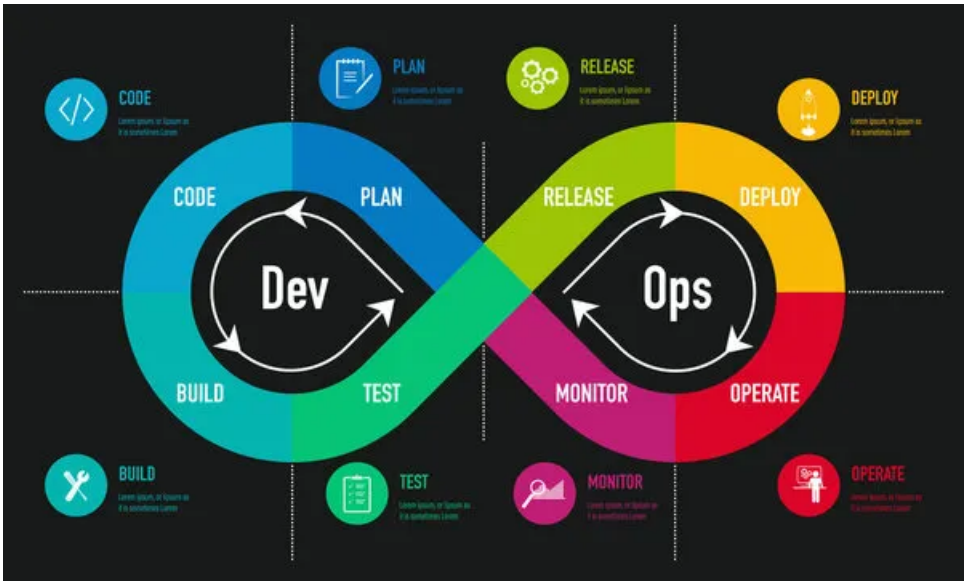
DevOps Modulo 1 - IT Talent

Escrito por: Gabriel Oliveira dos Santos

GIthub: https://github.com/Hypothasis

O que é DevOps?

DevOps é a combinação entre a filosofias culturais, práticas e ferramentas que aumentam a capacidade de uma empresa de distribuir aplicativos e serviços em alta velocidade, assim otimizando e aperfeiçoando produtos em um ritmo mais rápido do que o das empresas que usam processos tradicionais de desenvolvimento de software e gerenciamento de infraestrutura.



Os três pilares do DevOps sao a cultura, práticas e ferramentas.

Cultura

Criar um ambiente altamente colaborativo

A cultura do DevOps são a parte do indivíduo e as soft skills, o DevOps tira o ciclo das empresas (o costume e manias da empresa), visando melhorar as comunicações com a operação e os desenvolvedores, já que Ops visa a confiabilidade e o Dev visa a produtividade.

Automação, Foco no Cliente e Entrega

O DevOps tem como objetivo tirar todo ciclo ocioso e automatizar processos visando a eficiência do projeto, assim automatizando quando possível e com foco nas necessidades dos clientes, assim não produzindo coisas a menos e nem a mais para projeto, assim tirando o fato de desenvolver em excesso e perdendo tempo com coisas não necessárias.

Desenvolver e Lançar frequentemente

O DevOps tem o objetivo de produzir, com eficiência, e lançar frequetemente para monitorar e obter feedbacks de clientes, assim mantendo o ciclo vivo, de produção , testes, monitoramento, assim trazendo ideias novas para o projeto e visando a melhoria do mesmo.

Segurança e Entrega Continua

A segurança vai oferecer suporte para a entrega contínua, em que e um processo que o DevOps tem que educar as equipes de operaçoes e desevolvimentos para identificar os processos vulneraveis e resolver problemas com eficiência , antes que virem um problema mais grave.

Experimentar e aprender continuamente, perguntas e hipoteses estão ligados com a inovação para a mentoria das melhoria dentro da empresa, brainstorms para solução de problemas, em que quando a empresa enxerga os problemas como forma de aprendizado , ocorre a evolução, pois é inevitavel que a inovação não tenha falhas.

Assim o DevOps tem como objetivo a melhoria contínua, visando a inovação e aprendizado, assim definindo padrões da qualidades para a empresa.

Práticas

Comunicação e Colaboração

Times em que a informação em que a comunicação é claramente divulgada e compreendida, e ao inves de esconder os erros, é comunica-los e trabalhados em equipe, visando a melhoria do padrao de qualidade. Trabalho em equipe e que os problemas de cada um é o problema do projeto, então todo mundo deve auxiliar para a resolução desse problema geral, e não compactuar com a individualidade dos problemas.

Monitoramento e Observabilidade

Como as mudanças afetam a performance do projeto e a experiencia geral do usuario, assim visa melhorar o programa com o monitoramento e a observabilidade.

Integração Contínua (CI)

A AWS, a integração continua é uma prática continua em que os desenvolvedores executam seus codigos em um repositorio central. A integração continua tem objetivo como encontrar erros mais rapidamente, melhorar a qualidade do software e reduzir o tempo para a produção de atualização do software.

Entrega Continua (CD)

A AWS define a entrega continua é uma prática de desenvolviemento de software em que cada alteração de codigo é implantada em um ambiente de teste qualificado.

Arquitetura de Microsserviços

Monolito é quando um é feito todo em uma base de serviço único, já o microsserviços tem suas aplicações separadas, em qua cada parte do código tem sua finalidade especifica. A Arquitetura de Microsserviços vem com a forma de como o sistema é organizado em uma abordagem de projeto que cria uma aplicação de serviços acoplados.

Ferramentas

Nuvem

Serviços que estão ligados a internet, assim viabiliza a evolução da integração do projeto e tambem com a economia de recursos dentro da empresa.A Nuvem é uma ferramenta para o DevOps

Desenvolviemento

O Dev do DevOps, o clean code, práticas de padroes de desenvolvimentos.

CI/CD

Entreda Continua e Integração Continua, essas ferramentas automatizam o codigo com a equipe, e definem padroes de qualidades, executam codigos em diferentes ambientes de software. Assim essas ferramentas auxiliam na velocidade de produção dos projetos e implantaçao de aplicações.

Automação de Infraestrutura

Com programação voce pode definir sua infraestrutura, voce pode ter um sandbox de desenvolvimento e testar vários tipos de execução de códigos, configurar um ambiente de trabalho e automatizar para criar um padrao para a infraestrutura do ambiente de desenvolvimento

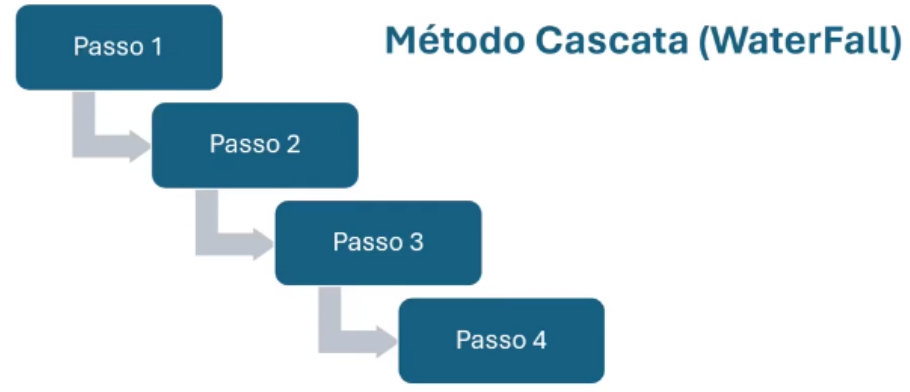
Contêineres e Serverless

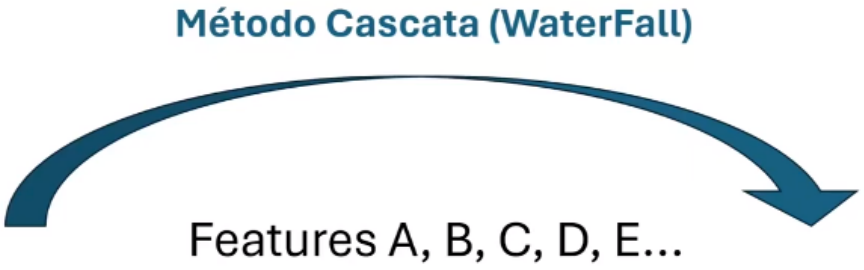
Serviço serverless (sem servidores), permitem com que os desenvolvedores apenas foquem na produção do código do programa,e não com os detalhes com o ambiente de hospedagem, assim a aplicação é portatil, podendo ser executado em qualquer servidor. já os contêineres são como se fossem máquinas virtuais, so que sao mais leves, executam qualquer coisas, desde microsserviços ate aplicações grandes, já que as politicas de seguranças são aplicadas com o tamanho dos contêineres.

Monitoramento e Observabilidade

Ajuda ver antes problemas, ver trafego de uso de aplicações, ajuda a rastrear transações de ponta-a-ponta por meio do sistema distribuido.

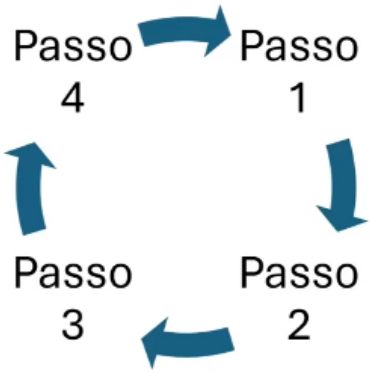
Waterfall e Agile



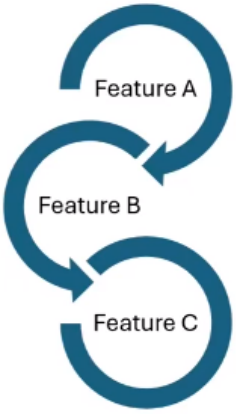


Waterfall é a forma sequencial de desenvolver um projeto para empresa, no método cascada, as informações vão passando de um processo para o outro,

assim existe uma sequência de passos a se seguir na elaboração dos projetos, assim tendo a entrega do produto no final. A entrega dessas features é mais demorado que o Agile, pois não há uma entrega parcial para o cliente, assim não tem feedback na produção, só na entrega, semelhante as jogos, em que normalmente os bugs e erros dos jogos são descritos pela comunidade só na venda do jogo final, pois não existe uma entrega contínua do jogo para com a comunidade.



Método Ágil (Agile)



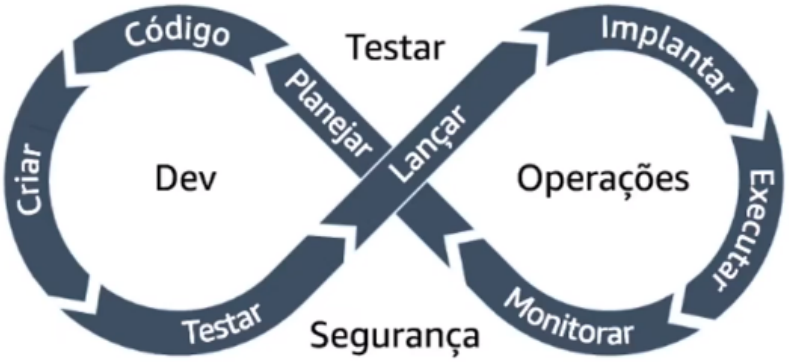
Método Ágil (Agile)

Iterativa => por iterações

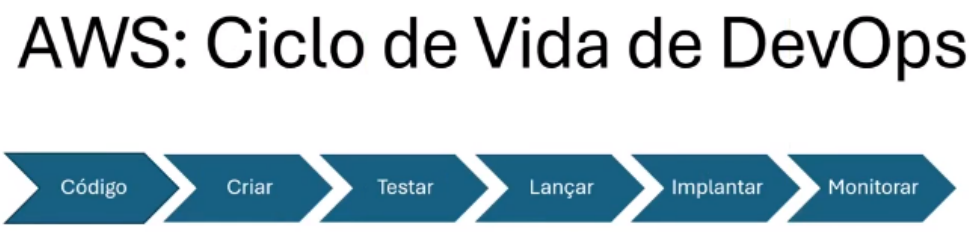
Agile é uma sequencia cíclica para o desenvolvimento dos projetos, em que cada passo tem um próximo passo sucessor, assim ao entregar o produto final, temos o ciclo que volta para o passo 1 de novo. Assim a diferença é que a entrega é de forma dinâmica, assim visa a aceitação do feedback do cliente, assim aumentando a segurança do projeto e o padrão de qualidade, pois no método Agile, o ciclo entre a produção, testes, monitoramento e segurança, fazem com que este método tenha uma vantagem em relação ao método Waterfall. Uma interação é feito cada feature, assim a criação do progama visa

a opnião do cliente para com a aceitação dessa feature, mudaça ou correção da mesma. No mundo dos games, é como se a empresa a cada atualização mandasse uma versão demo para a comunidade avaliar, assim tendo um feedback, a empresa junto com o DevOps tem a noção de quais as necessidades atendem melhor o projeto, mantendo o padrão de qualidade superior.

Pipeline ou Ciclo de Vide de Software



Pipeline funciona como uma esteria, é como fosse uma linha de produção em que ambiente ideal, seja o mais automatizado possível. Com tudo, as mudanças nos códigos seja mais padronizados, deixando-os as mudanças mais estáveis cheguem ao cliente de forma segura e testada, ampliando a vantagem competitiva da empresa. DevOps não se preocupa só com desenvolvimento e operações, mas também com testes, qualidade e segurança, assim monitorando para o incremento do padrão de qualidade, assim visando sempre melhorar a cada ciclo



Código

pequenas e frequentes mudanças visando melhorar o código fonte, em que devem estar presente a um repositório. A revisão do código é o ponto crucial dessa categoria, pois com a revisão coletiva do código, trazemos ideias novas, resolução de erros que não foram percebidos pelo programador, permitindo uma analise imparcial.

Criar (Build)

Pessoas devem trabalhar em conjunto para criar a mesma funcionalidade, e esses códigos devem trabalhar em conjunto, assim esse codigo deve ser compilado (esse objeto, é chamado de **artefato**, um **executavel**, que deve estar pronto para a execução de outros ambientes de desenvolvimento) para ser testado na próxima etapa.

Testar

Avaliação do projeto, para checar se o projeto atende aos requisitos do cliente e com o padrão de qualidade da empresa. Exemplos de testes são:

Funcional, checa se o artefato realmente funciona e atende os requisítos.

Integração, juntar duas ou mais partes para trabalhar como um todo.

Regressão, checa se depois de juntar as partes, não houve um erro na execução do projeto, sem nenhum novo surgindo.

Teste de Aceitação, checa se o programa executa como o cliente quer, sendo o último olhar para checar se o programa roda bem e tem uma aceitação com a interface para com o cliente.

Teste de Carga, checa quanto um programa aguenta antes de dar um problema, como colocar pesos em um elevador e checar até qunado aguenta. Assim verificando quanto o programa aguenta com interção de vários usuários.

Teste de Segurança, se há buracos no softwares que possibilitem que uma pessoa possa invadir, e tampar esses buracos.

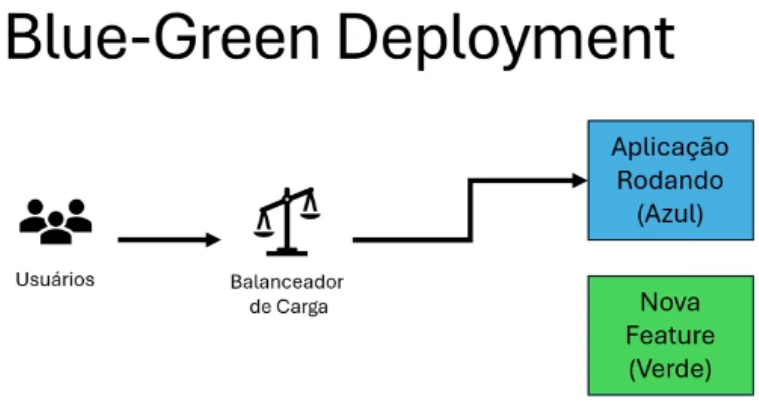
Vale a pena ressaltar que esses testes devem ter uma abordagem em que esses testes sejam executados desde a etapa do código, e vai se espalhando por todo ciclo de vida, que no caso seria o Test Driven Development (Desenvolvimento Orientado a Testes).

Lançar

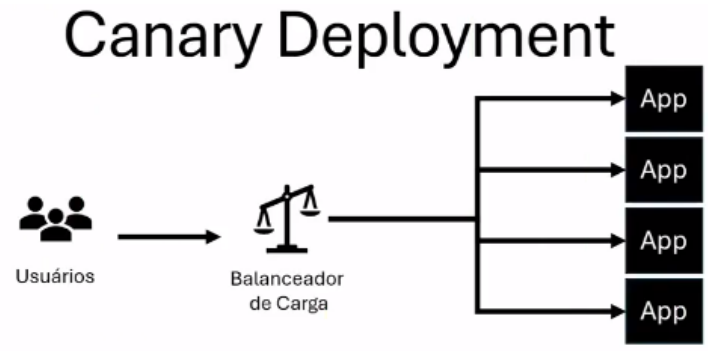
Erros comuns é confundir com implementação, lançar vem do ingles **Realise,** ja implementação vem do ingles **Deploy**. Deploy e a movimentação de um ambiente para o outro, pode ir para teste, para outro ambiente de outro desenvolvedor. Já a etapa Lançar, liberar o código para se executar no passo seguinte, ou seja, lançar pode liberar para fazer deployment. É como fazer o Realise de uma versão estavel e não estavel para que possam fazer o deploy. Lançar é liberar o programa para baixar, e deploy e outro usuario usar esse programa em outro ambiente.

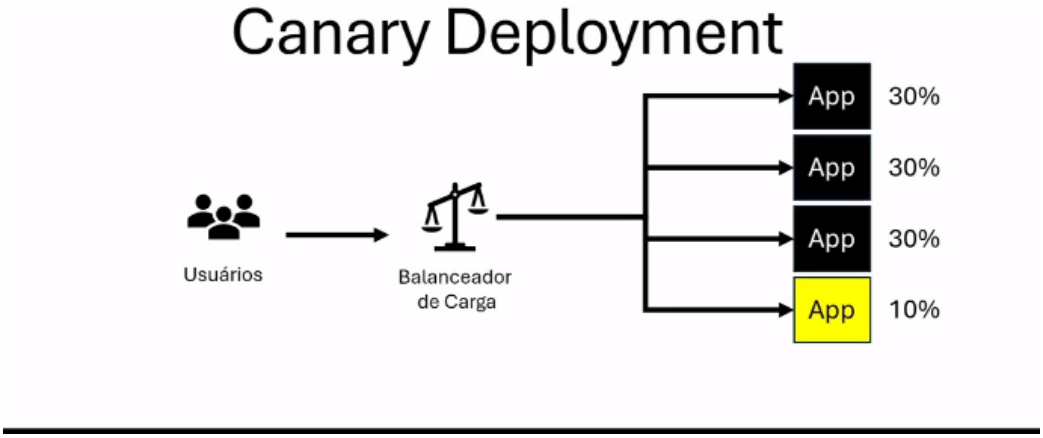
Implementar

O deploy é fato de liberar a versão final para o cliente, e deixar pronto para uso



Blue-Green Deployment, se desligarmos o servidor para atualizar o programa, nosso programa fica fora do ar por um tempo, isso não é legal. Nesse metedo o usuario é passado para um balaciador de carga, que ao acessar o programa antigo, ele é direcionado para o programa atual, direciona para a nova feature.





Canary Deployment, essas técnicas são uadas para diminuir os riscos de erro ao atualizar nosso programa no momento do deploy. Canary faz a mudança em um desses servidores, em que afeta apenas uma parte desses usuários. Assim monitoramos os índices de erros em comparação com as versões anteriormentes, e se caso aja mais erros do que as versões anteriores é necessário checarmos.

Monitorar

Entender e medir a confiabilidade dos nossos padrões de qualidade dos nossos projetos, a observabilidade é a forma de entender quão fácil é a sua forma de trabalhar e os sinais que ele emite, já o monitoramento é enxergar esses sinais para visualizar a execução desse sistema. Não adianta observar se não temos o que analisar, nao adianta enviar dados se ninguem vai capturar.

Ambientes de Software

Cada projeto ao clicar um artefato para teste, é necessário checar qual a finalidade de cada testador tem com o produto. Na imagem acima, o cientistas podem lidar com robos sem estar completos, so com os circuitos eletronicos, pois cientistas tem capacidade intelectual para olhar esses robos, mas esses robos nao podem ser enviados para clientes e nem testadores normais, pois os mesmos não possuem conhecimento para manujear robos como cientistas, assim robos para testadores , testes de aceitação e clientes, devem estar próximo ao resultado final, para que não haja um ambiente de teste fora do planejado. Com tudo isso vale para softwares, um software teste para desenvolvedor e para um cliente são diferentes, o dev le o código fonte , já o cliente avalia o software mais próximo do resultado final.