CI/CD

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Revisão Cloud e DevOps
- CI/CD
- LAB

Revolução na Nuvem

- A primeira revolução é a criação da nuvem
  - O que é nuvem ?
- A segunda revolução é o surgimento do DevOps
  - O que é DevOps ?
- A terceira revolução é o surgimento dos contêiners.
  - Tema do próximo módulo

Primeira nuvem ...

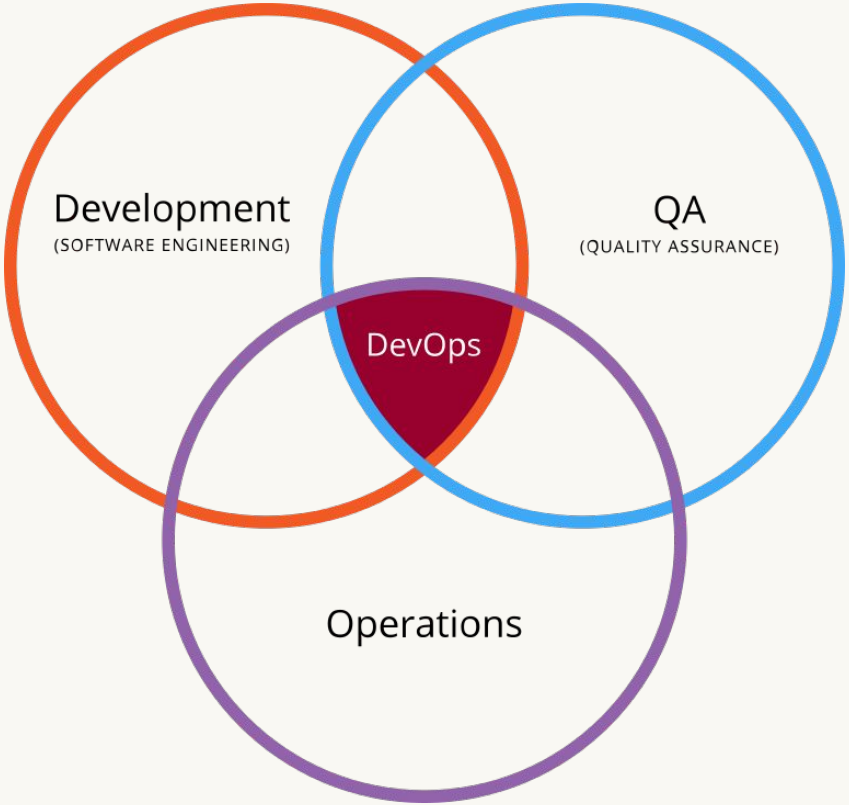
IBM System/360 Modelo 91

- Em 1960 do ponto de vista de custos, não era vantajoso que cada empresa comprasse e mantivesse o próprio hardware. Assim surgiu um modelo de negócio ao qual os usuários compartilhavam a capacidade de processamento de máquinas remotas pertencentes e administradas por terceiros.

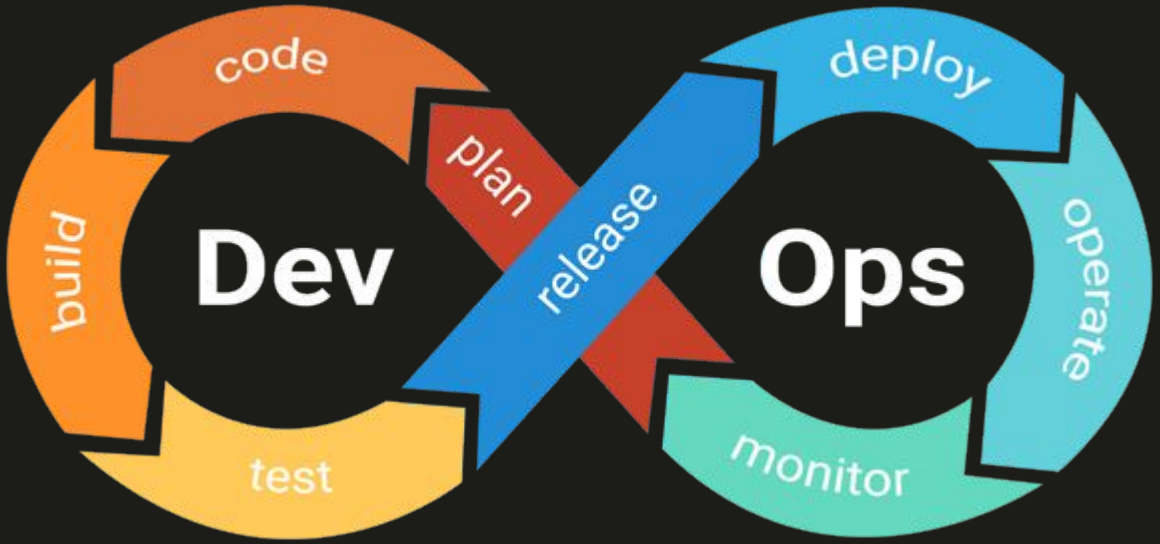
Surgimento do DevOps

- Antes do DevOps os desenvolvedores escreviam o software e passavam para a equipe de operações, que executavam e mantinham o software em ambiente de produção

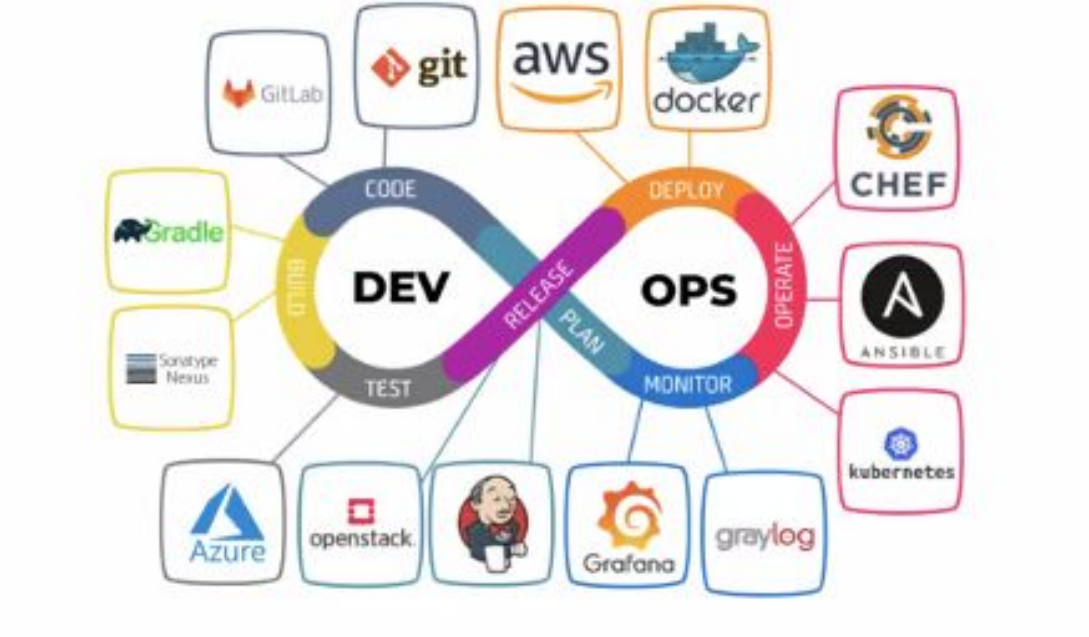
Revisão DevOps



Revisão DevOps



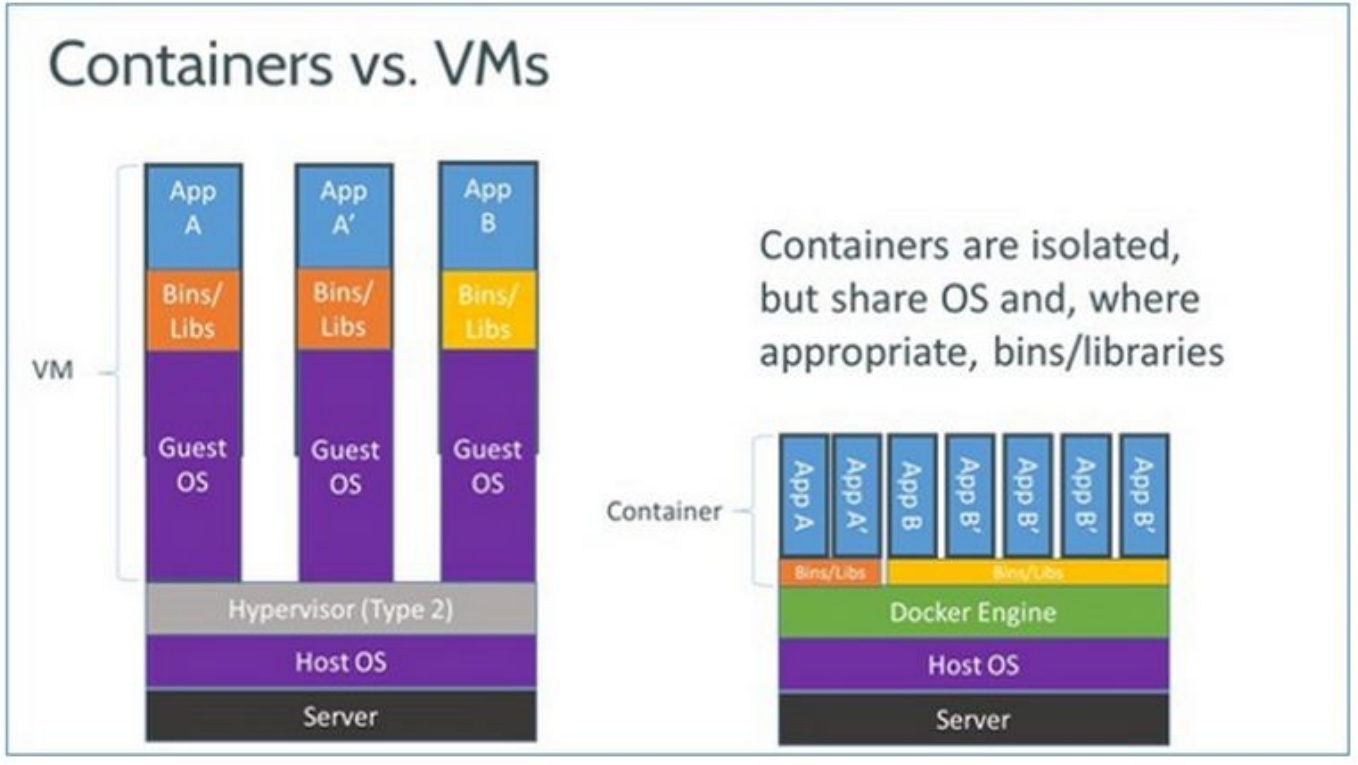
Revisão DevOps



Containers

- Os **containers** compõe o provisionamento de uma aplicação de forma escalonável
- Entre os container temos o **Docker** como a ferramenta mais popular para criar serviços.
- Os containers precisam ser **orquestrados** por outros serviços, no caso os serviços da **GCP**, mas em um ambiente local podemos usar um orquestrador como o **Kubernetes**.

Revisão DevOps



Introdução CI

- Continuous integration ou Integração contínua.
- É uma prática de **desenvolvimento de software** na qual você compila e testa software toda vez que um desenvolvedor envia código para a aplicação, e isso pode acontecer várias vezes ao dia
- Após enviarmos o código para um **repositório de versionamento** ele passa por um processo onde é feito o **Build**, após isso os testes são executados, entre esses podemos ter Unitários, de Integração, ou de qualquer outro tipo.

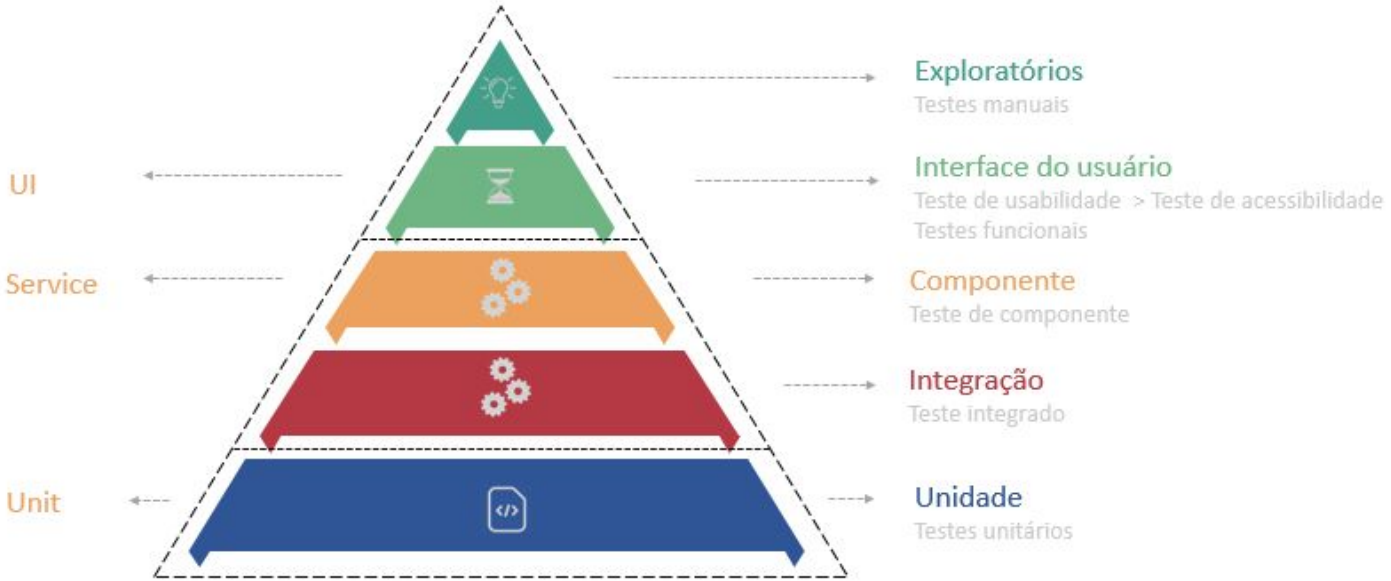
CI - Testes

- Um dos pontos chave do **CI** é o **teste**, esse pode ter vários níveis de granularidade, mas deve ser feito tanto na esteira quanto antes de ser enviado para ela. Sendo assim o desenvolvimento é feito no modelo **TDD - Test Driven Design**

- A quantidade de testes recomendados segue um modelo de **pirâmide**, onde a base são os testes unitários e no topo temos os testes manuais ou de exploração

- Uma unidade no código depende do tipo de código desenvolvido, no caso de um programa Orientado a Objeto, essa unidade pode ser um método.

CI - Testes


UI
Service
Unit
Exploratórios
Testes manuais
Interface do usuário
Teste de usabilidade > Teste de acessibilidade
Testes funcionais
Componente
Teste de componente
Integração
Teste integrado
Unidade
Testes unitários

Introdução CD

- **Continuous Delivery** ou Entrega Contínua
- É uma abordagem de **engenharia de software** na qual a integração contínua, testes automatizados e recursos de implantação automatizada permitem que o software seja desenvolvido e implementado de forma rápida, confiável e repetitiva com intervenção humana mínima.
- A implantação na produção é definida **estrategicamente** e é acionada manualmente.

CD - Ambientes

- Após realizarmos a parte de **Build** e **Testes** do **CI** podemos fazer a entrega de software em um ambiente, os mais comuns encontrados são:
    - **Desenvolvimento**
    - **Homologação**
    - **Produção**

CD - Ambientes

- **Desenvolvimento -** É onde colocamos o código para ser executado em um ambiente parecido com o de Produção pela primeira vez, porém o objetivo é apenas averiguar o funcionamento da aplicação.

CD - Ambientes

- **Homologação -** É o ambiente mais próximo de Produção, sendo que os dados são feitos para serem o mais parecido possível com produção e é onde testamos o comportamento junto da funcionalidade da aplicação.

CD - Ambientes

- **Produção** - É o ambiente que é acessado pelo usuário final, este pode ser interno ou externo a empresa, nessa fase nós apenas monitoramos o comportamento do serviço e verificamos se não há falhas no comportamento dele.

Outro significado de CD

- Podemos ter outro significado para **CD**, **Continuous Deploy** ou **Deploy Contínuo.**

- É uma prática de desenvolvimento de software na qual cada alteração de código passa por todo o **pipeline** e é colocada em produção automaticamente, resultando em muitas implantações de produção todos os dias.

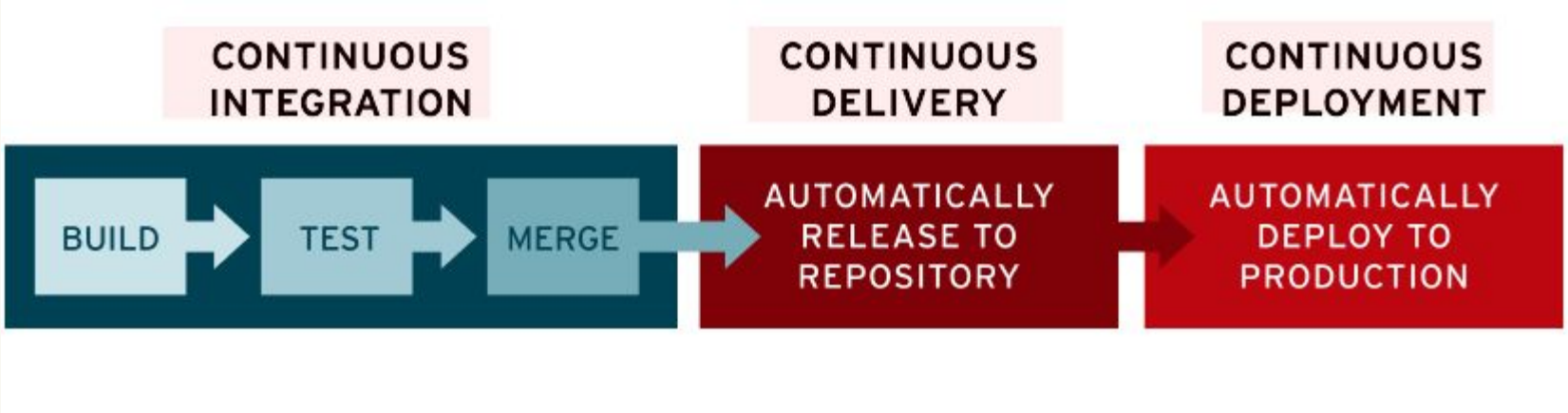
- Faz tudo o que a **entrega contínua faz**, mas o processo é totalmente **automatizado**, não há intervenção humana.

Vantagens do Continuous Deploy

- Entrega rápida;
- Menos trabalho na entrega;
- Escala;

Pipelines

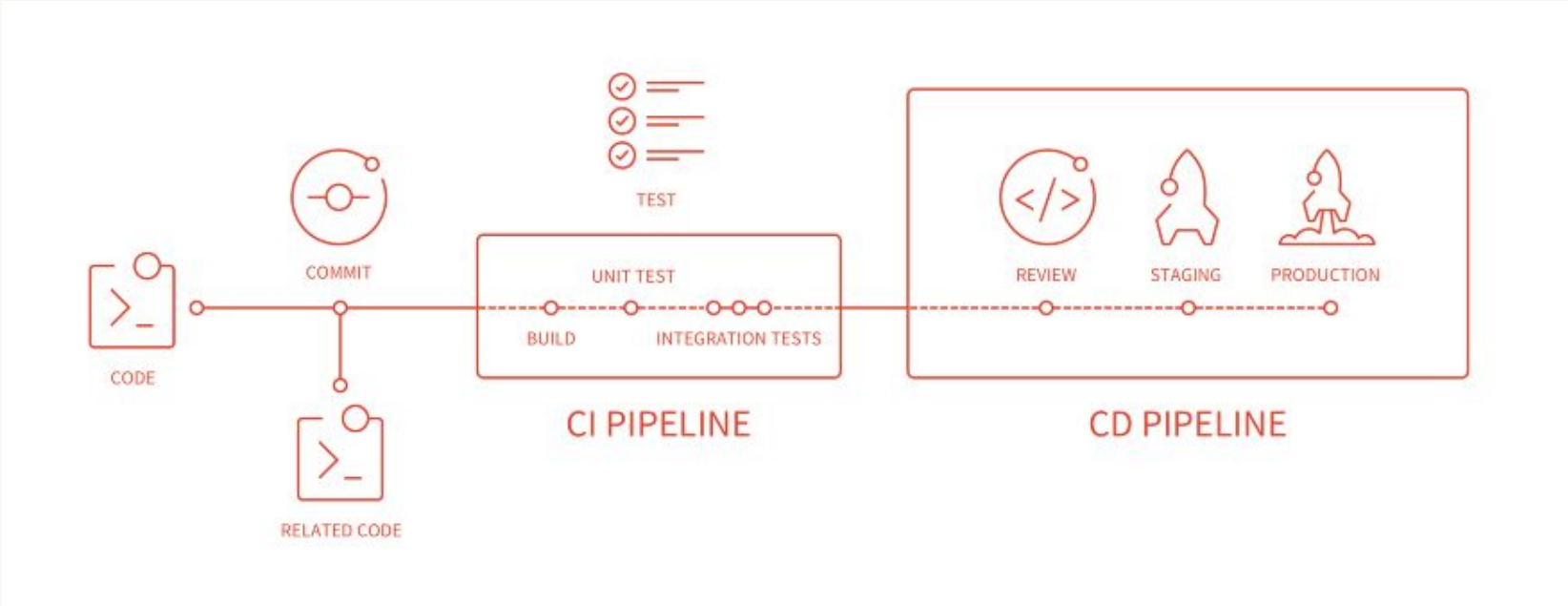
- As pipelines **CI** e **CD** são **automatização** do processo de entrega de software;
- São uma **sequência** de passos que são utilizadas;
- É rápida, confiável e precisa.



Pipelines

- As pipelines geram um constante **fluxo de feedback**, sendo assim sempre temos a informação mais recente sobre o programa e o seu comportamento.
- Exemplos de tecnologias para pipelines:
  - Jenkins
  - GitHub
  - Gitlab
  - **Bitbucket**

Pipelines


TEST
COMMIT
UNIT TEST
REVIEW
STAGING
PRODUCTION
CODE
BUILD
INTEGRATION TESTS
CI PIPELINE
CD PIPELINE
RELATED CODE

LAB

GCP (Google Cloud Platform)

- Para a nossa nuvem onde será feito o deploy da nossa aplicação vamos usar o **GCP**. Portanto precisaremos criar uma conta na plataforma.

Bitbucket

- O Bitbucket será a nossa ferramenta principal para exercitarmos os conceitos de **devOps**.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>