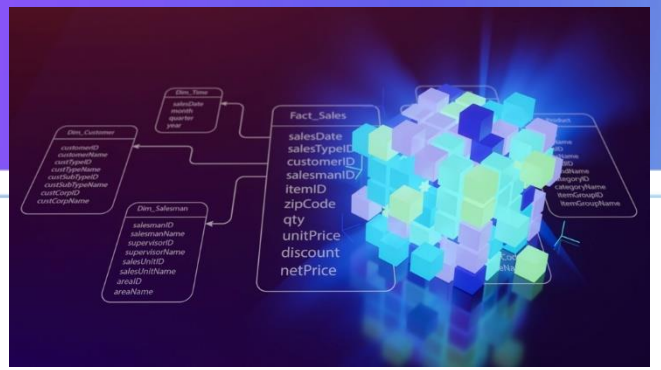
Camadas e Serviços de Consumo de Dados

Camadas em uma Arquitetura de Dados

Arquitetura de Dados



Arquitetura de dados é...

"...a prática de examinar a estratégia empresarial e identificar os principais pontos de integração de dados que precisam ser ativados para executar essa estratégia, e estabelece um roteiro para criar os principais recursos para entregar essas integrações, para que as empresas possam aproveitar os dados como um ativo estratégico."

"...a ciência de avaliar onde estão os dados da sua empresa e a arte de projetar a melhor maneira futura de armazenar dados em toda a empresa."

"...a disciplina para ajudar as empresas a gerenciar seus dados da maneira mais eficaz e maneira segura, compatível e lucrativa."

Arquitetura de Dados

Arquitetura de dados é...

"...tudo que uma empresa faz para garantir que as informações sejam precisas e disponíveis para facilitar propósitos comerciais válidos, conforme definido por regulamentos/legais requisitos, proposições de valor das partes interessadas e clientes (internos/externo)."

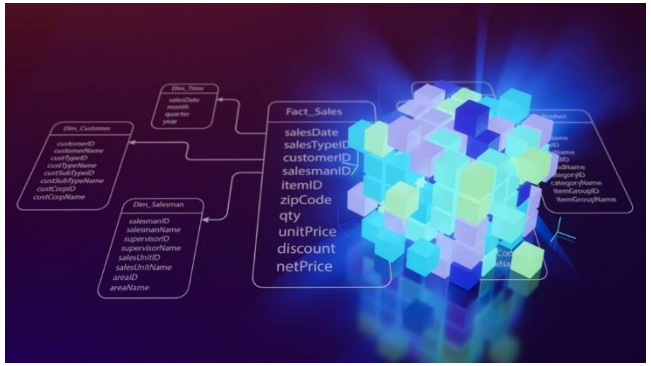
"...um conjunto integrado de artefatos de especificação que definem requisitos de dados estratégicos, orienta a integração de ativos de dados e alinha investimentos de dados com a estratégia de negócio.

"...o projeto que analisa todo o cenário de dados e governa o ciclo de vida dos dados. Descreverá a linguagem a ser usada pelas partes interessadas. Inclui as pessoas, processos, ferramentas, tecnologias e artefatos necessários".

Arquitetura de Dados

Arquitetura de dados é…

“…regras, políticas e padrões que regem:



1. Dados e documentação de requisitos para sistemas;

2. Projeto de banco de dados e modelagem;

3. Documentação, gestão e disseminação/acesso de metadados;

4. Coleta, uso e integração de dados em toda a empresa;

5. Qualidade dos dados, definição e aplicação da Governança de Dados;

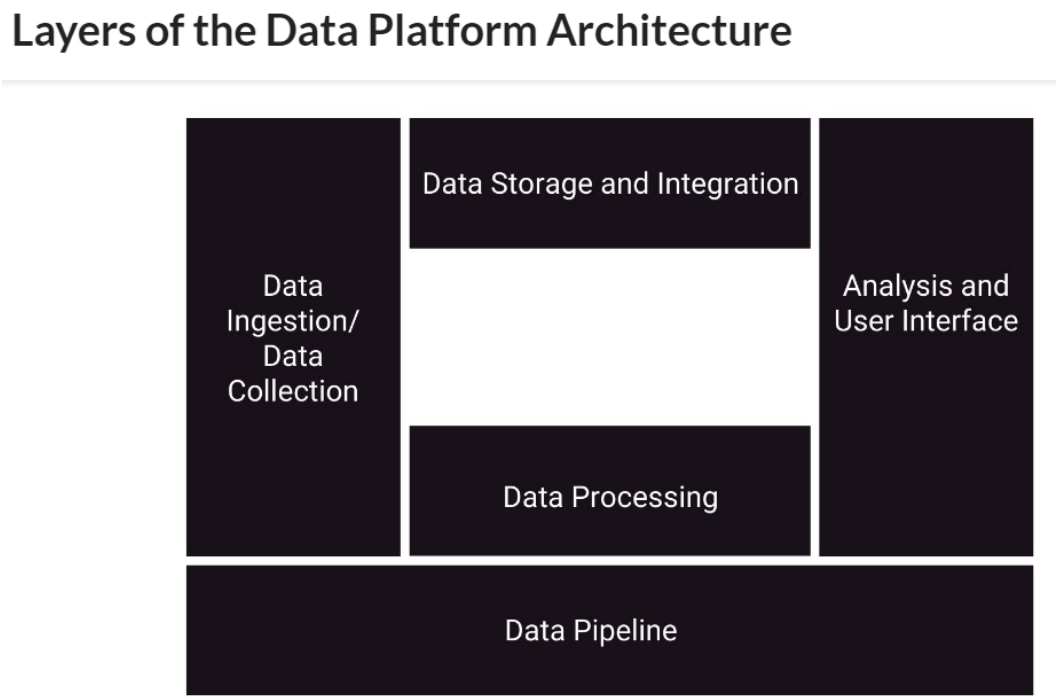
6. Conformidade e retenção de arquivamento de dados - LGPD;

7. Gerenciamento de Dados Mestres.”

Camadas em uma Arquitetura de Dados

As diferentes camadas da arquitetura da plataforma de dados incluem:

- **Camada de ingestão de dados.**
- **Camada de armazenamento de dados.**
- **Camada de processamento e análise de dados.**
- **Camada de interface do usuário.**
- **Camada de pipeline de dados.**



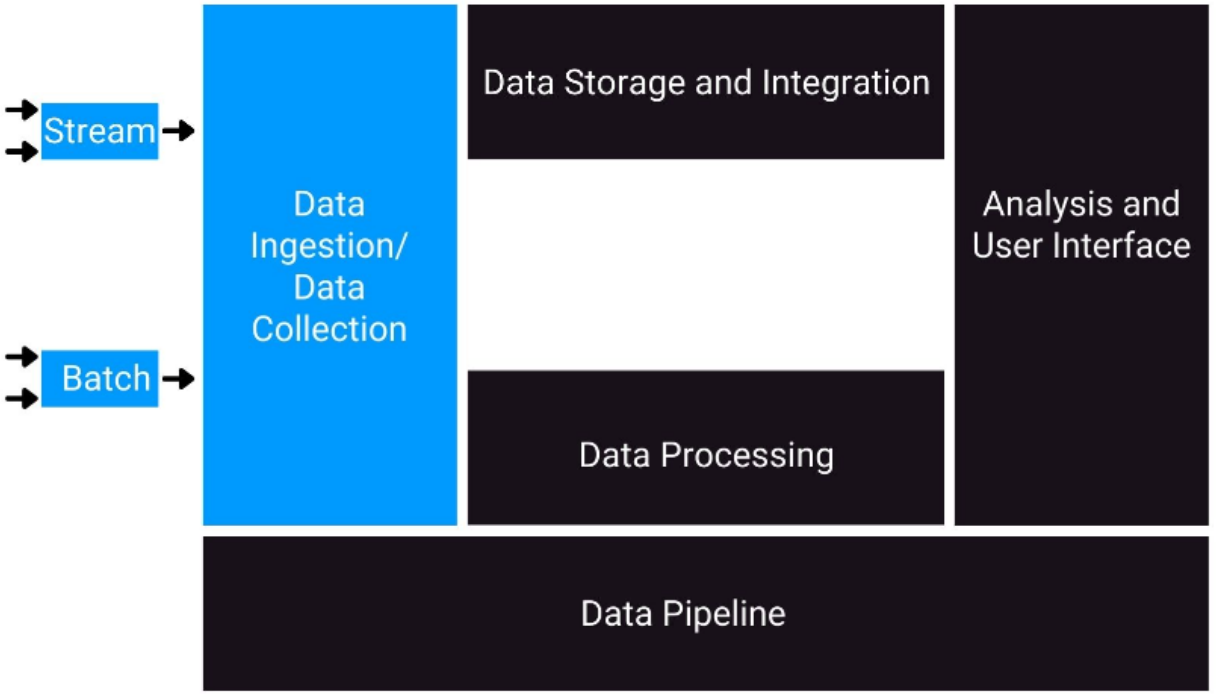
Fonte: https://www.analyticsvidhya.com

Camada de Ingestão / Coleta de Dados

Ingestão / Coleta de Dados

Esta é a primeira camada da arquitetura da plataforma de dados.

A camada de coleta de dados, como o nome sugere, é responsável por conectar-se aos sistemas de origem e trazer dados para a plataforma de dados de maneira periódica.



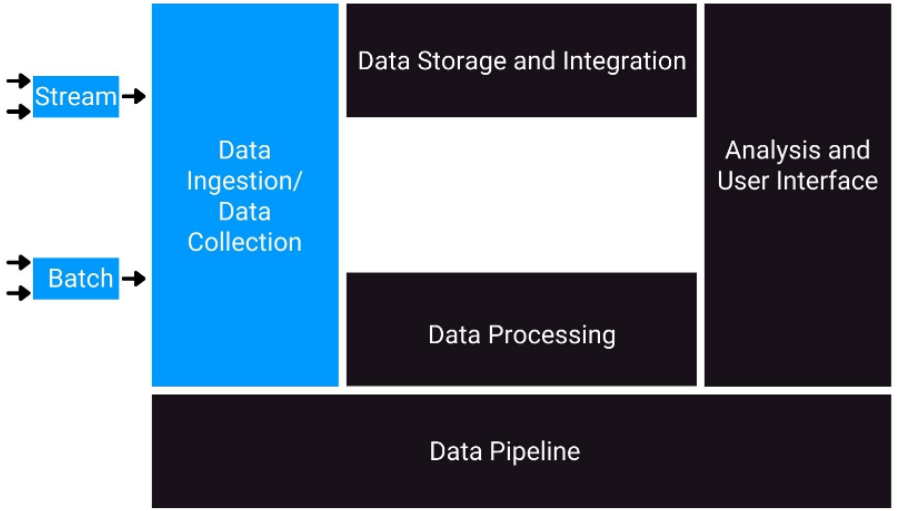
Fonte: https://www.analyticsvidhya.com

Ingestão / Coleta de Dados

Esta camada executa as seguintes tarefas:

- **É responsável pela conexão com as fontes de dados.**
- **É responsável pela transferência de dados das fontes de dados para a plataforma de dados no modo streaming, no modo batch ou em ambos.**
- **É responsável por manter as informações sobre os dados coletados no repositório de metadados.**

Por exemplo, quantos dados são coletados na plataforma de dados e outras informações descritivas?



Fonte: https://www.analyticsvidhya.com

Ingestão / Coleta de Dados

Existem várias ferramentas disponíveis no mercado.

Como ferramentas de podemos citar:

- **Google Cloud Data Flow**
- **IBM Streams**
- **Amazon Kinesis**
- **Apache Kafka**

Fonte: https://www.analyticsvidhya.com

São algumas das ferramentas mais usadas para ingestão de dados que suportam modos batch e streaming.

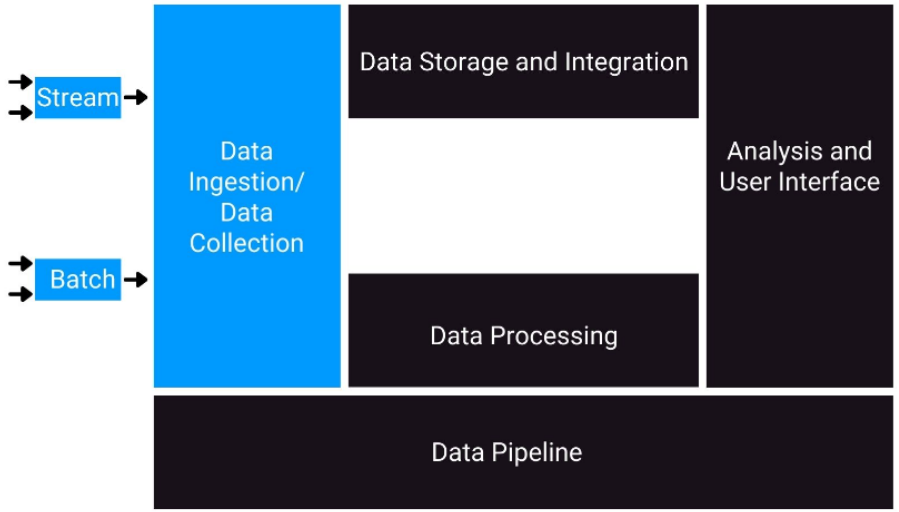
Batch e Stream

Batch e Stream

Qual é a diferença entre batch e stream?

O batch é um lote de pontos de dados que foram agrupados em um intervalo de tempo específico. Outro termo frequentemente usado para isso é uma janela de dados.

Já o processamento de dados em stream lida com dados contínuos e é essencial para transformar de grandes a rápidos.

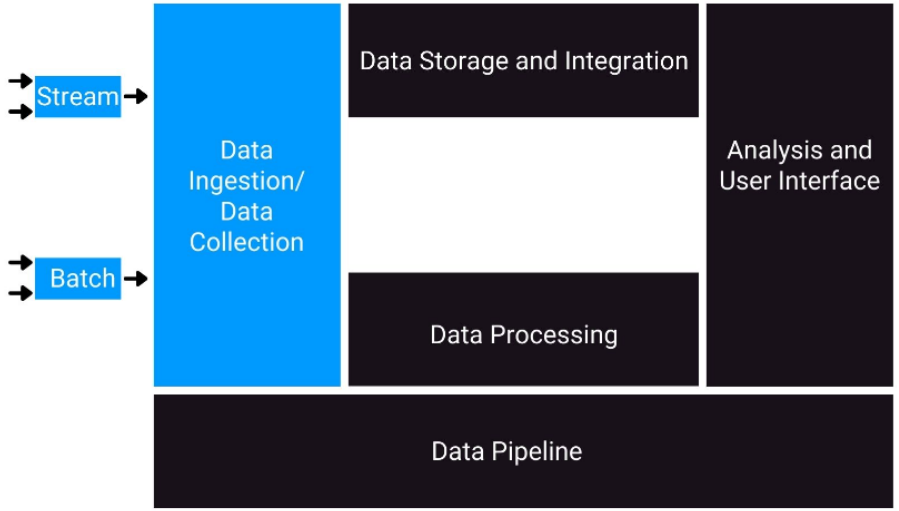


Fonte: https://www.analyticsvidhya.com

Batch e Stream

Embora o modelo de processamento em batch exija um conjunto de dados coletados ao longo do tempo, o processamento de stream requer que os dados sejam inseridos em uma ferramenta de análise, geralmente, em lotes menores e em tempo real.

Nesse caso, é comum acontecer que as equipes de TI fiquem paradas esperando que todos os dados sejam carregados antes de iniciar a fase de análise.



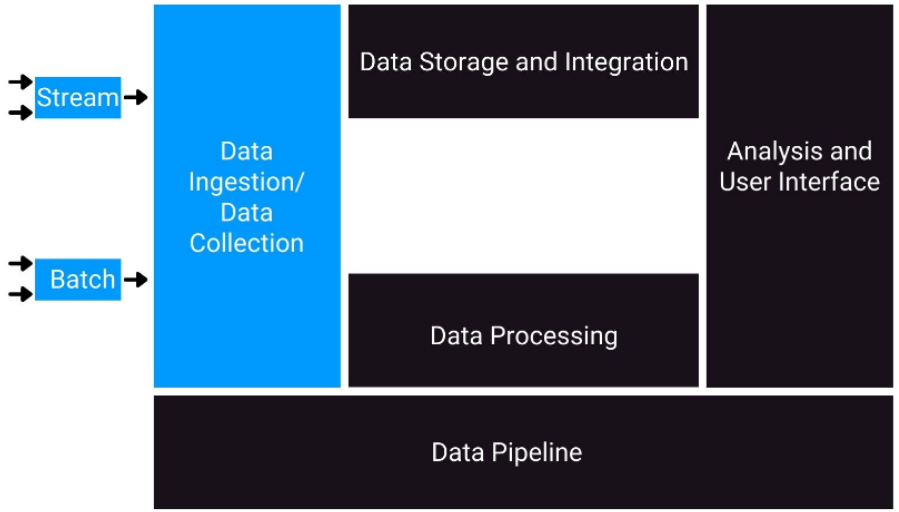
Fonte: https://www.analyticsvidhya.com

Batch e Stream

O stream também podem estar envolvido no processamento de grandes quantidades de dados.

Já o batch funciona melhor quando você não precisa de análises em tempo real.

Como o processamento stream é responsável pelo processamento de dados em movimento e pelo rápido fornecimento de análise, ele gera resultados quase instantâneos, por exemplo usando plataformas de Business Intelligence.



Fonte: https://www.analyticsvidhya.com

Quando escolher entre Batch e Stream

Por que não podemos apenas o batch como nos sistemas tradicionais ?

Você certamente pode, mas se tiver enormes volumes de dados, não é uma questão de quando você precisar extraí-los, mas quando precisará usá-los.

As empresas veem os dados em tempo real como um divisor de águas, mas ainda pode ser um desafio chegar lá sem as ferramentas adequadas, principalmente, porque elas precisam trabalhar com volumes, variedades e tipos de dados cada vez maiores de diversos sistemas, como mídias sociais.

Normalmente, as organizações querem ter processos de dados mais ágeis para poderem passar da imaginação para a inovação mais rapidamente e responder às ameaças da concorrência.

Quando escolher entre Batch e Stream

Por exemplo, os dados enviados dos sensores de uma turbina eólica estão sempre ligados. Assim, o fluxo de dados é ininterrupto e flui o tempo todo. Uma abordagem em batch aqui seria obsoleta, pois não há início ou parada do fluxo. Esse é um caso de uso perfeito em que o processamento de stream é o caminho mais adequado.

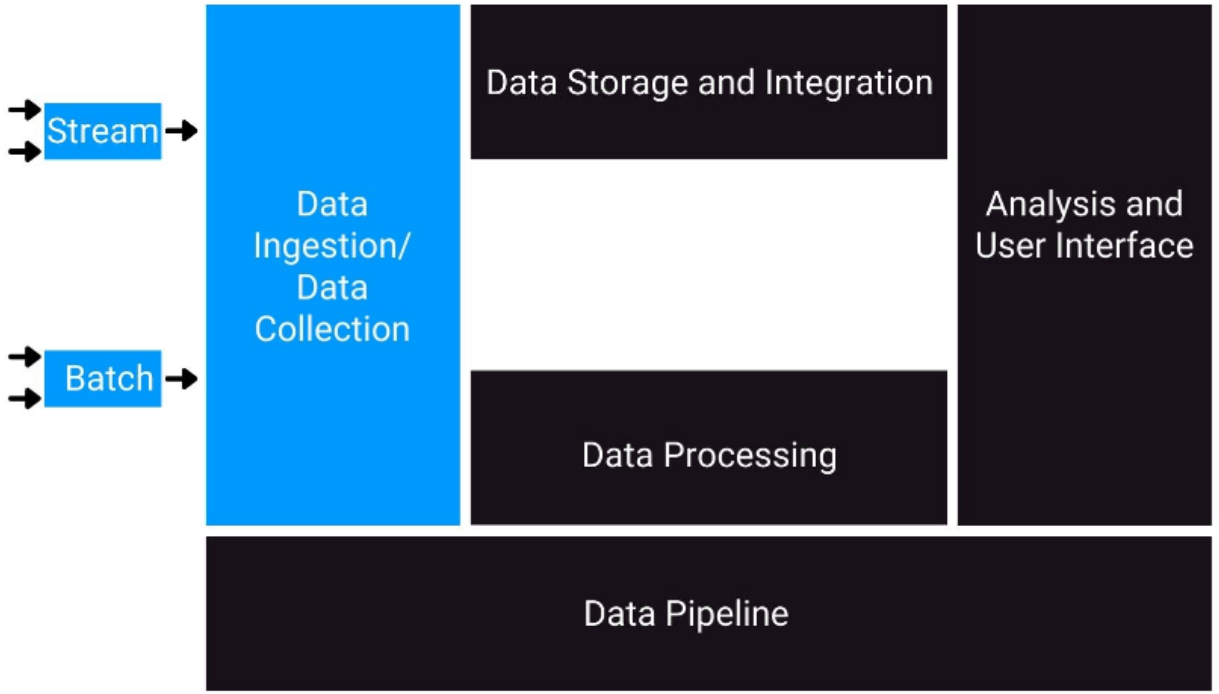
O processamento de dados em batch e stream são dois modelos diferentes — não é uma questão de escolher um sobre o outro, é sobre ser assertivo e determinar qual é o melhor para cada caso de uso.

Camada de Armazenamento e Integração de Dados

Ingestão / Coleta de Dados

Depois que os dados são coletados (data ingestion), eles precisam ser armazenados e integrados à plataforma de dados.

Para armazenar e integrar os dados, nos dirigimos para a segunda camada da plataforma de dados que é a camada de armazenamento de dados ou camada de integração de dados.



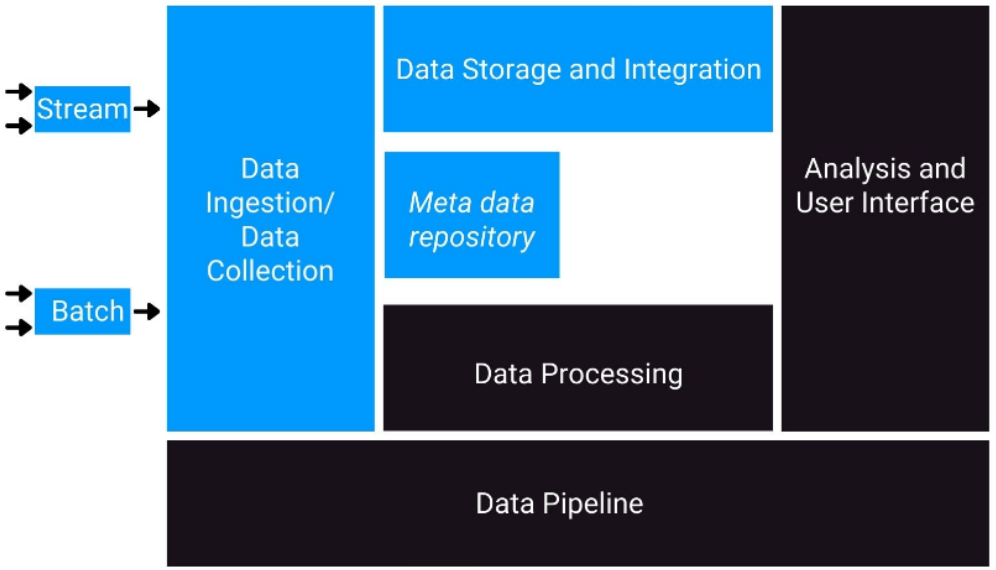
Fonte: https://www.analyticsvidhya.com

Armazenamento e Integração de Dados

A camada de coleta de dados, como o nome sugere, é responsável por armazenar dados para processamento e uso a longo prazo.

Além disso, essa camada também é responsável por disponibilizar os dados para processamento nos modos streaming e batch.

Como essa camada é responsável por disponibilizar os dados para processamento, ela precisa ser confiável, escalável, de alto desempenho e econômica.



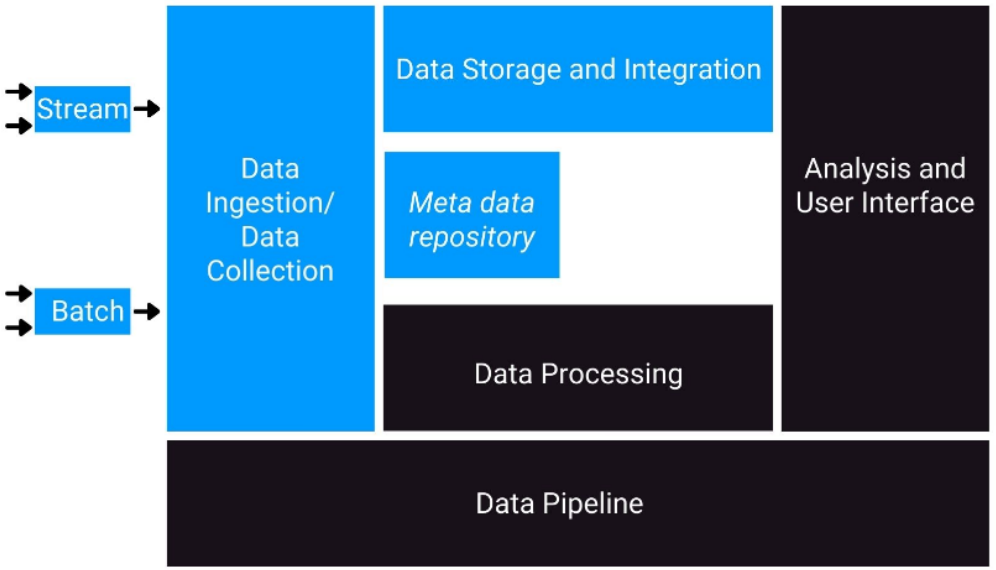
Fonte: https://www.analyticsvidhya.com

Armazenamento e Integração de Dados

IBM DB2, Microsoft SQL Server, MySQL, Oracle Database e PostgreSQL são alguns dos bancos de dados relacionais populares.

Nos últimos anos, os bancos de dados relacionais baseados em nuvem vem ganhando popularidade - IBM DB2, Google Cloud SQL e SQL Azure.

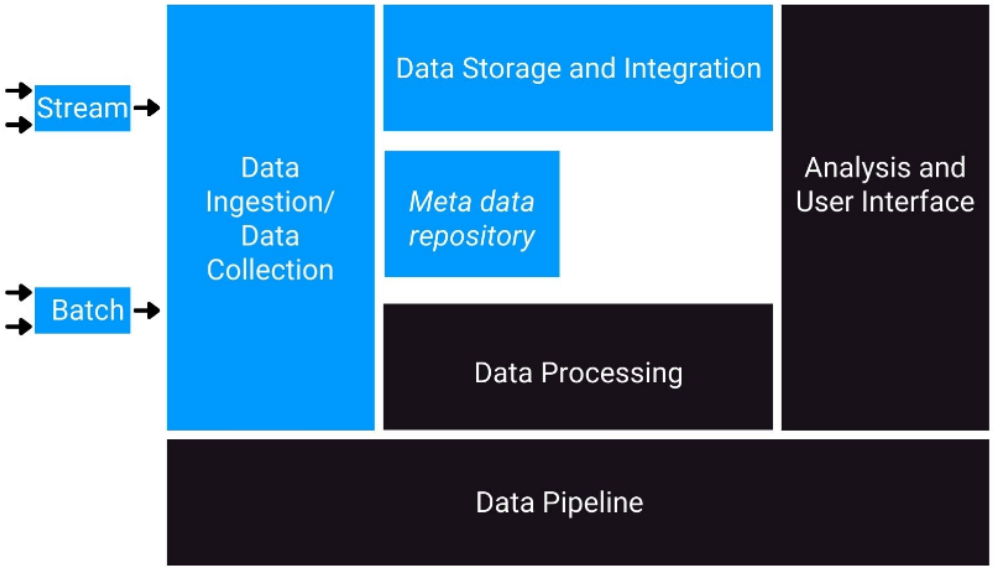
Também temso os sistemas de banco de dados NoSQL ou não relacional na nuvem, temos IBM Cloudant, Redis, MongoDB, Cassandra e Neo4J.



Fonte: https://www.analyticsvidhya.com

Armazenamento e Integração de Dados

As ferramentas para integração incluem o IBM Cloud Pak for Integration e o Open Studio da IBM. Uma vez que os dados tenham sido ingeridos, armazenados e integrados, eles precisam ser processados. Com isso, avançamos para a Camada de Processamento de Dados.



Fonte: https://www.analyticsvidhya.com

Armazenamento e Integração de Dados

Por exemplo, o software de integração do Cloud Pak for Integration fornece um conjunto abrangente de ferramentas de integração, conectando aplicações e dados em qualquer ambiente de nuvem ou no local. É uma plataforma cloud impulsionada por IA que permite a automação das operações de rede para que os provedores de serviços de comunicação (CSPs) possam fornecer serviços mais rapidamente.

A plataforma oferece interações de eventos em tempo real, transfere dados em qualquer nuvem, implementa e escala com arquitetura nativa em nuvem e disponibiliza serviços básicos compartilhados, tudo com segurança e criptografia de classificação corporativa de ponta a ponta.

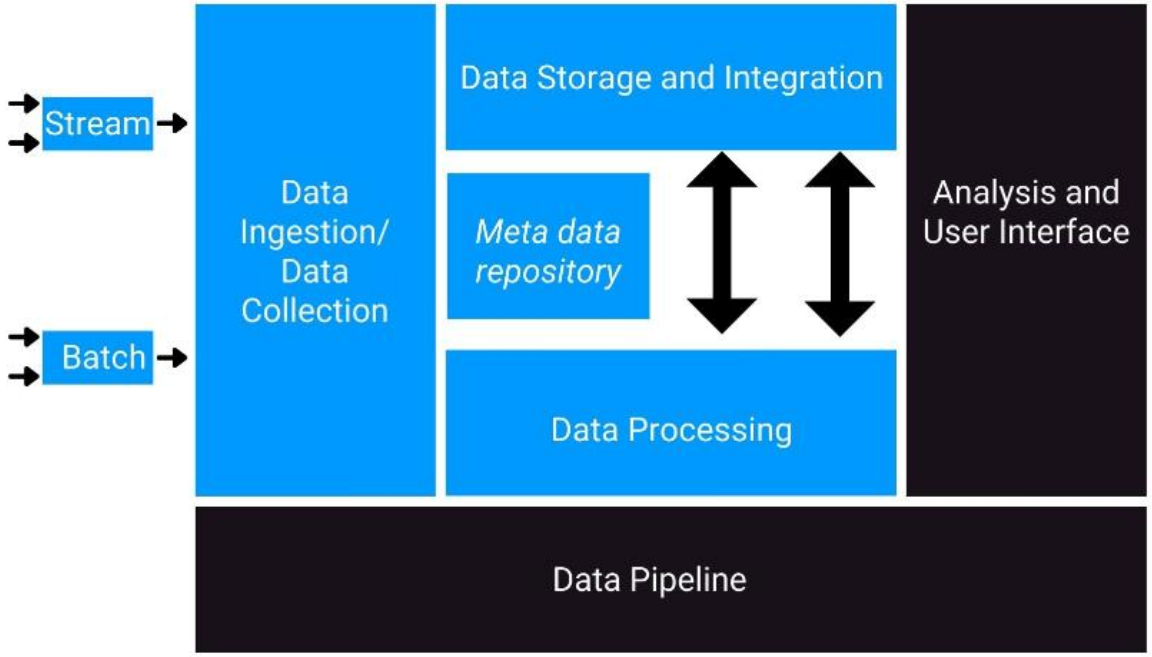
O Cloud Pak for Integration usa serviços e recursos da AWS, inclusive Virtual Private Clouds (VPCs), zonas de disponibilidade, grupos de segurança, o Amazon Elastic Block Store (Amazon EBS), o Amazon Elastic Compute Cloud (Amazon EC2) e o Elastic Load Balancing para criar uma plataforma de nuvem confiável e escalável.

Camada de Processamento de Dados

Processamento de Dados

Depois que os dados são coletados (data ingestion), eles precisam ser armazenados e integrados à plataforma de dados.

Para armazenar e integrar os dados, nos dirigimos para a segunda camada da plataforma de dados que é a camada de armazenamento de dados ou camada de integração de dados.



Fonte: https://www.analyticsvidhya.com

Processamento de Dados

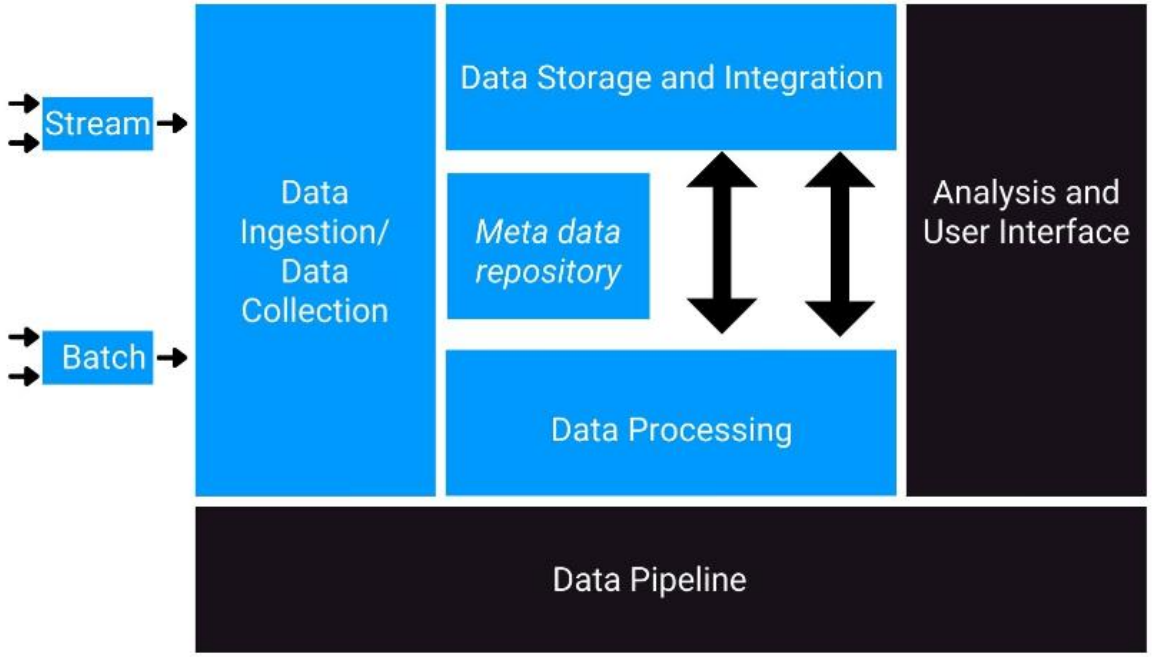
Esta camada é responsável por uma tarefa de processamento.

O processamento inclui validações de dados, transformações e aplicação de lógica de negócios aos dados.

A camada de processamento deve ser capaz de executar algumas tarefas que incluem:

Leia dados em batch (lote) ou modos de streaming do armazenamento e aplique transformações.

Suporte a ferramentas de consulta e linguagens de programação populares.



Fonte: https://www.analyticsvidhya.com

Processamento de Dados

A camada de processamento deve ser capaz de executar algumas tarefas que incluem:

- **Ler dados em batch (lote) ou modos de streaming do armazenamento e aplique transformações.**
- **Suporte a ferramentas de consulta e linguagens de programação populares.**
- **Escale para atender às demandas de processamento de um conjunto de dados crescente.**
- **Forneça uma maneira para analistas e cientistas de dados trabalharem com dados na plataforma de dados.**

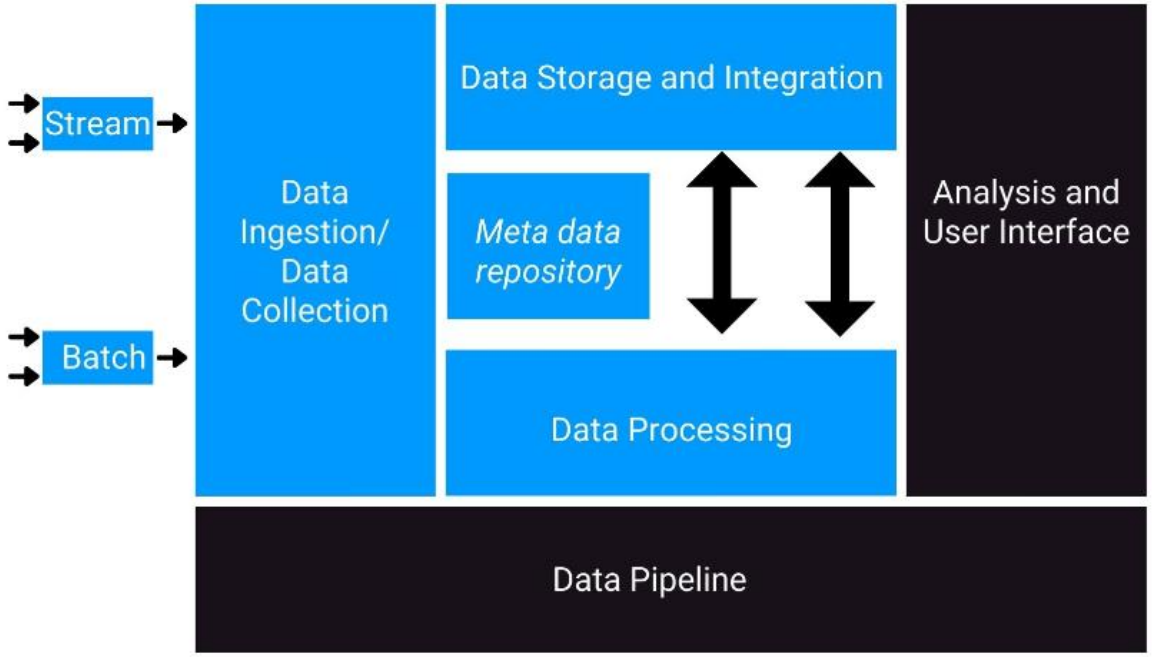
Processamento de Dados

As tarefas de transformação que geralmente ocorrem nesta camada incluem:

- **Estruturação: são as ações que alteram a estrutura dos dados. Essa mudança pode ser de natureza simples ou complexa. O simples também pode ser como alterar a disposição dos campos dentro do registro ou conjunto de dados ou complexo como combinar estruturas complexas de campos usando junções e uniões.**
- **Normalização: esta parte se concentra na redução da redundância e inconsistência. Ele também se concentra na limpeza do banco de dados de dados não utilizados.**
- **Desnormalização: a desnormalização é a tarefa de combinar dados de várias tabelas em uma única tabela para que ela possa ser consultada com mais eficiência para fins de relatório e análise.**
- **Limpeza de dados: corrige irregularidades nos dados para fornecer dados confiáveis para aplicativos e usos posteriores.**

Processamento de Dados

Existem inúmeras ferramentas disponíveis no mercado para realizar essas operações nos dados, incluindo planilhas, OpenRefine, Google DataPrep, Watson Studio Refinery e Trifacta Wrangler. Python e R também oferecem várias bibliotecas e pacotes criados explicitamente para processar dados.

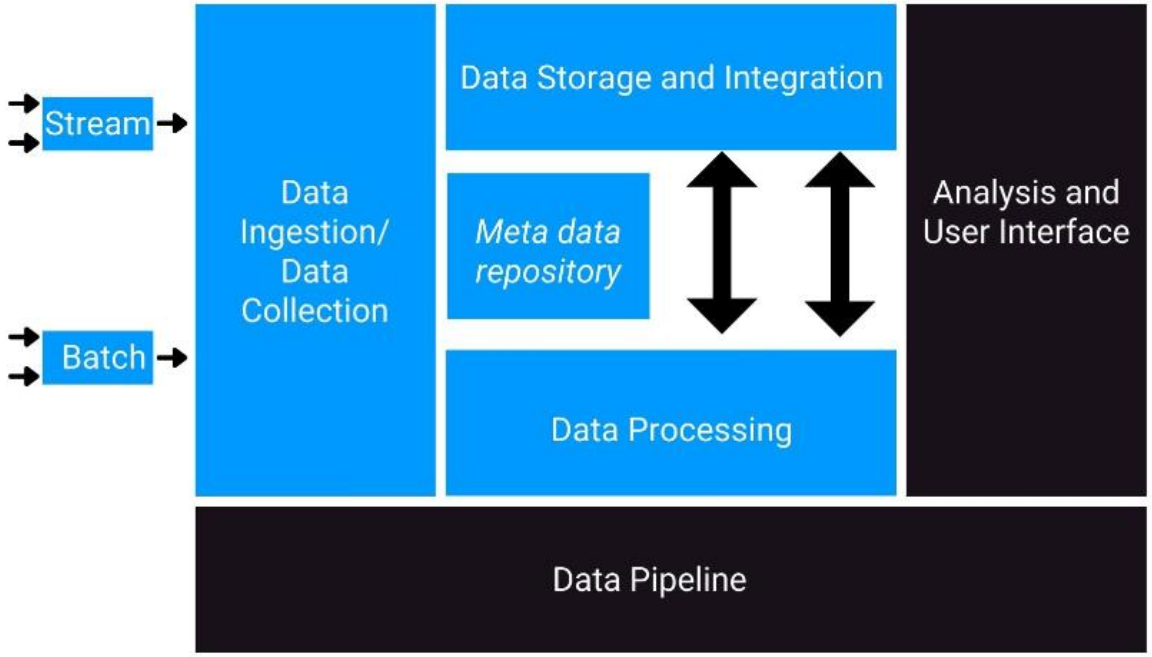


Fonte: https://www.analyticsvidhya.com

Processamento de Dados

É muito importante saber que o armazenamento e o processamento nem sempre são realizados em camadas separadas.

Por exemplo, em bancos de dados relacionais, o armazenamento e o processamento ocorrem na mesma camada, enquanto nos sistemas de big data, os dados são primeiro armazenados no sistema Hadoop File Distributed e depois processados no mecanismo de processamento de dados, como o Spark.



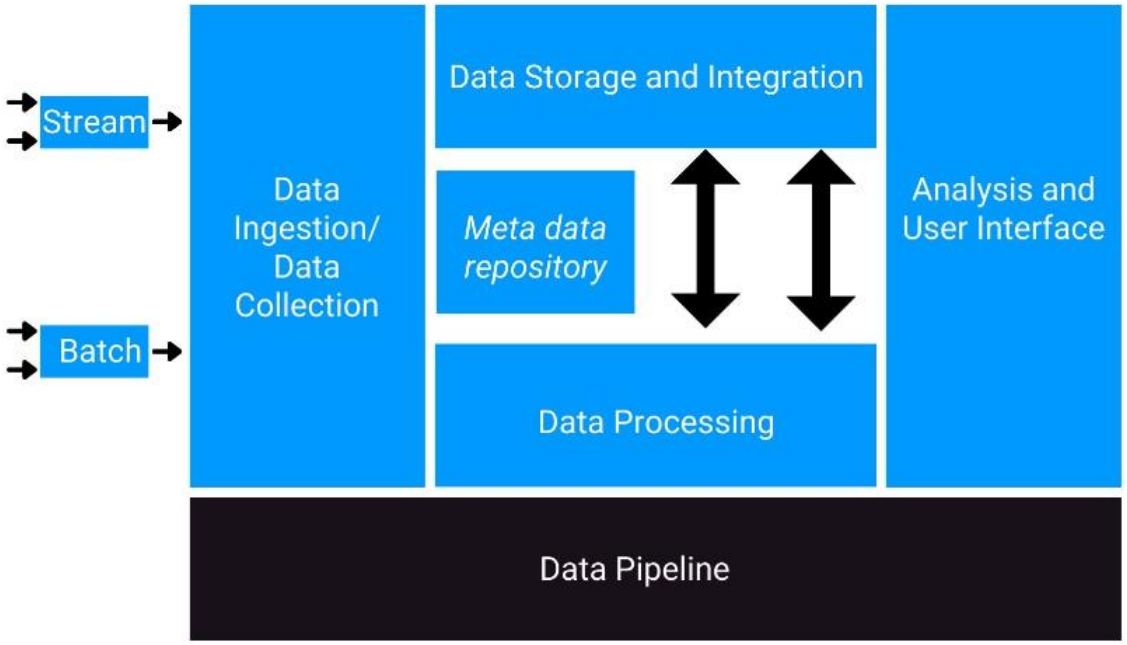
Fonte: https://www.analyticsvidhya.com

Camada de Análise e Interface do Usuário

Análise e Interface do Usuário

Essa camada é responsável por fornecer os dados do processo aos usuários finais, incluindo analistas de inteligência de negócios e partes interessadas nos negócios, que consomem esses dados com a ajuda de painéis e relatórios interativos.

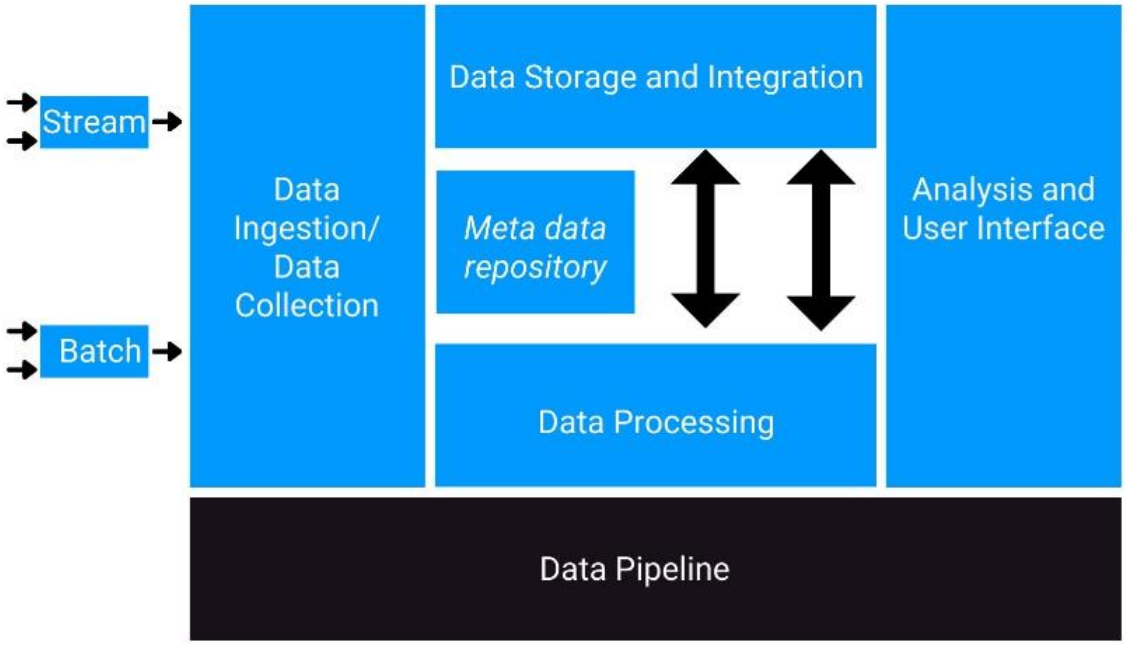
Além disso, cientistas de dados e analistas de dados se enquadram nessa categoria de usuário final que processe posteriormente esses dados para o caso de uso específico.



Fonte: https://www.analyticsvidhya.com

Análise e Interface do Usuário

Essa camada precisa oferecer suporte a ferramentas de consulta, como ferramentas SQL e ferramentas No-SQL e linguagens de programação como Python, R e Java e, além disso, essas camadas precisam oferecer suporte a APIs que podem ser usadas para executar relatórios sobre dados para processamento online e offline.



Fonte: https://www.analyticsvidhya.com

Tableu

https://www.tableau.com

A empresa Tableau Software, foi fundada em 2003, por Chris Stolte, Pat Hanrahan e Christian Chabot que tinham como intenção fornecer as visualizações dos dados do departamento de computação da Universidade de Stanford. Pat Hanrahan foi um dos membros fundadores da empresa de animação digital Pixar, então eles tinham a visão que a computação gráfica poderia ser de ajuda enorme na compreensão dos dados nas organizações (Tableau, 2017).

Como acontece com o Power BI, o grupo queria que o software fosse de fácil adaptação, onde qualquer pessoa familiarizada com o Excel poderia criar análises consistentes dos resultados de uma organização. Ficando a cargo da equipe de TI apenas as questões que envolvem a estruturação dos dados, como o DW e se necessário a criação de cubos OLAP.

Power BI

https://powerbi.microsoft.com/pt-br

Juntamente com o Tableau são os líderes no mercado de Business Intelligence. Começou a ser desenvolvido em 2010 com o codinome “Project Crescent”, porém só teve seu lançamento em 2013 e com outro nome Microsoft Power BI para o Office 365. No começo, era como uma parte do software de gerenciamento de planilhas da empresa, Microsoft Excel, com as funções PowerView, PowerQuery e Power Pivot. Desde lá o Power BI se tornou um software completo para uma implementação de BI, não só contando com apenas a visualização das informações, mas também com a criação do DW e processo de ETL e do cubo OLAP. A figura 11, mostra a logo adotada pela Microsoft para o Power BI.

Camada de Pipeline de Dados

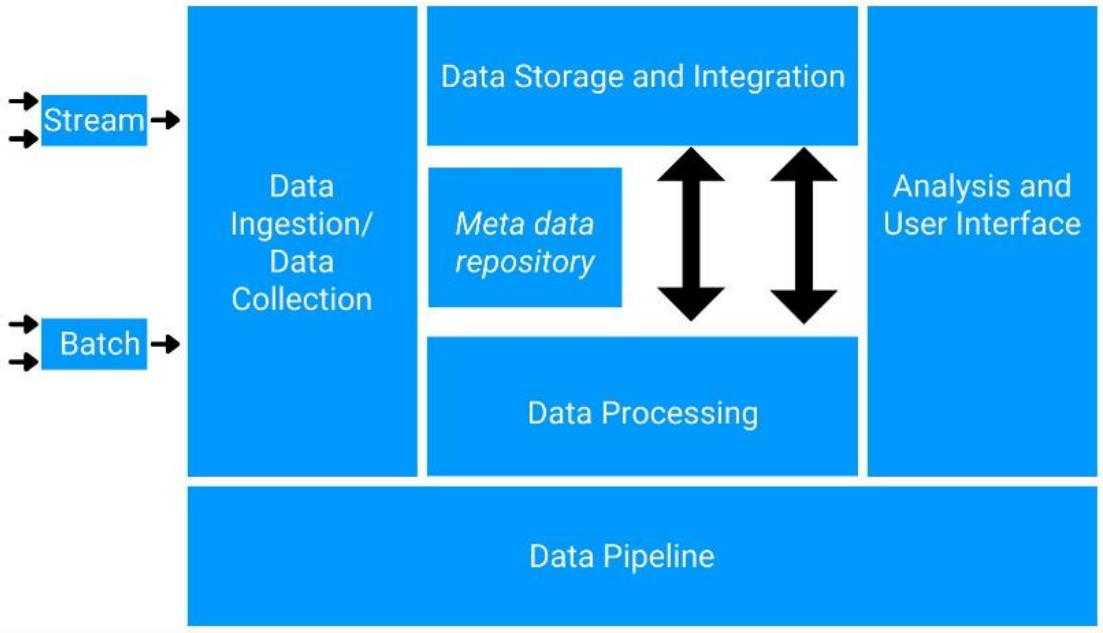
Pipeline de Dados

Esta é a última camada desta arquitetura

Esta camada é responsável por implementar e manter um fluxo contínuo de dados através deste pipeline de dados.

É a camada que tem a capacidade de extrair, transformar e carregar ferramentas.

Há uma série de soluções de pipeline de dados disponíveis, sendo as mais populares Apache Airflow e DataFlow.



Fonte: https://www.analyticsvidhya.com

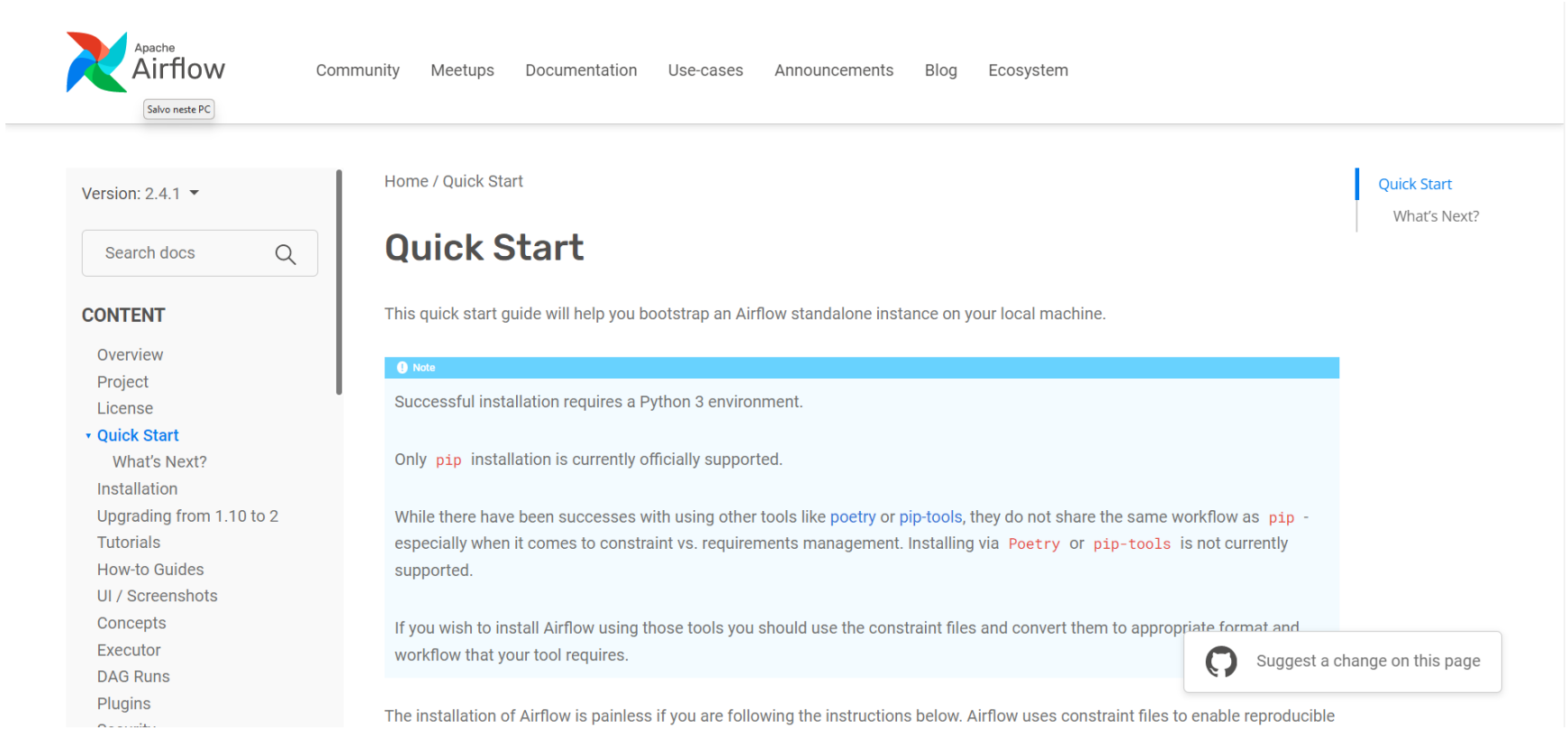
Apache Airflow

- https://airflow.apache.org/

- O Airflow é uma plataforma criada pela comunidade para criar, agendar e monitorar de forma agendada os fluxos de trabalho.
- O Airflow tem uma arquitetura modular e usa uma fila de mensagens para orquestrar um número arbitrário de trabalhadores. O fluxo de ar está pronto para escalar até o infinito.
- Os pipelines de fluxo de dados são definidos em Python, permitindo a geração dinâmica de pipeline. Isso permite escrever código que instancia pipelines dinamicamente.

Apache Airflow

- https://airflow.apache.org/docs/apache-airflow/stable/start.html

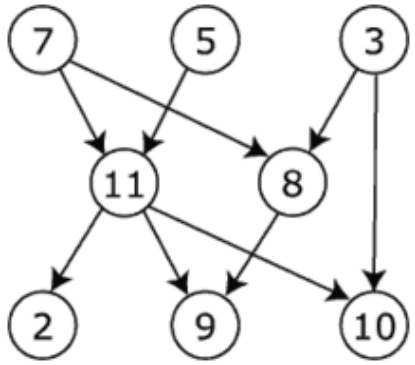


Apache Airflow

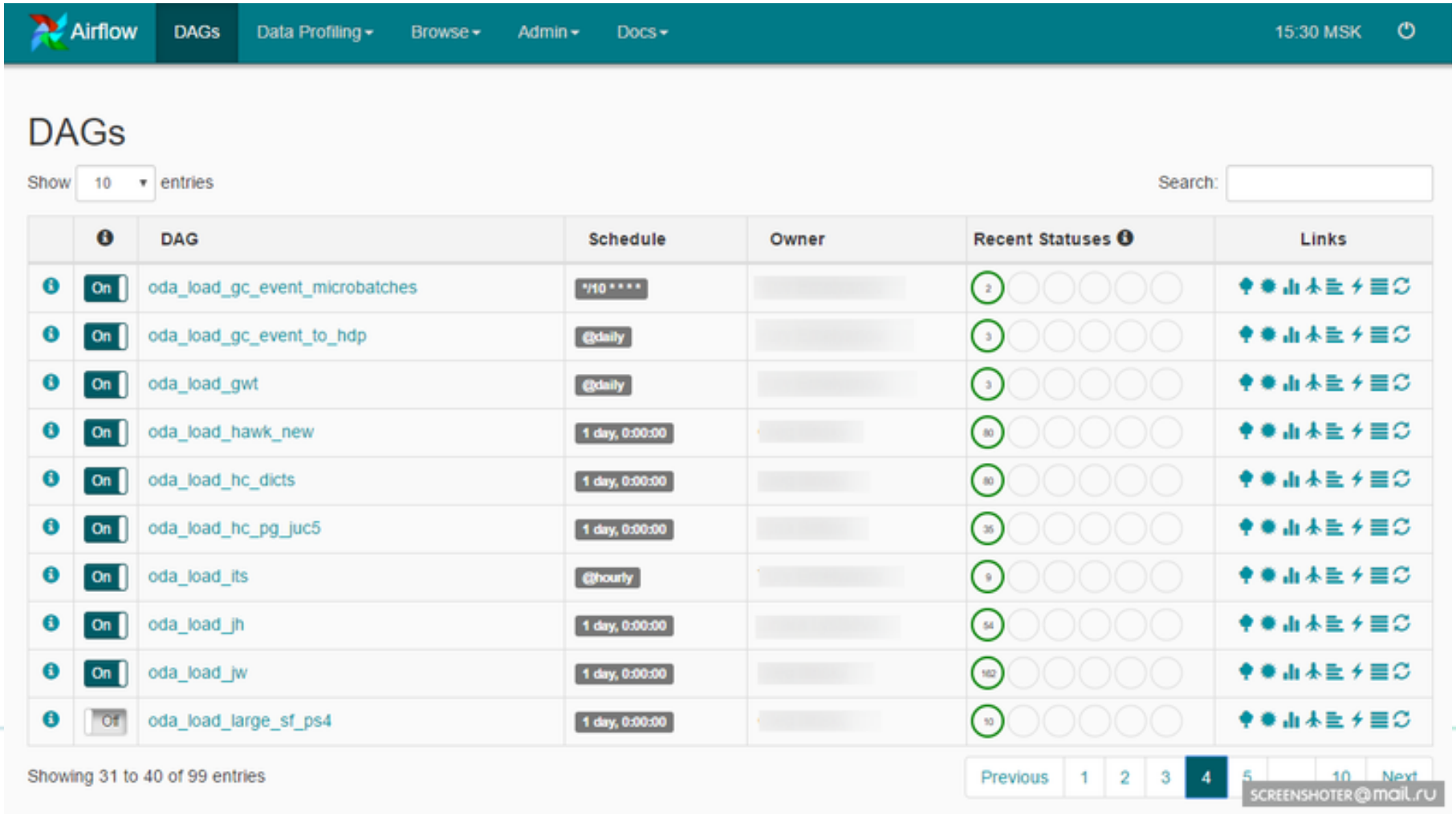
- https://airflow.apache.org/
- DAG - Grafos de tarefas/ usos.
- OPERATOR - O operador refere-se à etapa de transformação ainda dividida em
  - Sensor - Este tipo de operador realiza uma função de polling com frequência/timeout.
  - Executor - Este tipo de operador realiza operações de trigger, por exemplo, HiveOperator, Pig Operator.
  - TASK - Task é a principal entidade do DAG. O principal aqui é a instância de tarefa considerada para executar uma tarefa em um ponto do tempo.
  - HOOK - É considerado como a Interface para o Sistema externo como um gancho de JDBC e HTTP.

Apache Airflow - DAG

- Em ciência da computação e matemática, um grafo acíclico direcionado (DAG – directed acyclic graphs) é um grafo direcionado que não possui ciclos direcionados.
- Na teoria dos grafos, um grafo refere-se a um conjunto de vértices conectados por linhas chamadas arestas.
- Em um grafo direcionado, cada aresta está associada a uma direção de um vértice inicial a um vértice final.
- Se viajarmos ao longo da direção das arestas e descobrirmos que não se formam laços fechados ao longo de qualquer caminho, diz-se que não há ciclos direcionados.
- E o gráfico formado é um DAG.



Apache Airflow



Benefícios do Apache Airflow

- **Dinâmico** - O pipeline construído pelo **Airflow dynamic**, construído na forma de código que dá uma vantagem para ser dinâmico.
- **Extensível** - é fácil iniciar os operadores, executores devido aos quais a biblioteca foi impulsionada para que possa se adequar ao nível de abstração para suportar um ambiente definido.
- **Elegante** – um pipeline desenvolvido com a ajuda do Airflow é angular e inequívoco porque o mecanismo de modelo Jinja usado para parametrizar os scripts são incorporados ao núcleo do Airflow.
- **Escalável** - a arquitetura do Airflow é composta por unidades padronizadas que também utilizam a técnica de mensagens para enfileirar o número de trabalhadores e além disso é escalável ao infinito.

Apache Airflow

- O Apache Airflow é compatível com gerenciamento de dependências, extensível, **escalável e de código aberto.**
- As principais funções do Apache Airflow são **agendar fluxo de trabalho, monitorar e criar**.
- Essas funções são alcançadas **com Grafos Acíclicos Dirigidos (DAG) das tarefas.**

Barramentos de Mensageria de Dados

Mensageria de Alta Performance

O mundo tem gerado um volume cada vez maior de informações.
Um cenário comum é que estes dados, uma vez gerados, precisam ser transportados para diversas aplicações.
Por exemplo, quando fazemos uma compra com cartão de crédito a operadora precisa avisar várias aplicações diferentes para contabilizar a compra, os benefícios e até a análise contra fraudes.
Neste caso, é desejável que cada aplicação seja responsável pela implementação das regras do negócio. E apenas isso.
Uma aplicação não deveria se preocupar com a entrega ou captura dos dados. Este é um problema complexo envolvendo **data streaming.**

Mensageria de Alta Performance

Data streaming

Um fluxo de dados constante e sem controle é chamada de data streaming.
Sua principal característica é que os dados não tem limites definidos, assim, não é possível dizer quando começa ou acaba o fluxo (stream).
Os dados são processados à medida em que chegam no destino.
Alguns autores chamam de processamento em tempo real, ou on-line.

Uma abordagem diferente é o processamento em bloco, batch, ou off-line, na qual blocos de dados são processados em janelas de tempo.

Mensageria de Alta Performance

Sistema de Mensagem

O sistema de mensagens permite desacoplar a geração dos dados do seu processamento de forma assíncrona, assim, uma aplicação pode continuar funcionando sem esperar a outra terminar seu processamento.

Mensageria de Alta Performance com Apache Kafka

O que é o Apache Kafka?

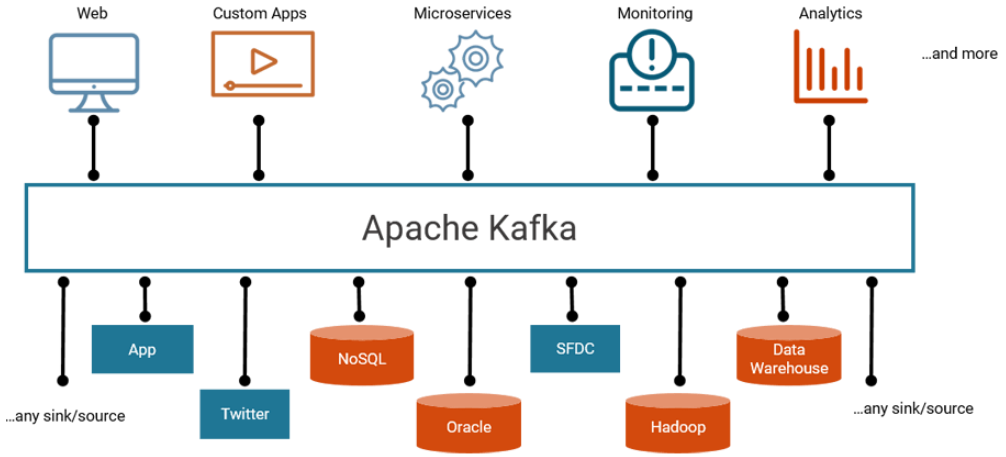
O Apache Kafka é uma plataforma open-source de mensageria usado para criar aplicações de streaming, ou seja, aquelas com fluxo de dados contínuo.

Kafka é baseado em logs, algumas vez chamado de write-ahead logs, commit logs ou até mesmo transaction logs. Um log é uma forma básica de armazenamento, porque cada nova informação é adicionada no final do arquivo. Este é o princípio de funcionamento do Kafka.

O Kafka é adequado para soluções com grande volume de dados (big data) porque uma das suas características é a alta taxa de transferência.

Data Engineering- Apache Kafka

- Apache Kafka é uma plataforma open-source de processamento de streams desenvolvida pela Apache Software Foundation, escrita em Scala e Java. O projeto tem como objetivo fornecer uma plataforma unificada, de alta capacidade e baixa latência para tratamento de dados em tempo real.



Fonte: https://kafka.apache.org/

Mensageria de Alta Performance com Apache Kafka

O Apache Kafka tem 3 funcionalidades principais:

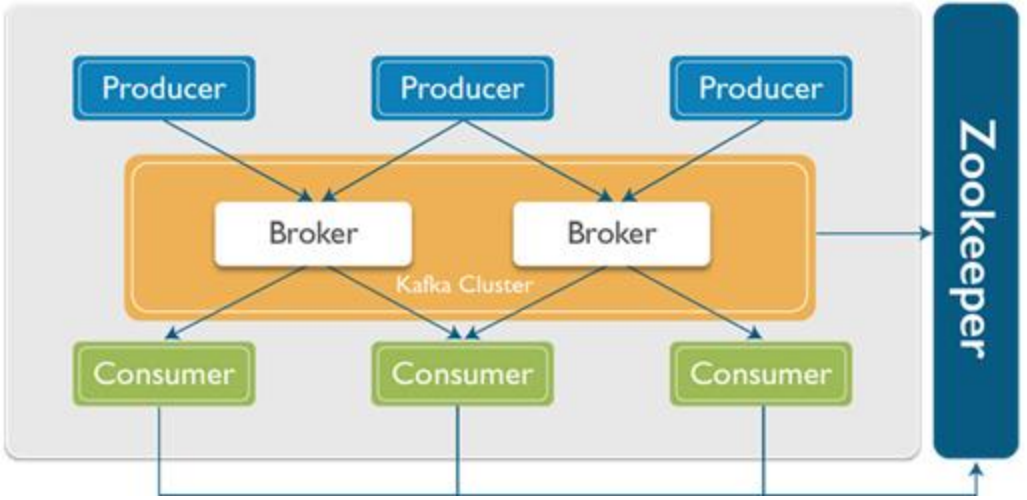
- **Sistema de mensagens**: Usado para desacoplar sistemas com escopos diferentes. Sistema de mensagem do tipo *publish/subscribe*;

- **Sistema de armazenamento**: Usado para guardar os eventos (logs) de forma consistente, permitindo reconstruir o estado do sistema a qualquer momento a partir destes logs. As mensagens ficam armazenadas por um período de tempo pré-definido.

- **Processamento de _stream**: Usado para transformar dados em tempo real, como mapeamentos, agregações, junções, etc. É possível transformar a mensagem imediatamente após o seu recebimento.

Apache Kafka

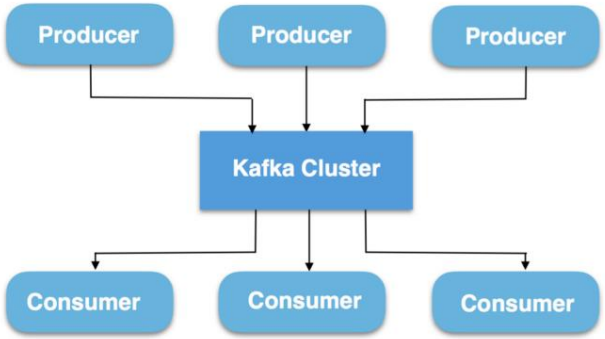
Basicamente, o Kafka é um intermediário que coleta os dados da fonte e entrega para uma aplicação que consumirá esses dados, como visto na imagem:



Fonte: https://kustavo.github.io/guia-rapido/aplicativos/kafka/

Apache Kafka - Arquitetura Pub/Sub

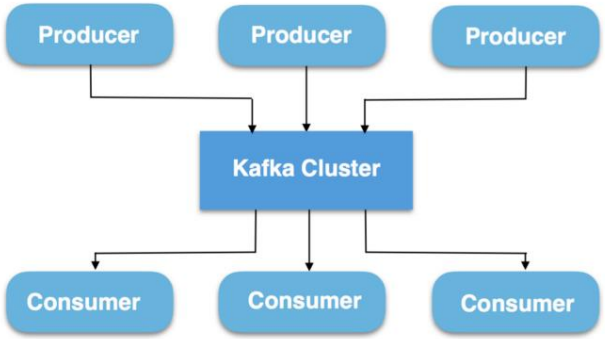
O Kafka segue a arquitetura de mensageria no estilo Publisher/Consumer, em que um componente **publica** uma mensagem em uma fila chamado de **tópico** a um intermediário e um outro componente **consome** essa mensagem a partir do intermediário, no **tópico** que essa informação está.
Essas mensagens não são direcionadas a determinados consumidores, então para que eles possam acessar os dados eles se **inscrevem** nessa fila do intemediário para aguardarem novas informações.



Fonte: https://medium.com/data-hackers/mensageria-de-alta-performance-com-apache-kafka-1-c801d60a9299

Apache Kafka - Arquitetura Pub/Sub

Um cluster Kafka é composto por vários brokers. Um broker é um servidor Kafka que recebe mensagens dos producers e as grava no disco. Cada broker gerencia uma lista de tópicos e cada tópico é dividido em diversas partições.
A arquitetura do Apache Kafka é composta por producers, consumers e o próprio cluster.
O producer é qualquer aplicação que publica mensagens no cluster. O consumer é qualquer aplicação que recebe as mensagens do Kafka. O cluster Kafka é um conjunto de nós (brokers) que funcionam como uma instância única do serviço de mensagem.



Fonte: https://medium.com/data-hackers/mensageria-de-alta-performance-com-apache-kafka-1-c801d60a9299

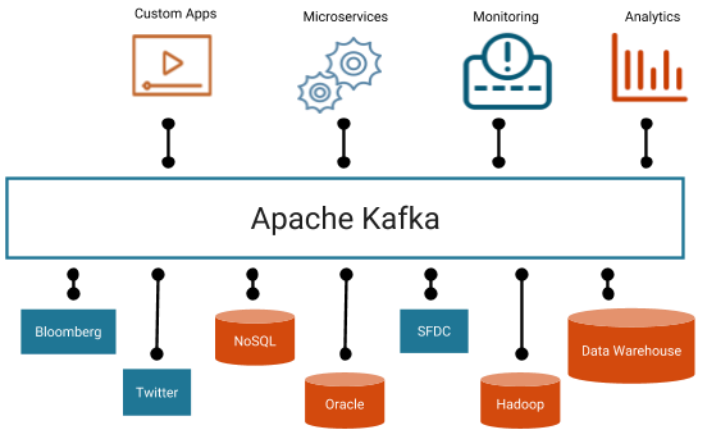
Problema Real - Mensageria de Dados

Mensageria de Alta Performance

Caso de uso com o Apache Kafka
Resolver problemas de integração entre a plataforma de marketplace e o e-commerce com seguintes problemas:

Lentidão na integração das ofertas dos lojistas marketplace com e-commerce que refletiam em:

- Um preço ou disponibilidade de uma oferta demorava até horas para refletir no site.
- O processo síncrono, e muitas vezes o alto volume das integrações gera indisponibilidade no site.
- O time precisava desligar as integrações, ligando somente à noite, para não haver risco de gerar indisponibilidade no site.
- A maioria das atualizações eram feitas por meio de processos *batch*.



Fonte: https://medium.com/analytics-vidhya/apache-kafka-architecture-getting-started-with-apache-kafka-771d69ac6cef

Mensageria de Alta Performance

Esses problemas faziam com que os produtos dos lojistas marketplace que estivessem sem estoque fossem exibidos no site como disponível, ou pior, produtos com preços desatualizados.

O Apache Kafka permitiu tornar essas integrações assíncronas e em tempo real, ao invés de integrar com processos batch.

Foi possível gerar eventos cada vez que uma oferta era criada, atualizada ou deletada, possibilitando atualizações distribuídas, entregando mais throughput.

Mensageria de Alta Performance

Pode agir como um buffer para que os sistemas não travem:

- Muitos sistemas ainda fazem a transformação e o processamento de dados em lote no período da noite, o Apache Kafka resolve esse processo lento em várias etapas agindo como um intermediário recebendo os dados de um sistema de origem, e em seguida disponibilizando para o sistema de destino em tempo real.

Mensageria de Alta Performance

Redução da necessidade de múltiplas integrações:

- Imagine um cenário contendo 4 sistemas de origem e 6 sistemas de destino, são necessárias a criação de 24 integrações. Esse é um processo complicado, sem mencionar também que é uma maneira lenta e propensa a erros no fornecimento de dados, com vários sistemas fazendo ***polling*** de um mesmo dado, consumindo recursos desnecessários dos nossos sistemas de gerenciamento de bancos de dados.

- Com o Apache Kafka, ao invés de construir múltiplas integrações, é necessário apenas uma integração para cada sistema produtor e consumidor, diminuindo a fricção entre times e tornando mais flexível a substituição de sistemas legados.

Mensageria de Alta Performance

Baixa latência e alto *throughput:*

- O Apache Kafka torna mais fácil a criação de arquiteturas de sistemas em tempo real (*near real time*), reduz o tempo de espera pela disponibilidade de um determinado dado pela área usuária, e possibilita a tomada de decisões próximo ao tempo real, um exemplo seria o marketing de precisão.

Mensageria de Alta Performance

Todos podem acessar os dados:

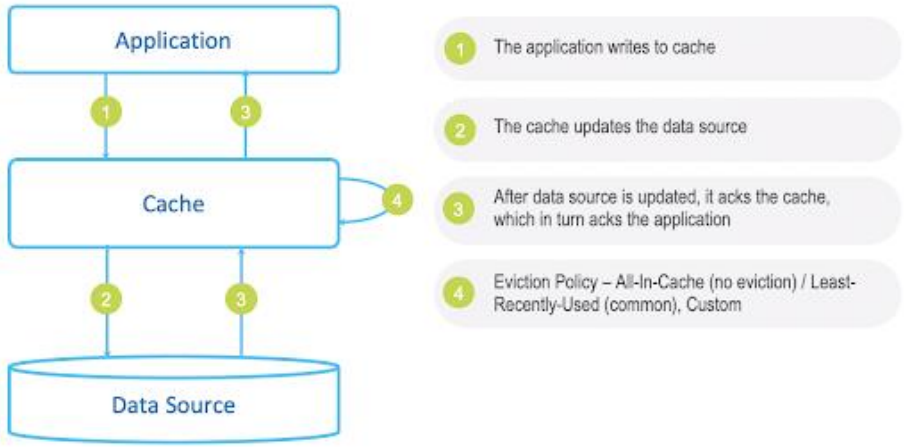
- Como os dados estão centralizados no Apache Kafka, qualquer outro sistema pode acessar esses dados sem impactar nos sistemas de origens. A maioria das integrações geralmente busca os dados no sistema de origem, podendo às vezes causar lentidão ou indisponibilidade na origem.

Armazenamento em Cache

O que é o armazenamento em cache?

Na área de computação, um cache é uma camada de armazenamento físico de dados de alta velocidade que guarda um subconjunto de dados, geralmente temporário por natureza, para que futuras solicitações referentes a esses dados sejam atendidas de modo mais rápido do que é possível fazer ao acessar o local de armazenamento principal de dados.

O armazenamento em cache permite reutilizar com eficiência dados recuperados ou computados anteriormente.



Fonte: https://www.gigaspaces.com/blog/in-memory-cache/

Como funciona o armazenamento em cache?

Os dados em um cache geralmente são armazenados no hardware de acesso rápido, como uma Random-access memory (RAM – Memória de acesso aleatório), e também podem ser usados em paralelo com um componente de software.
O principal objetivo de um cache é aumentar a performance da recuperação de dados ao reduzir a necessidade de acessar a camada subjacente mais lenta de armazenamento.

A substituição de capacidade por velocidade geralmente faz com que um cache armazene um subconjunto de dados de modo temporário, em comparação com bancos de dados, cujos dados são, de modo geral, completos e duráveis.

Visão geral do armazenamento em cache

RAM e mecanismos de memória: devido às altas taxas de solicitação ou IOPS (operações de entrada/saída por segundo) com suporte por mecanismos de RAM e em memória, o armazenamento em cache resulta em um desempenho de recuperação de dados otimizado, além de reduzir custos em grande escala.

Para apoiar a mesma escala com bancos de dados tradicionais e hardware de disco, seriam necessários recursos adicionais.

Esses recursos adicionais aumentam os custos e ainda assim não têm condições de atingir a performance de baixa latência proporcionada por um cache de memória.

Visão geral do armazenamento em cache

Aplicações: os caches podem ser aplicados e utilizados entre várias camadas de tecnologia, como sistemas operacionais, camadas de redes, incluindo Redes de entrega de conteúdo (CDN) e DNS, além de aplicativos web e bancos de dados.

As cargas de trabalho com alto consumo computacional que trabalham com conjuntos de dados, como mecanismos de recomendação e simulações computacionais de alta performance, também se beneficiam de uma camada de dados de memória atuando como um cache. Nesses aplicativos, conjuntos de dados muito grandes devem ser acessados em tempo real em clusters de máquinas que podem abranger centenas de nós. Devido à velocidade do hardware subjacente, manipular esses dados em um armazenamento baseado em disco é um gargalo significativo para esses aplicativos.

Visão geral do armazenamento em cache

Padrões de projeto: em um ambiente computacional distribuído, uma camada de armazenamento em cache dedicada permite que sistemas e aplicativos sejam executados de forma independente do cache com seus próprios ciclos de vida, sem o risco de afetar o cache.

O cache serve como uma camada central que pode ser acessada a partir de sistemas diferentes com seu próprio ciclo de vida e topologia arquitetônica. Isso é especialmente relevante em um sistema em que nós de aplicativo podem ser dimensionados dinamicamente na direção horizontal ou vertical.

Armazenamento em Cache – Exemplos AWS

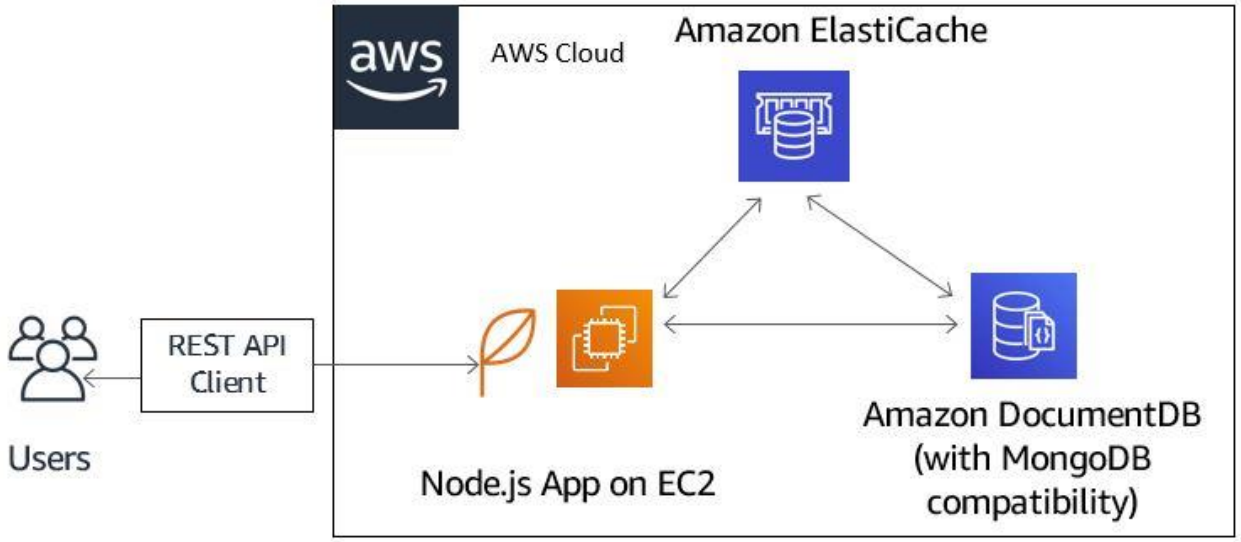
Visão geral do armazenamento em cache

Melhores práticas de armazenamento em cache: ao implementar uma camada de cache, é importante entender a validade dos dados que estão sendo armazenados em cache. Um cache bem-sucedido resulta em uma alta taxa de acertos. Isso significa que os dados estavam presentes quando foram obtidos. Um erro de cache ocorre quando os dados obtidos não estão presentes no cache.

Visão geral do armazenamento em cache

Os controles, como Time to live (TTLs – Vida útil), podem ser aplicados para expirar os dados conforme for necessário. Outra consideração a fazer pode ser avaliar se o ambiente de cache precisa ou não ser altamente disponível, o que pode ser atendido por mecanismos na memória, como o Redis. Em alguns casos, uma camada de memória pode ser usada como uma camada de armazenamento físico de dados independente, em vez de usar o cache de dados por meio de um local principal. Nesse cenário, é importante definir o RTO (Objetivo do tempo de recuperação), ou seja, o tempo necessário para a recuperação após uma interrupção, e o RPO (Objetivo de ponto de recuperação), que é o último momento ou transação capturada na recuperação, pertinentes aos dados armazenados no mecanismo em memória para determinar se isso é indicado. Estratégias e características de projeto de diferentes mecanismos em memória podem ser aplicadas para atender à maioria dos requisitos de RTO e RPO.

O Amazon ElastiCache é um serviço web que facilita a implantação, a operação e o dimensionamento de um armazenamento de dados ou cache na memória na nuvem. O serviço melhora a performance de aplicativos web, o que permite recuperar informações de datastores rápidos e gerenciados na memória, em vez de depender inteiramente de bancos de dados armazenados em disco e sua performance mais lenta.



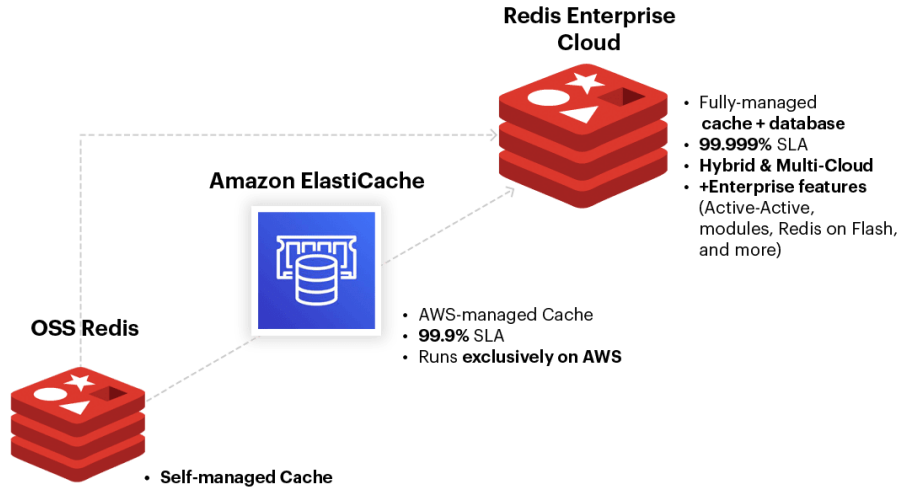
Fonte: https://aws.amazon.com/blogs/database/caching-for-performance-with-amazon-documentdb-and-amazon-elasticache/

Redis

O Redis é um armazenamento de estrutura de dados de chave-valor de código aberto e na memória.

O Redis oferece um conjunto de estruturas versáteis de dados na memória que permite a fácil criação de várias aplicações personalizadas. Os principais casos de uso do Redis incluem cache, gerenciamento de sessões, PUB/SUB e classificações.
É o armazenamento de chave-valor mais conhecido atualmente.

Ele tem a licença BSD, é escrito em código C otimizado e é compatível com várias linguagens de desenvolvimento. Redis é um acrônimo de REmote DIctionary Server (servidor de dicionário remoto).



Fonte: https://redis.com/cloud-partners/aws/

Redis

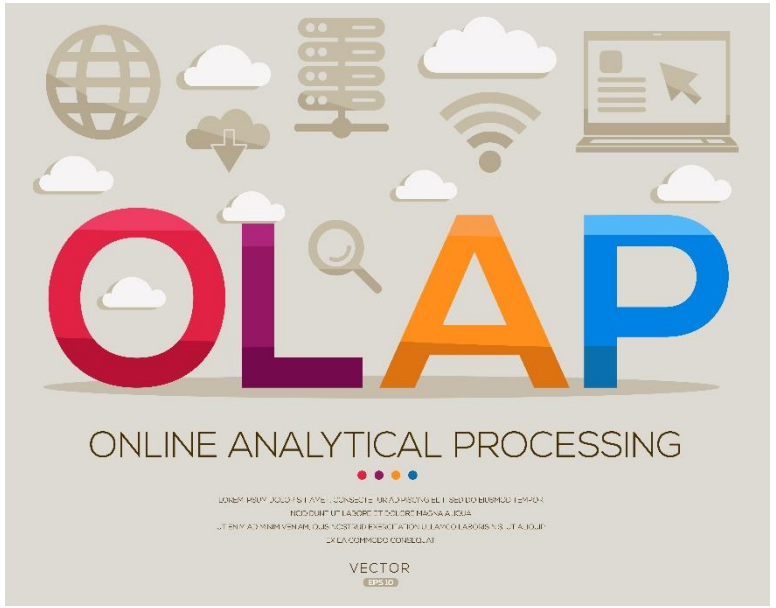
Estruturas de dados na memória

O Redis permite que os usuários armazenem chaves que fazem o mapeamento para vários tipos de dados. O tipo de dados fundamental é uma string, que pode ser em formato de dados de texto ou binários e ter no máximo 512 MB. O Redis também é compatível com listas de strings na ordem em que foram adicionadas, conjuntos de strings não ordenadas, conjuntos classificados ordenados de acordo com uma pontuação, hashes que armazenam uma lista de campos e valores e HyperLogLogs para contar os itens exclusivos de um conjunto de dados.
Praticamente qualquer tipo de dados pode ser armazenado na memória usando o Redis.

Consumo de alta volumetria de dados

OLAP

Sempre que se fala em carga massiva de dados há uma grande preocupação em como executá-la com alto índice de desempenho e baixo índice de consumo de recursos. É uma preocupação legítima e também um grande desafio. Esta situação ocorre, por exemplo, com cargas de dados diárias em ambientes OLAP (Online Analytical Processing) ou mesmo cargas em bancos de dados tipicamente OLTPs (Online Transaction Processing) que costumam receber arquivos do tipo texto (os famosos flat files).



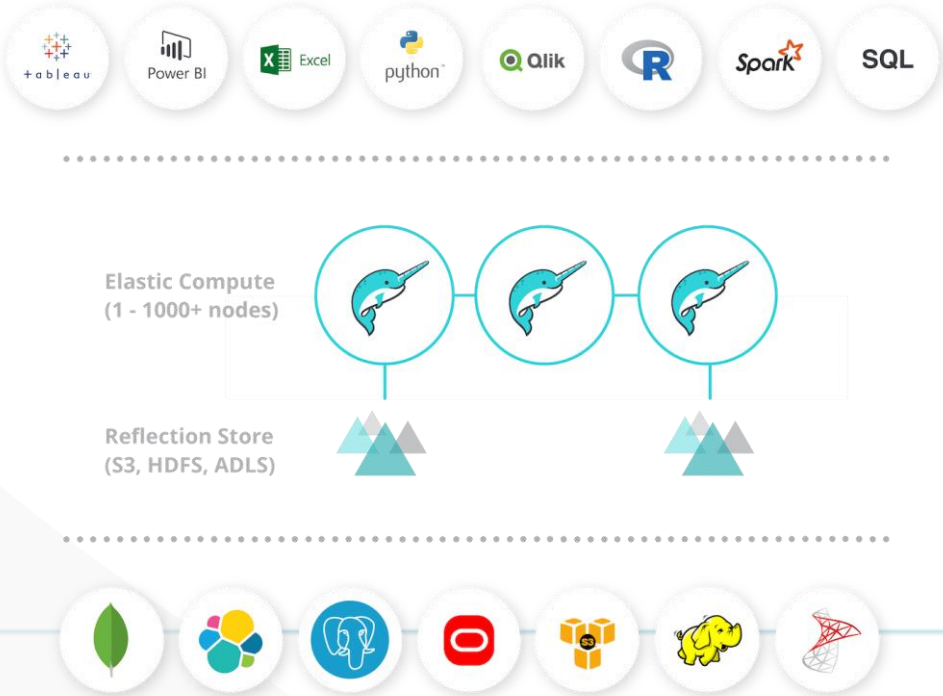
Visão geral do armazenamento em cache

O processamento analítico on-line, ou OLAP, é uma abordagem para responder rapidamente a consultas multidimensionais analíticas. OLAP é parte da categoria mais ampla de BI - Business Inteligence, que engloba também relatórios relacionais e mineração de dados. Aplicações típicas de OLAP incluem relatórios de negócios para vendas, marketing, comunicação, gestão de processos de negócios, orçamento e previsão, e relatórios financeiros.

Ferramentas OLAP permitem aos usuários analisar de forma interativa dados multidimensionais. Os bancos de dados configurados para ambientes OLAP usam um modelo de dados multidimensional, permitindo consultas analíticas complexas e ad-hoc com um tempo de execução rápido.

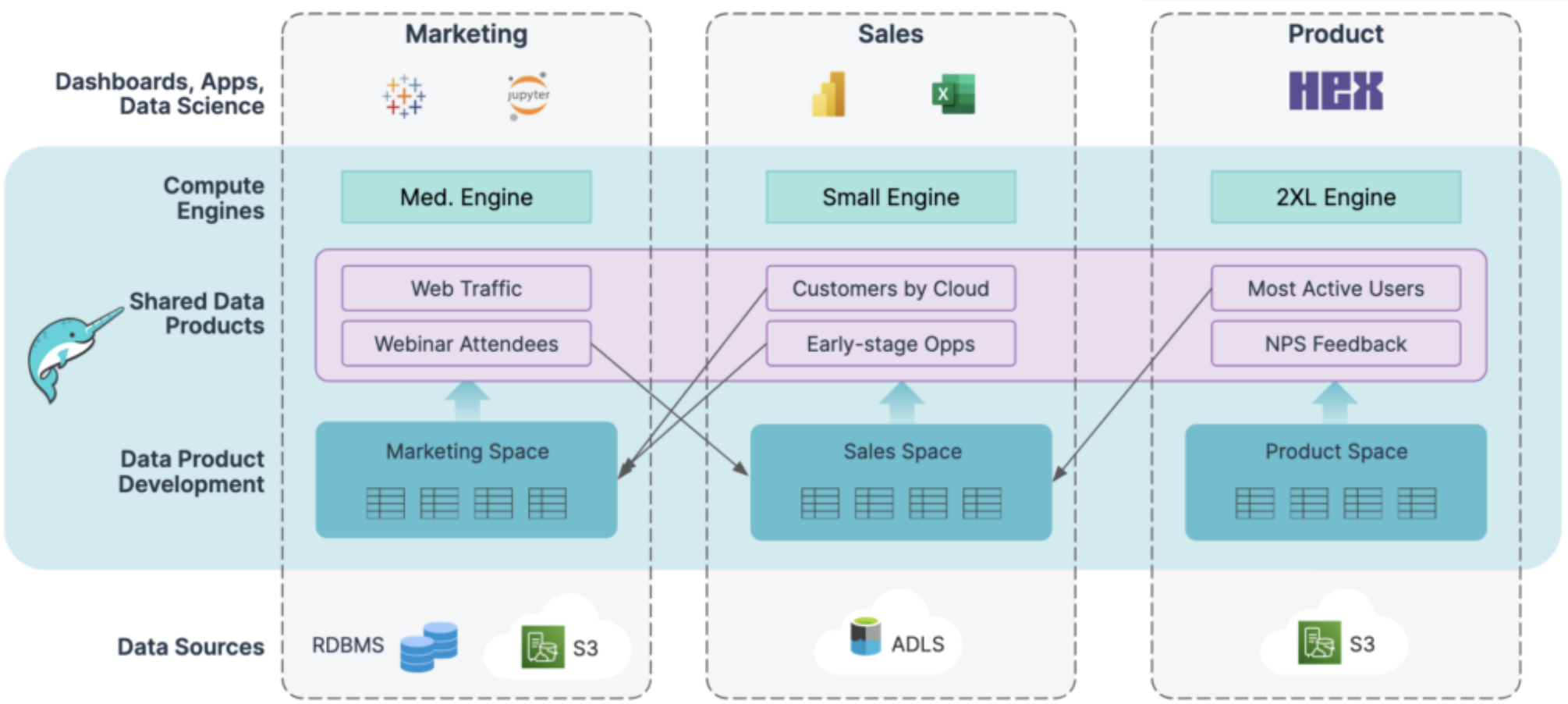
Dremio

Plataforma Self-service data. É uma plataforma de dados como serviço que capacita os usuários a descobrir, selecionar, acelerar e compartilhar quaisquer dados a qualquer momento, independentemente da localização, volume ou estrutura. Os dados modernos são gerenciados por uma ampla variedade de tecnologias, incluindo bancos de dados relacionais, datastores NoSQL, sistemas de arquivos, Hadoop e outros.



Fonte: www.dremio.com

Dremio



Fonte: www.dremio.com

Dremio - Segurança

Ferramentas dessa natureza prezam pela Conformidade.

O Dremio atende aos requisitos de controle de segurança da informação para uma variedade de estruturas e certificações de conformidade.

SOC 2 Tipo 2
O Dremio mantém a conformidade com os controles de sistema e organização do Instituto Americano, o AICPA - Critérios de Serviços de Confiança, comumente conhecidos como SOC 2.

ISO 27001
O Dremio opera um sistema de gerenciamento de segurança da informação (ISMS) que está em conformidade com os requisitos estabelecidos no padrão: ISO/IEC 27001:2013.

Fonte: www.dremio.com

ISO 27001

ISO 27001 é a norma que define os requisitos para um Sistema de Gestão da Segurança da Informação (SGSI).

O SGSI é descrito como um sistema parte do sistema de gestão global da organização, com base em uma abordagem de risco do negócio, para estabelecer, implementar, operar, monitorar, revisar, manter e melhorar a segurança da informação.

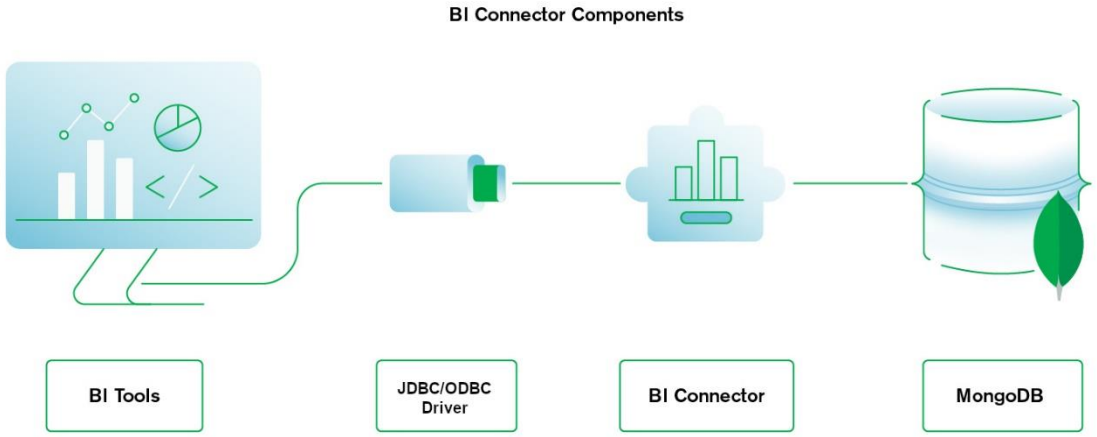
A ISO 27001 é a principal norma que uma organização deve utilizar como base para obter a certificação empresarial em gestão da segurança da informação. Por isso, é conhecida como a única norma internacional auditável que define os requisitos para um Sistema de Gestão de Segurança da Informação (SGSI).

A ISO 27002 é um código de práticas com um conjunto completo de controles que auxiliam aplicação do Sistema de Gestão da Segurança da Informação. É recomendável que a norma seja utilizada em conjunto com a ISO 27001, mas pode ser também consultada de forma independente com fins de adoção das boas práticas.

Camadas de consumo por serviços

Camada de Consumo

Simplificando, uma camada de consumo é uma ferramenta que fica entre os usuários de dados e as fontes de dados. Essa camada recebe uma consulta SQL como entrada (de uma ferramenta de BI, CLI (Call-Level Interface), ODBC/JDBC, etc.) e lida com a execução dessa consulta o mais rápido possível, consultando as fontes de dados necessárias e até unindo dados entre fontes quando necessário.



Fonte: https://www.mongodb.com/docs/bi-connector/current/reference/odbc-driver/

Data Services Layers

O que exatamente é uma DSL?

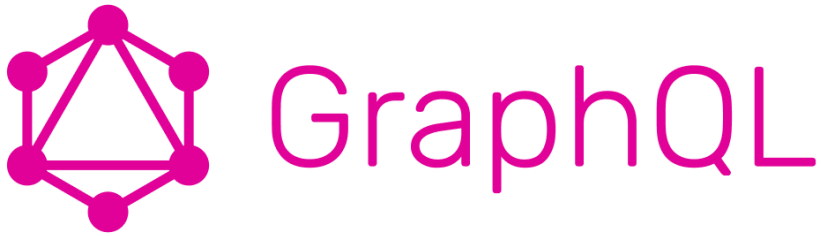
É uma camada de dados virtual que expõe fontes de dados subjacentes como serviços de dados, mais comumente serviços da web.

Os serviços de dados discretos, por si só, são projetados para responder a consultas específicas de uma maneira específica, o que lhes dá velocidade, agilidade e capacidade de gerenciamento, além de serem fáceis de gerenciar.

No entanto, seus recursos são muito restritos em escopo para fornecer entrega de dados em toda a empresa.

GraphQL

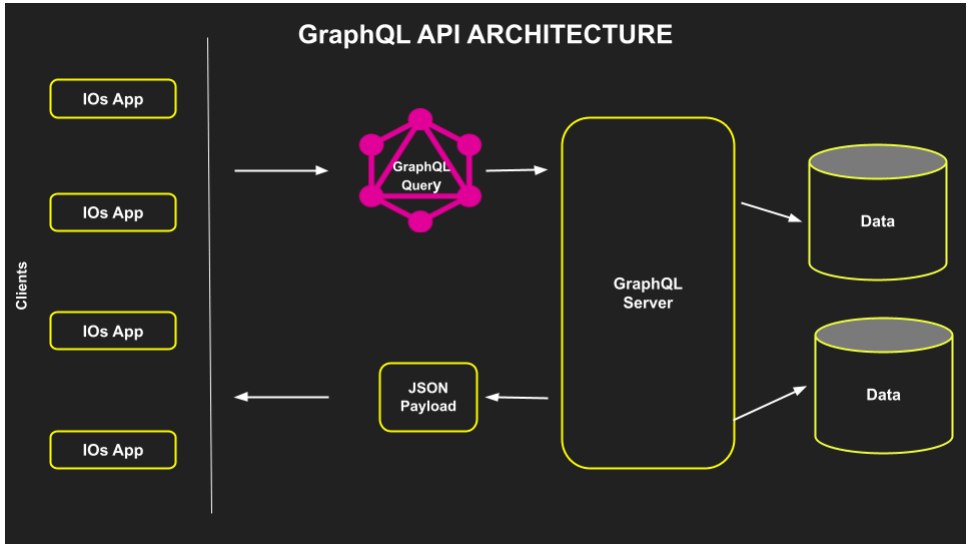
GraphQL é uma linguagem de consulta flexível que usa um sistema de tipo para retornar dados de forma eficiente com consultas dinâmicas. SQL (linguagem de consulta estruturada) é um padrão de linguagem mais antigo e mais adotado usado especificamente para sistemas de banco de dados tabulares/relacionais. Considere o GraphQL se quiser que sua API seja criada em um banco de dados NoSQL.



Fonte: https://graphql.org/

GraphQL

GraphQL significa Graph Query Language e como o nome sugere **é uma linguagem de query assim como SQL** (Structured Query Language) **porém seu uso não envolve implementar banco de dados, mas sim, definir dados seja para API ou servidor.**



Fonte: https://kinsta.com/pt/blog/graphql-vs-rest/

Graph

O Graph em GraphQL é para gráfico. O time do Facebook responsável pela linguagem aposta num modelo mental gráfico como uma forma de definir os dados. Dessa forma, a linguagem oferece a habilidade de modelar dados usando schemas. O resultado disso é um paradigma muito parecido com o formato JSON. Por exemplo, para definir um dado que representa um usuário com id, name, email e bio, teríamos o seguinte schema:

```
type User {
  id: ID
  name: String
  email: String
  bio: String
}
```

QL

A proposta do GraphQL, porém, não é apenas definir dados em schemas mas também fazer possível que consumidores desses dados disponham de uma forma padronizada de consumi-los. É aí que entra o QL do termo, já que podemos usar uma linguagem de query para fazer chamadas e consumir tais dados. Dessa forma, para consumir dados usando o exemplo de usuário acima, podemos fazer queries como:

```
query getUsers {
  users {
    name
    bio
  }
}
query getUserById {
  user(id: "123") {
    name
    bio
  }
}
```

Query e Mutation

Dados de um servidor GraphQL são consumidos através de queries feitas pelo cliente. Assim como query é usada para bucar dados, mutation é usada para criar, alterar ou deletar dados do banco de dados. Seguindo o exemplo da query de usuários definida acima, as mutations para usuários ficariam da seguinte maneira:

```
mutation createUser(name: "João", email: "...", bio: "...") {
  id
  name
  email
  bio
}
mutation updateUser(id: "123", ...) {
  id
  name
  email
  bio
}
mutation deleteUser(id: "123") {
  id
  name
  email
  bio
}
```

GraphQL como Alternativa a REST

GraphQL está se tornando uma forte concorrente de REST e é uma alternativa que está conquistando cada vez mais o mercado.

Em uma API REST os dados são dispostos através de diferentes rotas. Desta forma, se quisermos disponibilizar um array de um usuários fazemos uma rota GET /users, se quisermos disponibilizar um usuário a partir de seu id fazemos mais uma rota GET /users/:Id e para criar, atualizar e deletar um usuário precisamos de uma rota para cada ação com seus respectivos método HTTP. E isso é só para um tipo de dado, para cada novo tipo de dado é necessário as definições de rotas para cada método HTTP novamente.

Além de rotas e métodos HTTP, a arquitetura REST tem mais exigências para contemplar os requisitos RESTful, como definição de códigos de status para diferentes respostas do servidor, versionamento, etc.

Em contrapartida, com GraphQL, não há definições de rotas. Na verdade, existe apenas uma rota, e nela são feitas todas as requisições usando queries e mutations. Além disso, GraphQL também elimina a necessidade de código de status uma vez que e as mensagens de erro ou sucesso vêm anexadas ao corpo da resposta. Versionamento também fica desnecessário já que se pode adicionar novos campos e tipos sem alterar queries existentes.

GraphQL como Alternativa a REST

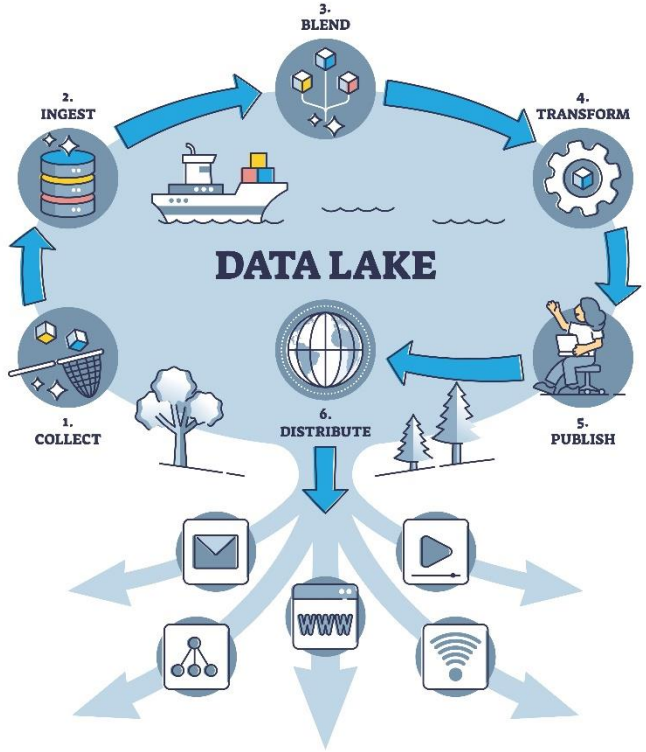
- GraphQL é uma especificação para consumo de dados;
- Graph é de gráfico pois os dados são modelados de forma visual com schemas;
- QL é de linguagem de query pois podemos usar query para fazer chamadas para um servidor;
- Schemas e types são usados para modelar os dados;
- Queries e Mutations são usados para ler a alterar dados;
- GraphQL é uma alternativa a REST com uma mudança de paradigma;
- GraphQL precisa ser implementado para poder ser usado em linguagens de programação.

Plataforma self-service

Importância do Self-service

Em dados, o self-service é um tipo de sistema de análise de dados que ganhou força no mercado pela autonomia que dá aos usuários não técnicos.

O nome “self-service” vem justamente dessa característica autossuficiente que permite a extração de diferentes relatórios, de acordo com os indicadores estabelecidos pelo usuário.



Importância do Self-service

Os meios tradicionais demandam muito tempo da equipe executiva e de TI, por isso, a análise de dados de começou a tomar força nas empresas e organizações.

As plataformas passaram a se tornar cada vez mais interativas, e hoje, é muito simples usá-las, não necessitando de um profissional com expertise em TI para tal.

O Self-service têm um grande papel nas organizações, já que eles desatolaram a equipe de TI, e fizeram o processo ser mais simples e usável.

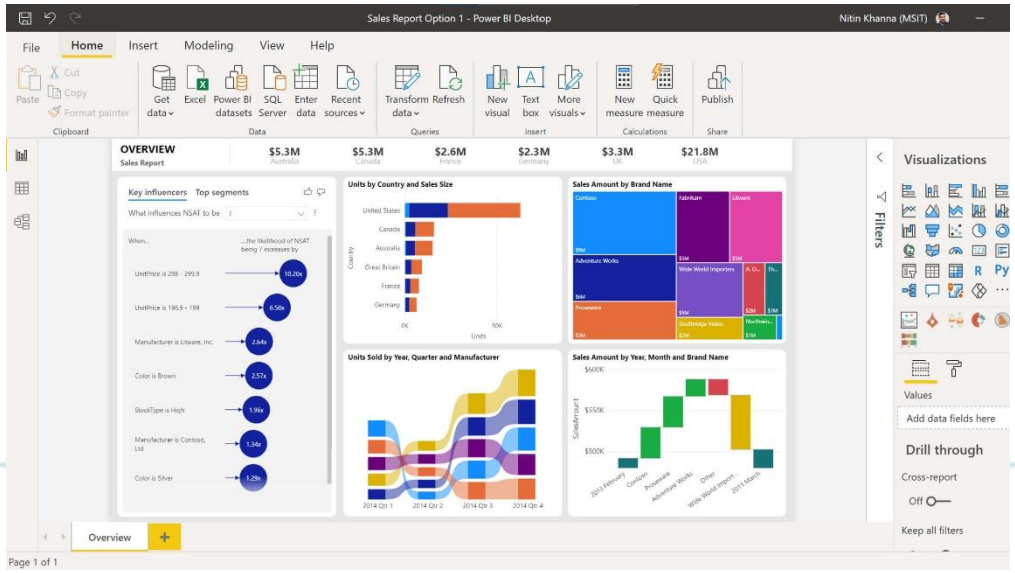
Vantagens do Self-service

- Permite que os usuários criem e editem seus próprios relatórios, painéis e outras visualizações para uso em fluxos de trabalho diários, descoberta de dados e tomada de decisões.
- Torna o processo de desenvolvimento mais rápido e fácil para os usuários, uma vez que não necessitam do suporte de TI.
- Abrange uma ampla variedade de conjunto de recursos interativos que tornam a descoberta de dados mais rápida e fácil.
- Distribui as informações de maneira simples.
- Análises disponíveis a qualquer momento – perto de tempo real.

Exemplo de Ferramentas Self-Service - Power BI

- O Power BI, que representa a entrada da Microsoft na briga de BI self-service, inclui uma interface Web para um serviço hospedado no Azure e um aplicativo Power BI Desktop para a área de trabalho do Windows.

- O Power BI é uma opção razoável para empresas que usam o ecossistema Windows / Office / Azure. Também é uma boa opção para empresas sensíveis ao custo que desejam fornecer BI a todos na organização. No lado negativo, o Power BI não oferece tanta capacidade de análise ou controle sobre seus gráficos como o Qlik Sense ou o Tableau.

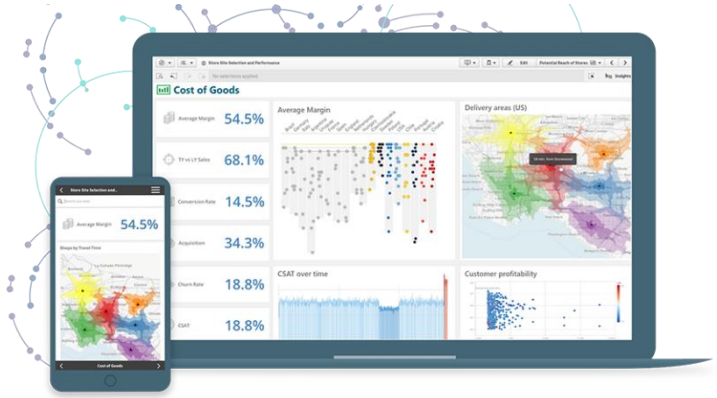


Fonte: https://powerbi.microsoft.com/pt-br/

Exemplo de Ferramentas Self-Service – Qlik Sense

O Qlik tinha um “Mode 1” ou um produto de BI tradicional no QlikView e expandiu para BI self-service com o Qlik Sense. Introduzido em 2014, o Qlik Sense é um BI e um produto de visualização do tipo ‘faça você mesmo’, baseado no mesmo mecanismo de indexação associativa de dados na memória usado pelo QlikView. Em 2016, o Qlik adicionou ao Qlik Sense o seu mecanismo de relatórios, anteriormente disponível apenas com o QlikView.

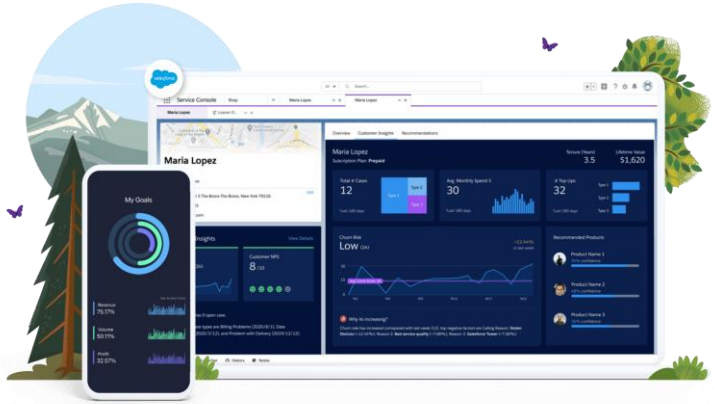
O Qlik Sense 2.0 é uma ferramenta de análise de dados e descoberta muito capaz. Pode se conectar a praticamente qualquer banco de dados SQL e oferece um bom controle sobre as visualizações. No entanto, não é tão fácil de aprender, tão fácil de usar ou flexível na apresentação de visualizações como o Tableau.



Fonte: https://www.qlik.com/pt-br/products/qlik-sense

Exemplo de Ferramentas Self-Service - Tableau

A Tableau descreve seus produtos como oferecendo “análises que funcionam da maneira como você pensa” e diz que essas ferramentas aproveitam a “capacidade natural das pessoas de identificar padrões visuais rapidamente”. Há uma certa verdade nisso, embora você possa dizer quase a mesma coisa sobre muitas outras ferramentas de BI.



Fonte: https://www.tableau.com/

Exemplo de Ferramentas Self-Service – QuickSight

O Amazon QuickSight é executado inteiramente na nuvem da AWS, tem bom acesso às fontes de dados da Amazon e acesso justo a outras fontes de dados e oferece análise básica e manipulação de dados a um preço básico. Dos outros produtos discutidos aqui, o QuickSight é mais parecido com o Power BI, somente sem a dependência de um produto de desktop para criar conjuntos de dados ou com o nível de poder de análise fornecido pela combinação do Power BI Desktop / Service.

Assim como o Power BI, o Qlik Sense e o Tableau, o QuickSight se conecta a uma infinidade de fontes de dados e permite preparar conjuntos de dados. Depois de ter conjuntos de dados, você pode criar análises com uma ou mais visualizações, e organizar em painéis e histórias.

Fonte: https://aws.amazon.com/pt/quicksight/

Dremio

Plataforma Self-service data.

Plataforma Data Lakehouse
Análise de autoatendimento com funcionalidade de data warehouse e flexibilidade de data lake em todos os seus dados.

Fonte: www.dremio.com

Design convergente para dados

Arquitetura de Dados Convergente

Nos sistemas e nas aplicações que são criadas hoje em dia, os maiores problemas com Dados em geral são a quantidade de informações geradas, os múltiplos formatos que existem, e a velocidade em que eles se modificam.

Arquitetura de Dados Convergente

Quando você armazena dados com objetivo de ser consumido, você precisa usar um banco de dados ou um sistema de arquivos.
Se você quer acelerar o resultado de uma busca complexa, leituras, você usa um mecanismo de cache.
Para permitir que os usuários façam buscas por palavras-chave específicas, textos-livres, ou consultas ad-hoc, você usa índices.
Para trocar mensagens entre processos, ou entre sistemas, você usa um broker de eventos.
E para periodicamente manipular uma grande quantidade de dados acumulados e processar, você faz um processamento batch.

Arquitetura de Dados Convergente

Nós tipicamente pensamos que bancos de dados, caches, filas, que são todas ferramentas de diferentes categorias.

Por exemplo, um banco de dados e um broker de eventos têm uma certa similaridade, porque ambos armazenam dados por algum tempo, mas eles são diferentes.

Eles têm padrões de acesso diferentes, e características de performance e implementação diferentes.

Arquitetura de Dados Convergente

Portanto é interessante pensar em “um sistema de dados convergente”.

É uma plataforma onde você tem quase todas as ferramentas, quase todos os tipos de dados, seja schema on write, seja schema on read, seja um cache, uma fila.
E quando esse sistema não suporta alguma coisa, ele virtualiza o acesso e expõe a mesma interface de acesso pra você consumir aquela informação da forma mais transparente possível.


PUC Minas
Virtual