Arquitetura de sistemas avançado

[Subtítulo do documento]

Digital Innovation One

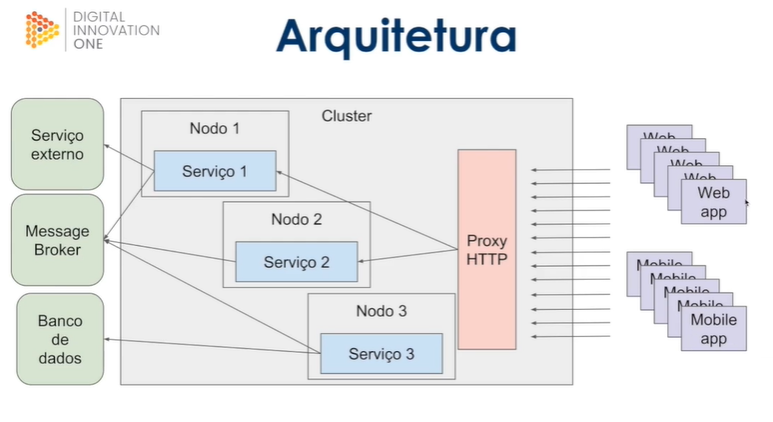
Arquitetura de sistemas avançado

1. O que é arquitetura em mensageria
2. Comunicação assíncrona entre serviços
3. Gerenciamento de erro

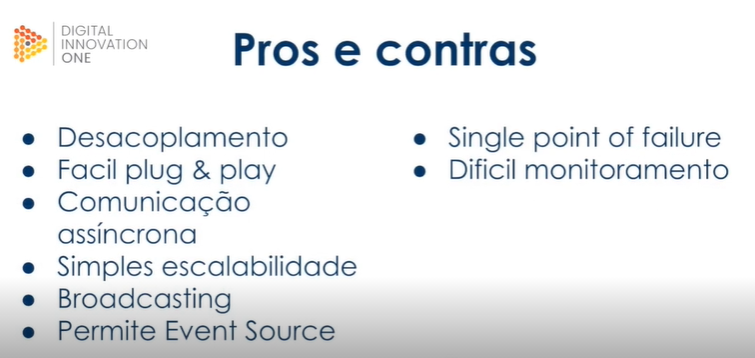
Requisitos básicos

* Entendimento sobre RestAPI
* Conhecimento básico sobre message brokers

**O que é arquitetura em mensageria**

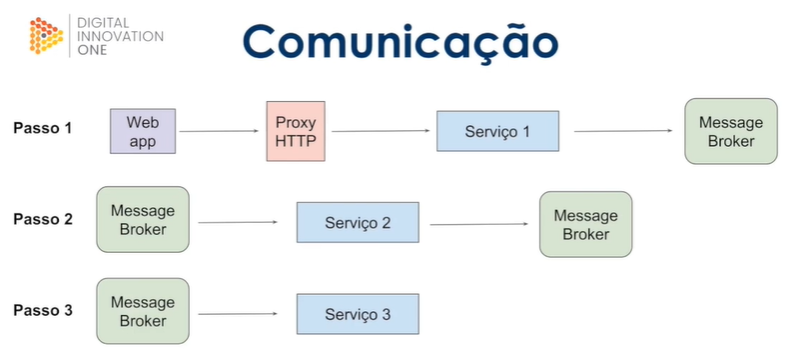


A arquitetura em mensageria nos permite ter serviços que se comunicam com aplicações externas e aplicações que não se comunicam diretamente, mas estão consumindo mensagens de serviços que tiveram esse contato. Como o nodo3 (serviço 3) que não se comunica com o externo, porém se alimenta de message broker dos ou tros nodos (serviços) que tiveram esse contato externo.

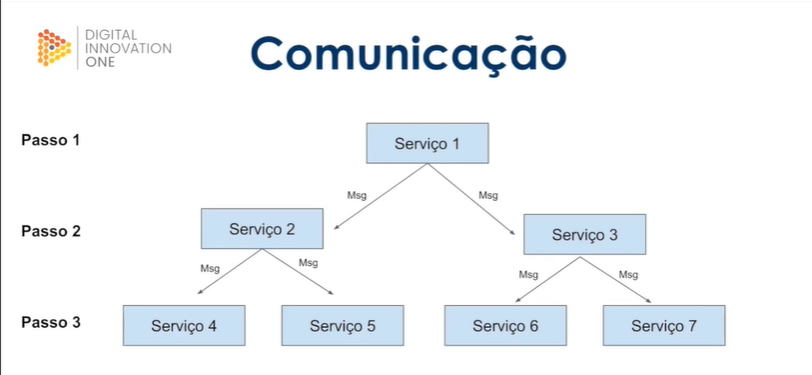


**Comunicação assíncronas entre serviços**

Comunicação tipo simples, linear ou pipeline.



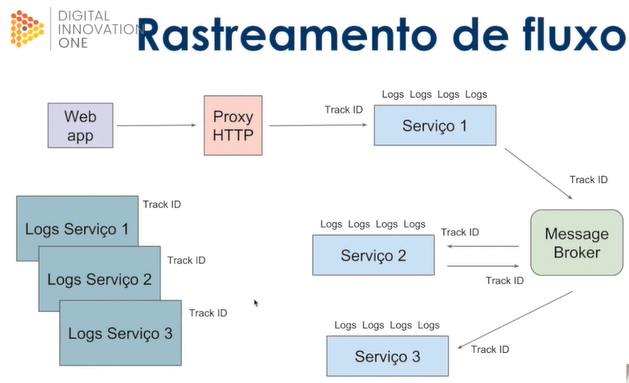
Comunicação mais complexa



**Gerenciamento de erro**

* Dead Letter Queue (Filas de re-tentativas)
* Monitoramento entre serviços
* Rastreamento de fluxo

**Rastreamento de Fluxo**



**Arquitetura de dados não estruturados e business intelligence**

Business Intelligence aplicado a modelo de dados

Data Warehouse

Big Data – Dados não estruturados

Data Lake

Requisitos básicos

* Conhecimento de bancos de dados
* Conhecimentos de Lógica

**Business Intelligence aplicado a modelo de dados**

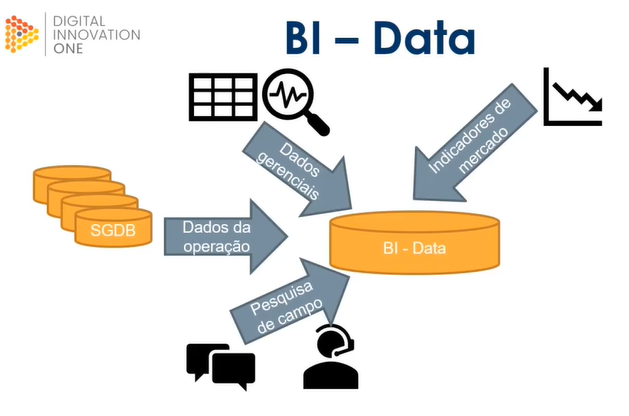
* **Business Intelligence** ou **BI**, em um modelo de dados, é algo que vai nos dar um norte para modelar os nossos dados em apoio a decisões estratégicas de uma organização de uma empresa.

O **BI** é composto basicamente por

* Ferramentas
* Infraestrutura
* Profissionais (corpo técnico)
* Dados

**De onde vem os dados para BI?**

Os dados essencialmente dentro de uma corporação de BI, vem essencialmente dos sistemas de SGBD ou qualquer outro sistema de operação.



O que uma infraestrutura completa de BI deve ter:



Ferramentas de BI

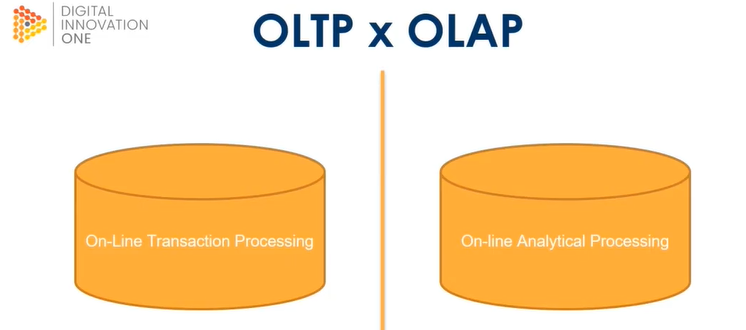


**Aprenda sobre os conceitos de Data Warehouse**

É uma estratégia, um estilo de modelagem de dados que serve para acomodar os dados de uma forma que eles sejam eficientes e satisfatórios para apoio ao BI. O conceito de Data Warehouse surgiu antes de conceitos como Big Data e data Lake.

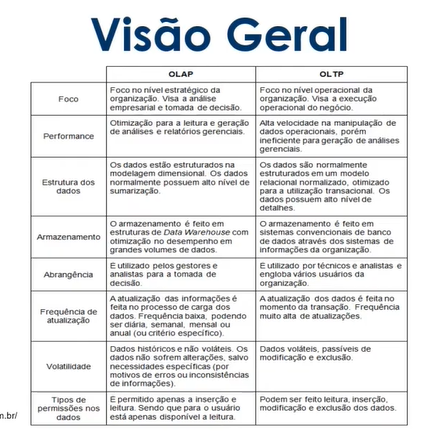
OLTP – Representa as transações online, pont-a-ponto que acontecen-no meu sistema, atualização do meu produto inserção de novo cliente.

OLAP – Análise das informações do OLTP, ou seja, a consolidação desses dados para relatórios e ferramentas estratégicas.

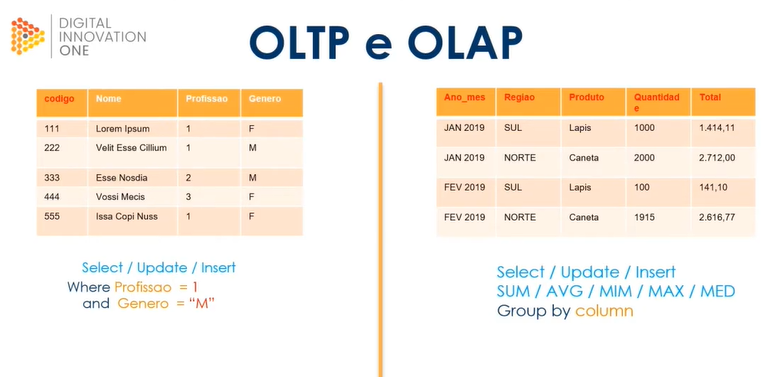


OLTP geralmente tende a atender uma grande gama de usuários. Vendedores, sistemas,etc conectados ao sistema OLTP realizando várias transações ao mesmo tempo. Ao ponto de o OLAP atender uma gama mais reduzida de usuários como responsáveis por setores, coordenadores, enfim, um público alvo mais reduzido em relação ao OLTP.

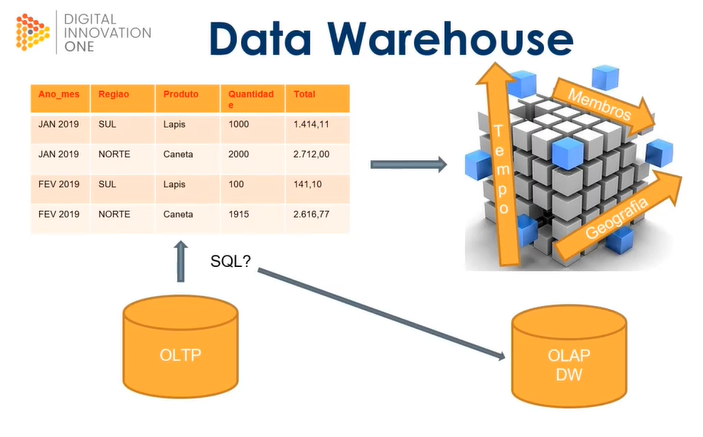
OLAP é mais focado no nível estratégico, OLTP mais no nível operacional da organização.



Nota na questão da **Volatilidade**!

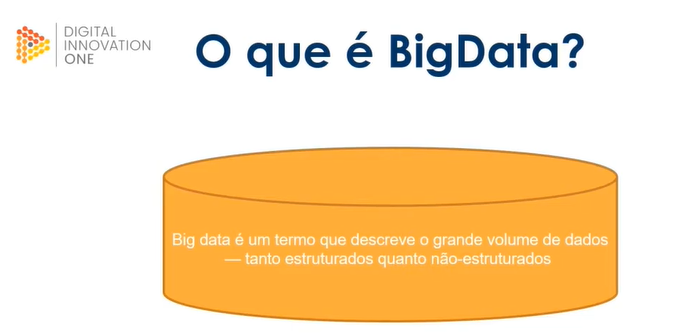


Em data Warehouse eu tenho uma ideia de que eu posso obter essas informações através dos meus bancos de dados OLTP. E como eu posso conseguir essas informações? O primeiro modelo conhecido de Data Warehouse foi através de sql, de consultas. Mas essas consultas geralmente são muito pesadas, exigem grandes volumes de processamento concorrente, e prejudicando com o online ou OLTP. Em outros aspectos, vc tem ferramentas que vão acessar essas informações que vão consumi-las, exibi-las de diversas formas e diversas vezes, isso é conhecido em Data Warehouse como **cubos de visões**, então meu relatório podem gerar cubos de visões que são projetados através de dimensionabilidade.



Você pode implementar a estrutura de dados de um Data Warehouse em um banco de dados relacional ou no SQL. Essa estrutura de relacionamento ela ocupa processamento dentro do meu sistema. Os insertes em banco de dados no DW é mais rápido em relação a banco de dados relacional.

O que é Big Data e dados não estruturados



Big Data então, passa a ser formatos diversos de informação onde nós precisamos retirar algum tipo de conhecimento. Essas informações podem ser: áudio, imagem, email, outros bancos de dados, log de sistema, satélites, etc...

A internet é um grande Big Data

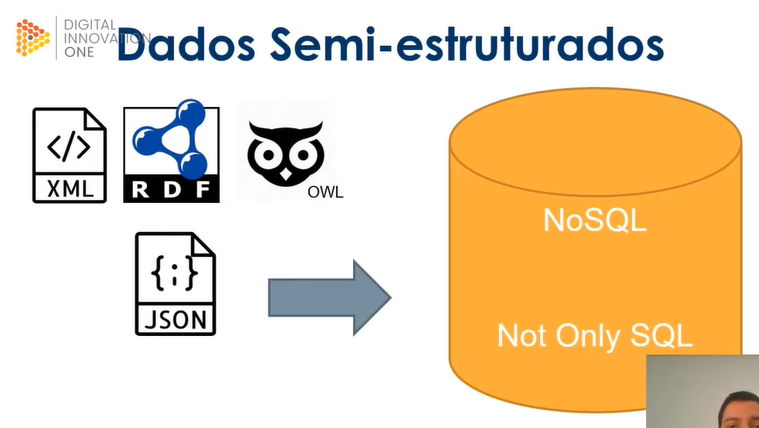
Big Data trata de grandes volumes de dados e ferramentas que podem extrair e trabalhar essa informação.

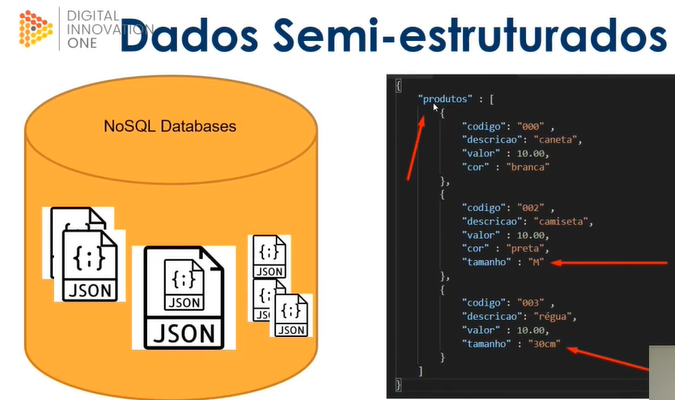
**Dados Estruturados**

São geralmente armazenados em banco de dados que mantem uma estrutura rígida. Em banco de dados estruturados eu preciso seguir uma regra no meu dicionário de informações para eu inserir e manusear as informações. Isso gera segurança na manutenção dos dados mais também enrijece a facilidade de mudança de dados.

Dados Semi-Estruturados

Nesse tipo eu tenho que obedecer uma estrutura básica, mas eu posso ter e não ter a informação.





**Banco de Dados no SQL**



**Dados Não Estruturados**

Apenas são a coleção de um grande tipo de informações estruturadas, semi estruturadas, vindas de redes sociais, dispositivos, etc. Dados não estruturados é apenas a conexões entre esses dados. Você precisa de ferramentas que possam pesquisar, gerir e trazer resultado dessas informações.



Imagem de dados não estruturados

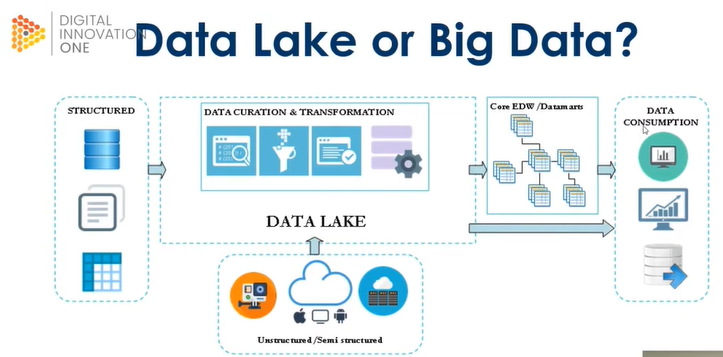


Nessa imagem, ferramentas que lidam com dados não estruturados

**Data Lake versus Big Data**

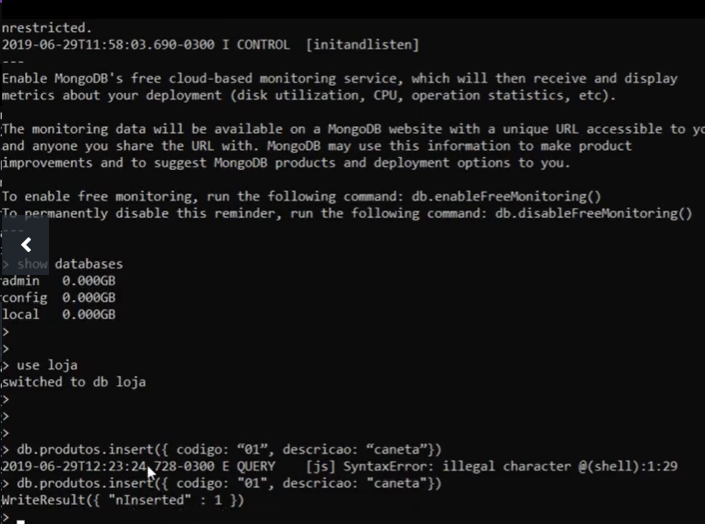
Data Lake

Data Lake é um big data, é um big data mais tratado, mais reservado, mais corporativo. Ou seja, você tem entradas no Data Lake que são de dados estruturados, os dados não estruturados e semi estruturados, e toda essa informação é armazenada em repositórios de data lake. O Data Lake precisa de um pouco mais que o Big Dat de Data Curation, que é da manutenção desses dados e algumas transformações, como catalogar, limpar dados, etc...enfim, um pouco mais de trabalho para manter esses dados.



**Exemplo Prático MongoDB**

É um banco de dados semiestruturado baseado em JSON.



**Fundamentos da Arquitetura de Aplicações em Nuvem**

Arquitetura em nuvem e sua evolução

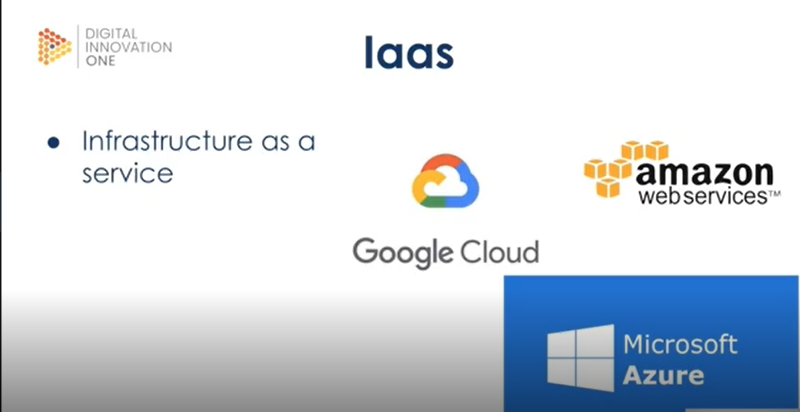
* Cloud Computing
* Disponibilidade
* Serveless

**Cloud Computing**

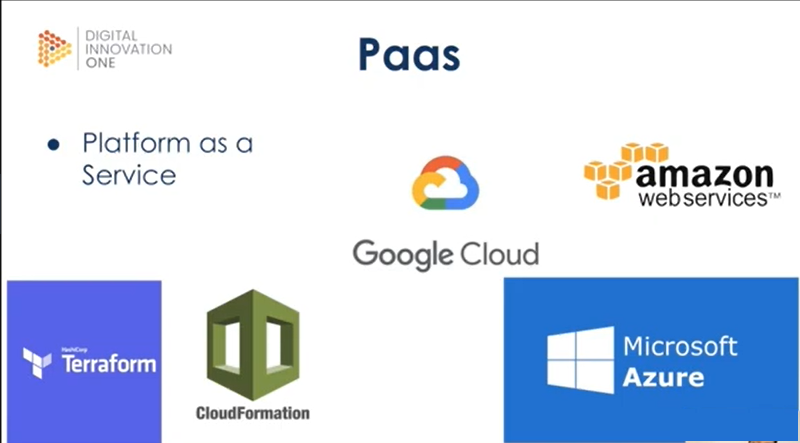
É um provedor ou um serviço que faz o gerenciamento de hardware e software para nós.

* Gerenciador de hardware e software
* Provedores de servidor e armazenamento
* Pague o que consumir
* Iaas / Paas / Baas

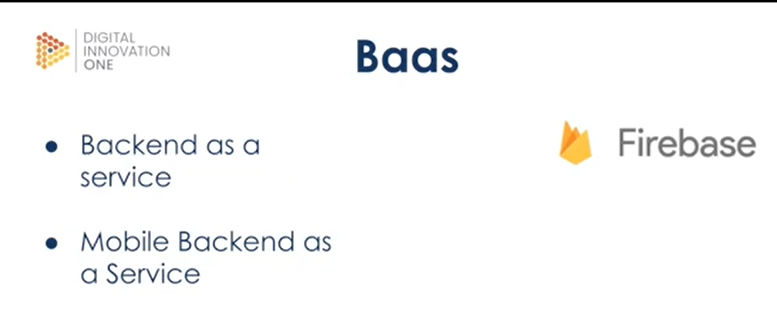
**Iaas**



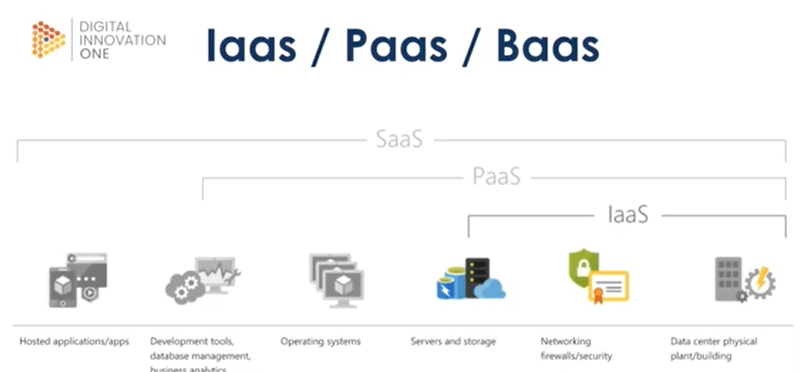
**Paas**



**Baas**



Visão Geral Iaas, Paas, Baas



Disponibilidade

* Iaas – hardware e internet
* Paas – auto scale on the go
* Baas – não há backend servisse
* Kubernets (K8S)
* Multiplos nodos
* Load balancer

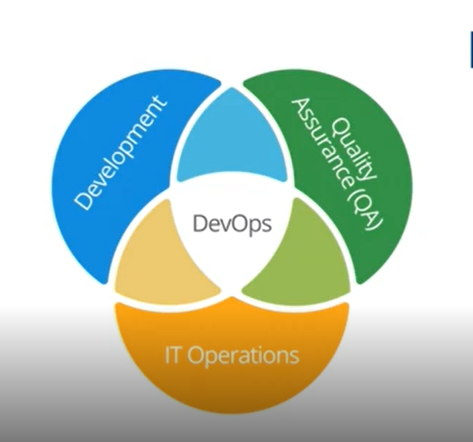
Serverless

* Sem servidor
* Sem serviço “rodando”
* Sem down time

**Desenvolvimento e operação de software integrado**

O que é DevOps?

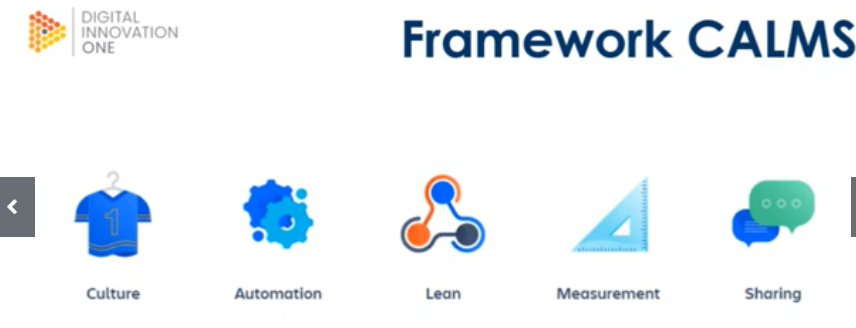
**DevOps** é um termo criado para definir o **conjunto de práticas** que integram e automatizam os processos entre as equipes de **desenvolvimento, operações e de apoio** (como QA) para a produção rápida e confiável de software.



O conceito de DevOps baseia-se em criar uma cultura de colaboração entre as equipes que sempre trabalharam separadas.

DevOps é uma mudança de mentalidade, uma cultura, um movimento, uma filosofia.

Framework Calms – framework cuja função é testar se o DevOps está sendo aplicado ou não.



* **Culture** – todas as ferramentas e automação são inúteis se não forem acompanhadas pela verdadeira disposição da área de desenvolvimento e operações em trabalhar juntos. Porque o DevOps não resolve problemas de ferramentas. Ele resolve problemas humanos.
* **Automatização** – elimina o trabalho manual repetitivo, produz processos repetitivos e cria sistemas confiáveis. Automatizar gera velocidade e entrega e tornam os envolvidos mais produtivos. Normalmente, compilação, teste, implementação e provisionamento automatizados são o ponto de partida típico para equipes que ainda não tem isso implantado.
* **Lean** – precisamos focar nas entregas de valor ao cliente. Precisamos sermos objetivos e enxutos. Precisamos conhecer as nossas limitações e os gargalos do processo. Precisamos ser lean. A mentalidade DevOps vê oportunidades de melhoria contínua em toda parte. Identificando as limitações, podemos otimizar o fluxo, entregando mais velocidade e maior eficiência.
* **Measurement** – DevOps é cíclico e infinito. Mensurar é obter métricas, é o ponto de partida para novas melhorias, seja para o processo de desenvolvimento, o software produzido ou as regras de negócios. Além de gerar conhecimento, as métricas criam previsibilidade sobre possíveis incidentes que possam vir a surgir. Assim, temos insumos suficientes para analisar falhas e gerar melhorias constantemente.
* **Sharing** – O compartilhamento de informações, além de ser saudável, auxilia a descentralização de conhecimento, em pessoas dos times, evitando que os processos se tornem dependentes. Compartilhar conhecimento ajuda na criação de times genéricos, com conhecimentos básicos em diversos assuntos de negócios e tecnologias. Assim o time se torna autossustentável.

**Os três caminhos**

* Flow
* Feedback
* Learning

**Flow**

A otimização do fluxo visa eliminar desperdícios, gargalos no processo, transferência de responsabilidades e tempos de espera. Esse caminho é trilhado entre a demanda e a entrega em produção. A chave para esse caminho é a aplicação de metodologias ágeis e a automatização dos processos do desenvolvimento à release, como a integração contínua e/ ou entrega contínua.

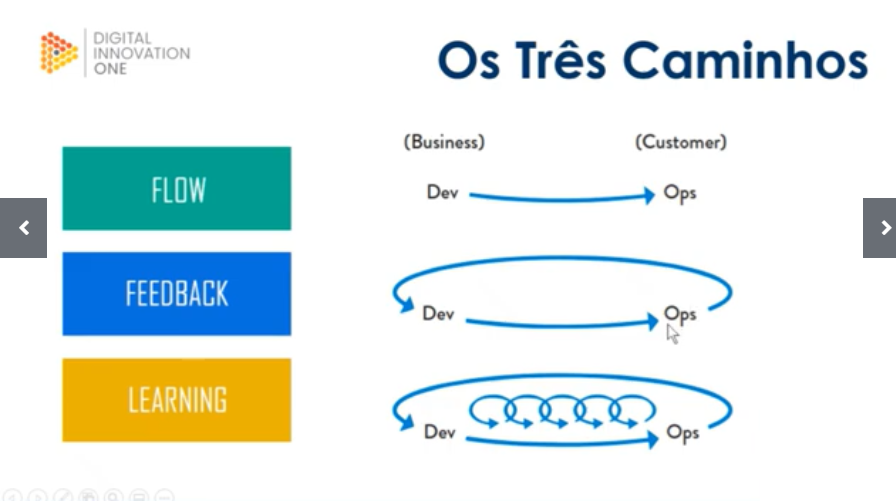
**Feedback**

Ciclos rápidos de feedbacks visam resolver problemas o quanto antes, testando tudo, alertando em qualquer falha, considerando todas as métricas coletadas no ambiente produtivo sobre o valor entregado. O monitoramento é a chave, ajudando a gerar informações relevantes constantemente. Com feedbacks constantes, o negócio consegue falhar rápido, e logo retornar o rumo, caso necessário.

**Learning**

A aprendizado contínuo visa gerar conhecimento através da experimentação. Hipóteses são melhores do que uma certeza imediata. Este caminho é fruto de processo científico e produz segurança psicológica. A chave é o trabalho dinâmico, com times realizando experimentos no seu trabalho diário para gerar novas melhorias.

Elimine a cultura da culpa e aumente a colaboração e o compartilhamento de conhecimento.

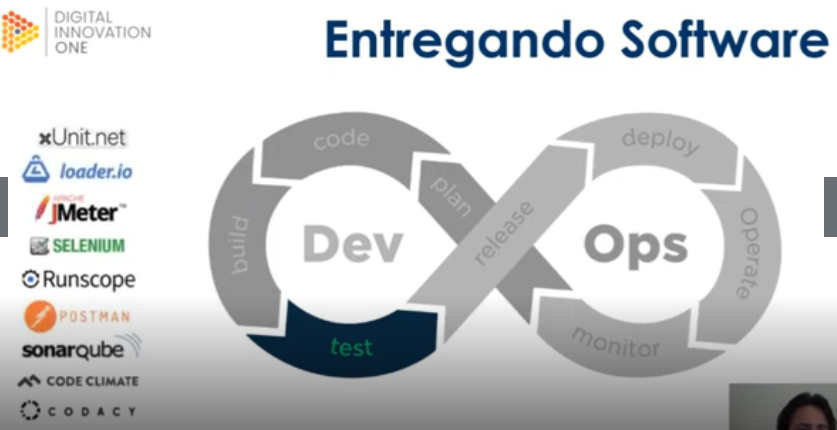


Entrega de software





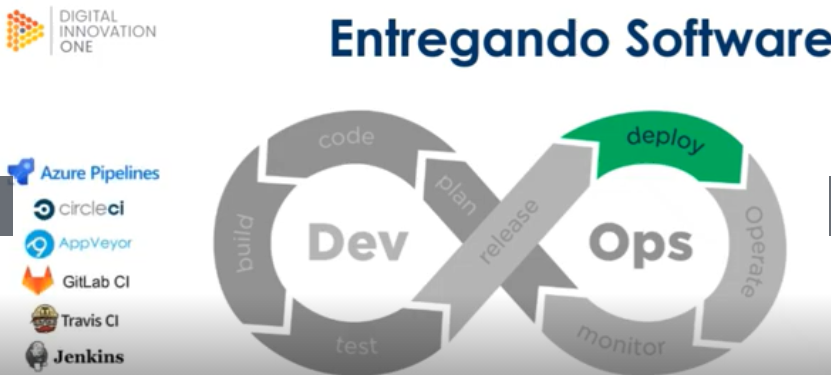




A etapa da **release** significa que depois de todas as fases: plan, code, build, test, terem passado, entrega-se ao cliente.



**Deploy** seria a implantação definitiva em produção

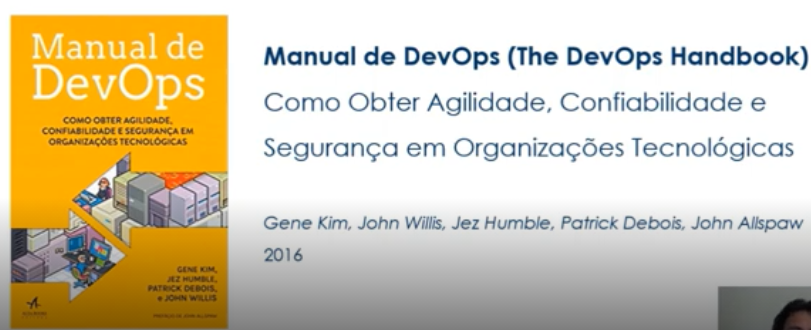


**Implantação** ou **operate** suporta todas as implantações que foram feitas

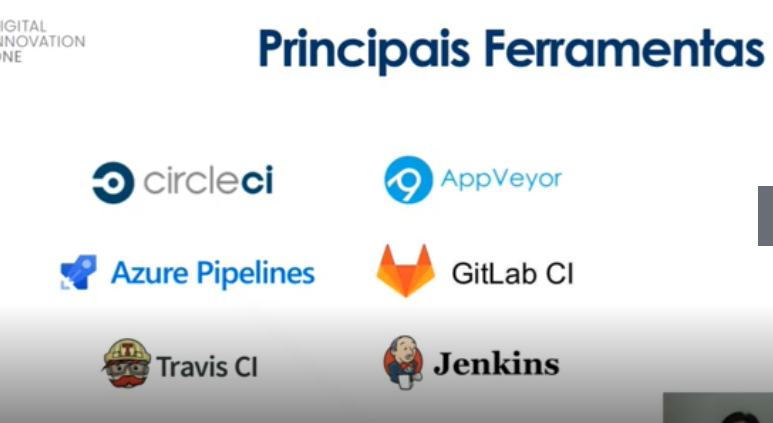




Livro de apoio



**Continuous Integration** ou **Integração Contínua**



**Continuous Inspection**



**Complexidade ciclomática** é o quanto de caminhos independentes o seu código pode seguir em um determinado pedaço da sua aplicação.

**Código duplicado**

**Vulnerabilidades / Code smell** é por exemplo ter um password amarrado ao seu código. Não é um boa prática pois você está expondo sua senha.

**Padronização e estilo**

**Débito técnico**

**Cobertura de testes**

Métricas