Implementando DevOps: Desafios e Soluções

Mudança Cultural	Destacar a necessidade de uma mudança cultural para adotar a filosofia DevOps, incentivando a colaboração e a comunicação eficaz entre as equipes.
Desafios de Automação	Enfrentar a complexidade da automação de ponta a ponta, desde a integração até a entrega, e garantir a correta implementação para melhorar a eficiência e reduzir erros.
Adoção e Adaptação	Superar resistências à mudança e garantir a adesão de toda a organização aos princípios e práticas do DevOps para alcançar benefícios a longo prazo.

A Cultura DevOps

DEV x OPS


WORKED FINE IN DEV
OPS PROBLEM NOW

Cultura DevOps: Colaboração e comunicação

A Cultura DevOps promove a colaboração, comunicação e responsabilidade compartilhada entre equipes de desenvolvimento e operações.

Colaboração

Fomenta a colaboração entre equipes para o desenvolvimento, teste e implantação de software de forma ágil.

Comunicação

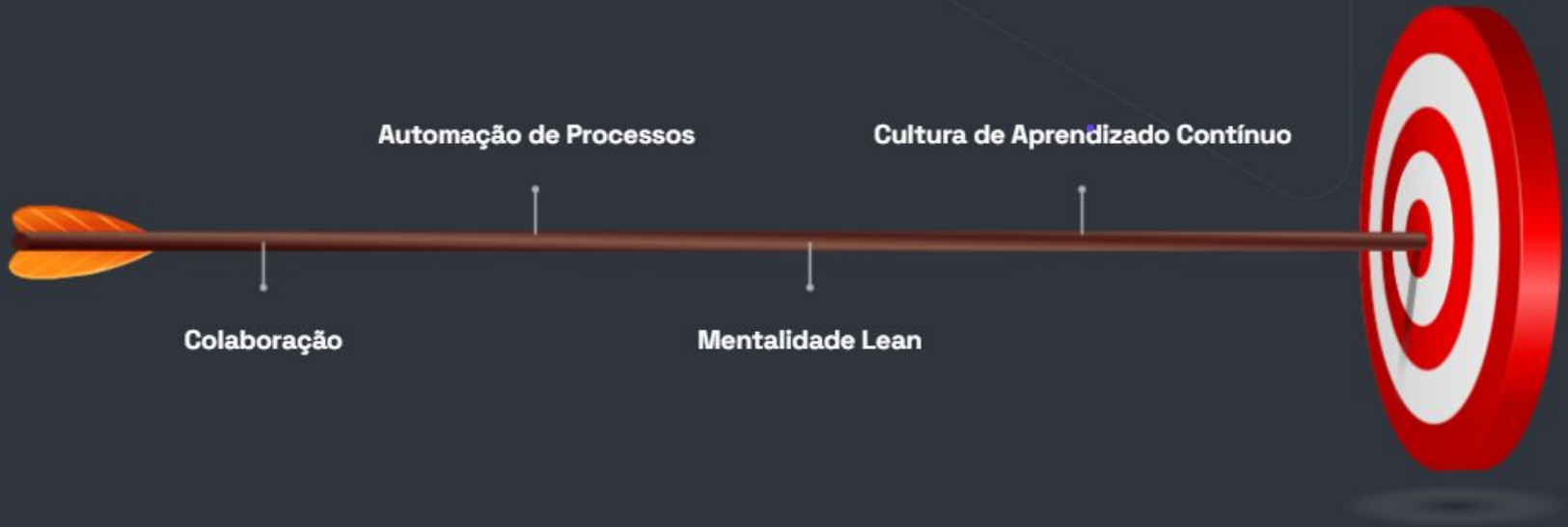
Estimula a comunicação efetiva para alinhar objetivos e garantir a entrega de software de alta qualidade.

Responsabilidade Compartilhada

Enfatiza a responsabilidade coletiva pela qualidade e desempenho do software em produção.

DevOps: Uma abordagem de desenvolvimento de software

A metodologia DevOps promove a integração e colaboração entre as equipes de desenvolvimento e operações para melhorar a entrega de software.



Pilares da Cultura DevOps

Colaboração Intensiva

Promove um ambiente de colaboração intensiva, quebra de barreiras e responsabilidade compartilhada.

Automação de Processos

Foco na automação de CI/CD, testes e IaC para reduzir erros, aumentar eficiência e consistência.

Mentalidade Lean

Adoção de uma abordagem Lean para minimizar desperdícios e melhorar processos continuamente.

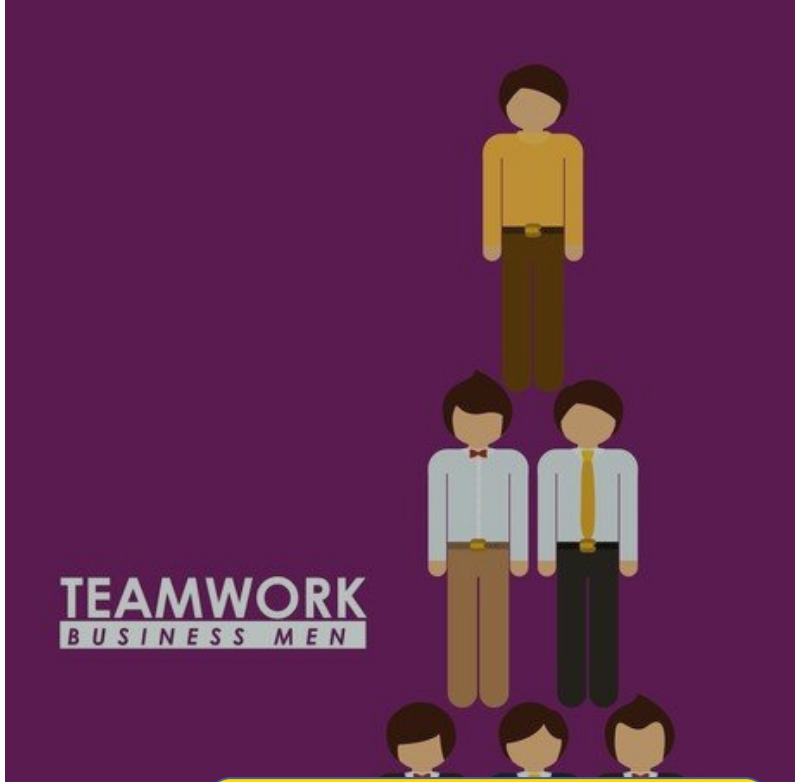
Cultura de Aprendizado Contínuo

Estímulo a experimentação, aprendizado com falhas e sucessos, e inovação constante nas práticas e ferramentas.

A Importância da Colaboração

Colaboração como Espinha Dorsal

- A colaboração no contexto DevOps implica em compartilhar responsabilidades, desafios e sucessos de maneira transparente, promovendo uma mudança significativa na mentalidade organizacional.
- Implica em substituir silos departamentais por equipes multidisciplinares focadas em objetivos comuns, iniciando com a liderança promovendo ativamente uma cultura de abertura.
- Adotar ferramentas e práticas que fomentem a comunicação contínua e o compartilhamento de conhecimento, como as de integração contínua e entrega contínua, facilita a colaboração efetiva.



Plus tip:

Incentive a colaboração ativa, destacando a importância do compartilhamento de responsabilidades e sucessos em toda a equipe.

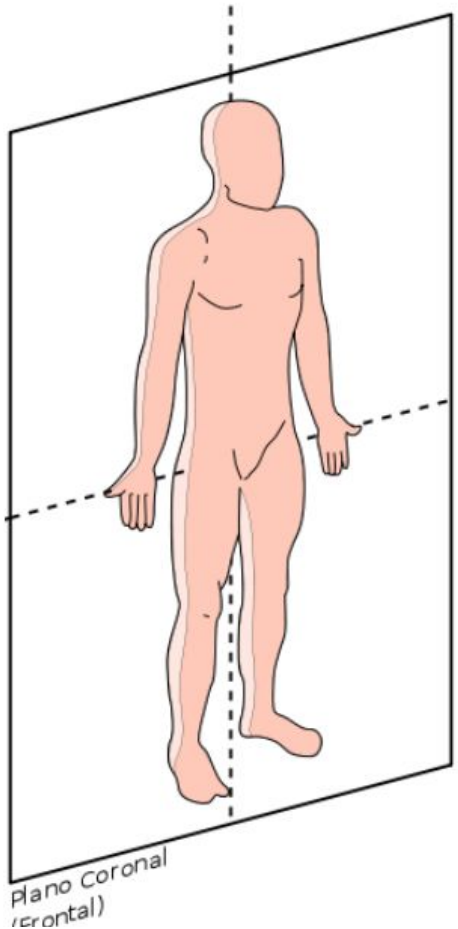
Dinâmica Partes do Corpo

1 - Organizem-se em equipes e escolham uma parte do corpo a ser construída.

2 - Cada equipe deverá selecionar uma parte do corpo humano para construir.

Opções: Cabeça, Tronco com Braços e Par de pernas.

3 - Ao término da construção cada equipe deverá apresentar seus resultados



Estratégias para Implementação

01

Estabelecer canais eficientes de comunicação para promover a transparência e o alinhamento entre as equipes.

02

Selecionar **ferramentas que suportem práticas como CI/CD e IaC,** essenciais para a automação e colaboração eficazes.

03

Fomentar **a cultura de melhoria contínua,** incentivando a busca por otimizações nos processos existentes.

04

Reforçar comportamentos desejados ao reconhecer e **celebrar as conquistas** alcançadas através da abordagem DevOps.

Mudança na Mentalidade Organizacional

Substituindo Silos por Equipes Multidisciplinares

- Promover a colaboração requer a transição de estruturas departamentais isoladas para equipes que compartilham responsabilidades e objetivos.
- A mudança para uma abordagem de equipes multidisciplinares facilita a inovação e a resolução de problemas complexos de forma mais eficaz.
- Equipes multidisciplinares têm uma visão mais ampla do processo de desenvolvimento, promovendo uma maior sinergia e eficiência.

Promoção de uma Cultura de Abertura pela Liderança

- A liderança deve fomentar um ambiente de confiança e abertura, onde o feedback é valorizado e a comunicação é transparente.
- Uma cultura de abertura estimula a criatividade e o engajamento, essenciais para o sucesso da colaboração dentro do modelo DevOps.
- Líderes que demonstram vulnerabilidade e receptividade ao feedback incentivam a inovação e o aprimoramento contínuo.

Construindo uma Cultura Inclusiva

Estabelecer um ambiente de confiança que incentive a diversidade de pensamento e opiniões, essencial para a inovação e crescimento das equipes.

Promover sessões de treinamento regulares e compartilhamento de conhecimento para fomentar uma cultura de aprendizado constante .

Facilitar workshops interativos e atividades de treinamento cruzado que e**stimulem a colaboração e a resolução de problemas** em equipe.

Incentivar a participação de todos os membros da equipe, valorizando a diversidade de experiências e perspectivas para enriquecer o ambiente de trabalho.

I**mplementar práticas de feedback construtivo** para apoiar o desenvolvimento profissional e o crescimento pessoal dos colaboradores.

Reforçar os valores de inclusão e aprendizado através de exemplos e histórias que promovam uma cultura organizacional coesa e acolhedora.

Desafios e Benefícios

+ Benefícios

- Maior velocidade de entrega de software, impulsionando a eficiência e competitividade no mercado.
- Melhoria da qualidade do produto e estabilidade dos ambientes de produção, resultando em maior satisfação do cliente.
- Promoção de um ambiente de trabalho mais integrado, colaborativo e satisfatório para as equipes.

× Desafios

- A resistência às mudanças nas práticas estabelecidas pode ser um obstáculo significativo para a adoção da cultura DevOps.
- A transformação cultural profunda exigida para implementar DevOps com sucesso pode ser um processo desafiador e demorado.
- Enfrentar a necessidade de educação e capacitação das equipes em novas ferramentas e práticas, demandando tempo e recursos.

Exercício de rápido de maturidade DEVOPs

1. https://www.devops-research.com/quickcheck.html
2.
3.

Ciclo de Desenvolvimento DevOps - Entendendo o Pipeline de Entrega

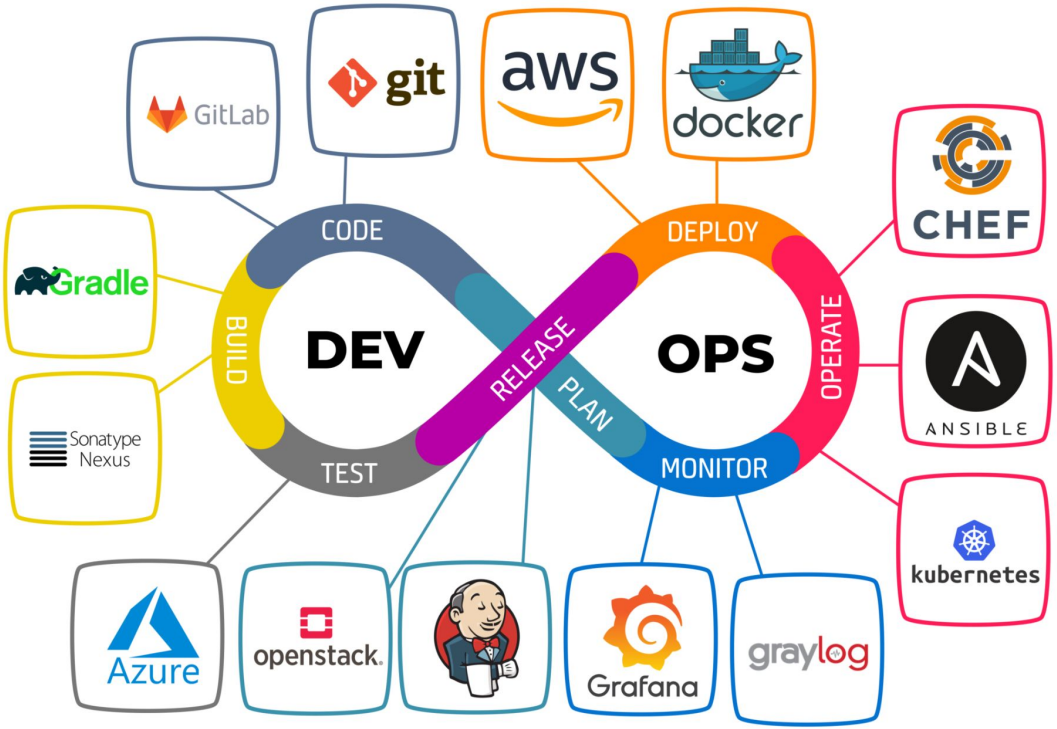
Introdução ao Ciclo de DevOps

O ciclo de DevOps descreve o processo contínuo de desenvolvimento, integração, entrega e operação de software.

Enfatiza a colaboração entre equipes de desenvolvimento (Dev) e operações (Ops) para garantir uma entrega rápida, confiável e contínua de software.

Visa a entrega rápida e eficiente de software de alta qualidade, promovendo a automação e a cultura de feedback contínuo.

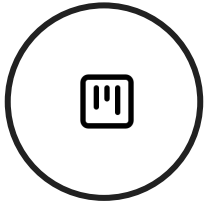
Introdução ao Ciclo de DevOps



Planejamento

Estratégias de Planejamento

Define requisitos, prioridades e metas do projeto. Utiliza Kanban, metodologias ágeis e reuniões de planejamento.



Ferramentas de Planejamento

Facilita o processo de planejamento. Comum no DevOps para envolver as partes interessadas.

Colaboração

Promove a colaboração entre equipes. Fundamental para o sucesso do ciclo de DevOps.

Desenvolvimento

Escrita e Revisão de Código

Os desenvolvedores escrevem e revisam o código-fonte para implementar novas funcionalidades ou corrigir problemas existentes.

Integração Contínua

A prática de integração contínua (CI) é fundamental, permitindo que as alterações de código sejam mescladas regularmente e verificadas por testes automatizados.

Implantação Automatizada

Garante que o código aprovado seja automaticamente implantado em ambientes de teste para validação adicional, seguindo o princípio de entrega contínua (CD).

Testes e Integração

Testes Abrangentes

O código desenvolvido é testado em vários níveis, incluindo testes unitários, testes de integração e testes de aceitação do usuário (UAT).

Entrega Contínua

A prática de entrega contínua (CD) garante que o código aprovado seja automaticamente implantado em ambientes de teste para validação adicional.

Implantação Eficiente

Após a conclusão bem-sucedida dos testes, o código é implantado em ambientes de produção, garantindo uma implantação rápida, consistente e livre de erros.

Implantação

Processo de Implantação

Após a conclusão dos testes, o software é implantado em ambientes de produção. A automação da implantação garante rapidez e consistência.

Transição de Código

A fase de implantação requer uma transição suave do código testado para os servidores de produção. A automação desempenha um papel crucial na garantia de uma implantação sem erros.

Preparação da Infraestrutura

É essencial garantir que a infraestrutura de produção esteja preparada para receber o novo software. A automação ajuda a verificar e configurar a infraestrutura conforme necessário.

Operações e Monitoramento

Monitoramento em Tempo Real

Após a implantação, o software é monitorado em tempo real para garantir sua estabilidade e desempenho contínuo.

Coleta e Análise de Métricas

Métricas de desempenho, logs e eventos são essenciais para identificar e resolver problemas, garantindo a estabilidade do software.

Automatização e Colaboração

A prática de DevOps busca promover a automação e a colaboração, facilitando a detecção e resolução de problemas para garantir a estabilidade e performance do software.

Feedback e Melhoria

Feedback Contínuo

O ciclo de DevOps é iterativo e baseado em feedback contínuo. Os dados coletados são usados para identificar áreas de melhoria e priorizar futuras iterações do produto.

Melhoria Iterativa

As equipes buscam constantemente otimizar e aprimorar o ciclo de desenvolvimento e entrega de software, promovendo uma cultura de aprendizado contínuo e melhoria.

Papel do Feedback

O feedback dos usuários é essencial para impulsionar as melhorias no processo de desenvolvimento e entrega de software.

Automação no DevOps

A automação desempenha um papel fundamental na eficiência do ciclo de DevOps, permitindo a ráp entrega de software com qualidade.

Ferramentas de automação como Chef e Puppet facilitam a configuração e gerenciamento de infraestrutura complexa.

A automação de testes e implantações garante a consistência e ajuda a identificar e corrigir rapidamente possíveis falhas no processo.

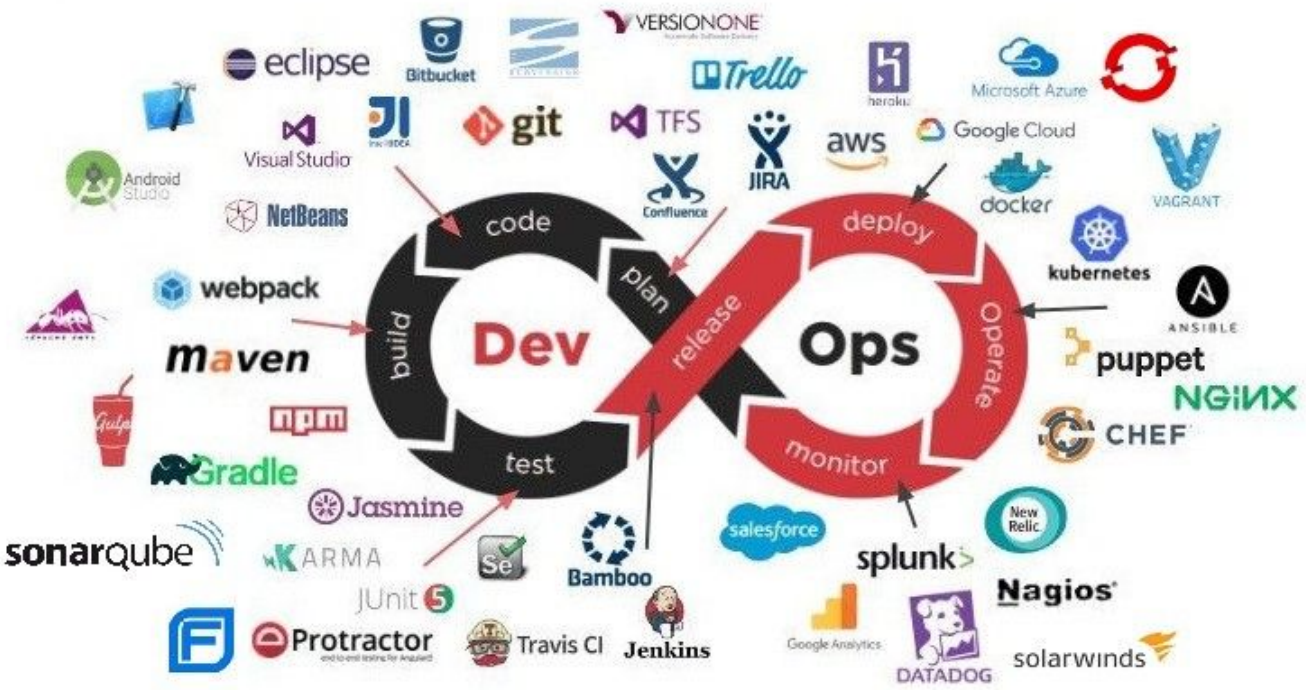
Pontos Relevantes

- O ciclo de DevOps, do planejamento ao monitoramento, representa a jornada contínua de desenvolvimento, entrega e operação de software, promovendo colaboração, automação e feedback contínuo.

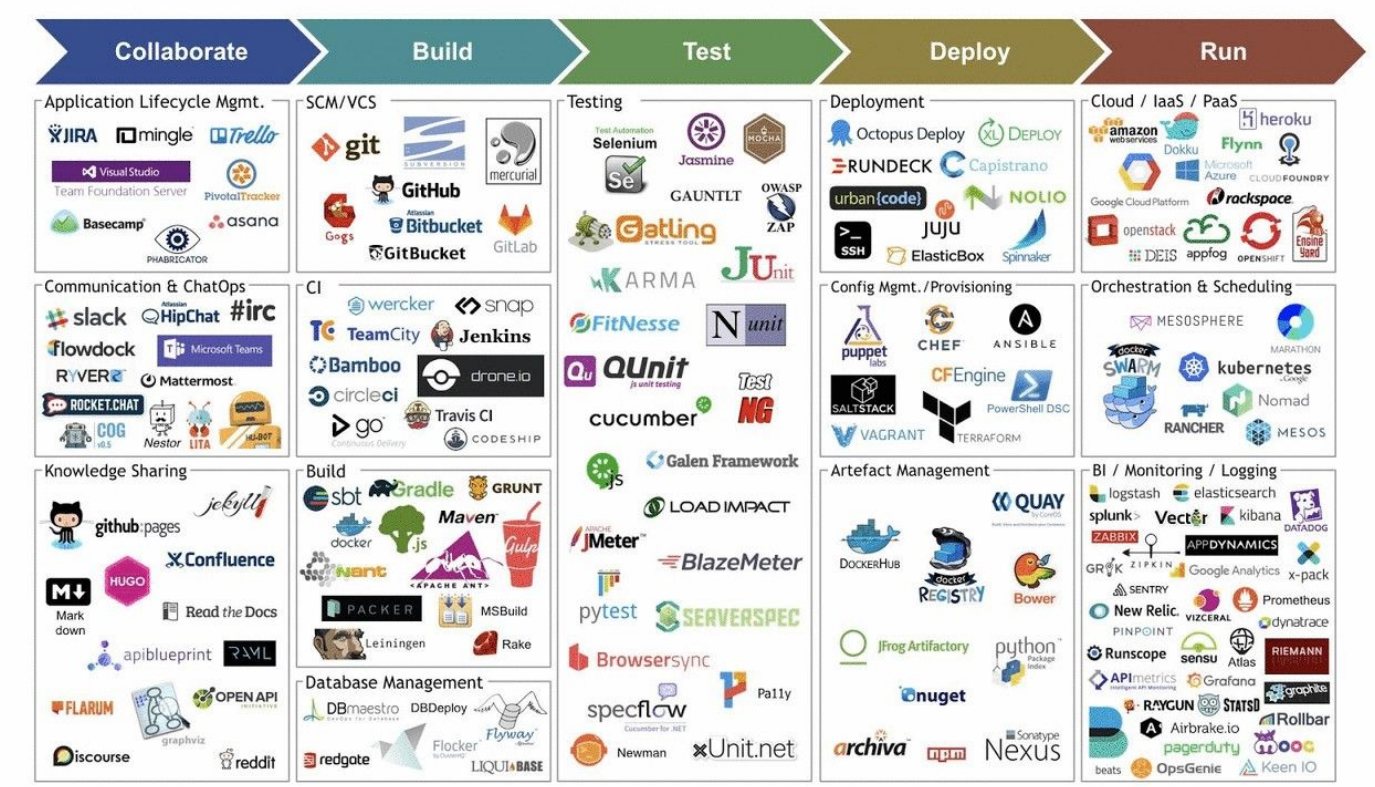
- A abordagem holística do DevOps visa acelerar a entrega de software de alta qualidade, mantendo eficiência e confiabilidade.

- A prática de DevOps busca aprimorar constantemente o ciclo de desenvolvimento e entrega de software, promovendo uma cultura de aprendizado contínuo e melhoria.

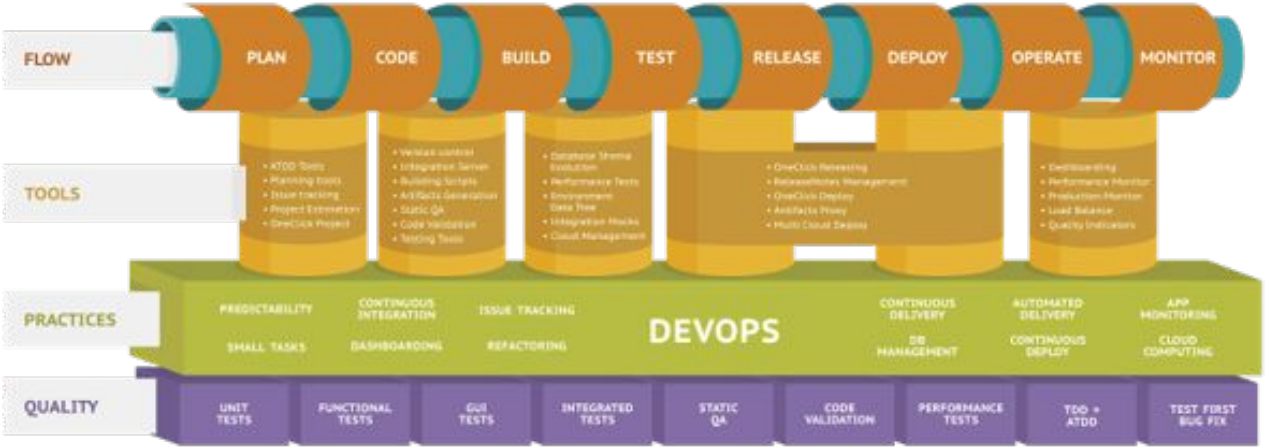
Ferramentas DevOps



Ferramentas DevOps



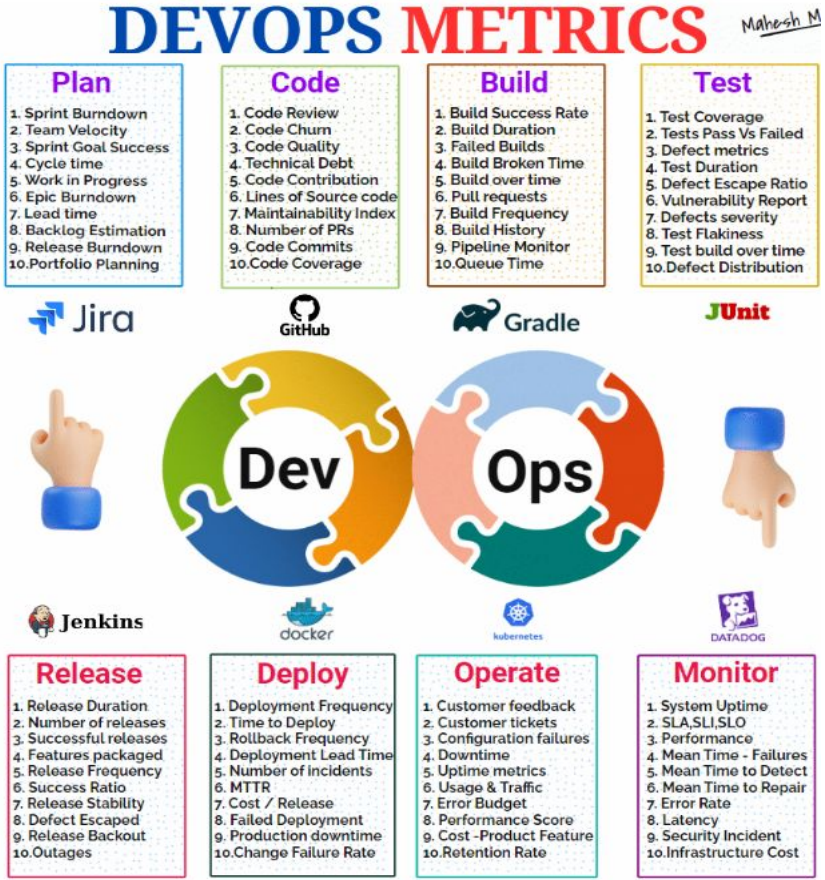
Maturidade DevOps



https://www.objective.com.br/insights/devops-e-testes-automatizados/

	Inicial	Gerenciado	Definido	Gerenciado Quantitativamente	Otimizando
Cultura e Organização	• Times organizados baseados na plataforma/tecnologia • Processos definidos e documentados	• Um backlog por time • Adota metodologias ágeis • Remove os limites do time	• Time com alta colaboração • Remove limites entre dev e ops • Processo comum para todas as mudanças	• Melhoria continua entre times • Times responsáveis por todo o caminho até produção	• Times funcionais cruzados
Build e Deploy	• Controle de versionamento centralizado • Scripts de build automatizado • Sem gerenciamento de artefatos • Deploy manual • Provisionamentos de ambientes manual	• Builds agrupados • Qualquer build pode ser recriado dos fontes • Gerenciamento de artefatos de build • Scripts de deploy automatizados • Provisionamentos de ambientes automatizados	• Gatilhos de build no commit • Bulid é interrompido se qualidade não for cumprida (analise de código, performance, etc.) • Deploy em um clique e implantação de qualquer artefato pronto em qualquer ambiente • Processo de deploy padronizado em todos os ambientes	• Prioridade do time é manter uma versão estável ao invés de nova trabalho. • Builds não são deixados quebrados • Deploy orquestrado • Blue-Green deploy	• Deploy Continuo sem necessidade de ação manual
Release	• Release sem frequência definida e sem confiança • Processo manual de release	• Release trabalhoso e sem frequência porém confiável	• Release sem frequência mas automatizado e confiável em qualquer ambiente	• Release frequente e todo automatizado • Deploy desconectado do release • Releases para subconjunto de usuários incrementando até disponibilizar para todos	• Sem rollback, sempre fazendo novo deploy
Gerenciamento de dados	• Migração de dados são feitas manualmente e sem scripts	• Migração de dados usando scripts versionados, feitos manualmente	• Mudanças em bases de dados feitas automatizadas e versionadas	• Mudanças em bases de dados feitas como parte do processo de deploy	• Mudanças em bases de dados feitas em qualquer ambiente de forma automática e com rollback testado
Testes e verificações	• Testes unitários automatizados • Ambiente de testes separado	• Testes integrados automatizados • Analise estática de código • Analise de cobertura de código	• Testes funcionais automatizados • Testes de segurança e de performance feitos manualmente	• Testes de aceitação todos automatizados • Testes de segurança e de performance automatizados • Testes manuais exploratórios baseados em analise de risco	• Verificação de expectativa de valor do negocio • Defeitos identificados e corrigidos imediatamente
Relatórios e Informações	• Métricas de referencia do processo • Relatórios manuais • Visível para informar execução	• Medidas de processo • Relatórios automáticos • Visível pelo time	• Geração automática de release notes • Rastreabilidade do pipeline • Histórico de relatórios • Visível pelo time	• Relatórios com analise de tendência • Gráficos em tempo real com métricas do pipeline	• Informações dinamicamente disponíveis • Gráficos customizáveis • Informações ultrapassando limites da organização

Ciclo DevOps Métricas



https://www.linkedin.com/feed/update/urn:li:activity:7167440031345831937/?utm_source=share&utm_medium=member_android

Ciclo DevOps Métricas Fundamentais

1. Tempo de espera para mudanças
2. Taxa de falha em alterações
3. Frequência de implementação (Janela)
4. Tempo médio de recuperação

Ciclo DevOps Métricas Adicionais

1. NPS
2. Tempo médio para restaurar o serviço (TMPRS)
3. Rotatividade da equipe
4. Taxa de erros

Práticas e Ferramentas do DevOps

Práticas e ferramentas do DevOps

O DevOps adota diversas práticas e ferramentas para automatizar processos e melhorar a colaboração entre equipes.

Microserviços

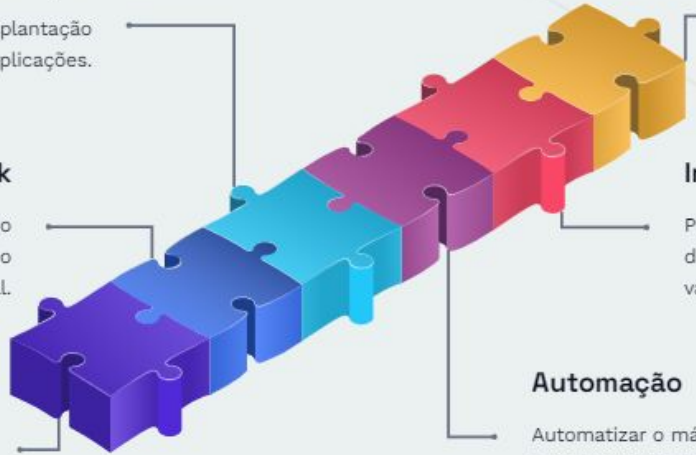
Adota arquitetura de microserviços para facilitar a implantação e manutenção de aplicações.

Monitoramento e Feedback

Utiliza ferramentas para monitorar o desempenho do software em produção e obter feedback em tempo real.

Infraestrutura como Código

Automatiza a gestão de infraestrutura por meio de scripts e configurações versionadas.



Entrega Contínua (CD)

Estender a integração contínua, automatizando também a entrega do software em ambientes de teste e produção de forma incremental e confiável.

Integração Contínua (CI)

Prática que envolve integrar o código produzido por diferentes membros da equipe várias vezes ao dia, validando cada integração com testes automatizados.

Automação

Automatizar o máximo possível do processo de desenvolvimento, teste e implantação de software para reduzir erros humanos e acelerar o tempo de entrega.

Automação: A chave para o sucesso do DevOps

A automação de processos é um pilar fundamental do DevOps, permitindo a entrega rápida e confiável de software.

Introdução à Automação no Desenvolvimento

- Aceleração da entrega de software.
- Redução de erros no desenvolvimento.
- Aumento da eficiência no processo de desenvolvimento.

Compilação e Integração Contínua (CI)

Automação desde a fase de compilação e integração do código fonte.

Uso de ferramentas como Jenkins, Travis CI e CircleCI.

Automatização da compilação do código e execução de testes.

Integração de diferentes partes do código produzido por diferentes membros da equipe.

Testes Automatizados

1

Automação de testes é essencial no processo de desenvolvimento, abrangendo testes unitários, de integração e de aceitação do usuário (UAT).

2

Ferramentas como JUnit, Selenium e pytest são comumente utilizadas para automatizar a execução desses testes.

3

Automatizar os testes permite uma detecção rápida de erros e uma validação contínua do software em desenvolvimento.

4

A automação de testes contribui significativamente para a qualidade do software e para a eficiência do desenvolvimento.

Entrega Contínua (CD)

1

A automação na entrega contínua permite a implantação automatizada do software em ambientes diversos.

2

Ferramentas como Docker, Kubernetes e Ansible são utilizadas para automatizar o processo de implantação.

3

A prática de Entrega Contínua agiliza a disponibilização do software em ambientes de teste e produção de forma consistente.

4

Automatizar a entrega do software traz benefícios como redução de erros e padronização nos processos.

Implantação e Orquestração de Infraestrutura

1. Automação na criação de ambientes de desenvolvimento, teste e produção.

2. Configuração automatizada dos recursos de infraestrutura conforme especificações definidas.

3. Gerenciamento automatizado de infraestrutura para garantir consistência e escalabilidade.

3. Ferramentas como Terraform e AWS CloudFormation simplificam e agilizam o processo de implantação e orquestração.

Monitoramento e Logging Automatizados

1

Automatizar o monitoramento do software em produção é essencial para garantir a visibilidade contínua do desempenho e a pronta detecção de problemas.

2

A coleta automatizada de logs é crucial para analisar eventos, identificar falhas e realizar ações corretivas de forma eficiente.

3

Ferramentas como Prometheus, Grafana e ELK Stack (Elasticsearch, Logstash, Kibana) são amplamente utilizadas para automatizar o monitoramento e logging.

4

A integração dessas ferramentas permite a configuração de alertas, visualização de métricas e análise de logs de forma automatizada.

Benefícios da Automação

1

Redução do tempo gasto em tarefas manuais repetitivas, permitindo foco em atividades mais estratégicas.

2

Minimização de erros humanos, resultando em software mais confiável e estável.

3

Aumento da qualidade do software devido à execução consistente de testes automatizados.

Desafios da Automação

1

Implementar a automação no processo de desenvolvimento requer habilidades específicas que nem todas as equipes possuem.

2

A complexidade da configuração das ferramentas de automação pode ser um obstáculo para a sua implementação eficaz.

3

A resistência à mudança e a adaptação a novos processos automatizados podem ser desafios culturais a serem superados.

Integração Contínua

Prática que envolve integrar o código produzido por diferentes membros da equipe várias vezes ao dia, validando cada integração com testes automatizados.

Integração Contínua: Integrando o trabalho da equipe

A Integração Contínua envolve a automação de testes e compilações, garantindo a integração suave do código produzido pela equipe.

Compilação Automatizada

Realiza a compilação automatizada do código fonte para identificar erros de integração.

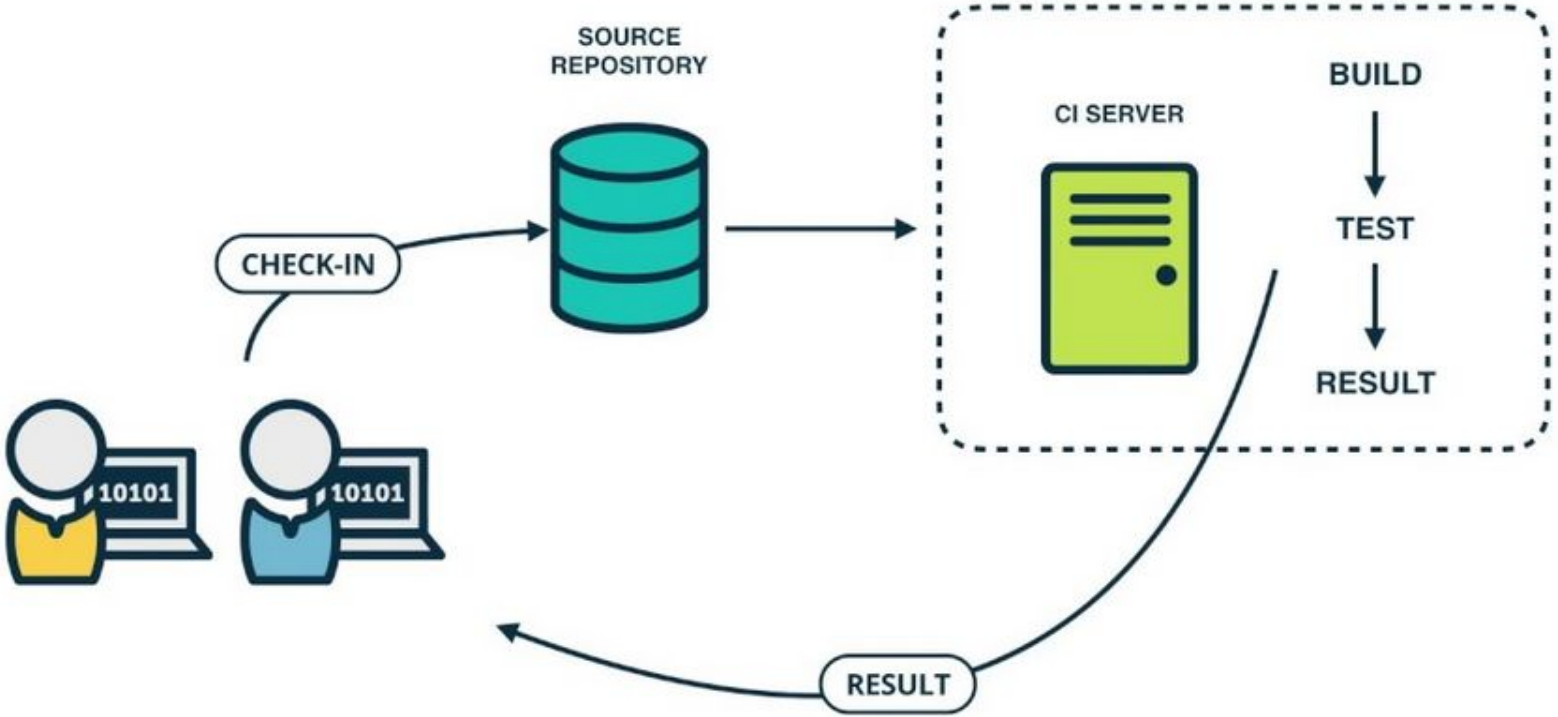
Feedback Contínuo

Fornece feedback imediato aos desenvolvedores sobre a qualidade do código integrado.

Automatização de Testes

Automatiza a execução de testes para verificar a integridade do código produzido.

Integração Contínua (CI)



Integração Contínua (CI)

A Integração Contínua (CI) é uma prática de desenvolvimento de software na qual os membros da equipe integram seu trabalho frequentemente, verificando cada integração com compilação automatizada e testes automatizados.

As etapas-chave da CI incluem commit frequente, build automatizada, testes automatizados e feedback imediato para corrigir problemas rapidamente.

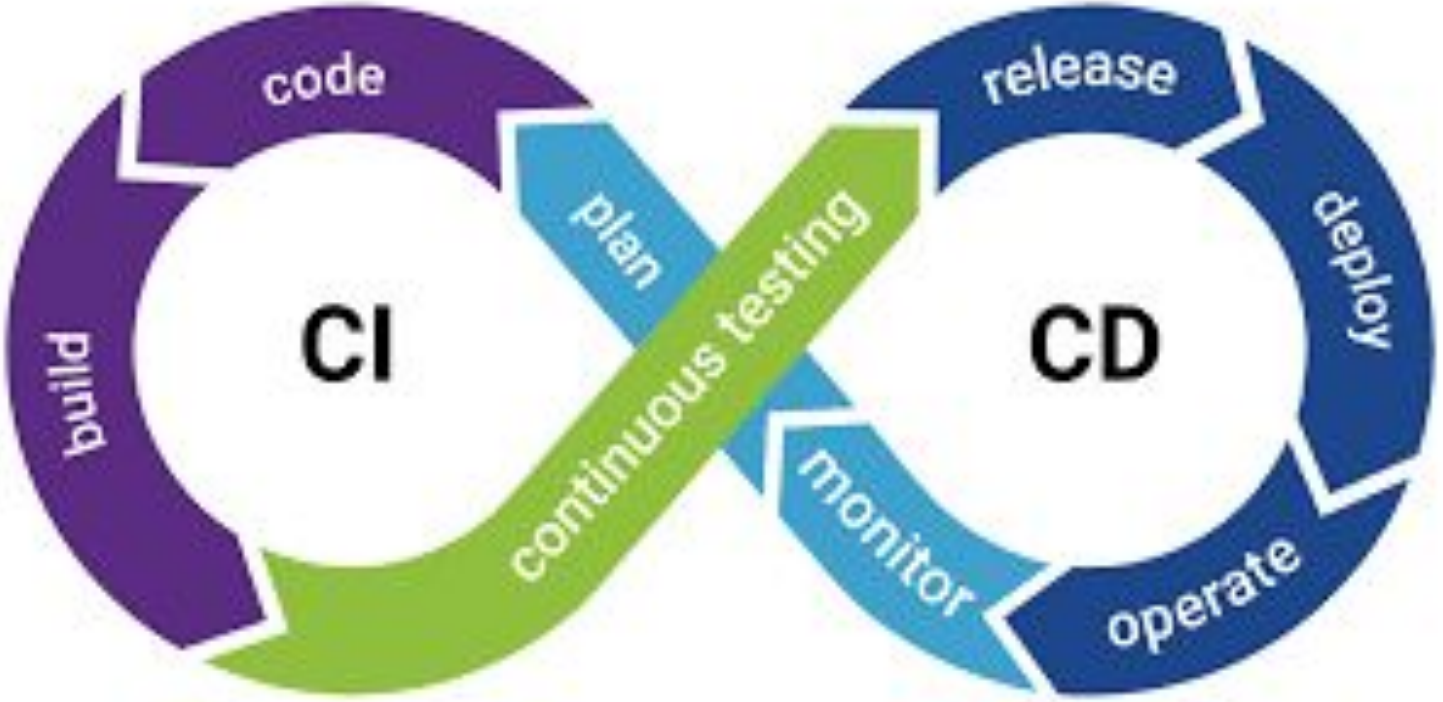
A CI é fundamental no desenvolvimento de software moderno, pois ajuda a garantir a qualidade do código, reduzir o risco de problemas de integração e acelerar o ciclo de desenvolvimento.

Ao adotar a Integração Contínua, as equipes podem construir software de alta qualidade de forma mais eficiente e identificar e corrigir problemas de integração mais cedo no ciclo de desenvolvimento.

Entrega Contínua

Estender a integração contínua, automatizando também a entrega do software em ambientes de teste e produção de forma incremental e confiável.

Entrega Contínua (CD)



Entrega Contínua: Automatizando o processo de lançamento

A Entrega Contínua automatiza o processo de implantação de software, permitindo lançamentos frequentes e confiáveis.

Implantação Automatizada

Description of a primary heading

Rollbacks Automatizados

Description of a primary heading

Ambientes Padronizados

Description of a primary heading

Entrega Contínua (CD)

1. Automatizar e simplificar o processo de lançamento de software de forma rápida, segura e confiável.

2. **Características:** automação de implantação, implantação incremental, ambientes isolados, feedback imediato e rollback automático em caso de falhas.

3. A Entrega Contínua estende os princípios da Integração Contínua (CI)

Para implementar a Entrega Contínua, as equipes de desenvolvimento utilizam:

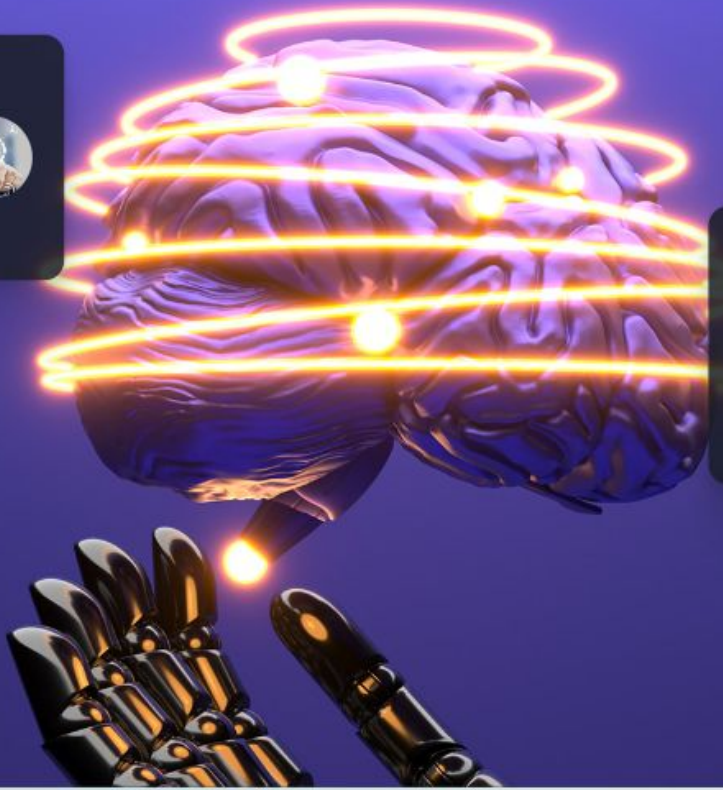
- Automação de compilação e testes.
- Controle de versão.
- Infraestrutura como código (IaC).
- Integração com ferramentas de CI/CD.
- Cultura colaborativa e interativa.

Infraestrutura como Código

Gerenciar a infraestrutura de TI usando código, permitindo a automação e a reproducibilidade do ambiente de implantação.

Infraestrutura como Código

A gestão de infraestrutura como código permite a automação e versionamento das configurações de infraestrutura.



Automatização de Provisionamento

Automatiza o provisionamento de infraestrutura por meio de scripts e templates.

Orquestração de Recursos

Realiza a orquestração automatizada de recursos de infraestrutura conforme a demanda.

Versionamento de Configurações

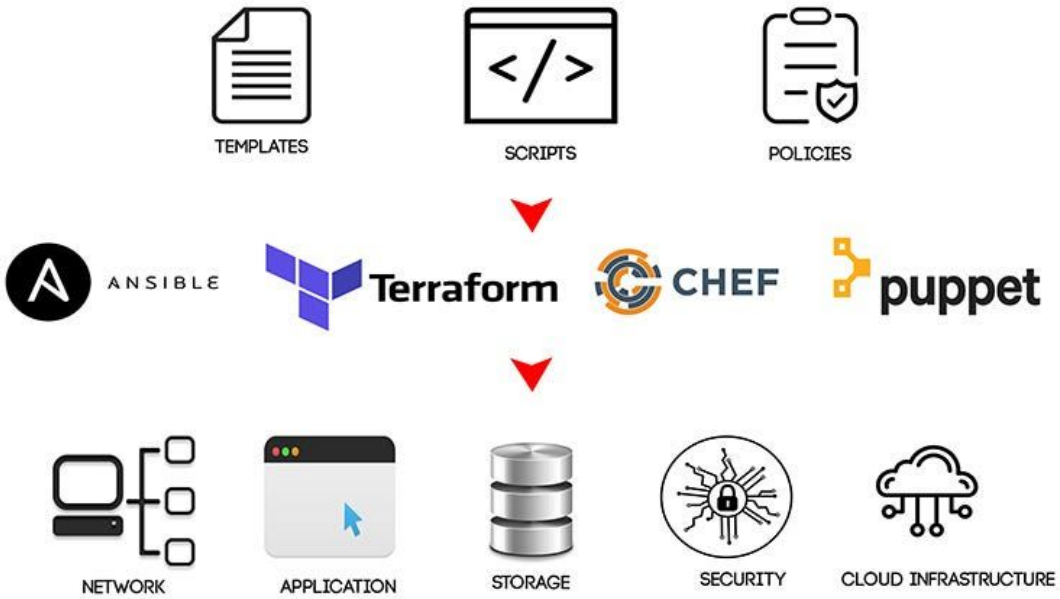
Utiliza controle de versão para as configurações de infraestrutura, garantindo rastreabilidade e consistência.

Infra Ágil - Pilares

Infra Ágil


Pilares da InfraÁgil
Automação
puppet
ANSIBLE
RUNDECK
Terraform
CHEF
CloudFormation
Entregas/Deploy
SERVERSPEC
git
Jenkins
Métricas
Prometheus
New Relic.
APACHE
JMeter™
Pessoas
Kanban
Scrum
Lean IT

Infra Ágil


TEMPLATES
SCRIPTS
POLICIES
ANSIBLE
Terraform
CHEF
puppet
NETWORK
APPLICATION
STORAGE
SECURITY
CLOUD INFRASTRUCTURE

Infra Ágil

Terraform, que permite criar, alterar e melhorar a infraestrutura de forma rápida e previsível.

Ansible, que permite automatizar a instalação e a atualização de aplicativos dentro dos nossos servidores e também gerenciar algumas configurações, além de ferramentas de entrega contínua.

Chef e ***Puppet***, que fazem a mesma função do Ansible.

Microserviços

Arquitetura de software na qual um aplicativo é dividido em componentes independentes e interconectados, facilitando a implantação e a manutenção contínua.

Microserviços: Facilitando a implantação e a manutenção

A arquitetura de microserviços promove a modularidade e escalabilidade das aplicações, facilitando sua implantação e manutenção.

01

Desenvolvimento Modular

Permite o desenvolvimento de pequenos serviços independentes, facilitando a manutenção e evolução do sistema.

02

Escalabilidade

Facilita a escalabilidade horizontal dos serviços, atendendo a demandas variáveis de forma eficiente.

03

Resiliência

Promove a resiliência do sistema, isolando falhas e prevenindo a propagação de problemas.

Monitoramento e Feedback

Monitorar constantemente o desempenho do software em produção, coletando métricas e feedbacks dos usuários para identificar e corrigir problemas rapidamente.

Monitoramento e Feedback: Garantindo a qualidade do software

O monitoramento contínuo do desempenho do software e a obtenção de feedback são essenciais para garantir a qualidade e confiabilidade do produto.

1

Monitoramento em Tempo Real

Utiliza ferramentas de monitoramento para acompanhar o desempenho do software em produção.

2

Coleta de Feedback

Obtém feedback dos usuários e das operações para identificar oportunidades de melhoria.

3

Análise de Métricas

Realiza análise de métricas para identificar gargalos de desempenho e áreas de atenção.

Monitoramento e Feedback

1

Monitoramento de Desempenho: Acompanhamento contínuo do sistema, uso de recursos e métricas relevantes.

2

Monitoramento de Logs: Coleta, análise e visualização de logs para identificar problemas e padrões.

3

Alertas e Notificações: Configuração de alertas automáticos para condições específicas.

4

Análise de Tendências: Identificação de padrões ao longo do tempo para dimensionamento e ajustes.

5

Feedback do Usuário: Importância do feedback para avaliar a usabilidade do software.

6

Iteração e Melhoria Contínua: Uso de informações para aprimorar o software continuamente.

O Futuro do DevOps

Tendências Emergentes

Novas práticas e tecnologias como DevSecOps, AIOps, e GitOps estão moldando o futuro do DevOps. Integração de IA, automação avançada e segurança integrada são áreas em destaque.

Impacto no Desenvolvimento de Software

O DevOps continuará a acelerar a entrega de software, melhorar a colaboração entre equipes e impulsionar a inovação. Maior foco na segurança, qualidade e escalabilidade será essencial.

Adaptação Contínua às Novas Tecnologias

Empresas devem estar preparadas para adotar e adaptar-se às mudanças tecnológicas rapidamente. Cultura de aprendizado contínuo e flexibilidade serão fundamentais para o sucesso no cenário de evolução constante.