UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
PROGRAMA DE RESIDÊNCIA EM TECNOLOGIA DA INFORMAÇÃO

Análise comparativa entre ferramentas *low-code* para ETL: Um estudo de caso da DPE/RN

Andressa Silva de Souza

Natal – RN, Brasil

2025

Andressa Silva de Souza

Análise comparativa entre ferramentas *low-code* para ETL: Um estudo de caso da DPE/RN

Trabalho de Conclusão de Curso apresentado ao Programa de Residência em Tecnologia da Informação do Instituto Metrópole Digital da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Especialista em Tecnologia da Informação. Área de Concentração: Business Intelligence & Analytics.

Orientador: João Carlos Xavier Junior

Natal – RN, Brasil

2025

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Souza, Andressa Silva de.
Análise comparativa entre ferramentas low-code para ETL: um estudo de caso da DPE/RN / Andressa Silva de Souza. - 2025.
82 f.: il.

Trabalho de Conclusão de Curso - TCC (especialização) - Universidade Federal do Rio Grande do Norte, Instituto Metrópole Digital, Programa de Residência em Tecnologia da Informação.
Orientação: Prof. Dr. João Carlos Xavier Junior.

1. ETL - TCC. 2. Low-code - TCC. 3. Airbyte - TCC. 4. KNIME - TCC. I. Xavier Junior, João Carlos. II. Título.

RN/UF/BCZM CDU 004

Elaborado por Jackeline dos Santos Pinheiro da Silva Maia Cavalcanti - CRB-15/317

Andressa Silva de Souza

Trabalho de Conclusão de Curso apresentado ao Programa de Residência em Tecnologia da Informação do Instituto Metrópole Digital da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Especialista em Tecnologia da Informação. Área de Concentração: Desenvolvimento de software.

Trabalho aprovado. Natal – RN, Brasil, 05 de novembro de 2025.

_____

Professor Dr. Joao Carlos Xavier Junior

Orientador

_____

Professora Drª. Anne Magaly de Paula Canuto

Examinadora

_____

Professor Dr. Sérgio Ramiro Rivero Guardia

Examinador

Natal – RN, Brasil
2025

Resumo

O crescente volume de dados nas organizações públicas e privadas exige soluções que tornem os processos de integração, transformação e análise mais ágeis e eficientes. Nesse cenário, ferramentas low-code para ETL (Extract, Transform, Load) ganham relevância por facilitarem a construção de pipelines de dados, oferecendo flexibilidade para múltiplas demandas e reduzindo a dependência de conhecimentos técnicos avançados por parte dos desenvolvedores. Este trabalho apresenta um estudo comparativo entre duas ferramentas dessa categoria, Airbyte e KNIME, aplicadas a um banco de dados real da Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN). Foram construídos pipelines equivalentes em ambas as plataformas e submetidos a cargas de dados idênticas, sendo o monitoramento das execuções realizado com o Datadog, que permitiu a coleta sistemática de métricas tanto durante os processos de sincronização quanto em estado ocioso. Para viabilizar a análise, foi configurado um ambiente controlado que replica, em escala reduzida, a infraestrutura utilizada pela DPE/RN. Os resultados obtidos visam fornecer subsídios práticos para a escolha da solução mais adequada a diferentes contextos de ETL, contribuindo para a adoção consciente de ferramentas low-code em ambientes organizacionais. A análise evidenciou que, embora o KNIME tenha demonstrado bom desempenho e aplicabilidade, o Airbyte mostrou-se mais alinhado ao stack tecnológico e às necessidades operacionais da DPE/RN, confirmando que sua adoção institucional é tecnicamente coerente com o cenário atual.

Palavras-chave: ETL. *Low-code*. Airbyte. KNIME.

Abstract

The growing volume of data in public and private organizations demands solutions that make integration, transformation, and analysis processes more agile and efficient. In this context, low-code ETL (Extract, Transform, Load) tools gain relevance by facilitating the construction of data pipelines, offering flexibility for multiple demands, and reducing the dependence on advanced technical knowledge from developers. This study presents a comparative analysis between two tools in this category, Airbyte and KNIME, applied to a real database from the Public Defender's Office of the State of Rio Grande do Norte (DPE/RN). Equivalent pipelines were built in both platforms and submitted to identical data loads, with execution monitoring performed through Datadog, which enabled the systematic collection of metrics both during synchronization processes and in idle states. To support the analysis, a controlled environment replicating, on a reduced scale, the infrastructure used by DPE/RN was configured locally. The results obtained aim to provide practical guidance for selecting the most suitable solution for different ETL contexts, contributing to the conscious adoption of low-code tools in organizational environments. The analysis showed that although KNIME demonstrated solid performance and applicability, Airbyte proved to be more aligned with the existing technological stack and operational needs of DPE/RN, confirming that its institutional adoption is technically consistent with the organization's current scenario.

Keywords: ETL. Low-code. Airbyte. KNIME.

Lista de abreviaturas e siglas

IMD	Instituto Metrópole Digital
UFRN	Universidade Federal do Rio Grande do Norte
TIC	Tecnologias de Informação e Comunicação
ETL	*Extract, Transform and Load*
DW	*Data Warehouse*
DL	*Data Lake*
MDS	*Modern Data Stack*
DPE/RN	Defensoria Pública do Estado do Rio Grande do Norte
JVM	Java Virtual Machine
APM	Application Performance Monitoring

Lista de figuras

Lista de quadros

Sumário

1 Introdução

A transformação digital tem impulsionado de forma significativa a maneira como organizações públicas e privadas lidam com seus dados. O volume de informações disponíveis cresce de forma exponencial, exigindo soluções tecnológicas capazes de coletar, integrar, processar e analisar dados em escala cada vez maior. Esse fenômeno acompanha a evolução das Tecnologias da Informação e Comunicação (TICs), que modificaram profundamente processos sociais, econômicos e administrativos, tornando os dados um ativo estratégico para a tomada de decisão (Marr, 2016). Nessa seara, a chamada "explosão de dados" impacta diretamente órgãos públicos, que necessitam estruturar seus fluxos informacionais para garantir eficiência, transparência e qualidade nos serviços prestados (Belli et al., 2024).

Ainda neste cenário, soluções de ETL (Extract, Transform, Load) assumem papel central. Tradicionalmente, o processo de ETL envolve extrair dados de múltiplas fontes heterogêneas, transformá-los em formatos consistentes e, por fim, carregá-los em um repositório analítico, como *data warehouses* (DW) ou *data lakes* (DL) (Kimball & Ross, 2013). Assim, vale destacar que tecnologias clássicas de ETL muitas vezes necessitam de alto grau de especialização técnica, o que pode aumentar custos de desenvolvimento e manutenção, além de gerar dependência de equipes altamente qualificadas.

Para mitigar barreiras como essas, que demandam além de outras bases, mão de obra especializada que geralmente possui custo mais elevado, ganham destaque as plataformas *low-code*, que permitem a construção de *pipelines* de dados de maneira mais ágil, visual e acessível, reduzindo a necessidade de programação extensiva (Richardson *et al.*, 2020). Esse tipo de solução vem sendo cada vez mais utilizada em empresas e instituições públicas que buscam modernizar seus processos de integração de dados sem depender exclusivamente de especialistas e/ou engenheiros de dados. Entre as vantagens das ferramentas *low-code* de ETL estão a redução do tempo de implementação, a flexibilidade para diferentes cenários de uso e a possibilidade de ampliar a autonomia de equipes multidisciplinares (Gartner, 2025).

Tecnologias como Airbyte e KNIME são exemplos de plataformas *low-code* que viabilizam a criação de *pipelines* de dados como o próprio nome sugere, com pouco ou nenhum código. A primeira solução é voltada à integração de diversas fontes heterogêneas, com foco em conectores pré-construídos e facilidade de orquestração em ambientes modernos de nuvem e contêineres. Além disso, oferece suporte à execução *on-premise*, garantindo também a flexibilidade para atender tanto a infraestruturas locais quanto a arquiteturas em nuvem. O KNIME, similarmente, possibilita a integração de dados a partir de diferentes fontes, sejam elas arquivos locais no desktop, bancos de dados relacionais ou grandes DWs corporativos. Sua arquitetura, no entanto, é baseada em execução local como aplicação tradicional, sem suporte nativo ao uso em contêineres, o que influencia diretamente nas estratégias de implantação e monitoramento da ferramenta.

O presente trabalho propõe um estudo comparativo entre as ferramentas Airbyte e KNIME, aplicadas a um banco de dados real da Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN). Devido às limitações de desempenho do ambiente local utilizado nos testes, não foi possível utilizar a base completa, optando-se por um subconjunto composto por tabelas representativas do banco original.

A análise envolveu a construção de pipelines equivalentes e seu monitoramento em um ambiente controlado, com infraestrutura replicada, conforme o ambiente institucional da DPE/RN. O monitoramento foi realizado por meio do Datadog, uma plataforma de observabilidade com abordagem low-code, que permite coletar, visualizar e analisar métricas sem a necessidade de scripts complexos, otimizando o esforço de instrumentação.

Durante os testes, foi possível acompanhar o comportamento das ferramentas tanto durante a execução da carga de dados quanto em estado ocioso. As métricas foram coletadas de acordo com as particularidades arquiteturais de cada ferramenta: enquanto o Airbyte foi acompanhado a nível de container, o KNIME foi monitorado como uma aplicação local em execução Java. Dessa forma, não houve comparação direta entre valores absolutos, mas sim uma observação contextualizada do comportamento de cada solução dentro do seu modelo operacional.

A relevância desta pesquisa reside no fato de que instituições públicas, como a DPE/RN, enfrentam o desafio de lidar com grandes volumes de dados provenientes de diferentes sistemas e fontes, muitas vezes em um contexto marcado por processos burocráticos e limitações de agilidade tecnológica.

Nesse cenário, a escolha consciente de uma ferramenta *low-code* adequada pode representar ganhos expressivos em produtividade, eficiência de recursos e qualidade analítica, além de apoiar a transformação digital no setor público. Para que essa escolha seja feita de maneira mais precisa e embasada, esta pesquisa adota o monitoramento como recurso central na comparação entre as ferramentas, permitindo que a análise se apoie em métricas reais de desempenho e comportamento.

A abordagem fundamentada em evidências técnicas torna possível identificar qual solução responde melhor a cada tipo de cenário, contribuindo para decisões mais alinhadas às necessidades específicas das organizações. Dessa forma, o estudo amplia o conhecimento sobre a adoção de tecnologias de integração de dados em ambientes organizacionais, oferecendo subsídios técnicos e práticos para gestores e profissionais de tecnologia da informação.

1.1 Justificativa

O volume de dados gerados e processados por organizações públicas cresce de forma acelerada, acompanhado pela necessidade de integrar informações provenientes de diferentes sistemas, bases e formatos. Segundo o relatório *Data and Analytics Trends* da Gartner (2023), até 80% do tempo de cientistas e analistas de dados é consumido em tarefas de preparação e integração, o que reforça a importância de soluções que automatizem e simplifiquem esses processos. No caso da DPE/RN, essa realidade é ainda mais evidente, uma vez que a instituição lida com bases de dados heterogêneas — desde pequenos conjuntos locais até grandes volumes extraídos de APIs — utilizadas na geração de relatórios, análises e apoio à gestão institucional.

Nesse contexto, ferramentas *low-code* como o Airbyte e o KNIME surgem como alternativas viáveis para a construção de *pipelines* de dados, oferecendo conectores pré-configurados e mecanismos de orquestração que reduzem significativamente a

necessidade de codificação manual. Ao tornarem o processo de integração mais acessível as equipes com diferentes níveis de especialização, essas soluções contribuem diretamente para a agilidade e a eficiência dos fluxos de dados nas organizações. A importância dessa acessibilidade se intensifica diante de fatores como o volume de dados processado, o tipo de algoritmo utilizado e a eficiência no consumo de recursos computacionais — elementos que, como aponta Manzoor (2021), impactam diretamente os custos operacionais e a sustentabilidade dos sistemas de ETL.

A presente pesquisa ganha relevância por adotar uma abordagem fundamentada em evidências operacionais para comparar ferramentas *low-code* de ETL em um contexto real de instituição pública. Utilizando cargas compostas por tabelas representativas do banco de dados da DPE/RN, foi possível executar *pipelines* equivalentes e coletar indicadores de desempenho por meio da plataforma Datadog. O monitoramento contínuo permitiu observar o comportamento de cada solução em diferentes condições de execução, respeitando suas arquiteturas nativas, o que torna a análise mais aderente à realidade de uso e fortalece a base para decisões técnicas mais precisas.

Além de fornecer subsídios práticos para a escolha de tecnologias de integração de dados no setor público, este estudo destaca a importância da observabilidade como aliada na gestão eficiente de ferramentas ETL. Em contextos com recursos limitados e alta complexidade operacional, como ocorre em muitas instituições governamentais, tornar visível o desempenho das soluções adotadas é essencial para promover decisões mais racionais, otimizar a infraestrutura e qualificar os serviços prestados à sociedade.

1.2 Objetivos

Aqui são apresentados o objetivo geral e os objetivos específicos que orientam a realização do estudo comparativo entre as ferramentas *low-code* de ETL, Airbyte e KNIME, a partir da aplicação a dados reais da DPE/RN e da análise de desempenho com base nos pontos monitorados.

1.2.1 Objetivo geral

Comparar as ferramentas Airbyte e KNIME com base em métricas de desempenho — como uso de CPU, memória e tempo de execução — monitoradas pelo Datadog, além de identificar os pontos fortes e as limitações de cada solução em diferentes cenários de carga de dados.

1.2.2 Objetivos específicos

- Construir *pipelines* equivalentes nas ferramentas Airbyte e KNIME, utilizando tabelas representativas do banco de dados da DPE/RN;
- Monitorar os *pipelines* em ambiente controlado por meio do Datadog, coletando indicadores relevantes de desempenho conforme as particularidades arquiteturais de cada ferramenta;
- Analisar os resultados obtidos, identificando o perfil operacional de cada ferramenta e os contextos em que sua adoção pode ser mais vantajosa, considerando diferentes condições de uso e limitações de recursos.
- Fornecer subsídios técnicos que auxiliem gestores de TI na escolha da solução mais adequada às necessidades da organização;
- Contribuir para a literatura acadêmica ao apresentar um estudo comparativo aplicado ao contexto de instituições públicas brasileiras.

1.3 Organização do trabalho

O presente trabalho está estruturado em seis capítulos, organizados de modo a conduzir o leitor desde a contextualização do problema até a apresentação dos resultados e considerações finais.

O Capítulo 1 apresenta a introdução, expondo o contexto da pesquisa, sua justificativa, objetivos e relevância no âmbito da Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN). Também descreve os desafios enfrentados pelas

instituições públicas na integração e tratamento de dados, destacando a motivação para a comparação entre as ferramentas *low-code* Airbyte e KNIME.

O Capítulo 2 reúne o referencial teórico que fundamenta a pesquisa. São discutidos os conceitos essenciais de ETL e ELT, a estrutura e funcionamento dos *pipelines* de dados, bem como os princípios das plataformas *low-code*. Além disso, o capítulo apresenta uma visão geral das ferramentas utilizadas — Airbyte, KNIME e Datadog — destacando suas características, funcionamento e relevância no contexto da engenharia de dados.

O Capítulo 3 contempla os trabalhos relacionados, trazendo estudos e publicações que servem de base comparativa para o desenvolvimento da pesquisa. São exploradas abordagens acadêmicas e práticas que analisam o uso de ferramentas *low-code* em processos de integração de dados, permitindo situar o presente estudo dentro do panorama atual da literatura.

O Capítulo 4 descreve o estudo de caso aplicado à DPE/RN, apresentando detalhadamente a infraestrutura de testes construída, a metodologia adotada e o ambiente experimental de execução. Este capítulo também abrange as etapas de instalação, configuração e desenvolvimento dos *pipelines* nas ferramentas Airbyte e KNIME, além do monitoramento de desempenho realizado via Datadog.

O Capítulo 5 concentra-se na análise e discussão dos resultados, abordando o comportamento das ferramentas durante os testes, os indicadores de desempenho obtidos e as implicações práticas das diferenças arquiteturais observadas. O capítulo ainda destaca subsídios que podem orientar a escolha de ferramentas ETL *low-code* em ambientes institucionais semelhantes.

Finalmente, o Capítulo 6 apresenta as considerações finais, sintetizando as principais conclusões do estudo, suas limitações e sugestões de trabalhos futuros, especialmente quanto ao uso de ambientes de produção e à ampliação da observabilidade para outros tipos de processos e métricas.

2 Referencial teórico

No contexto contemporâneo – outubro de 2025 – observa-se que a sociedade está bastante inserida na denominada era dos dados. Seja no ambiente domiciliar ou corporativo os indivíduos produzem e transmitem dados a todo momento. A título de exemplo, podem ser citados os assistentes virtuais baseados em Inteligência Artificial (IA) desenvolvidos por grandes empresas de tecnologia, como a Alexa (Amazon) e a Siri (Apple), além dos modelos de linguagem de larga escala, que se apresentam amplamente difundidos no cotidiano moderno. Em se tratando do ambiente corporativo, o alto fluxo de atendimentos diários resulta na geração contínua de grandes volumes de dados, inclusive de natureza sensível, destacando-se nessa seara a Lei Geral de Proteção de Dados (LGPD), criada no Brasil para regulamentar o uso e garantir a proteção dos dados pessoais frente à crescente escala de sua coleta e processamento.

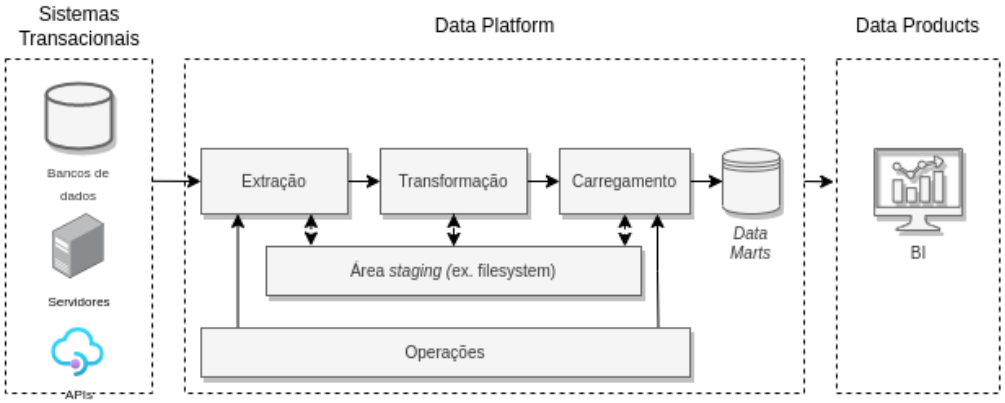
Diante do exposto, não há dúvidas de que um volume massivo de dados disseminados necessita de tratamento para que se transformem em informações relevantes que contribuam efetivamente para a vida em sociedade, especialmente em aspectos como apoio à tomada de decisões, produtividade e aprimoramento da assertividade estratégica. Nesse cenário, a etapa de Extração, Transformação e Carga (do inglês, ETL para *Extract, Transform and Load*) desempenha papel central nesse processo.

2.1 Fundamentos de ETL

A etapa de ETL ou ELT costuma ser a mais onerosa em termos de tempo dentro das organizações. Pesquisas do setor indicam que profissionais como analistas, gestores e cientistas de dados chegam a dedicar entre 60% e 80% de sua rotina exclusivamente à preparação de dados. Nesse contexto, dispor de ferramentas e serviços de ETL adequados, capazes de automatizar e acelerar essas atividades, torna-se fundamental para que engenheiros e cientistas de dados concentrem seus esforços

em tarefas de maior relevância estratégica para a organização (Castellanos *et al.*, 2021). Segundo Kimball (2002), *Extract, Transform and Load* (ETL) representa uma camada intermediária que fica entre os sistemas operacionais de origem e a camada de apresentação dos dados (KIMBAL, 2002). Conforme mencionado, essa etapa é considerada uma das mais críticas em todo o processo de construção de um *Data Warehouse (DW),* uma vez que é nesse momento que os dados brutos são preparados para uso analítico. Em outras palavras, a partir da conclusão dessa fase os dados passam a estar disponíveis para consumo e se tornam a base para a geração de *insights* que apoiam diretamente a tomada de decisões. O termo ETL descreve as etapas e a sequência tradicional de processamento de dados, que vai desde a extração até o carregamento. Esse processo é ilustrado por meio da Figura 1 e tem suas fases descritas a seguir.

Figura 1 – Processo de ETL

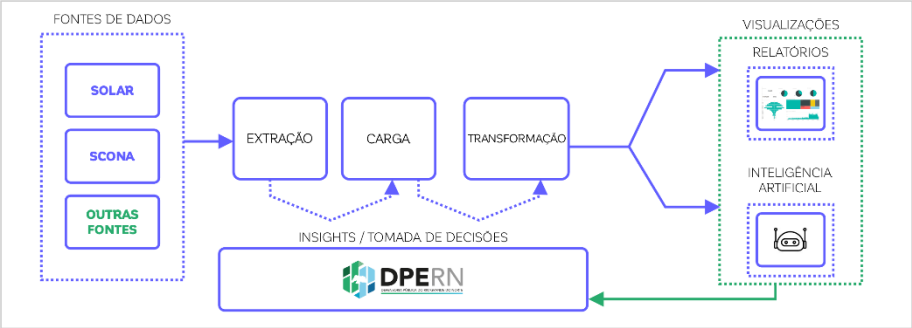


Fonte: Avancini (2022).

- **Extract (Extração):** É nessa fase que os dados são coletados das fontes diversas, que incluem bancos de dados transacionais, APIs, arquivos, *web scrapping*, entre outros.
- **Transform (Transformação):** A fase de transformação é responsável por realizar uma limpeza dos dados, seleção daqueles que são de fato necessários para análise, aplicação de regras de negócio, enriquecimento, agregação e padronização para garantir que esses dados cheguem consistentes ao DW.
- **Load (Carga):** Inserção dos dados transformados no depósito de dados de destino, prontos para análise.

Com a evolução das arquiteturas analíticas e o surgimento do *Modern Data Stack* (MDS), consolidou-se uma abordagem alternativa denominada ELT (*Extract, Load, Transform*). No contexto da DPE/RN, a estratégia adotada segue o paradigma ELT, em substituição ao modelo clássico de ETL cuja abordagem é descrita por meio da imagem a seguir:

Figura 2 – Processo de ELT



Fonte: Elaborado pela autora (2025).

A arquitetura ELT, adotada neste estudo como estratégia principal de integração de dados, é também a abordagem recomendada pela plataforma Airbyte, ferramenta que será detalhada em seções posteriores. Nesse modelo, os dados são inicialmente extraídos das fontes e carregados em seu formato bruto. A etapa de transformação ocorre posteriormente, já dentro do *Data Warehouse* (DW).

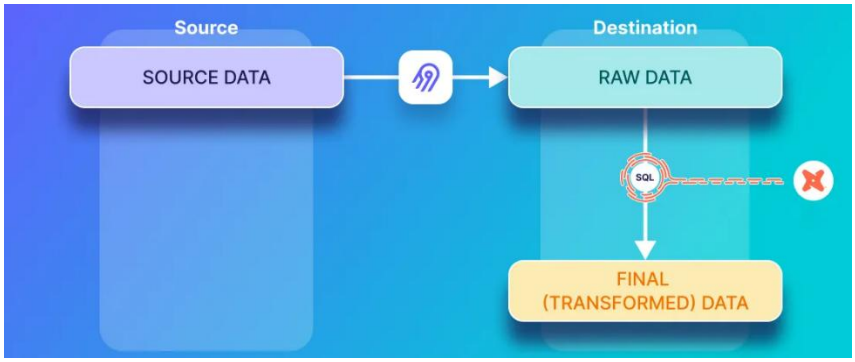
A abordagem de transformação no DW não apenas simplifica a arquitetura do *pipeline*, como também proporciona maior flexibilidade e escalabilidade na manipulação e análise dos dados. Historicamente, a difusão da arquitetura ETL ocorreu há cerca de três décadas, em um contexto de elevados custos de armazenamento e processamento. Nesse cenário, a transformação intermediária cumpria a função de reduzir o volume de dados antes de seu carregamento no repositório final, minimizando assim os gastos com infraestrutura.

Com o avanço tecnológico e a crescente necessidade de manter o histórico completo dos dados, tornou-se mais eficiente carregá-los primeiro em sua forma bruta para, em seguida, aplicar as transformações necessárias diretamente no DW. Essa mudança possibilitou que apenas os dados relevantes fossem preparados para análise, sem perda do conteúdo original. Além disso, à medida que novos requisitos de

negócio surgem — o que é bastante comum —, a presença dos dados já carregados permite realizar novas transformações e enriquecimentos de forma ágil, evitando a necessidade de repetir a etapa de extração (Vasconcelos, 2025).

Com o advento do MDS surgiram também tecnologias especializadas para cada etapa do processo de integração de dados. Por exemplo, o Airbyte atua na extração e ingestão, enquanto o dbt (data build tool) desempenha papel central na transformação dos dados por meio de instruções SQL estruturadas — processo que o próprio Airbyte denomina normalização. No caso específico da DPE/RN, o dbt também é utilizado para essa finalidade. Contudo, como não constitui o foco central deste trabalho, seus aspectos técnicos não serão detalhados.

Figura 3 – Arquitetura ELT



Fonte: Airbyte (2025).

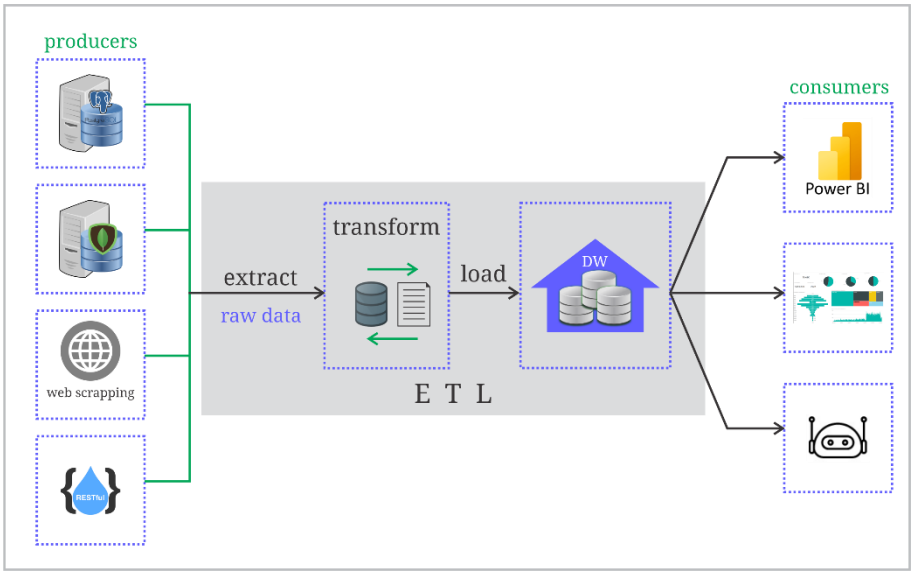
No campo da engenharia de dados, não existe uma resposta definitiva sobre qual abordagem, ETL ou ELT, é superior. A escolha depende das necessidades e características de cada instituição, bem como dos objetivos do projeto em questão. Assim, não há uma "melhor" solução universal, mas sim a que se mostra mais adequada ao contexto específico de aplicação. A literatura aponta que o ETL tende a ser mais apropriado em cenários que envolvem pequenos ou médios volumes de dados estruturados ou semiestruturados, sobretudo quando já existem padrões bem definidos de qualidade e governança. O ELT, por outro lado, revela-se mais vantajoso para grandes volumes de dados, geralmente não estruturados ou de maior complexidade, em situações que demandam flexibilidade e critérios variáveis de qualidade.

De modo geral, o ETL atende melhor a necessidades relacionadas a relatórios específicos e análises previamente definidas, enquanto o ELT oferece maior liberdade para explorações analíticas e consultas ad hoc. Em síntese, o ETL tende a se adequar a ambientes com infraestrutura mais limitada, ao passo que o ELT se beneficia de arquiteturas escaláveis, capazes de explorar plenamente o poder de processamento dos DWs modernos (Halakhellow, 2023). Considerando o escopo da DPE/RN, a arquitetura ELT é a que melhor se qualifica para atender às demandas propostas.

2.2 *Pipeline* de dados

Um *pipeline* de dados é composto por diversas etapas, incluindo a de ETL. Vale mencionar que todo processo de ETL (ou ELT) representa um *pipeline* de dados, mas nem todo *pipeline* de dados se compõe exclusivamente da etapa mencionada. O termo *Data Pipeline* (ou *pipeline* de dados) representa um conceito mais amplo e moderno do que o tradicional ETL. Enquanto no passado ambos eram utilizados quase como sinônimos, o *pipeline* de dados hoje compreende toda a estrutura que conecta as fontes geradoras de dados (como sistemas transacionais, plataformas de vendas e aplicações corporativas) aos seus consumidores finais (como *dashboards*, relatórios analíticos, aplicações de *machine learning*, entre outros). A Figura 4, a seguir, exemplifica um *pipeline* completo.

Figura 4 – *Pipeline* de dados



Fonte: Elaborado pela autora (2025).

Conforme ilustrado, do lado esquerdo estão os *producers*, que representam sistemas e bancos de dados responsáveis pela geração de registros. Essas fontes diversas alimentam a etapa de extração de dados. Em seguida, os dados extraídos passam pelo processo de transformação, onde são estruturados e ajustados, e logo após são carregados em um Data Warehouse (DW). A partir desse ponto, os dados armazenados no DW tornam-se disponíveis para consumo pelas tecnologias analíticas. Do lado direito da figura, estão os *consumers*, que podem incluir painéis de *Business Intelligence* (BI), relatórios customizados ou mesmo aplicações de *machine learning*, que exploram e utilizam as informações para gerar valor analítico.

De acordo com Weller (2020), a automação das etapas de um *pipeline* permite reduzir erros manuais, aumentar a eficiência e garantir maior consistência nos dados tratados. Além disso, *pipelines* bem projetados são capazes de lidar com grandes volumes de dados em tempo quase real, o que é fundamental para organizações que dependem de informações atualizadas para a tomada de decisão.

2.3 Plataformas *Low-Code*

Em um cenário marcado por rápidas transformações, a busca por agilidade e inovação torna-se cada vez mais essencial. Para quem não possui amplo conhecimento em programação, desenvolver um aplicativo do zero pode parecer uma tarefa complexa e distante (Garcia, 2024). Nesse contexto, ganham destaque as abordagens *low-code* e *no-code* (LC/NC).

As plataformas *low-code* (LC) têm como objetivo reduzir a complexidade do desenvolvimento de soluções, permitindo que usuários com diferentes níveis de conhecimento técnico construam aplicações com mínima necessidade de programação, como sugere o próprio termo em tradução livre, “pouco código”. Já as plataformas *no-code* (NC) avançam um passo além, ao eliminar completamente a necessidade de escrever código, oferecendo um processo de criação totalmente visual e simplificado.

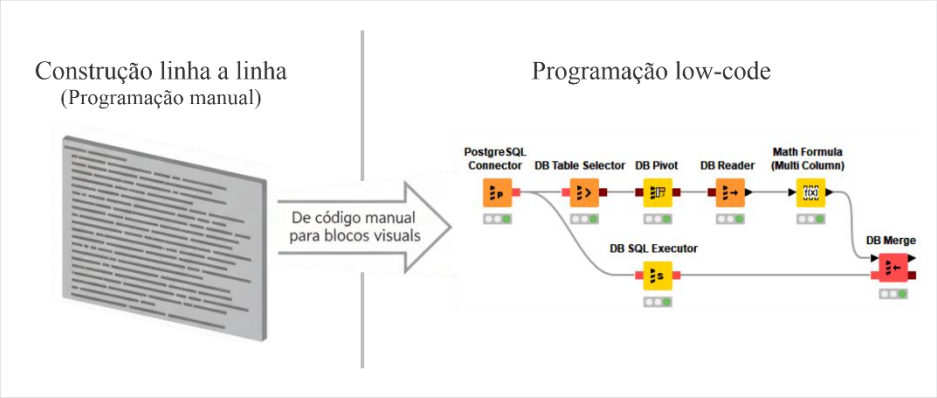
Embora largamente utilizadas, as definições de *low-code* e *no-code* ainda não são plenamente consensuais. Saarinen (2024) aponta que a ausência de uma conceituação

precisa pode dificultar a tomada de decisão sobre qual plataforma adotar e até mesmo sobre a viabilidade de optar por esse tipo de abordagem. De forma prática, o autor explica que as plataformas *low-code* reduzem a necessidade de programação ao permitir que boa parte do desenvolvimento ocorra por meio de interfaces visuais e componentes pré-construídos, enquanto as soluções *no-code* eliminam completamente o contato com código, restringindo a construção às interações gráficas. Essa diferenciação mostra como tais ferramentas variam em profundidade e público-alvo, indo desde usuários técnicos até profissionais de negócio.

Um exemplo que evidencia o impacto das tecnologias *low-code*/*no-code* é apresentado no estudo conduzido pela Forrester Consulting sobre a Mendix Platform. Os resultados demonstraram que, enquanto no desenvolvimento tradicional uma aplicação exigiria em média 2,26 desenvolvedores, com a utilização da Mendix esse número foi reduzido para menos de um profissional por aplicação (0,96), o que representa um ganho significativo de eficiência. No contexto da ciência de dados, plataformas modernas de LC/NC se diferenciam das abordagens tradicionais de ETL voltadas ao processamento em lote por oferecerem suporte à ingestão de dados em fluxo contínuo (*streaming*) proveniente de diferentes fontes, como Apache Kafka, AWS Kinesis e APIs RESTful (Heritage, 2025).

Um ponto destacado por Saarinen (2024) é a amplitude de aplicações possíveis dentro do universo das LCDPs (*Low-Code Development Platforms*). O autor classifica essas ferramentas em categorias que vão desde plataformas de gerenciamento de dados e fluxos de trabalho até ambientes integrados de desenvolvimento e soluções multiuso para configuração e integração. Além disso, revisões recentes da literatura indicam que o uso mais frequente dessas plataformas ocorre em aplicações de negócios e gestão de bancos de dados, seguidos pela criação de interfaces de usuário e aplicações web e móveis. Essa variedade reforça a flexibilidade das abordagens LC/NC e sua relevância em diferentes domínios organizacionais. De forma análoga à construção com blocos, a figura a seguir ilustra como pode ser vista a programação *low-code*.

Figura 5 – Representação da programação manual vs *low-code*



Fonte: Desenvolvido pela autora (2025).

É importante destacar que as plataformas *low-code/no-code* não apenas aceleram o desenvolvimento de soluções, como também democratizam o acesso à criação de aplicações, permitindo que profissionais com diferentes níveis de conhecimento técnico participem ativamente do processo. Ademais, tais soluções possibilitam que especialistas da área de negócios construam *pipelines* e implementem projetos que normalmente exigiriam suporte da equipe de TI, reduzindo essa dependência. Além da agilidade, essas ferramentas oferecem integração simplificada, redução de custos e maior flexibilidade, possibilitando às organizações responderem com rapidez às demandas do mercado.

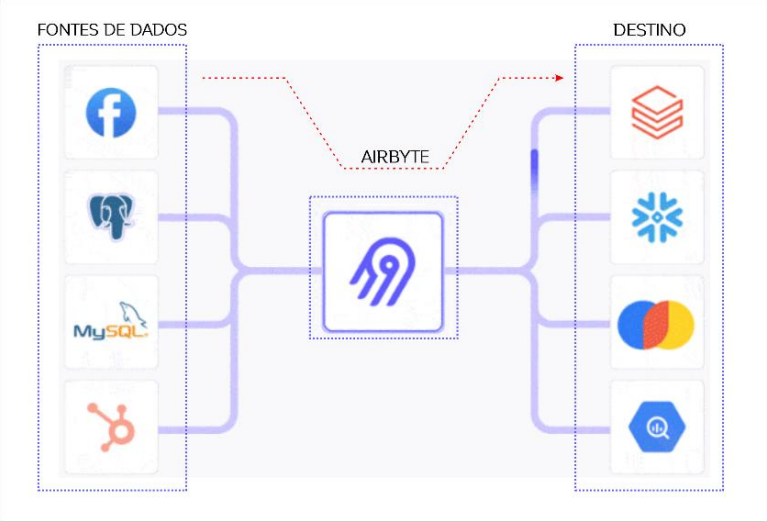
Cabe ressaltar que, apesar do ganho de velocidade proporcionado pelas plataformas LC/NC, uma vez que se usam componentes pré-construídos e interfaces visuais, estudos alertam que é fundamental implementar governança adequada para mitigar riscos de segurança e garantir a qualidade do código (Bodicherla, 2023). Observa-se que as plataformas *low-code/no-code* apresentam significativa relevância no contexto atual. Contudo, é importante salientar que o uso reduzido ou inexistente de código não dispensa o conhecimento técnico, mas atenua sua complexidade, conferindo maior agilidade e acessibilidade ao processo de desenvolvimento.

2.4 Airbyte

No contexto do ELT, essa tecnologia atua na etapa inicial do processo, realizando a extração e o carregamento dos dados. O Airbyte é uma plataforma de

integração de dados de código aberto que permite a configuração de *pipelines* de forma ágil e intuitiva, apoiando-se em uma extensa biblioteca de conectores pré-construídos para diversas fontes e destinos (AIRBYTE, 2025). Classificada como uma solução *low-code*, essa plataforma disponibiliza uma interface gráfica de usuário (GUI, do inglês *Graphic User Interface*) que possibilita a criação de fluxos de integração com poucos cliques, reduzindo de maneira significativa a necessidade de programação manual. Ao mesmo tempo, mantém a flexibilidade para cenários mais complexos, permitindo o desenvolvimento de conectores customizados em linguagens como Python ou Java.

Figura 6 – Fluxo de movimentação do Airbyte



Fonte: Airbyte (2025) (Adaptado).

Como mostra a imagem, o Airbyte atua como intermediário entre as fontes de dados e os destinos de armazenamento. À esquerda estão as fontes — como redes sociais, bancos de dados e aplicações — de onde os dados são extraídos. Em seguida, a plataforma orquestra e integra essas informações por meio de conectores pré-construídos, enviando-as aos destinos à direita, como *data warehouses*, plataformas analíticas ou sistemas em nuvem.

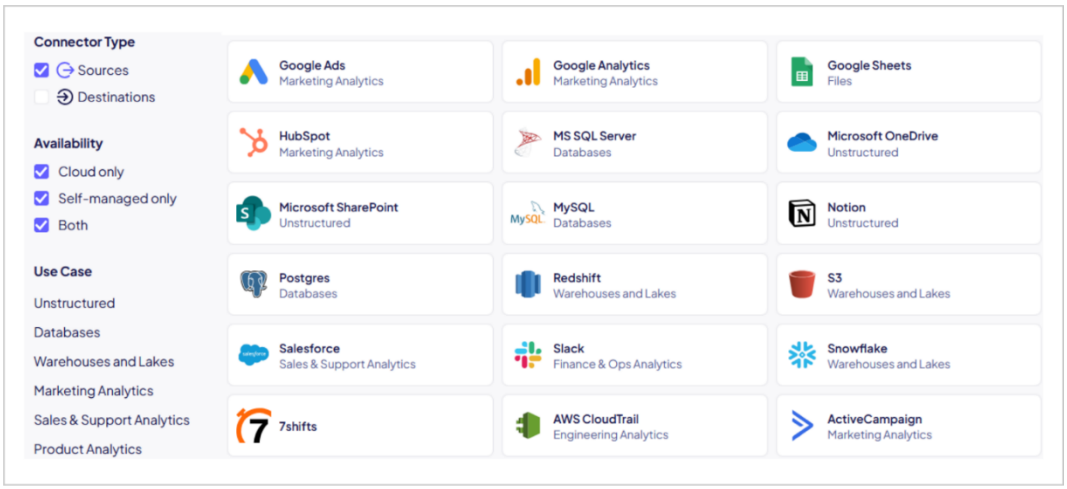
O fluxo ELT representa a extração, carga e posterior transformação dos dados, evidenciando a capacidade do Airbyte de simplificar integrações entre ambientes distintos. Em síntese, sua combinação de simplicidade operacional e flexibilidade

técnica o consolida como uma solução robusta para instituições que buscam agilidade, escalabilidade e controle na integração de dados.

2.4.1 Conectores do Airbyte

Semelhantes às tomadas e plugues que permitem conectar diferentes aparelhos em redes elétricas de padrões distintos, os conectores do Airbyte podem ser compreendidos como adaptadores universais, ou seja, na prática, esses conectores são os módulos que possibilitam a comunicação entre sistemas de origem e de destino, viabilizando a integração de dados de forma padronizada. Em síntese, um conector é o componente que permite extrair dados de uma fonte ou enviá-los a um destino. Segundo a documentação, há dois tipos: *source connectors*, que realizam a extração, e *destination connectors*, que fazem o carregamento (Airbyte, 2025). Um mesmo sistema pode exercer ambos os papéis — como o PostgreSQL, que pode atuar tanto na origem quanto no destino dos dados. A Figura 7 ilustra alguns dos *source conectors* disponibilizados pela plataforma.

Figura 7 – *Source conectors* do Airbyte

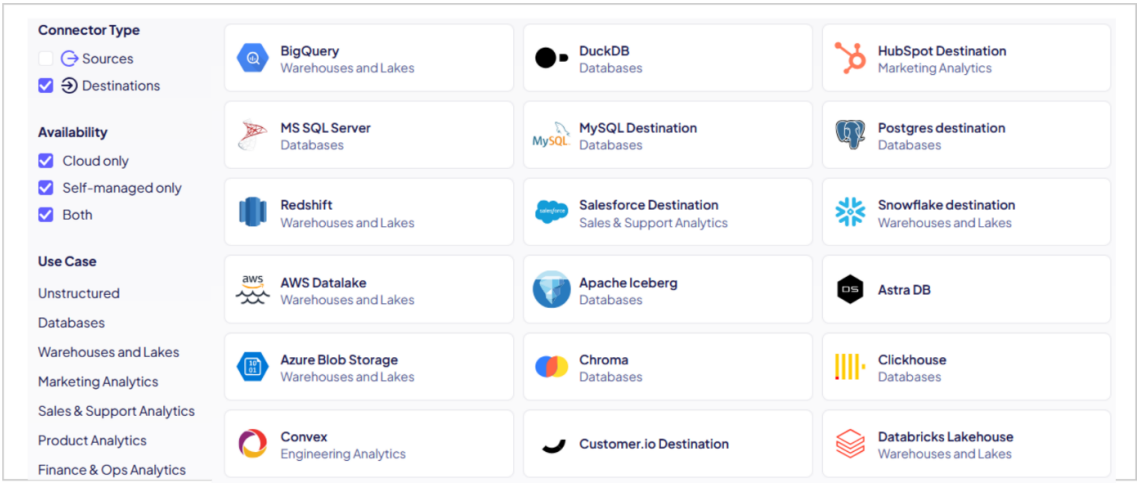


Fonte: Airbyte (2025).

A versatilidade do Airbyte evidencia-se na ampla gama de conectores compatíveis com diferentes fontes e destinos, permitindo sua adoção em variados contextos arquiteturais. Conectores como MySQL e PostgreSQL exemplificam essa

flexibilidade ao integrar sistemas legados e soluções modernas em nuvem. Os *destination connectors* têm a função de receber e armazenar os dados extraídos, tornando-os disponíveis para análise e visualização. Segundo a documentação oficial, incluem *Data Warehouses* (BigQuery, Snowflake), *Data Lakes* (S3, Azure Blob), bancos relacionais (PostgreSQL, MySQL) e ferramentas analíticas (Google Analytics). (Airbyte, 2025). AFigura 8, a seguir, exemplifica alguns dos *destination connectors* mais utilizados na plataforma.

Figura 8 – *Destination conectors* do Airbyte



Fonte: Airbyte (2025).

Os conectores do Airbyte desempenham papel central na integração de dados, permitindo tanto a extração a partir das fontes (*source connectors*) quanto o carregamento em repositórios de destino (*destination connectors*). Em muitos casos, uma mesma tecnologia pode assumir ambos os papéis — como PostgreSQL e MySQL.

2.4.2 Conexões e *pipelines* de dados no Airbyte

Enquanto os conectores representam os componentes responsáveis por extrair dados de uma fonte ou enviá-los para um destino, as conexões no Airbyte constituem, de fato, os *pipelines* de extração automatizados de dados. Cada conexão é formada pela combinação de um *source connector* configurado e um *destination connector*, estabelecendo o fluxo de replicação entre origem e destino.

Essas conexões permitem definir parâmetros fundamentais, como a frequência de sincronização (manual, horária, diária, entre outros), os conjuntos de dados ou *streams* a serem replicados e o formato de carregamento. Na prática, é por meio da configuração de uma conexão que os *pipelines* ganham vida, possibilitando o transporte contínuo de dados entre sistemas distintos de forma confiável. Essa abordagem pode ser comparada a uma linha de produção em uma fábrica: os conectores funcionam como máquinas especializadas em receber insumos e entregar produtos intermediários, enquanto a conexão atua como a esteira que organiza o fluxo de trabalho, garantindo que os insumos saiam de um ponto e cheguem a outro no tempo e forma desejados.

O Airbyte não apenas simplifica a integração de dados heterogêneos, mas também assegura flexibilidade para ajustar o processo conforme as necessidades específicas de cada organização (Airbyte, 2025). Conforme já mencionado, a criação de um *pipeline* no Airbyte segue uma sequência lógica de configurações que, combinadas, possibilitam a automação da movimentação de dados entre sistemas. A seguir, são detalhados os principais elementos que compõem um *pipeline* de dados no Airbyte.

1. **Identificação e configuração do conector de origem**

   Nesta etapa, busca-se definir o tipo de sistema ou armazenamento que contém os dados de origem — ou seja, o ambiente de onde serão extraídos. O Airbyte suporta diversas fontes, como bancos de dados relacionais (PostgreSQL, MySQL), APIs (Google Analytics, Stripe), arquivos estruturados (CSV, JSON) e plataformas em nuvem (Salesforce, BigQuery).

   O conector certificado do Airbyte para PostgreSQL permite a replicação de dados a partir de tabelas, *views* e *views* materializadas, oferecendo suporte a diferentes modos de sincronização, incluindo técnicas como Change Data Capture (CDC) e leitura incremental via coluna *xmin*. A ferramenta também garante replicações confiáveis, mesmo em tabelas de grande porte, por meio de mecanismos como *checkpointing* e divisão das leituras em blocos (Airbyte, 2025).

2. **Identificação e configuração do conector de destino**

Nesta etapa, define-se o destino dos dados, isto é, o ambiente onde serão armazenados após a extração. Considerando que os sistemas do estudo de caso utilizam bancos relacionais, adotou-se o PostgreSQL como destino, garantindo compatibilidade e facilitando a análise. A ferramenta assegura integridade e consistência durante a carga, sendo adequada para grandes volumes e cargas incrementais com alta confiabilidade (Airbyte, 2025).

É importante mencionarar que o Airbyte possui limitações ao usar o PostgreSQL como destino, sendo recomendado pela própria documentação que o banco não ultrapasse 10 GB de dados, a fim de assegurar estabilidade e bom desempenho na replicação. No entanto, essa recomendação não se alinha à realidade da Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN), onde o banco de dados do sistema SOLAR possui aproximadamente 60 GB de informações armazenadas (dados de setembro de 2025). Ainda assim, optou-se por utilizar o PostgreSQL como conector de destino para fins de padronização e compatibilidade com os sistemas analisados no escopo desta pesquisa. A seguir, são apresentadas as principais configurações necessárias para a adição do conector do PostgreSQL como destino de armazenamento, no Airbyte:

- ***Host***: endereço do servidor onde o banco está hospedado;
- **Porta**: número da porta utilizada para comunicação (por padrão, a 5432);
- **Usuário e senha**: credenciais de acesso ao banco;
- **Nome do *schema* padrão**: o(s) *schema*(s) que serão utilizados para localizar os objetos referenciados nas instruções SQL, podendo ser um ou mais, separados por vírgulas;
- **Nome do banco de dados**: o banco ao qual será feita a conexão;
- **Parâmetros JDBC (opcional)**: utilizados para configurações adicionais na *string* de conexão.

Esses elementos são fundamentais para garantir uma conexão segura, estável e corretamente direcionada ao ambiente de armazenamento dos dados replicados.

3. **Definir as formas de sincronização dos dados**

O conector PostgreSQL, quando utilizado como destino no Airbyte, oferece suporte a diferentes modos de sincronização. A escolha do modo adequado depende dos requisitos do *pipeline* e da natureza dos dados a serem replicados. Os modos disponíveis incluem:

Full Refresh Sync: neste modo, todos os dados da origem são carregados novamente a cada sincronização. É útil em cenários em que não há controle sobre alterações incrementais ou quando se deseja garantir que os dados estejam sempre atualizados de forma completa, mesmo com maior custo computacional.

Incremental - Append Sync: realiza a carga apenas dos dados novos ou modificados desde a última sincronização, sem excluir os registros anteriores. É indicado para cenários em que os dados são continuamente acrescentados, otimizando o tempo de sincronização e o uso de recursos.

Incremental - Append + Deduped Sync: semelhante ao modo anterior, mas com um processo adicional de desduplicação. O Airbyte identifica e remove registros duplicados com base em chaves primárias ou campos de controle, garantindo maior integridade dos dados no destino.

4. **Definição de frequência das *syncs***

Uma parte importante no planejamento de *pipelines* de ETL/ELT com Airbyte é determinar com qual periodicidade ocorrerá a sincronização dos dados entre a fonte e o destino. O Airbyte oferece diferentes formas de configurar esse agendamento para atender às necessidades de latência, carga de dados, disponibilidade do sistema de origem e impacto operacional.

Opções de agendamento disponíveis no Airbyte:

- **Sincronizações agendadas:** executadas automaticamente em intervalos definidos (1h, 2h, 6h, 8h, 12h ou 24h), com uma sincronização inicial imediata ao criar a conexão.
- **Sincronizações com CRON**: permitem definir horários exatos ou padrões específicos (ex.: todos os dias às 05h), utilizando o agendador **Quartz**.
- **Sincronizações manuais**: iniciadas sob demanda pelo usuário, pela interface ou via **API**, quando não há agendamento automático.

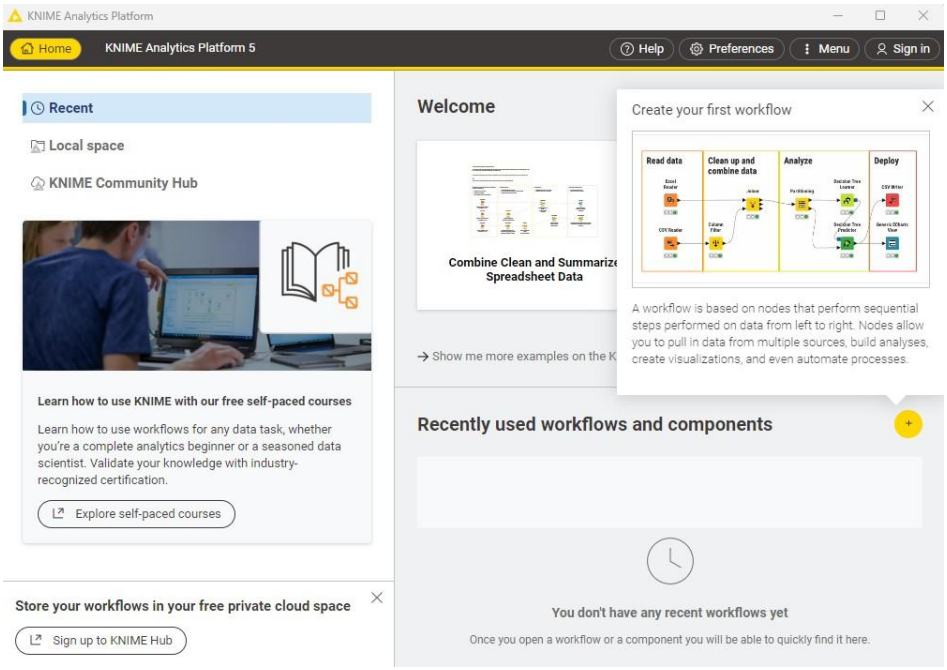
Em relação aos critérios para escolha da frequência, a periodicidade das sincronizações deve considerar: **(i)** o volume de dados e o impacto na performance; **(ii)** a sensibilidade ao atraso entre a fonte e o destino; **(iii)** os horários críticos de uso do sistema; e **(iv)** as janelas de manutenção ou disponibilidade do ambiente de origem. No contexto da DPE/RN, optou-se por um agendamento diário, com execução da sincronização programada para às 5h da manhã, todos os dias. Essa escolha justifica-se por:

1. **Redução de impacto:** como os bancos estão em produção, evita-se sincronizar em horários de uso para prevenir lentidão ou interferências nos serviços do órgão.
2. **Garantia de atualização:** a sincronização diária mantém os dados suficientemente atualizados para análises e relatórios, sem sobrecarregar o sistema.
3. **Facilidade operacional:** Na DPE/RN, o processo é complementado por automações na VM, que recria o contêiner diariamente via CRON, enquanto o Quartz CRON do Airbyte agenda as execuções internas, assegurando um fluxo contínuo e estável.

2.5 KNIME

O KNIME – *Konstanz Information Miner* – é uma plataforma voltada à análise de dados e à construção de fluxos de trabalho automatizados, cuja principal característica é a interface gráfica baseada em "nós" (*nodes*), permitindo que os usuários desenvolvam *pipelines* complexos de forma visual (Berthold, 2009). Cada nó representa uma etapa específica, como transformação, importação, junção, filtragem ou visualização dos dados.

Figura 9 – Interface inicial do KNIME



Fonte: Capturado pela autora (2025).

A ferramenta disponibiliza uma ampla variedade de conectores — mais de 300, conforme documentação oficial — que viabilizam a integração com diferentes tipos de fontes e destinos de dados, incluindo bancos relacionais, arquivos estruturados, APIs, serviços em nuvem e ambientes distribuídos. Além disso, o KNIME suporta tanto transformações executadas diretamente no banco de dados quanto processamentos realizados fora dele, de acordo com as necessidades do projeto. Disponibilizada gratuitamente em grande parte de seus recursos, a ferramenta foi desenvolvida em Java e consolidou-se como um ambiente robusto para integração, processamento e visualização de dados.

O aprendizado do KNIME é facilitado pela ampla disponibilidade de materiais instrucionais on-line — como guias, tutoriais e exemplos práticos — o que favorece sua adoção mesmo por usuários iniciantes. No entanto, apesar de ser uma plataforma acessível, é necessário dedicar tempo para explorar com atenção os muitos detalhes presentes nesses materiais, de modo que o aprendizado se torne realmente efetivo.

Uma das principais características do KNIME é o uso de uma interface gráfica baseada na lógica de *drag-and-drop*. Ao invés de programar manualmente cada instrução, os fluxos de trabalho podem ser construídos a partir de blocos chamados

nodes (ou "nós"), que encapsulam diferentes funcionalidades. Essa abordagem reduz a necessidade de programação formal, o que aproxima o KNIME das ferramentas *low-code* e *no-code*. Ainda assim, para casos mais avançados, a plataforma oferece suporte à escrita de trechos de código em linguagens como Python, R e JavaScript, o que amplia a flexibilidade da solução e favorece a integração com tecnologias modernas (Moreira, 2023).

O conceito de interface gráfica no KNIME remete às GUIs tradicionais, cujo modelo WIMP (*Windows, Icons, Menus, Pointing devices*) foi amplamente difundido entre as décadas de 1970 e 1990. Com base nessa lógica, a plataforma organiza seus recursos em fluxogramas visuais, nos quais ícones representam operações, janelas exibem resultados e menus permitem acesso a diferentes funções. Tal estrutura democratiza o uso da ferramenta, pois dispensa conhecimentos avançados em linguagens formais de programação, permitindo que analistas e profissionais de diferentes áreas construam soluções complexas com relativa facilidade. Outro destaque é o caráter colaborativo da plataforma, reforçado pelo KNIME Hub, pela comunidade ativa e pelo fórum oficial, nos quais usuários compartilham soluções, tiram dúvidas e divulgam boas práticas. Esses espaços funcionam como ecossistemas de apoio que estimulam tanto o aprendizado quanto a inovação, tornando o KNIME uma referência em acessibilidade e disseminação de conhecimento no campo da ciência de dados (Moreira, 2023).

No que se refere à organização dos fluxos de trabalho, a plataforma dispõe de recursos como *Metanodes* e *Components* (KNIME, 2025). Ambos permitem encapsular sequências de operações em blocos compactos, facilitando a visualização de fluxos mais extensos e complexos. Enquanto os *Metanodes* atuam como agrupadores que simplificam a interpretação do *workflow*, os *Components* oferecem funcionalidades adicionais, como maior capacidade de configuração, compartilhamento em servidores e manipulação de variáveis de fluxo. Esses recursos tornam os projetos mais modulares e escaláveis, evitando a sobrecarga visual e mantendo a clareza na execução das tarefas.

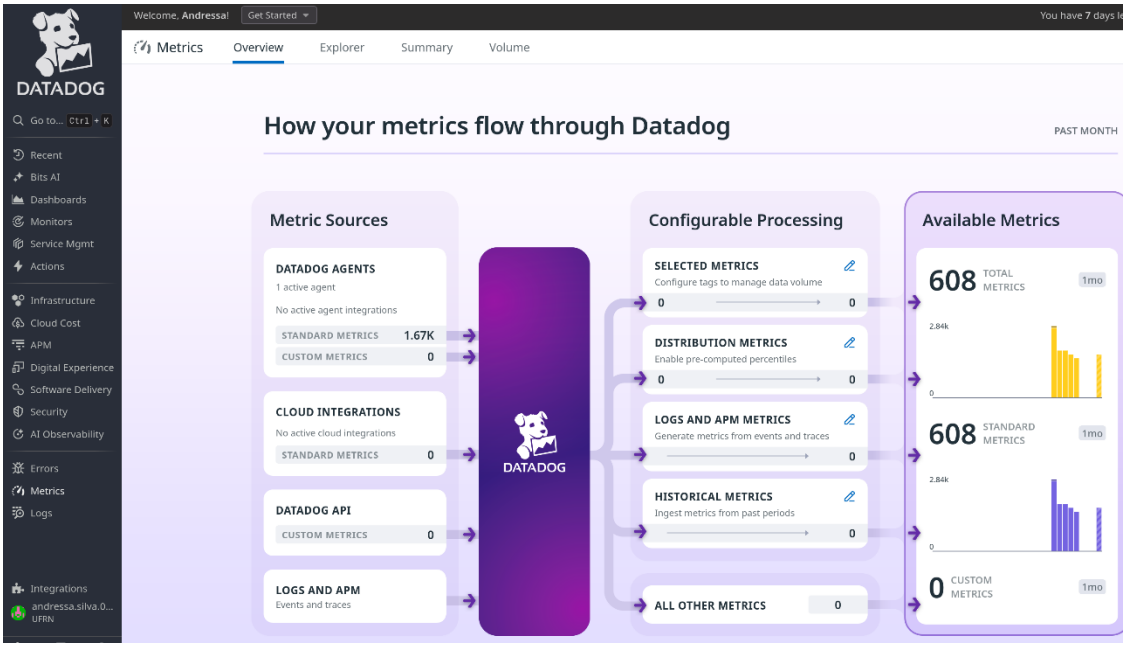
O KNIME suporta a importação de dados provenientes de múltiplas fontes e formatos, permitindo a criação e manipulação de bancos relacionais. Isso viabiliza a

exploração de padrões e relações entre variáveis de diferentes arquivos, o que o torna especialmente útil em projetos de mineração de dados e aprendizado de máquina. (KNIME, 2025). A combinação de interface gráfica intuitiva, extensibilidade por linguagens de programação e suporte comunitário consolida o KNIME como uma ferramenta versátil e poderosa para ciência de dados e integração de sistemas.

2.6 Datadog

O Datadog é uma plataforma unificada de monitoramento e segurança projetada para atender às exigências da era da computação em nuvem. Seu principal objetivo é centralizar, em um único ambiente, as ferramentas e recursos necessários para observar e analisar o desempenho de sistemas, aplicações e infraestruturas, reduzindo custos e complexidades associadas ao uso de múltiplas soluções isoladas. Desde sua criação, em 2010, o Datadog evoluiu de uma ferramenta de monitoramento em tempo real para uma plataforma completa de observabilidade, abrangendo métricas, logs e rastreamentos (*traces*), além de oferecer recursos integrados voltados às áreas de Desenvolvimento, Segurança e Operações (DevSecOps) (Datadog, 2025).

Figura 10 – Visão geral do painel de métricas do Datadog



Fonte: Capturado pela autora (2025).

Embora o Datadog seja um serviço pago, ele oferece opções gratuitas de avaliação, incluindo um período de teste de 14 dias, que permite acesso a recursos essenciais de monitoramento e análise. Neste trabalho, foi utilizada essa versão de avaliação do Datadog, suficiente para demonstrar suas principais funcionalidades e validar sua aplicabilidade no contexto dos experimentos realizados.

O funcionamento dessa tecnologia varia conforme o tipo de ambiente monitorado. Em sistemas locais, como o KNIME, o monitoramento é realizado por meio do Datadog Agent, um software de código aberto responsável por coletar métricas de CPU, memória, disco e rede, além de logs e processos em execução. Já em ferramentas contêinerizadas, como o Airbyte, a coleta de métricas pode ser realizada diretamente a partir do container airbyte-abctl-control-plane, que fornece informações sobre o desempenho e o comportamento do sistema. Essa abordagem foi necessária porque o Airbyte somente disponibiliza integração nativa com o OpenTelemetry — e, consequentemente, com o Datadog Collector — na versão Enterprise Self-Managed.

2.6.1 Métricas

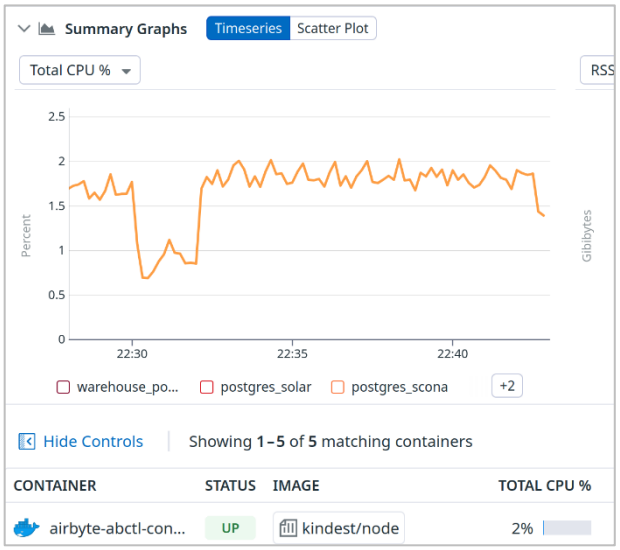
No Datadog, as métricas são valores numéricos coletados ao longo do tempo que permitem rastrear e quantificar o comportamento de sistemas, aplicações e infraestrutura (Datadog, 2025). Cada métrica representa um ponto de dado contendo um valor e um carimbo de data e hora, e a sequência desses pontos forma uma série temporal. O objetivo das métricas na ferramenta é oferecer uma visão abrangente da saúde e do desempenho do ambiente monitorado, permitindo acompanhar a operação dos sistemas em tempo real e identificar comportamentos anômalos antes que causem impacto significativo. Essas métricas proporcionam uma visão consolidada do estado do ambiente, auxiliando na identificação de padrões, detecção de anomalias e tomada de decisões baseadas em dados.

Figura 11 – Exemplo de como o Datadog armazena métricas

Fonte: Capturado pela autora (2025).

Como ilustrado na Figura 11, cada ponto de dado representa um valor numérico medido em um momento específico — por exemplo, o uso de CPU, memória ou tempo de execução de uma tarefa. Esse valor vem acompanhado de uma marca temporal, que registra a hora exata em que a medição foi feita. A Figura 12 ilustra um gráfico de série temporal do painel do Datadog que exibe a porcentagem de uso da CPU pelos containers monitorados.

Figura 12 – Métrica coletada pelo Datadog Agent



Fonte: Capturado pela autora (2025).

Assim como o Prometheus, uma ferramenta de observabilidade e monitoramento bastante difundida no mercado, o Datadog suporta diferentes tipos de

métricas, como *count*, *rate*, *gauge*, *histogram* e *distribution*, cada um adequado a um tipo de análise. A coleta dessas métricas pode ocorrer via Datadog Agent, DogStatsD, HTTP API, OpenTelemetry, ou ser gerada diretamente a partir de logs. Uma vez coletadas, as métricas são agregadas e organizadas pelo Datadog, podendo ser visualizadas em *dashboards* interativos e personalizáveis disponibilizados pela própria plataforma. No caso do Airbyte, o Datadog oferece um *dashboard* pré-configurado, que exibe automaticamente as principais métricas de desempenho e sincronização do sistema. No entanto, esse painel está disponível apenas na versão Enterprise Self-Managed do Airbyte. Assim, no presente trabalho, o *dashboard* específico do Airbyte não pôde ser exibido, uma vez que foi utilizada a versão on-premise 2.0, que não inclui essa integração nativa.

Essas métricas são classificadas em diferentes categorias, conforme descrito no Quadro 1 a seguir:

Quadro 1 – Tipos de métricas no Datadog

Tipo	Descrição
Count (Contagem)	Soma todos os eventos em um intervalo de tempo. Útil para medir quantas vezes algo aconteceu, como requisições ou erros.
Rate (Taxa)	Mostra quantos eventos ocorrem por segundo com base na contagem. Boa para medir velocidade de processamento ou requisições por segundo.
Gauge (Indicador)	Registra o valor atual de um recurso que pode variar, como uso de CPU, memória ou espaço em disco. Pode subir ou descer.
Histogram (Histograma)	Agrupa e resume valores de desempenho (como tempos de resposta) em faixas, mostrando média, mediana e máximos. Útil para analisar variação de tempo.
Distribution (Distribuição)	Parecido com o histograma, mas combina dados de vários hosts e mostra percentis. Ideal para comparar desempenho entre servidores ou containers.

Fonte: Elaborado pela autora (Adatpado da documentação do datadog) (2025).

3 Trabalhos relacionados

É sabido que o avanço das tecnologias de integração e processamento de dados tem impulsionado o surgimento de soluções voltadas à simplificação do desenvolvimento de *pipelines* e à automação das etapas do processo de ETL, destacando-se as ferramentas *low-code* e *no-code* por reduzirem a necessidade de programação e ampliarem o acesso tanto para desenvolvedores quanto para analistas de negócio. Apesar do crescente interesse por esse tipo de solução, observa-se que a literatura acadêmica ainda é incipiente no que diz respeito à comparação direta entre plataformas *low-code* aplicadas especificamente ao desenvolvimento de *pipelines* de dados.

Diversos estudos abordam temas correlatos, como a adoção de tecnologias *low-code* em processos de engenharia de dados, a avaliação de desempenho de ferramentas de integração e o monitoramento de cargas em ambientes produtivos e há pesquisas que realizam comparativos entre plataformas *low-code* voltadas ao desenvolvimento de software em geral. No entanto, não foram identificados na literatura trabalhos que tratem especificamente da engenharia de dados, isto é, estudos que comparem soluções *low-code* aplicadas à construção de *pipelines* de dados, com ênfase nas etapas de extração, transformação e carregamento (ETL). A seguir, são apresentados os trabalhos encontrados que mais se relacionam com o tema proposto neste estudo.

3.1 Avaliação de Métricas relacionadas ao desenvolvimento de projetos de *Business Intelligence*

O trabalho de autoria do Ivan Alisson é o que mais se relaciona com este estudo proposto. Ele se concentra na avaliação de métricas relacionadas ao desenvolvimento de projetos de *Business Intelligence* (BI), propondo uma solução para o desafio da integração e alta rotatividade de novos membros em equipes de dados. Para isso, o artigo realiza uma análise comparativa detalhada do processo de construção de *pipelines* de ETL.

O estudo confronta a ferramenta *open-source low-code* Airbyte, destacada por sua interface amigável e capacidade de permitir que equipes menos técnicas construam *pipelines* de dados de forma rápida e intuitiva, simplificando o processo de ETL, contra a abordagem tradicional que utiliza a combinação de Apache Airflow e Spark. A avaliação é dividida em etapas que englobam métricas de desempenho técnico (como uso de CPU, *throughput*, tempo de execução e uso de memória), métricas do processo de desenvolvimento (como o tempo gasto pelo desenvolvedor para criar a solução), além de aspectos qualitativos, como a utilidade percebida e a facilidade de uso percebida.

O objetivo desse trabalho é minimizar os impactos da integração de novos membros e otimizar o progresso dos projetos.

3.2 Construção de uma plataforma *Low-code* versátil para integração de dados

O trabalho, uma Tese de Mestrado cujo título é *Building a Low-Code Platform for versatile Data Integration*, propõe o conceito e a implementação parcial de uma plataforma *low-code* para integração de dados versátil, atuando como uma alternativa econômica e mais eficiente às custosas soluções proprietárias de *Data Warehouse*.

O objetivo central é capacitar as empresas a unirem e preparar dados diversos de sistemas atuais e futuros de maneira simples, econômica e com economia de tempo, enfrentando o desafio imposto pelo crescente volume de dados e pela variedade de softwares em uso. Para atender a requisitos complexos, como limpeza, verificação de razoabilidade e modelagem de dados, a arquitetura utiliza uma combinação de estilos de microsserviços e arquitetura orientada a eventos, incorporando soluções de código aberto, como o Airbyte para o Serviço de Integração de Dados, e softwares desenvolvidos sob medida.

A plataforma desenvolvida para esse trabalho é intencionalmente projetada como uma Plataforma *Low-Code* (do inglês *Low-code platform* - LCP), gerenciada por uma Interface Gráfica do Usuário (GUI), permitindo que usuários com pouca ou nenhuma experiência em engenharia de software modelem dados e gerenciem todo o

processo ETL. A avaliação do conceito, baseada em cenários de uso, confirma que o sistema atende aos requisitos desafiadores da empresa (uma fornecedora de energia local) e de outros domínios, cumprindo o objetivo de criar uma plataforma fácil de usar e modular.

3.3 Construção de um *pipeline* de dados para o Sistema Alerta Rio utilizando KNIME Analytics Platform

O presente trabalho, uma dissertação de mestrado, insere-se no paradigma da Ciência Aberta, que preconiza o compartilhamento e a colaboração científica impulsionados pelas Tecnologias Digitais da Informação e Comunicação (TDIC). O objetivo geral do estudo é construir um *pipeline* de dados para os dados meteorológicos e pluviométricos fornecidos pelo Sistema Alerta Rio utilizando a KNIME Analytics Platform. Por esse motivo, relaciona-se com o estudo aqui proposto. A necessidade desse processamento surge porque o Sistema Alerta Rio disponibiliza esses dados governamentais abertos sem a estruturação adequada, o que os classifica como semiestruturados, demandando mais etapas de processamento para adequação.

Para atingir objetivo deste trabalho, o estudo empregou técnicas de Engenharia de Dados, desenvolvendo um *pipeline* modular usando o KNIME, demonstrando que essa plataforma *low-code* facilitou o uso e reduziu o tempo de desenvolvimento. O *pipeline* foi responsável pela integração, limpeza, pré-processamento e análise dos dados, agregando valor por meio da classificação de leituras por conceitos estabelecidos (como a Escala Beaufort para velocidade do vento).

Os dados transformados foram consolidados em um novo banco de dados e disponibilizados publicamente no repositório Mendeley Data, obtendo Maturidade 4 estrelas em Dados Abertos e servindo como fonte estruturada para futuros estudos e aplicações de aprendizado de máquina.

3.4 Análise comparativa entre plataformas de desenvolvimento *low-code* para web

O trabalho visa avaliar e comparar plataformas de desenvolvimento *low-code* para desenvolvimento web. O autor menciona que diante da escassez de especialistas em TI e das restrições financeiras as LCDPs (do inglês *Low Code Developtment Platforms*), surgem como uma solução eficaz para a criação rápida de software, especialmente de Produtos Mínimos Viáveis (MVPs).

Esse estudo adota uma metodologia prática, que inclui a implementação de um MVP de controle financeiro em plataformas como Adalo, Bubble, FlutterFlow e Outsystems, avaliando critérios quantitativos (como recursos nativos e bancos de dados suportados) e qualitativos (como usabilidade, personalização, escalabilidade e capacidade de migração). O autor destaca, com base nos resultados obtidos, que a escolha da plataforma deve considerar as necessidades específicas de cada projeto, indicando quais ferramentas se mostram mais adequadas para ambientes de grande porte e quais apresentam melhor desempenho em projetos de menor escala.

A relação entre o trabalho mencionado e o estudo aqui proposto reside na validação do paradigma *low-code*, embora em domínios de aplicação distintos. Em ambos os contextos (desenvolvimento de aplicações web e Engenharia de Dados/ETL), as soluções *low-code* compartilham o objetivo de reduzir a barreira técnica, acelerar o desenvolvimento e promover a agilidade. Assim como o Airbyte e o KNIME foram analisados como ferramentas *low-code* que permitem que equipes com menor conhecimento técnico criem *pipelines* de dados de forma mais rápida e intuitiva, as plataformas apontadas no estudo, de maneira análoga, são mencionadas como ideais para criação rápida e simplificada de sistemas para web.

3.5 Comparative Analysis of ETL Tools in Big Data Analytics

O trabalho desenvolvido por Qaiser *et al.* apresenta uma análise abrangente sobre o papel das ferramentas ETL no contexto do Big Data, destacando a necessidade de soluções que integrem grandes volumes de dados de forma eficiente e escalável. Os

autores comparam diferentes categorias de ferramentas — incluindo as baseadas em código, *open source*, de processamento em lote e com interface gráfica (GUI) —, enfatizando que estas últimas, por oferecerem recursos *low-code* ou *no-code*, têm se tornado cada vez mais populares devido à facilidade de uso e à funcionalidade *drag and drop* para construção de *pipelines* de dados.

Essa abordagem se conecta diretamente ao tema do presente estudo, uma vez que ambos os estudos exploram o papel das plataformas *low-code* na simplificação dos processos de integração e transformação de dados. Enquanto o artigo de Qaiser *et al.* estabelece um panorama geral das principais soluções de mercado, este estudo busca aprofundar a análise comparativa em ferramentas específicas — Airbyte e KNIME —, ambas classificadas como *open source* e baseadas em interface gráfica, reforçando a relevância prática desse tipo de solução no ecossistema de engenharia de dados.

Dessa forma, o trabalho de Qaiser e colaboradores oferece um referencial teórico e técnico que sustenta a proposta deste estudo ao evidenciar a importância das ferramentas *low-code* e *no-code* para a democratização do desenvolvimento de *pipelines* ETL, contextualizando a escolha de Airbyte e KNIME como objetos de estudo pertinentes e alinhados às tendências contemporâneas em *Big Data Analytics.*

4 Estudo de caso: Defensoria Pública do Estado do RN – DPE/RN

A Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN) é um órgão institucional essencial para garantir o acesso à justiça gratuita à população hipossuficiente do estado. Sua missão inclui a prestação de assistência jurídica integral e gratuita, tanto no âmbito judicial quanto extrajudicial, especialmente para pessoas que não dispõem de recursos financeiros para custear advogados particulares. A sede administrativa da DPE/RN está situada em Natal-RN e além desta, a instituição opera várias unidades de atendimento distribuídos pelo estado, inclusive núcleos especializados (como Núcleo de Defesa da Criança e do Adolescente, Núcleo de Assistência às Vítimas etc.), para atender diferentes demandas jurídicas e grupos vulneráveis (DPE/RN, 2025).

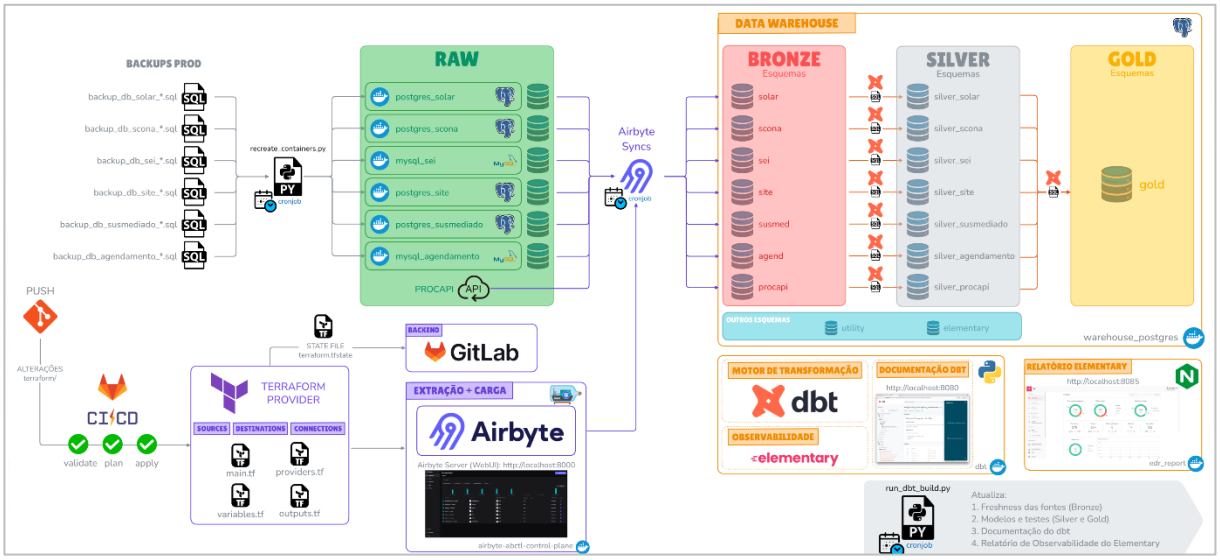
De acordo com o site institucional, a DPE/RN conta com dezenas de defensores públicos e milhares de atendimentos e processos em curso. Atualmente, mais de 40 núcleos de atendimento estão em operação em todo o estado, e a instituição já alcança aproximadamente 92% da população potiguar, índice que reflete não apenas a expansão significativa da cobertura do órgão, mas também a crescente geração de dados jurídicos, administrativos e sociais. Esse cenário reforça a necessidade de estratégias robustas de integração e análise de dados capazes de lidar com o grande volume e a diversidade de registros que são produzidas cotidianamente.

Com o advento da equipe de Residência em Tecnologia da Informação do IMD/UFRN, a DPE/RN deu início a um processo inédito de modernização de sua infraestrutura de dados. Até então, o órgão não dispunha de uma arquitetura voltada para o tratamento, integração e análise de dados, o que limitava a utilização estratégica dos subsídios disponíveis. O ambiente foi integralmente desenvolvido desde a sua concepção, fruto de pesquisas e do esforço coletivo dos residentes, que estudaram, avaliaram e implementaram as ferramentas mais adequadas ao contexto institucional. Esse trabalho resultou na construção de uma *pipeline* de dados robusta, flexível e escalável, capaz de sustentar processos de integração contínua, análises avançadas e

geração de *insights* estratégicos. Mais do que um avanço tecnológico, a iniciativa marca um divisor de águas na transformação digital da DPE/RN, colocando o órgão em patamar equivalente a instituições que já adotam a análise orientada por dados como base para suas decisões estratégicas. A partir desse esforço, foi estruturada uma solução completa de *Data Warehouse*, concebida para integrar e consolidar informações provenientes de sete sistemas corporativos críticos, anteriormente dispersos e heterogêneos. Essa consolidação permitiu a criação de uma única fonte da verdade institucional, centralizando dados jurídicos, administrativos e sociais em um repositório analítico de alta disponibilidade, capaz de subsidiar painéis gerenciais, relatórios dinâmicos e estudos estratégicos com maior consistência, confiabilidade e velocidade de acesso.

A arquitetura de dados foi concebida com base em uma abordagem moderna e *open source*, fundamentada no padrão *Medallion Architecture*, que organiza o fluxo de dados em três camadas: (i) Bronze, responsável pelo armazenamento dos dados brutos replicados das fontes originais; (ii) Silver, voltada ao tratamento, limpeza e padronização; e (iii) Gold, destinada à modelagem analítica e agregação para consumo em ferramentas de BI e Analytics, no contexto da DPE/RN foi utilizado o Power BI. Essa estrutura modular garante escalabilidade, governança e rastreabilidade dos dados, além de viabilizar a automação completa do processo de ELT.

Figura 13 – Infraestrutura de dados da DPE/RN



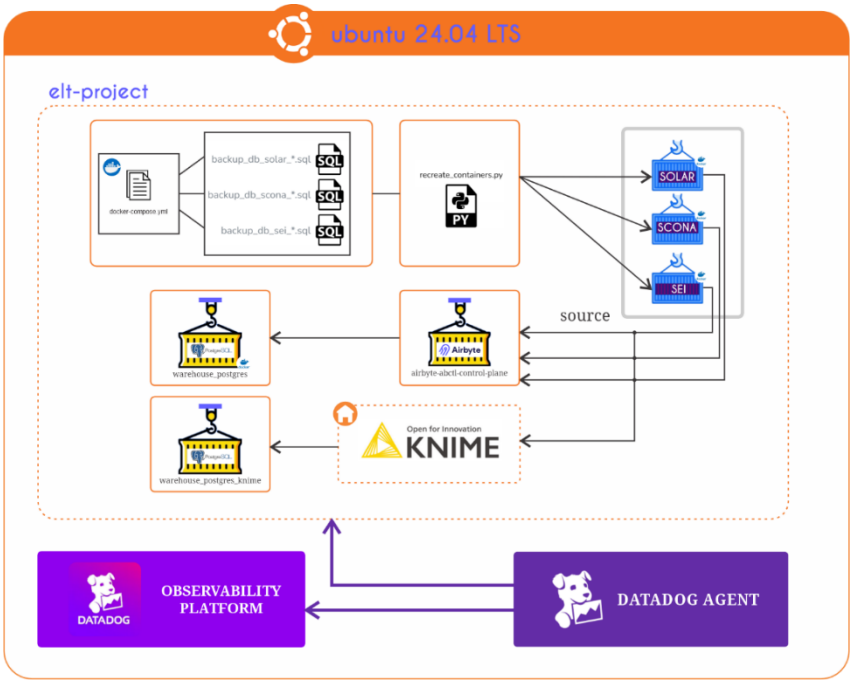
Fonte: Residentes de BI DPE/RN (2025).

Conforme ilustrado pela Figura 13, o ecossistema tecnológico implementado combina ferramentas amplamente reconhecidas no mercado — como Docker, PostgreSQL, Airbyte, Terraform, dbt, Elementary e GitLab — integradas em uma *pipeline* automatizada, observável e reprodutível. Essa combinação eliminou processos manuais, otimizou o tempo de preparação e validação de dados e fortaleceu a cultura de governança e confiabilidade da informação. Como resultado, a DPE/RN passou a dispor de uma infraestrutura de dados moderna, escalável e sustentável, que não apenas consolidou sua base analítica, mas também estabeleceu os fundamentos técnicos e operacionais para a adoção de soluções avançadas de *Business Intelligence, Analytics* e Inteligência Artificial no âmbito institucional.

4.1 Infraestrutura de testes e ambiente de execução

Para a realização dos experimentos e construção dos *pipelines* de dados com as ferramentas KNIME e Airbyte, foi desenvolvido um ambiente de testes controlado que reproduz, de forma minimizada, a infraestrutura de bancos de dados da Defensoria Pública do Estado do Rio Grande do Norte (DPE/RN).

Figura 14 – Arquitetura geral do ambiente de execução



Fonte: Elaborado pela autora (2025).

A Figura 14 apresenta a arquitetura geral do ambiente construído para a execução dos experimentos e *pipelines* de dados com o Airbyte e o KNIME. Por limitações de recursos computacionais e de acesso, a DPE/RN não disponibilizou uma máquina virtual para execução direta dos testes; dessa forma, foi necessário recriar o ambiente de forma controlada em uma estação de trabalho pessoal, levando em consideração a compatibilidade de sistemas e os recursos disponíveis. Em contrapartida, a máquina virtual (VM) da DPE/RN, utilizada como referência, opera em um servidor Ubuntu Server, com 32 GB de memória RAM e 500 GB de armazenamento, oferecendo um ambiente mais robusto e dedicado. A escolha desse ambiente buscou equilibrar realismo e viabilidade técnica, garantindo uma base suficientemente robusta para execução da pesquisa.

A partir do diretório elt-project, foi estruturado um arquivo docker-compose.yml responsável por definir os serviços dos bancos de dados que compõem as fontes de origem. Um *script* em Python (recreate_containers.py) foi desenvolvido para automatizar a recriação dos containers e a restauração dos *backups* SQL dos bancos SCONA, SOLAR e SEI, simulando o ambiente de produção da DPE/RN. Uma vez restaurados, esses bancos passam a funcionar como fontes de dados (*sources*) para os dois sistemas de extração. O Airbyte, executado em container, foi configurado para realizar as extrações e cargas no repositório warehouse_postgres, enquanto o KNIME Analytics Platform, instalado localmente — uma vez que não possui suporte gráfico para execução em container — realiza o mesmo processo de extração e carga em um segundo repositório, denominado warehouse_postgres_knime. Essa duplicidade foi planejada para permitir testes comparativos de desempenho e comportamento entre as duas ferramentas. Por fim, o ambiente conta com a integração do Datadog, que atua em duas camadas: o Datadog Agent, responsável pela coleta de métricas de desempenho tanto dos containers Docker quanto do processo local do KNIME, e a plataforma de observabilidade, onde as métricas são agregadas e visualizadas.

Importante destacar que, inicialmente, o ambiente foi configurado para trabalhar com os três bancos de origem (SOLAR, SCONA e SEI); contudo, devido às limitações de hardware da máquina local, os experimentos finais foram conduzidos

com apenas uma tabela de 11 GB proveniente do banco SOLAR, o que garantiu a execução estável dos testes e a coleta consistente das métricas de desempenho.

4.1.1 Desenvolvimento do ambiente

Para o desenvolvimento desta pesquisa, foram definidos e executados procedimentos que permitiram desde a preparação do ambiente até a coleta e análise das métricas de desempenho.

Inicialmente, foi configurado um ambiente controlado para a execução dos testes, utilizando uma máquina virtual (VM) com o sistema operacional Ubuntu Server, instalada em computador local. Essa escolha buscou reproduzir, de forma próxima ao real, o ambiente adotado pela DPE/RN, assegurando maior confiabilidade nos resultados. No entanto, como o ambiente oficial da DPE utiliza Ubuntu Server, que não dispõe de interface gráfica nativa, a execução direta do KNIME seria inviável. Nesse cenário, para viabilizar o uso em um servidor sem interface, seria necessário configurar soluções alternativas, como a instalação de um servidor X11 aliado ao uso de ferramentas de X11 *forwarding* via SSH, ou ainda a adoção de protocolos de desktop remoto (como VNC ou RDP). Essas abordagens permitiriam encaminhar a interface gráfica do KNIME para a máquina cliente, mas introduziriam maior complexidade na configuração, além de potenciais impactos de desempenho. Diante disso, a opção mais prática foi recriar o ambiente em Ubuntu 24.04 Desktop, que já oferece suporte gráfico nativo e possibilita o uso direto da ferramenta.

Na sequência, procedeu-se à instalação das ferramentas a serem avaliadas. O Airbyte foi instalado por meio do utilitário abctl, seguindo os critérios estabelecidos em sua documentação. Já o KNIME foi instalado diretamente na máquina anfitriã, uma vez que a tentativa de execução via contêiner apresentou limitações técnicas que inviabilizaram seu uso. A ferramenta de observabilidade foi configurada a partir da instalação do Datadog Agent na máquina local utilizada para a execução dos testes. Durante o processo, foram inseridas a API Key e a Application Key disponibilizadas pela plataforma, necessárias para autenticar o agente e estabelecer a comunicação com o ambiente de monitoramento em nuvem. Essa integração viabilizou a coleta

automática de métricas e a visualização em tempo real dos dados por meio dos *dashboards* interativos fornecidos pelo Datadog.

Com o ambiente devidamente configurado, iniciou-se a construção de *pipelines* equivalentes em ambas as ferramentas. A base de dados utilizada para os testes foi uma tabela específica do banco SOLAR, sistema central da DPE/RN que concentra a maior parte dos registros institucionais. Essa tabela possui aproximadamente 11 GB de dados e foi selecionada por sua relevância e volume significativo. A intenção inicial era utilizar a totalidade do banco, que atualmente ultrapassa 67 GB, mas devido às limitações de desempenho no ambiente local — que ocasionaram travamentos durante os testes — a carga de dados precisou ser reduzida. Ainda assim, a equivalência entre os *pipelines* foi mantida, assegurando condições comparáveis para a avaliação das ferramentas.

Após a construção dos *pipelines*, iniciou-se o monitoramento das ferramentas por meio da interface web do Datadog, acessada diretamente na plataforma. Foram observados o comportamento dos sistemas tanto em estado ocioso — quando não havia processos agendados ou em execução — quanto durante a sincronização dos dados. Essa etapa possibilitou observar o comportamento das ferramentas em contextos variados de operação, considerando suas particularidades técnicas e modos de execução.

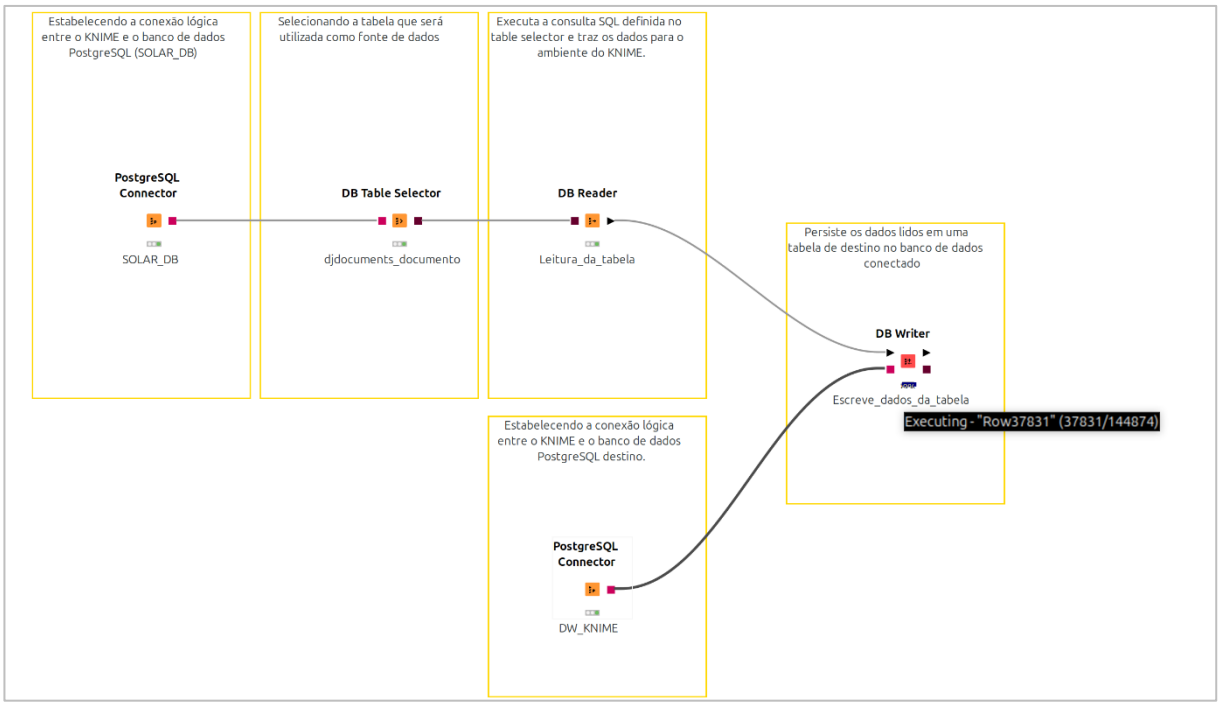
Os resultados obtidos foram sistematizados com o objetivo de viabilizar uma comparação entre as tecnologias analisadas. A partir dessa análise, buscou-se compreender as vantagens, limitações e comportamentos operacionais de cada solução, oferecendo subsídios concretos para orientar a escolha mais adequada a diferentes demandas. Além do valor prático para a instituição, o estudo também se propõe a enriquecer a produção acadêmica voltada à integração de dados em ambientes públicos por meio de ferramentas *low-code*.

4.2 KNIME: instalação, configuração e *pipeline*

O KNIME Analytics Platform foi instalado localmente no *host* Ubuntu 24.04 LTS, uma vez que a ferramenta não possui suporte gráfico adequado para execução

em containers. Optou-se pela versão estável mais recente compatível com sistemas Linux x86_64, garantindo estabilidade e compatibilidade com os demais componentes do ambiente. A instalação local possibilitou a execução e depuração dos *workflows* diretamente na interface gráfica, além de permitir o monitoramento do processo Java do KNIME pelo Datadog Agent.

Figura 15 – *Pipeline* desenvolvido no KNIME



Fonte: Elaborado pela autora (2025).

A Figura 15 ilustra o *pipeline* desenvolvido no KNIME Analytics Platform, responsável pela extração e carga dos dados provenientes do banco SOLAR para o repositório de destino DW_KNIME. Este fluxo foi resultado de diversas iterações e reconstruções ao longo do processo experimental, nas quais foram testadas diferentes combinações de nós e estratégias de leitura de múltiplas tabelas. Contudo, devido a limitações técnicas e de recursos computacionais enfrentadas durante a execução dos *workflows* mais complexos, optou-se por simplificar o *pipeline* de modo a garantir estabilidade e viabilizar a coleta precisa das métricas de desempenho — que constituem o foco central deste estudo.

O *workflow* é composto por cinco nós principais que são descritos conforme o Quadro 2 a seguir:

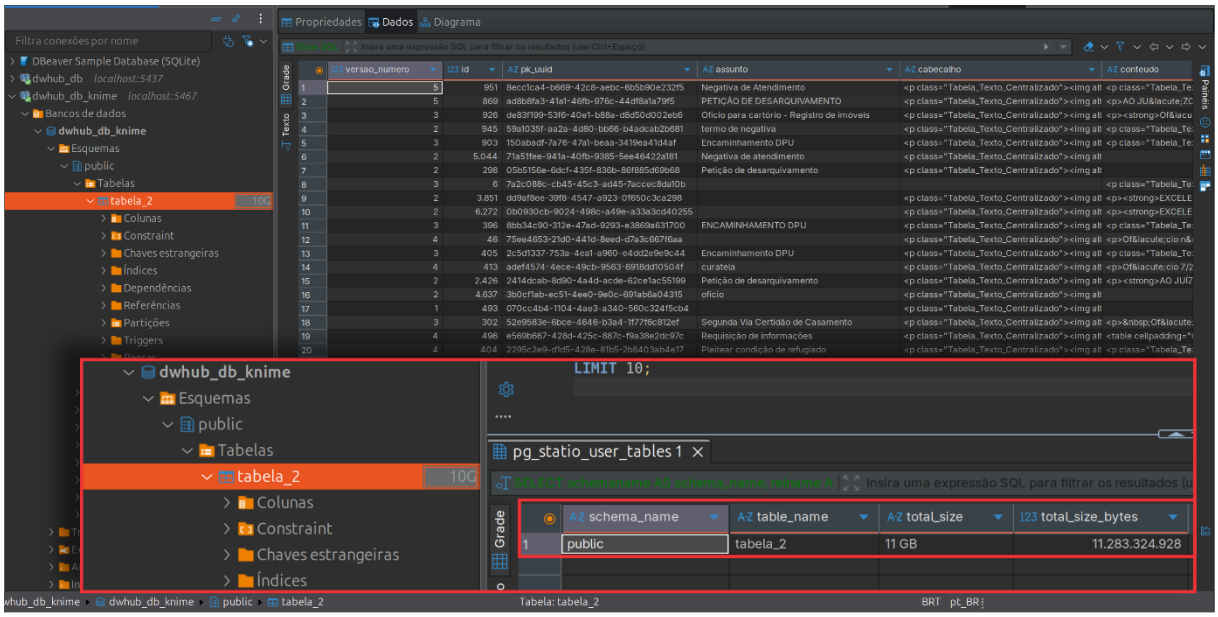
Quadro 2 – Nós utilizados no *workflow* do KNIME

1 PostgreSQL Connector	Estabelece a conexão entre o KNIME e o banco de dados PostgreSQL utilizando as credenciais e parâmetros definidos (host, porta, banco, usuário e senha) para criar um canal JDBC persistente, que será reutilizado pelos nós subsequentes do fluxo. (SOLAR_DB)
2 DB Table Selector	Responsável por selecionar a tabela que será utilizada como fonte de dados, operando sobre a conexão estabelecida pelo Connector e gera uma referência de consulta SQL interna, que aponta para o objeto escolhido no banco.
3 DB Reader	Responsável por executar uma consulta SQL no banco de dados conectado e trazer os resultados efetivos para dentro do ambiente de execução do KNIME. Ou seja, ele materializa os dados: transforma o resultado da query em uma tabela KNIME.
4 DB Writer	Este nó realiza a persistência dos dados lidos (ou transformados) em uma tabela de destino de um banco de dados conectado, consumindo da tabela produzida pela DB Reader e executando comando de inserção em lote no banco especificado.
5 PostgreSQL Connector	Estabelece a conexão entre o KNIME e o banco de dados PostgreSQL utilizando as credenciais e parâmetros definidos para ingestão no banco DW_KNIME.

Fonte: Capturado pela autora (2025).

Pela imagem – Figura 15 – é possível perceber o momento de execução do nó DB Writer no *workflow* do KNIME, etapa responsável por gravar os dados extraídos do banco SOLAR_DB no repositório de destino DW_KNIME. Durante o processo, o KNIME exibe em tempo real o andamento da escrita das linhas, como indicado pela mensagem "*Executing – Row 37831 (37831/144874)*", evidenciando a carga progressiva dos registros. O nome "tabela_2" foi atribuído manualmente à tabela de destino para diferenciá-la do ambiente utilizado pelo Airbyte, onde a mesma operação havia sido registrada sob o nome "tabela_1", antes da separação definitiva entre os dois *data warehouses*. Na Figura 16, visualizada no DBeaver, observa-se o resultado dessa operação: a tabela tabela_2 criada dentro do esquema *public* do banco dwhub_db_knime, contendo todos os registros replicados a partir da origem. Cada coluna e linha visível confirma a integridade e completude da carga, demonstrando que o fluxo de extração e escrita executado pelo KNIME foi concluído com sucesso. Essa etapa serviu como validação final do *pipeline* e como base para as coletas de métricas de desempenho que serão discutidas nos capítulos seguintes.

Figura 16 – Dados da tabela carregados no DW_KNIME



Fonte: Capturado pela autora (2025).

4.2.1 Observabilidade do KNIME via Datadog

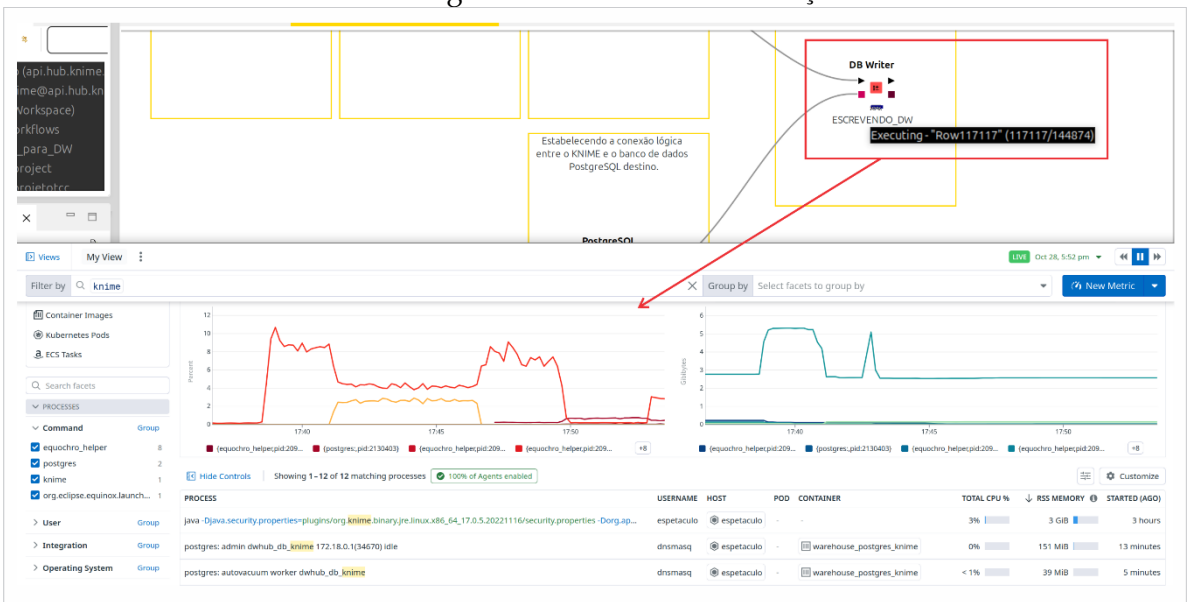
Para avaliar o comportamento do KNIME Analytics Platform durante a execução dos pipelines, foi realizado o monitoramento do processo Java responsável por sua execução. No Datadog, esse processo é identificado como o serviço org.eclipse.equinox.launcher_1.7.0.v20250519-0528. Esse nome corresponde ao *launcher* da JVM (Java Virtual Machine) que inicializa o ambiente gráfico e de execução do KNIME, uma vez que o KNIME é construído sobre a plataforma Eclipse Equinox OSGi — um contêiner modular de execução em Java. Assim, ao instalar o Datadog Agent no *host* Ubuntu e habilitar a integração JVM Metrics, todas as métricas coletadas referentes a esse serviço refletem diretamente o comportamento interno do KNIME (uso de *heap*, *garbage collector*, *threads*, classes carregadas, entre outros.) durante a execução dos fluxos de dados. Essa associação entre o processo monitorado e o ambiente interno do KNIME pode ser confirmada pelo arquivo de inicialização knime.ini, mostrado na Figura 17 abaixo. Esse arquivo, localizado no diretório de instalação (/Applications/knime_5.7.0), define os parâmetros de inicialização da plataforma e evidencia o uso do *launcher* já mencionado, que corresponde exatamente ao serviço identificado pelo Datadog.

Figura 17 – Arquivo knime.ini evidenciando o *launcher* org.eclipse.equinox responsável pela inicialização do KNIME Analytics Platform

Fonte: Capturado pela autora (2025).

A Figura 18 apresenta o monitoramento em tempo real do processo do KNIME Analytics Platform durante a execução do fluxo de extração de dados do banco SOLAR_DB para o DW_KNIME, conforme o *workflow* exibido parcialmente na parte superior da imagem (nó DB Writer em execução).

Figura 18 – KNIME em execução



Fonte: Capturado pela autora (2025).

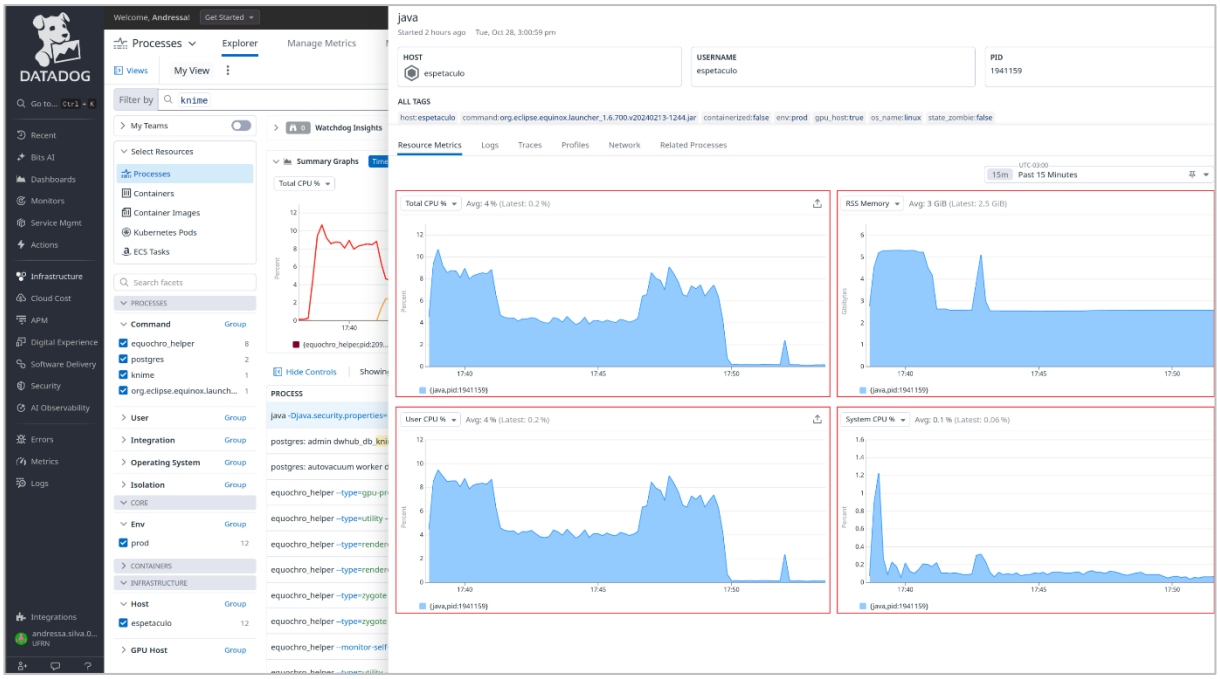
É importante destacar que o processo monitorado pelo Datadog, identificado como org.eclipse.equinox.launcher, pode aparecer com números de versão distintos (por exemplo, 1.6.x ou 1.7.x) em diferentes momentos ou capturas de tela. Essa variação ocorre porque o KNIME Analytics Platform é executado sobre o Eclipse

Equinox, um framework modular em Java que atualiza periodicamente seus componentes internos de inicialização (*launchers*). Assim, embora o identificador de versão varie conforme a atualização ou reinicialização do ambiente, todos os registros se referem ao mesmo serviço de execução da JVM responsável por iniciar e gerenciar o KNIME, sendo plenamente equivalentes para fins de análise das métricas e comportamento da aplicação. No painel inferior da imagem, o Datadog registra as métricas de CPU (à esquerda) e memória RAM (RSS Memory) (à direita) referentes aos principais processos ativos no *host* Ubuntu. Entre eles, destacam-se o processo que representa a execução principal do KNIME, e os processos do PostgreSQL responsáveis pela escrita dos dados no *data warehouse*. Os gráficos (Figura 18) mostram dois comportamentos distintos e complementares:

Uso de CPU (gráfico à esquerda): É possível observar picos contínuos de processamento, com valores variando entre 8% e 12% de utilização, exatamente no período em que o nó DB Writer estava ativo. Esse padrão indica que o KNIME estava realizando operações intensivas de leitura e transformação em memória, enquanto o PostgreSQL processava as inserções no banco de destino. O leve decréscimo no final do gráfico coincide com o encerramento da fase de escrita.

Uso de memória (gráfico à direita): O comportamento da memória apresenta um crescimento rápido e estabilização, atingindo cerca de 3 GB de uso pelo processo do KNIME. Essa curva indica que o KNIME aloca a memória necessária para carregar os dados da origem e mantê-los em buffer até a gravação final no destino. Pequenas quedas pontuais representam a atuação do Garbage Collector (GC), que libera blocos de memória já processados. Abaixo dos gráficos, a tabela de processos confirma que o KNIME manteve consumo predominante de CPU e RAM, enquanto os serviços PostgreSQL (warehouse_postgres_knime) permaneceram com carga mínima, refletindo que o esforço computacional maior está do lado do cliente (KNIME), e não do servidor de banco de dados.

Figura 19 – Variação de CPU e memória do KNIME durante a extração e escrita no DW



Fonte: Capturado pela autora (2025).

A Figura 19 ilustra com um pouco mais de detalhes os gráficos de Uso de CPU e Memória durante a execução do KNIME, também podendo ser percebido o comportamento após o período de escrita no banco de dados.

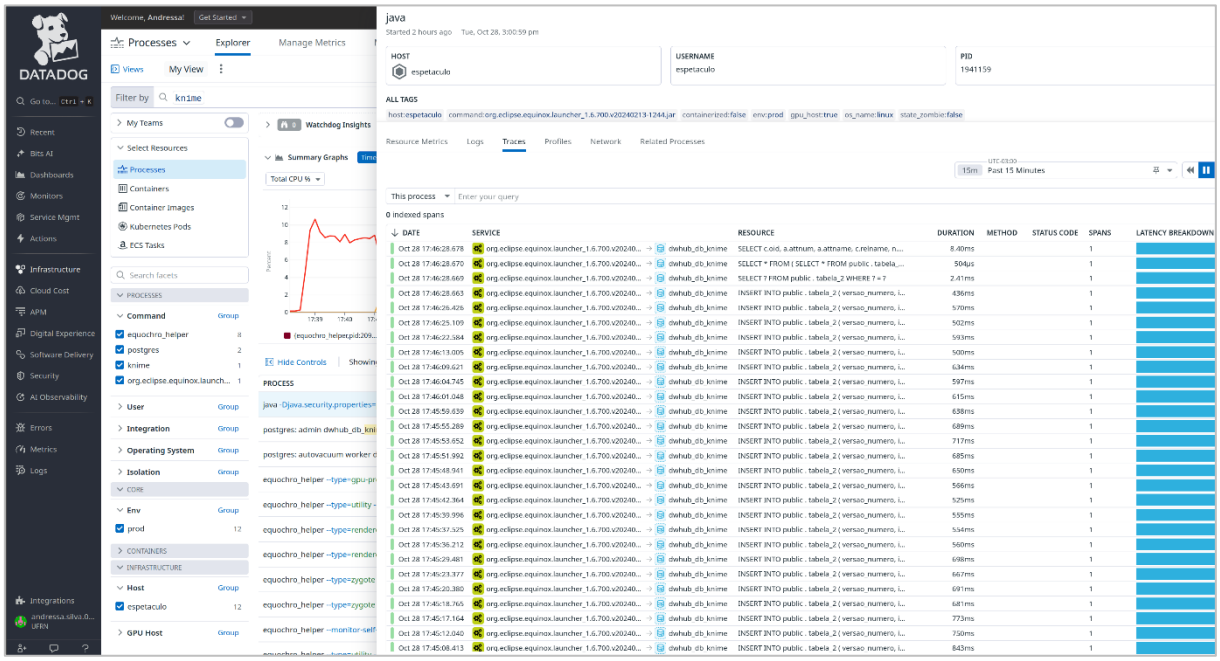
Uso de CPU (Total CPU % e User CPU %): Nos dois primeiros gráficos (superior e inferior à esquerda), o Datadog apresenta a utilização total da CPU pelo processo KNIME, bem como a fração correspondente ao tempo de processamento em modo usuário (User CPU) — isto é, o tempo efetivamente empregado pela JVM na execução das instruções do KNIME, excluindo atividades de sistema e espera por I/O. Entre 17:38 e 17:50, observa-se um comportamento caracterizado por picos sustentados entre 6% e 10%, típico de uma aplicação em fase de carga de dados — momento em que o KNIME executa operações intensivas de leitura e escrita sobre o banco de destino.

Esses picos coincidem precisamente com a ativação do nó DB Writer no *workflow*, confirmando que a CPU estava sendo utilizada de forma contínua para processar e transferir registros.

A inflexão abrupta no final da série temporal (por volta de 17:50) marca o encerramento completo do pipeline, quando a utilização da CPU retorna a níveis

residuais, refletindo o estado ocioso do processo Java. Em termos figurativos, o gráfico expressa o "ritmo de trabalho" do KNIME: um início vigoroso, um período de atividade constante e, finalmente, o repouso após a conclusão da tarefa.

Figura 20 – Execução de comandos SQL do KNIME exibida no Datadog



Fonte: Capturado pela autora (2025).

A Figura 20 ilustra a parte interna das operações realizadas pelos nós executados no KNIME. No painel central direito, a aba "Traces" está selecionada — ela apresenta os rastros (traces) das operações SQL realizadas durante o fluxo de execução, enquanto o processo Java estava ativo. Cada linha da tabela representa uma transação individual registrada pelo Datadog APM (Application Performance Monitoring), mostrando a interação do KNIME com o banco de dados dwhub_db_knime. Os registros indicam claramente comandos SQL sendo executados, como: **(i)** SELECT * FROM public.tabela_2 e **(ii)** INSERT INTO public.tabela_2 (...), o que reflete as operações realizadas pelos nós DB Reader e DB Writer dentro do *workflow* KNIME, responsáveis pela leitura e escrita de dados entre o banco de origem (SOLAR_DB) e o destino (DW_KNIME). À esquerda, o gráfico "Total CPU %" mostra um pico de utilização de CPU no mesmo intervalo (17:45–17:50), o que confirma a

correlação entre os períodos de execução SQL e o uso intensivo de recursos pelo KNIME. O Quadro 3 a seguir descreve cada coluna exibida.

Quadro 3 – Interpretação dos principais campos exibidos na aba *Traces* do Datadog

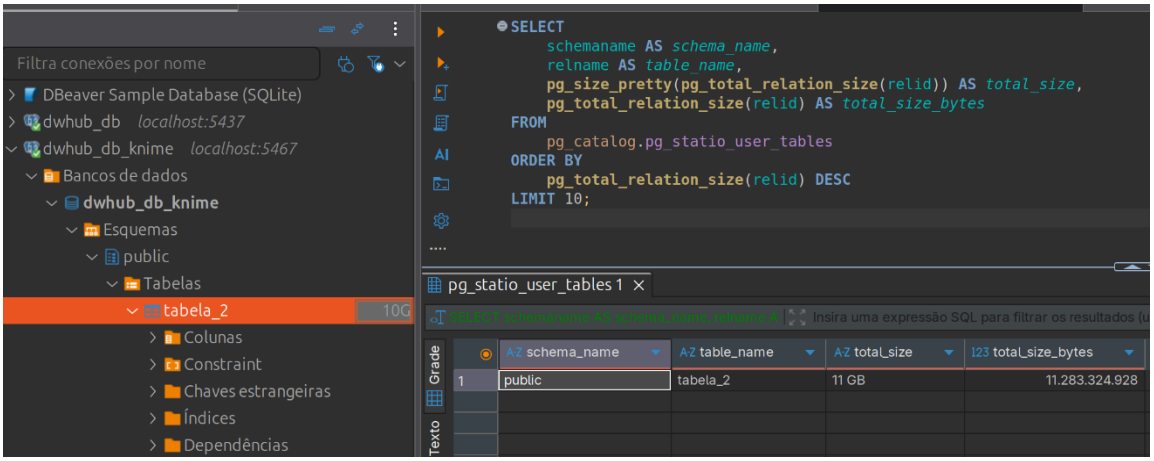
Campo	**Significado técnico**	**Interpretação no contexto do KNIME**
DATE	Data e hora em que o trace foi capturado.	Mostra o instante exato de cada comando SQL executado pelo *workflow*.
SERVICE	Nome do serviço que originou a operação monitorada.	Aqui, org.eclipse.equinox.launcher identifica o serviço Java que executa o KNIME.
RESOURCE	Recurso ou operação específica rastreada — normalmente o comando SQL ou endpoint acessado.	Exibe instruções SQL executadas (SELECT, INSERT, UPDATE). Permite identificar quais tabelas e ações o KNIME estava realizando.
DURATION	Tempo total de execução da operação (latência total).	Mede quanto tempo o KNIME levou para concluir cada comando SQL. Valores típicos de alguns milissegundos (400–800 ms) indicam boa performance de escrita/leitura.
METHOD	Tipo de operação ou verbo associado ao trace.	No caso de queries SQL, representa a execução direta de comandos via driver JDBC.
STATUS CODE	Código de status retornado pela operação.	Um valor "1" indica sucesso (sem erro). Valores maiores poderiam representar falhas ou exceções.
SPANS	Quantidade de spans (suboperações) dentro daquele trace.	Cada span é uma etapa dentro do trace completo — aqui, "1" significa que cada comando SQL foi registrado como uma única operação atômica.
LATENCY BREAKDOWN	Gráfico de barras que mostra a decomposição da latência.	Permite visualizar graficamente quanto do tempo total foi gasto em execução, espera, rede, entre outros. Nesse caso, todas as barras estão uniformes, indicando latências curtas e consistentes.

Fonte: Elaborado pela autora (2025).

O Quadro 3, portanto, relaciona os campos técnicos do Datadog à execução do KNIME, mostrando como métricas como tempo, operação e recurso acessado traduzem o comportamento do *workflow* ao processar os comandos SQL. A figura a seguir exibe o resultado da extração visualizado no DBeaver, evidenciando a

conclusão bem-sucedida do processo. O tempo total de execução do *workflow* no KNIME foi de cerca de 5 minutos, considerando as etapas de leitura e escrita dos dados.

Figura 21 – Visualização da tabela resultante no DBeaver



Fonte: Capturado pela autora (2025).

4.2.2 Conclusões a respeito da utilização do KNIME

A análise do comportamento do KNIME Analytics Platform, associada às métricas coletadas via Datadog e às observações empíricas durante os testes, permitiu identificar uma série de características técnicas e operacionais relevantes para compreender o posicionamento e o potencial dessa ferramenta no contexto de *pipelines* de integração de dados.

Em primeiro lugar, constatou-se que o KNIME apresenta uma arquitetura robusta e de ampla aplicabilidade, extrapolando o escopo de uma ferramenta puramente ETL/ELT. Sua natureza modular e a vasta biblioteca de nós disponíveis o tornam apto a desempenhar funções que vão desde a simples replicação de dados até análises avançadas, modelagem preditiva e automação de processos analíticos. Em contrapartida, percebe-se que a proposta do KNIME difere substancialmente da do Airbyte: enquanto o Airbyte adota uma arquitetura voltada exclusivamente à integração e sincronização de dados, com foco em escalabilidade e automação via contêineres, o KNIME se posiciona como uma plataforma analítica de propósito geral,

priorizando a flexibilidade, a interação visual e o controle granular das etapas de processamento.

A diferença de arquitetura das duas ferramentas reflete também o público-alvo e o nível de abstração de cada uma — o Airbyte busca simplificar e automatizar, enquanto o KNIME oferece maior poder de customização e análise exploratória. Vale destacar que, apesar de sua interface intuitiva, o KNIME apresenta uma curva de aprendizado mais acentuada em comparação ao Airbyte, principalmente por exigir maior familiaridade com sua extensa biblioteca de nós e constante consulta à documentação.

Na plataforma KNIME cada nó funciona como um componente modular, responsável por uma etapa específica do processo — leitura, transformação ou escrita — que deve ser combinado a outros para compor o fluxo completo. Por exemplo, a leitura e escrita de dados demandam nós distintos, como DB Reader e DB Writer, configurados de forma encadeada para que a operação se conclua. Essa lógica "dividir para conquistar" confere ao KNIME grande flexibilidade e transparência sobre o processo, mas também torna sua configuração mais detalhada. Em contraste, o Airbyte abstrai essa complexidade, permitindo realizar toda a sincronização de dados a partir de uma única conexão entre as fontes e os destinos.

Durante os testes, observou-se que o comportamento do KNIME varia entre ambientes Windows e Linux, o que trouxe desafios para a análise de desempenho. No Linux, foram notadas diferenças sutis na interface e na responsividade — como a ausência de descrições detalhadas em alguns nós e uma performance inferior nas ações de *drag and drop*. Essas divergências não comprometem o funcionamento da plataforma, mas reforçam que a experiência do usuário e a performance da interface ainda são aspectos dependentes do sistema operacional hospedeiro.

Em termos de recursos, o KNIME demonstrou eficiência e estabilidade muito superiores, mesmo em cargas moderadas de CPU e memória. As métricas revelaram um consumo previsível e controlado, sem picos anômalos de utilização, o que evidencia um bom gerenciamento interno da JVM. No entanto, diferentemente do Airbyte, o KNIME não opera em ambiente containerizado, sendo executado diretamente sobre a máquina hospedeira, o que limita sua escalabilidade.

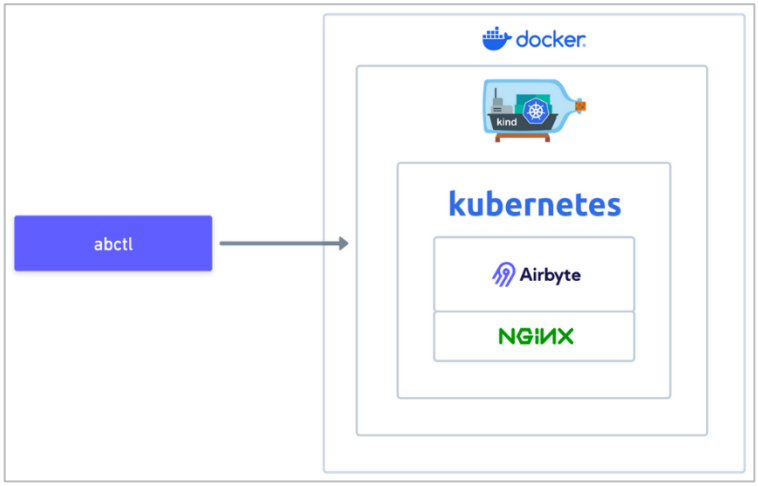
Sob uma perspectiva comparativa, enquanto o Airbyte se posiciona como uma solução voltada exclusivamente para integração e sincronização de dados (ETL/ELT), com forte aderência a arquiteturas modernas, o KNIME adota uma abordagem mais ampla, orientada à análise e ciência de dados. O Airbyte privilegia automação, escalabilidade e integração contínua, mas depende fortemente de recursos computacionais e infraestrutura robusta.

Em síntese, os resultados obtidos reforçam que o KNIME é uma ferramenta tecnicamente sólida, de alta capacidade exploratória e excelente custo-benefício, aplicável não apenas a cenários de pequeno e médio porte, mas também a contextos mais amplos — desde que alinhados à arquitetura de dados que se deseja implementar. Sua flexibilidade e riqueza de recursos o tornam adaptável a diferentes necessidades, embora sua curva de aprendizado mais elevada e a ausência de suporte nativo a contêineres o distanciem de soluções voltadas à escalabilidade corporativa, como o Airbyte.

4.3 Airbyte: instalação, configuração e *pipeline*

A utilização do Airbyte deu-se por meio de sua instalação utilizando o utilitário abctl, conforme orienta a documentação oficial. Esse utilitário automatiza todo o processo de implantação da plataforma, criando um ambiente isolado de execução baseado em Kubernetes. Especificamente, o abctl utiliza o Kind (Kubernetes in Docker) para gerar um *cluster* Kubernetes dentro de um contêiner Docker, sobre o qual são instalados, via Helm, os componentes essenciais do Airbyte e o NGINX Ingress Controller, conforme ilustrado na Figura 22 a seguir, retirada da documentação oficial da tecnologia (Airbyte, 2025).

Figura 22 – Estrutura de implantação do Airbyte via abctl



Fonte: Documentação do Airbyte (2025).

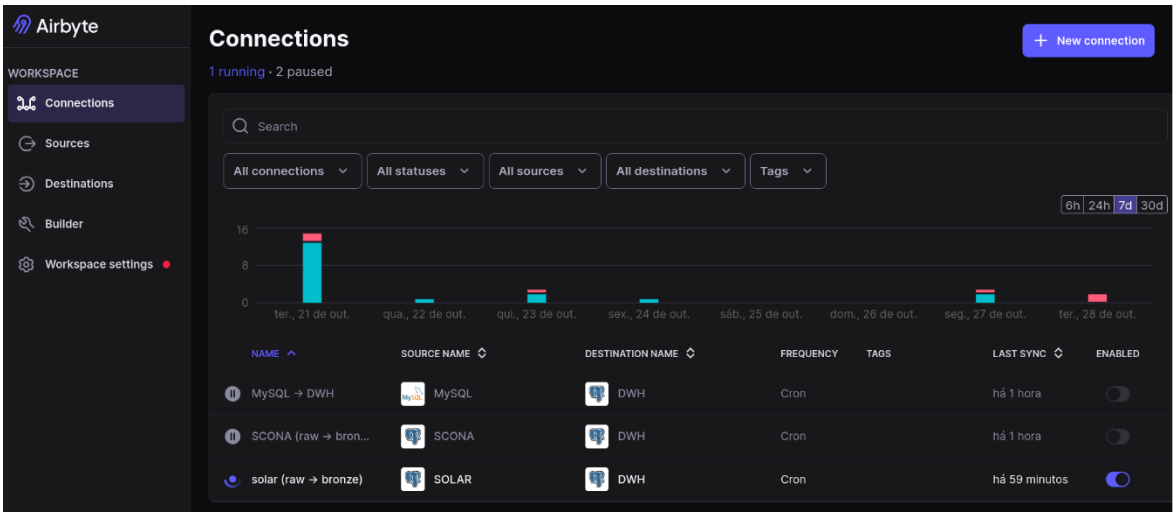
Importante mencionar que, em versões anteriores, o Airbyte era tradicionalmente instalado via Docker Compose, mas esse método foi descontinuado nas versões mais recentes. Atualmente, a instalação via abctl é o único meio oficialmente suportado, refletindo a transição do Airbyte para uma arquitetura mais robusta, modular e orientada a microserviços, alinhada a práticas de orquestração modernas.

O processo de instalação evidenciou também a alta demanda por recursos computacionais exigida pelo Airbyte. Isso pôde ser comprovado empiricamente: em uma estação pessoal equipada com Intel® Core™ i7 (11ª geração), 16 GB de RAM e 1 TB de armazenamento, a instalação completa — incluindo o download das imagens, criação do cluster e inicialização dos serviços — levou aproximadamente 10 horas para ser concluída. Em contraste, em uma estação de trabalho com Intel® Core™ 7 240H (10 núcleos, cache de 24 MB, até 5.2 GHz), o mesmo processo foi concluído em cerca de 10 minutos, demonstrando que o Airbyte depende fortemente de CPU moderna, múltiplos núcleos e memória abundante para inicialização eficiente. Esse comportamento reforça o posicionamento do Airbyte como uma plataforma projetada para ambientes de alto desempenho, idealmente executada em infraestrutura corporativa, servidores dedicados ou máquinas virtuais com recursos superiores.

Conforme já mencionado, foram configurados como *source connectors* três bancos de dados distintos: o SEI (MySQL) e os bancos SOLAR e SCONA (ambos em

PostgreSQL), todos executados em contêineres. A Figura 23 a seguir ilustra as conexões de origem no Airbyte UI:

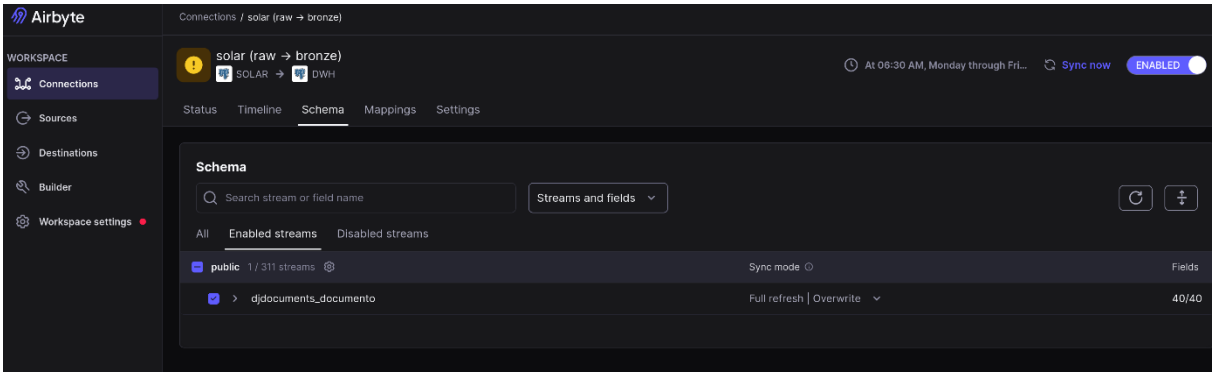
Figura 23 – Bancos de origem conectados no Airbyte



Fonte: Capturado pela autora (2025).

Entretanto, durante os experimentos, o Airbyte apresentou instabilidades e travamentos recorrentes no processo de sincronização envolvendo múltiplas fontes. Diante disso, optou-se por concentrar os testes apenas no banco SOLAR, selecionando uma tabela de tamanho significativo para realizar a sincronização e, assim, possibilitar a coleta de métricas e a análise do comportamento da ferramenta. A figura a seguir ilustra a tabela selecionada para o processo de sincronização, cujo volume corresponde a aproximadamente 11 GB de dados. A operação foi configurada no modo de sincronização Full Refresh | Overwrite, que consiste na reescrita completa dos dados no destino a cada execução, substituindo integralmente o conteúdo previamente existente.

Figura 24 – Tabela selecionada para sincronização no Airbyte

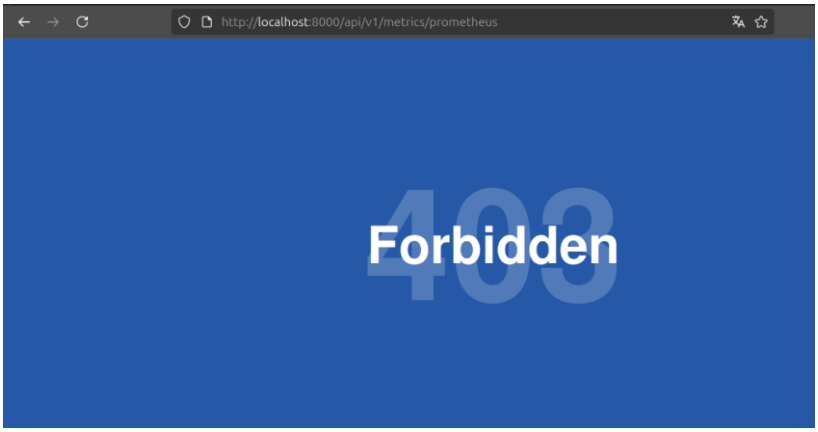


Fonte: Capturado pela autora (2025).

4.3.1 Observabilidade do Airbyte via Datadog

Em termos de observabilidade, o Airbyte mostrou-se uma ferramenta bastante restritiva. Inicialmente, foi configurado um ambiente de monitoramento composto pelos contêineres Prometheus e Grafana, com o objetivo de coletar e visualizar as métricas de desempenho do Airbyte — seguindo a mesma abordagem adotada para o monitoramento do KNIME. No entanto, mesmo com todo o ambiente devidamente configurado, verificou-se que o Airbyte não disponibiliza mais sua API de métricas nativa para acesso externo. Ao tentar consultar o *endpoint* padrão (/api/v1/metrics/prometheus), a interface retornou o erro 403 – Forbidden, como ilustrado na Figura 25, indicando a restrição de acesso a esse recurso.

Figura 25 – API de métricas do Airbyte mostrando acesso proibido



Fonte: Capturado pela autora (2025).

Conforme documentação oficial, o Airbyte passou a oferecer suporte à exportação de métricas apenas por meio do OpenTelemetry, no entanto, disponível exclusivamente para a versão Enterprise da plataforma — uma edição voltada a ambientes corporativos e de alto custo. Durante a fase de monitoramento, foram realizadas diversas tentativas de coleta de métricas do Airbyte por meio do Prometheus, utilizando diferentes abordagens de configuração e integração. Buscou-se inicialmente estabelecer a extração de dados de telemetria diretamente do *endpoint* interno da aplicação, mas sem sucesso, uma vez que o Airbyte, em suas versões mais recentes, não expõe mais métricas compatíveis com Prometheus.

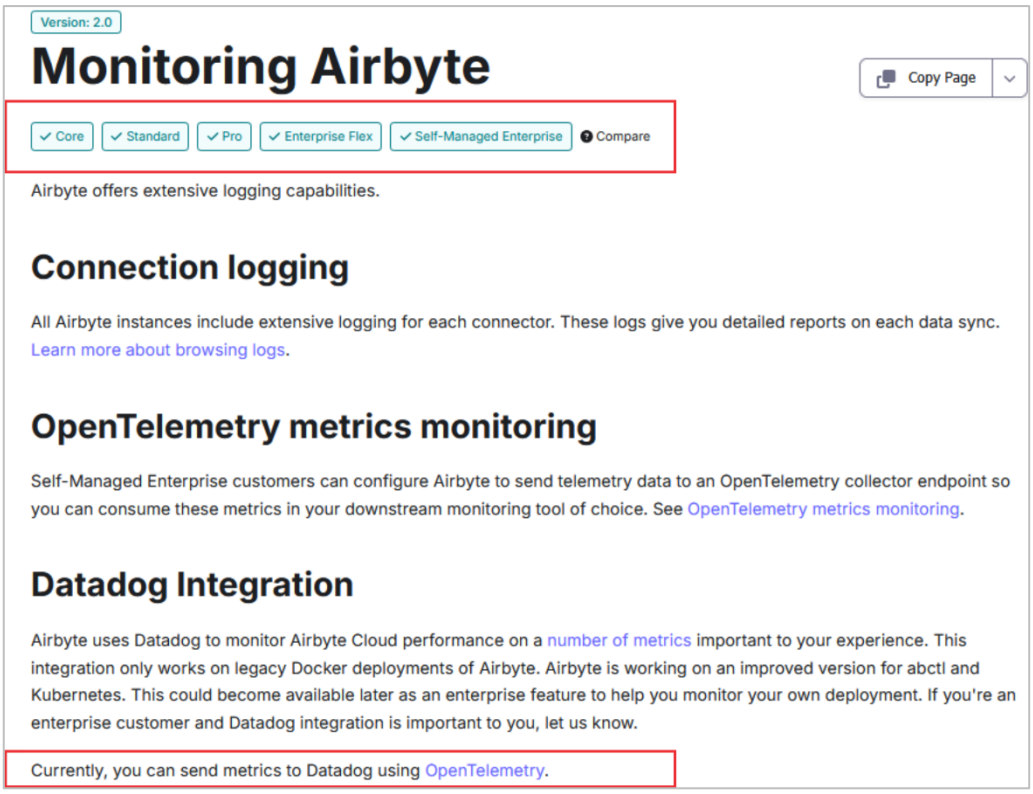
Diante dessa limitação, foi testada também a integração com o OpenTelemetry, seguindo as instruções disponíveis na documentação oficial. Para viabilizar o experimento, procedeu-se inclusive a um *downgrade* da ferramenta, reinstalando uma versão anterior do Airbyte obtida por meio do repositório oficial no GitHub, ainda distribuída via Docker Compose. Apesar das múltiplas tentativas de configuração — incluindo ajustes de portas, variáveis de ambiente e permissões de acesso —, a coleta de métricas permaneceu inviável, resultando em falhas recorrentes de autenticação e ausência de endpoints acessíveis.

Devido aos vários problemas enfrentados, mesmo após várias tentativas exaustivas e sem êxito, optou-se por interromper a instrumentação direta do Airbyte com ferramentas externas de observabilidade e prosseguir com a pesquisa por outros meios de análise, concentrando os esforços na coleta de métricas de infraestrutura (CPU, memória e rede) por meio do Datadog Agent instalado diretamente no *host*. Para isso, utilizou-se a versão de avaliação gratuita de 14 dias disponibilizada pela ferramenta, que se mostrou o único meio viável para dar continuidade à pesquisa e assegurar o monitoramento do ambiente de execução do Airbyte.

É relevante mencionar que apesar de todos os esforços para instrumentar o Airbyte com ferramentas de observabilidade externas, a integração com o Datadog revelou-se inviável nas versões não comerciais da plataforma. Inicialmente, a documentação oficial do Airbyte e do próprio Datadog indicava a possibilidade de uma integração direta, prometendo um *dashboard* completo e interativo capaz de exibir, em tempo real, métricas detalhadas sobre o desempenho das sincronizações —

algo que, à primeira vista, causava entusiasmo pela riqueza visual e pelo potencial analítico apresentado.

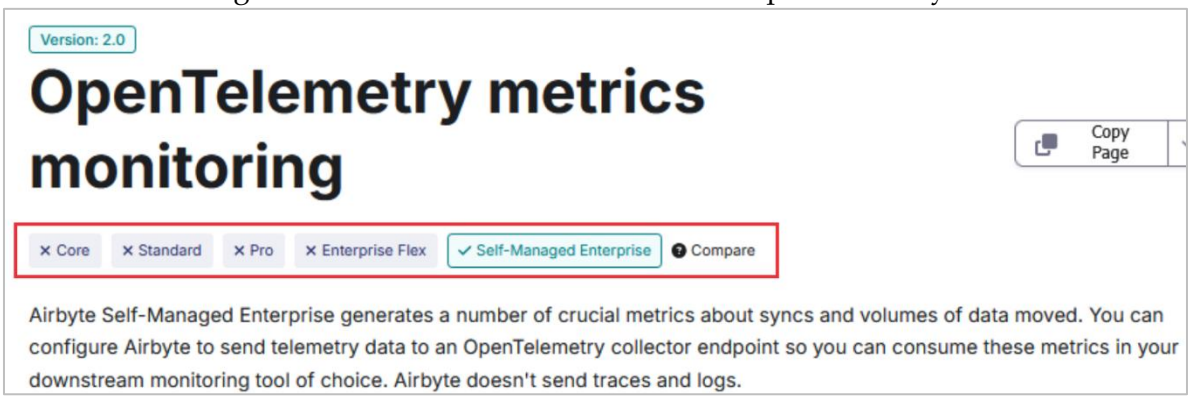
Figura 26 – Parte da documentação exibindo como monitorar o Airbyte



Fonte: Capturado pela autora (2025).

No entanto, durante a execução prática, verificou-se que essa integração está restrita exclusivamente à versão Enterprise Self-Managed do Airbyte, conforme destacado na documentação oficial.

Figura 27 – Como coletar métricas com o OpenTelemetry

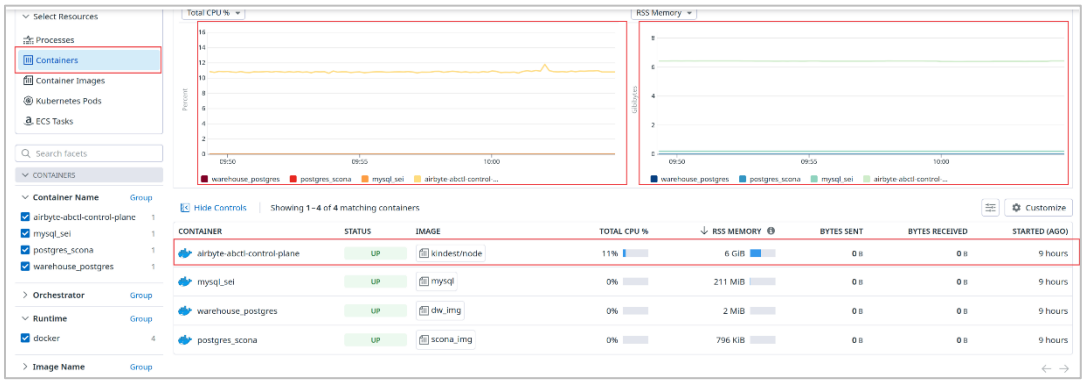


Fonte: Capturado pela autora (2025).

Ainda assim, em uma tentativa de validar o processo, foi realizada uma configuração completa do Datadog Agent, incluindo a edição do arquivo datadog.yaml e a habilitação do módulo OpenTelemetry para recepção de dados do Airbyte. Todo o procedimento seguiu rigorosamente o passo a passo da documentação oficial das duas ferramentas, com a expectativa de que o Datadog capturasse e exibisse as métricas geradas pela aplicação. Contudo, mesmo após a integração ser concluída com sucesso no nível de configuração, o *dashboard* permaneceu vazio, sem apresentar qualquer métrica proveniente do Airbyte. Essa constatação confirmou que, nas versões atuais distribuídas via abctl, o Airbyte não expõe dados de telemetria ou desempenho fora de sua camada interna de execução.

Nessa perspectiva, a única alternativa viável encontrada foi monitorar o contêiner de controle principal (airbyte-abctl-control-plane) por meio do Datadog, capturando métricas indiretas de uso de CPU, memória e rede — o que permitiu, ainda que de forma limitada, acompanhar o comportamento da ferramenta durante as sincronizações e prosseguir com a pesquisa de forma consistente.

Figura 28 – Contêiner airbyte-abctl monitorado via Datadog



Fonte: Capturado pela autora (2025).

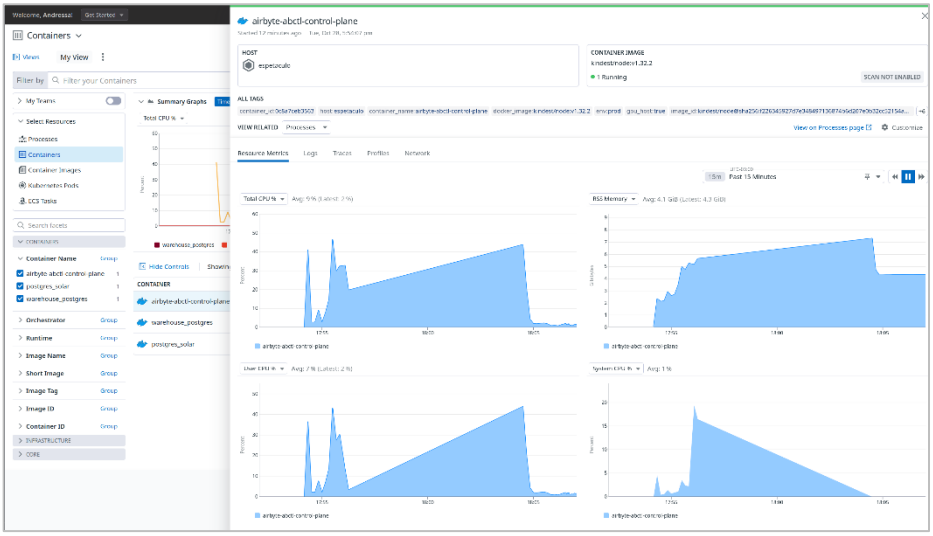
A imagem apresenta o monitoramento em tempo real dos contêineres do Airbyte e dos bancos de dados associados (MySQL_SEI, Postgres_SCONA e Warehouse_Postgres), capturado pelo Datadog Agent. No painel superior, são exibidos os gráficos de utilização de CPU (Total CPU %) e memória física (RSS Memory), enquanto a tabela inferior detalha o status e os recursos consumidos por

cada contêiner em execução. Mesmo em estado ocioso — isto é, sem nenhuma sincronização de dados em andamento —, observa-se que o contêiner airbyte-abctl-control-plane apresenta uso contínuo de CPU em torno de 11% e ocupação de memória de aproximadamente 6 GB.

O comportamento do Airbyte é característico em ambientes orquestrados via abctl, uma vez que o contêiner Kindest/Node, responsável por manter o cluster Kubernetes interno, permanece ativo mesmo sem tarefas em execução, garantindo a disponibilidade dos serviços essenciais. Os demais contêineres — correspondentes aos bancos de dados de origem e destino — exibem consumo mínimo de recursos, com uso de memória inferior a 250 MB e CPU praticamente nula, o que confirma a ausência de processos de leitura, escrita ou sincronização ativa.

Esse resultado evidencia que o Airbyte mantém uma sobrecarga de base mesmo em repouso, reflexo de sua arquitetura distribuída e dependência de múltiplos processos internos, o que reforça a necessidade de infraestrutura mais robusta, sobretudo quando comparado a ferramentas locais, como o KNIME.

Figura 29 – Uso de CPU e Memória pelo contêiner do Airbyte em estado ocioso

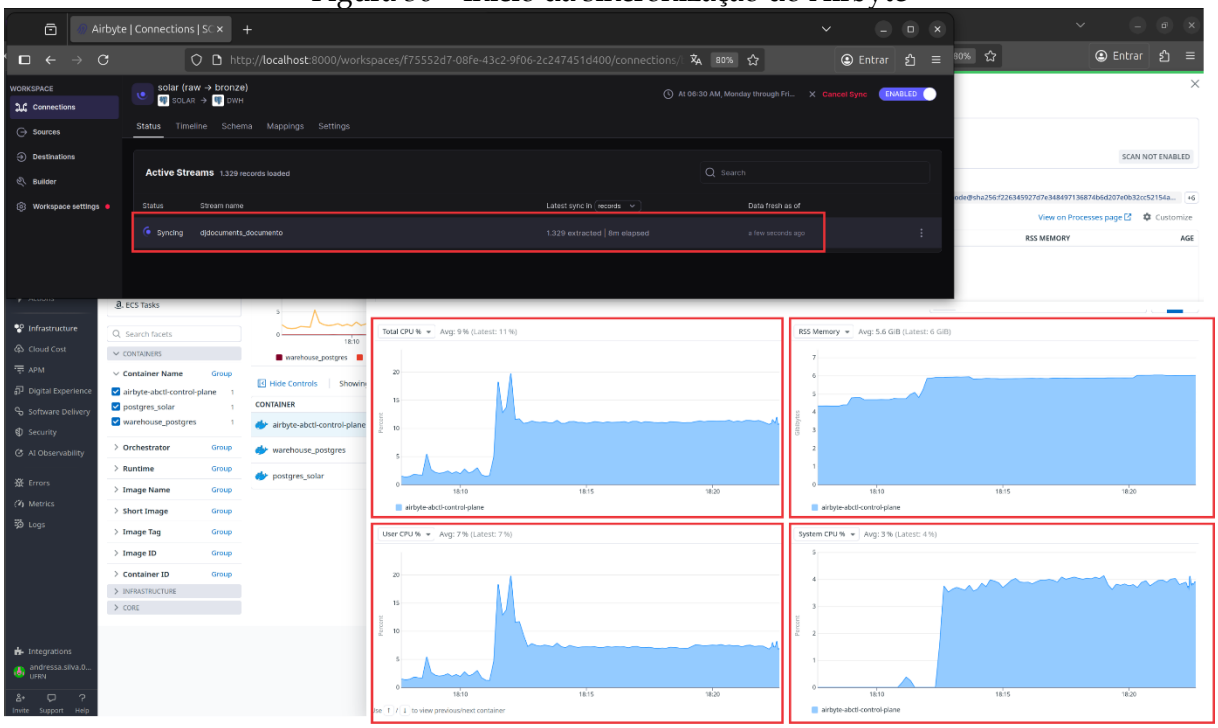


Fonte: Capturado pela autora (2025).

As métricas detalhadas na sequência – Figura 29 – complementam essa observação, abrangendo os principais indicadores de desempenho — uso de CPU total, CPU de usuário, CPU do sistema (kernel) e memória física (RSS Memory) — ao longo de uma janela de 15 minutos. Nota-se que o contêiner apresenta picos iniciais de

utilização de CPU próximos a 50%, seguidos por estabilização em torno de 9% (User CPU ~7%). Esses picos correspondem ao período de inicialização dos componentes internos do cluster Kubernetes, enquanto a posterior estabilização reflete o consumo constante necessário para manter os serviços de controle em execução. Esse padrão demonstra que o Airbyte mantém uma sobrecarga de base mesmo em repouso, resultado direto de sua arquitetura distribuída e dependência de múltiplos processos internos, o que reforça a necessidade de infraestrutura computacional mais robusta — especialmente quando comparado a ferramentas locais, como o KNIME, cujo consumo de recursos é mais diretamente vinculado à execução efetiva de fluxos de dados.

Figura 30 – Início da sincronização do Airbyte



Fonte: Capturado pela autora (2025).

A Figura 30 mostra o momento em que o Airbyte iniciou efetivamente a sincronização dos dados entre o banco de origem SOLAR e o destino DWH, operação realizada via interface web da ferramenta. Na parte superior da tela, o painel do Airbyte indica a conexão ativa (Status: Syncing), com a extração de 1.329 registros do fluxo configurado. Em paralelo, o Datadog Agent registrou o comportamento do contêiner airbyte-abctl-control-plane durante essa execução, exibindo métricas em

tempo real de CPU (Total, User e System) e memória física (RSS Memory). Observa-se, nos gráficos à direita, um aumento abrupto na utilização de CPU logo após o início da sincronização, atingindo picos próximos a 20% de uso total e estabilizando-se em torno de 9%, valor superior ao verificado durante o estado ocioso. O gráfico de User CPU % acompanha o mesmo padrão, indicando que a maior parte do processamento é executada no nível da aplicação, enquanto o System CPU % mantém média em torno de 3%, refletindo a atividade do kernel para operações de E/S e gerenciamento de processos.

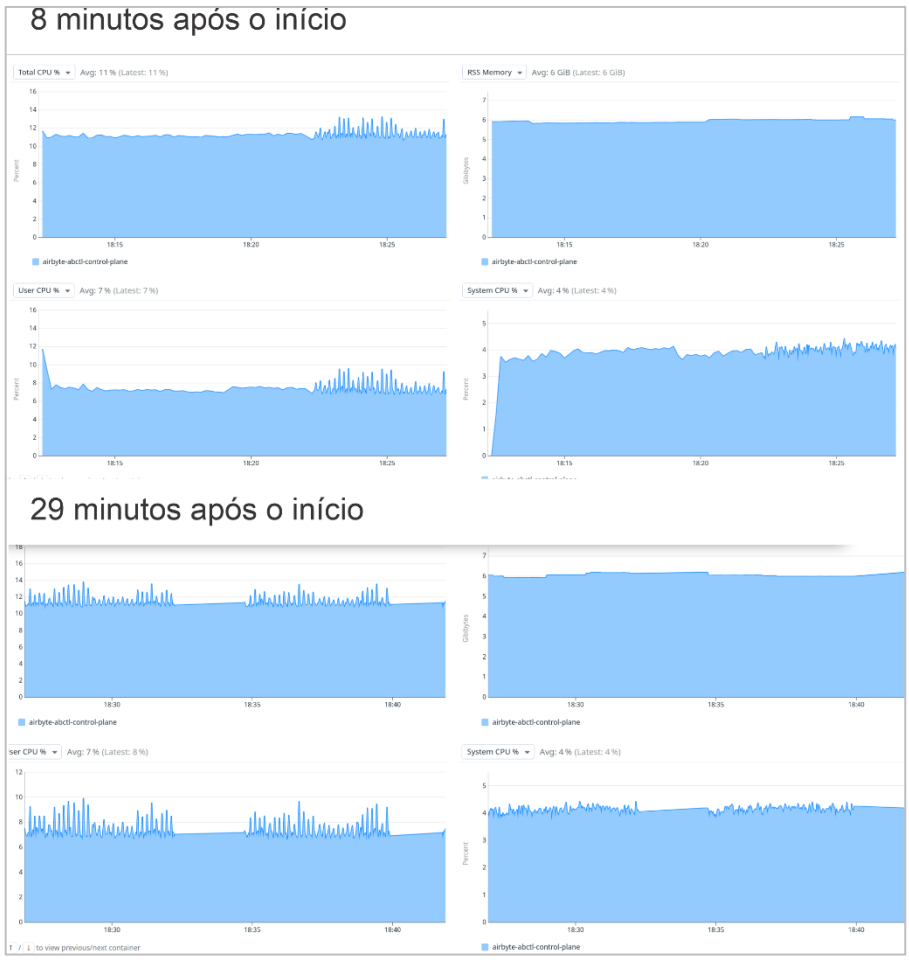
No gráfico de RSS Memory, percebe-se uma elevação progressiva no consumo de memória, partindo de aproximadamente 3 GB até estabilizar-se em torno de 6 GB, o que evidencia o carregamento temporário dos buffers de dados e a execução simultânea dos pods internos responsáveis pelo processamento e transferência entre os conectores. Esse comportamento confirma o impacto direto da execução do pipeline de sincronização nos recursos computacionais, demonstrando que o Airbyte, por operar sobre um cluster Kubernetes interno, mantém uma sobrecarga mais pronunciada e persistente, mesmo em tarefas de curta duração. Ainda assim, o padrão observado é consistente e controlado, sem indícios de vazamento de memória ou flutuações anômalas de CPU, o que indica estabilidade no gerenciamento de recursos durante a carga de trabalho.

As imagens a seguir representam a evolução temporal do comportamento do Airbyte durante o processo de sincronização dos dados do banco SOLAR para o destino DWH. Antes do início da sincronização, conforme mostrado anteriormente, o contêiner apresentava atividade residual, com CPU média de 9% e memória estável em torno de 6 GB, reflexo do funcionamento de base do cluster Kind. Essa condição de ociosidade foi fundamental como ponto de referência para identificar as mudanças decorrentes da execução efetiva do pipeline.

Na primeira imagem (Figura 30), correspondente ao início da sincronização, observa-se um aumento imediato e significativo na utilização de CPU, atingindo picos de até 20% e estabilizando-se em torno de 10%. Esse comportamento marca a fase de inicialização das tarefas internas do Airbyte, incluindo o carregamento dos conectores, a criação dos pods de execução e o estabelecimento das conexões JDBC com as bases

de dados de origem e destino. Paralelamente, o consumo de memória física cresce gradualmente até cerca de 6 GB, indicando o carregamento dos buffers de dados e estruturas temporárias de controle da transferência.

Figura 31 – Evolução temporal da sincronização do Airbyte



Fonte: Capturado pela autora (2025).

Na segunda imagem, já com o processo em andamento há alguns minutos, o uso de CPU se mantém em níveis estáveis, com média de 11% (User CPU 7%, System CPU 4%), apresentando pequenas oscilações rítmicas que refletem a atividade cíclica de leitura, transformação e escrita dos blocos de dados. O consumo de memória, por sua vez, estabiliza-se em um platô constante de aproximadamente 6 GB, sinalizando que o sistema alcançou um equilíbrio de carga, em que os recursos alocados são suficientes para manter o fluxo contínuo de processamento sem a necessidade de novas expansões. Na Figura 31, transcorridos quase 30 minutos de execução, verifica-

se a manutenção desse padrão de estabilidade, tanto na CPU quanto na memória. O uso total de CPU permanece entre 10% e 12%, com pequenas variações associadas à serialização e confirmação de lotes de dados.

De forma comparativa, nota-se que, enquanto no estado ocioso o contêiner já apresentava sobrecarga estrutural decorrente da arquitetura do cluster interno, durante a execução do pipeline o Airbyte mantém um perfil estável de consumo, sem degradação perceptível de desempenho ou anomalias de uso. Esse padrão reforça que, embora o Airbyte demonstre comportamento previsível e consistente, trata-se de uma previsibilidade que exige significativamente mais recursos computacionais, sobretudo em termos de CPU e memória, quando comparado a ferramentas locais como o KNIME. Assim, sua estabilidade vem acompanhada de um custo operacional elevado, inerente à natureza distribuída e contêinerizada de sua arquitetura.

4.3.2 Conclusões a respeito da utilização do Airbyte

A análise comparativa entre o Airbyte e o KNIME Analytics Platform evidencia que, embora ambas as ferramentas se enquadrem na categoria de soluções low-code para integração e transformação de dados (ETL/ELT), elas seguem caminhos arquitetônicos e conceituais distintos. O KNIME adota uma abordagem local e modular, baseada em execução direta sobre a JVM (Java Virtual Machine), enquanto o Airbyte é concebido para ambientes distribuídos e escaláveis, estruturado sob uma arquitetura contêinerizada voltada à orquestração em larga escala.

Do ponto de vista de usabilidade, o Airbyte apresenta uma experiência mais ágil e intuitiva, permitindo a configuração rápida de conexões, sincronizações e destinos por meio de uma interface web moderna. Já o KNIME requer maior atenção e familiaridade com seus componentes, uma vez que cada fluxo de dados é construído de forma visual e granular, exigindo o encadeamento lógico de nós e a parametrização manual das etapas. Essa diferença reflete diretamente a filosofia de cada plataforma: enquanto o Airbyte busca automatizar e simplificar os processos de integração, o KNIME privilegia o controle detalhado e a flexibilidade analítica.

Em termos de arquitetura e consumo de recursos, as divergências são ainda mais expressivas. O Airbyte, por operar dentro de contêineres e manter um cluster Kubernetes interno via abctl, exige infraestrutura robusta, com alto uso de CPU e memória mesmo em estado ocioso. Essa característica reforça seu posicionamento como uma ferramenta voltada à escala corporativa e à execução em nuvens privadas ou públicas, distanciando-se do perfil de uso local.

O KNIME, em contrapartida, apresenta comportamento mais leve e previsível, operando diretamente no *host* sem camadas intermediárias de orquestração, o que o torna mais acessível para ambientes de pequeno e médio porte, mas não limitando-se a isso. Outro aspecto observado refere-se ao impacto no banco de dados destino. Durante os testes, constatou-se que o Airbyte cria tabelas adicionais de controle e versionamento, o que aumenta consideravelmente o volume armazenado — em alguns dos experimentos, o tamanho da tabela original chegou a triplicar. No entanto, não foi possível registrar esse comportamento de forma visual ou quantitativa precisa, uma vez que ocorreram travamentos recorrentes durante as tentativas de captura, tanto no próprio Airbyte quanto em ferramentas auxiliares, como o DBeaver e os contêineres de banco de dados.

As observações empíricas e os logs de execução sustentam a inferência de que há um crescimento significativo do volume físico de dados gerado pela estrutura de versionamento interna do Airbyte.

Finalmente, destaca-se que ambas as ferramentas possuem alto potencial de crescimento e amadurecimento tecnológico. O KNIME, embora ainda pouco difundido no Brasil, já é amplamente utilizado por grandes corporações internacionais, conforme indicado em sua própria documentação, sendo reconhecido por sua versatilidade analítica. O Airbyte, por sua vez, continua a expandir sua base de usuários e integrações, consolidando-se como uma solução de integração moderna e escalável.

Sob essa ótica, conclui-se que o KNIME se mostra mais adequado para cenários locais e controlados, enquanto o Airbyte se destaca em contextos corporativos que demandam automação, escalabilidade e integração contínua em ambientes distribuídos.

5 Resultados discussões

Este capítulo apresenta a consolidação e a discussão dos resultados obtidos no estudo comparativo entre as ferramentas Airbyte e KNIME, aplicadas a um banco de dados real da DPE/RN. A análise se concentrou no monitoramento do comportamento operacional das ferramentas, tanto em estado ocioso quanto durante a sincronização de dados, utilizando a plataforma de observabilidade Datadog.

5.1 Subsídios práticos para a escolha consciente de ferramentas ETL *low-code*

A escolha de uma ferramenta *low-code* deve considerar o contexto organizacional, sobretudo em instituições públicas com restrições de recursos e agilidade tecnológica. O estudo evidenciou que essa decisão deve equilibrar escalabilidade corporativa e eficiência em ambientes limitados.

Quadro 4 – Comparativa entre as ferramentas

Critério de escolha	Airbyte	KNIME Analytics Platform
Arquitetura principal	Apresenta arquitetura distribuída e modular, executada em contêineres orquestrados por Kubernetes e controlados via abctl.	Arquitetura local e tradicional, executada sobre a Java Virtual Machine (JVM).
Cenário de uso ideal	Indicado para ambientes corporativos que exigem automação, escalabilidade e integração contínua em infraestruturas distribuídas ou baseadas em nuvem.	Indicado para cenários locais e controlados, voltados à análise exploratória e à ciência de dados, sendo especialmente adequado a ambientes de pequeno e médio porte, embora possa ser aplicado em contextos mais amplos.
Demanda de recursos	Demanda uma infraestrutura sólida, com elevado poder de processamento e disponibilidade de recursos avançados, como CPUs modernas, múltiplos núcleos	Mais eficiente e com uso previsível de recursos, ideal para ambientes com limitações de processamento ou memória.

Critério de escolha	Airbyte	KNIME Analytics Platform
	e grande quantidade de memória.	
Filosofia/Usabilidade	Busca automatizar e simplificar os processos de integração, apresentando uma curva de aprendizado mais suave, especialmente em cenários de replicação pura de dados.	Prioriza flexibilidade, controle granular e análise exploratória, com uma curva de aprendizado mais acentuada devido à vasta biblioteca de nós.

Fonte: Elaborado pela autora (2025).

Em síntese, a pesquisa mostra que não existe uma solução universal, mas sim a mais adequada a cada contexto. Em instituições com recursos limitados, tornar visível o desempenho das ferramentas é importante para decisões racionais e otimização da infraestrutura. Além disso, ferramentas *low-code* permitem que usuários de negócio participem da construção de *pipelines* de dados, o que torna sua adoção especialmente interessante quando essa colaboração é valorizada pela organização.

No caso específico da DPE/RN, o uso do Airbyte — já adotado institucionalmente — mostrou-se a escolha acertada. Embora o KNIME tenha se demonstrado plenamente capaz e bem aceito ao longo do projeto de residência, a análise do *stack* tecnológico existente revelou que sua adoção não seria a mais adequada ao cenário da instituição. A arquitetura atual, os fluxos operacionais estabelecidos e a integração nativa do Airbyte com o ecossistema já implantado reforçam que a ferramenta atende de forma mais eficiente às necessidades reais da DPE/RN.

5.2. Impacto das diferenças arquiteturais no desempenho e consumo de recursos

As divergências arquiteturais entre o Airbyte e o KNIME impactam de forma fundamental o desempenho e, sobretudo, o consumo de recursos computacionais, conforme evidenciado pelas métricas coletadas:

5.2.1 Arquitetura e sobrecarga de base

O Airbyte é estruturado sob uma arquitetura distribuída e contêinerizada, utilizando o utilitário abctl para criar um *cluster* Kubernetes interno (Kindest/Node). Essa arquitetura, embora privilegie a escalabilidade e a orquestração moderna, resulta em uma sobrecarga de base significativa.

Quadro 5 – Análise das ferramentas em período ocioso

Airbyte em Ociosidade	Mesmo sem *pipelines* em execução, o contêiner de controle principal (airbyte-abctl-control-plane) apresentava uso contínuo de CPU em torno de 11% e uma ocupação de memória física estável de aproximadamente 6 GB. Esse consumo é resultado da necessidade de manter o *cluster* Kubernetes ativo e seus serviços essenciais disponíveis.
KNIME em Ociosidade	Por ser uma aplicação local, o consumo de recursos do KNIME é mais diretamente vinculado à execução efetiva dos fluxos de dados, não apresentando essa sobrecarga estrutural permanente.

Fonte: Elaborado pela autora (2025).

5.2.2 Desempenho em execução

Durante a sincronização de dados – carga de 11 GB –, o comportamento de ambas as ferramentas revelou perfis de consumo distintos.

Quadro 6 – Análise das ferramentas em tempo de execução

Airbyte **Carga Distribuída**	1) A execução do pipeline exigiu CPU moderna, múltiplos núcleos e muita memória para um desempenho adequado; 2) No início da sincronização, o uso de CPU atingiu picos próximos a 20% e estabilizou-se em torno de 9% a 10% (valor superior ao estado ocioso); 3) O consumo de memória física elevou-se progressivamente, estabilizando-se em torno de 6 GB.

	4) O Airbyte demonstrou um padrão estável e consistente no gerenciamento desses recursos, mas essa previsibilidade exige significativamente mais recursos computacionais em comparação ao KNIME, o que implica um custo operacional elevado
KNIME **Carga Local/JVM**	1) O KNIME demonstrou eficiência e estabilidade superiores em cargas moderadas; 2) Durante a execução do nó DB Writer, o uso de CPU apresentou picos contínuos entre 8% e 12%, refletindo operações intensivas de leitura e transformação em memória 3) O uso de memória aumentou rapidamente, estabilizando-se em cerca de 3 GB, indicando que o esforço computacional maior está ao lado do cliente (KNIME, na JVM), e não do servidor de banco de dados.

Fonte: Elaborado pela autora (2025).

Em suma, a arquitetura distribuída do Airbyte (baseada em contêineres e Kubernetes) confere-lhe escalabilidade e automação, mas exige infraestrutura mais robusta. A arquitetura local do KNIME (baseada em JVM) o torna mais leve e eficiente em termos de consumo, sendo mais adequado para cenários com recursos limitados, embora limite sua escalabilidade nativa.

6 Considerações finais

Ao longo do desenvolvimento deste trabalho, emergiram questões relevantes que merecem atenção no contexto do tema abordado. A seguir, esses aspectos são discutidos, à luz das experiências obtidas ao longo do processo.

6.1 Limitações

A primeira limitação está relacionada à infraestrutura: a DPE/RN não forneceu uma máquina virtual (VM) dedicada para execução e testes; portanto, o desenvolvimento e a execução dos *pipelines* ocorreram, em um computador pessoal em ambiente residencial. Essa escolha impôs restrições óbvias — recursos de CPU, memória e I/O limitados, concorrência com outras tarefas locais e variabilidade na conectividade de rede — que podem afetar tanto o desempenho observado quanto a reprodutibilidade dos testes em ambientes corporativos. Em termos metodológicos, a falta de uma VM institucional impede a replicação em condições idênticas às de produção, tornando necessário interpretar os resultados com cautela e considerar que desempenho em infra dedicada (servidor ou cloud) tende a diferir substancialmente.

A curva de aprendizado constitui uma limitação também importante de ser mencionada. Tanto Airbyte quanto KNIME possuem documentação técnica majoritariamente em inglês e, embora contenham guias e exemplos, muitos conectores, parâmetros avançados e logs operacionais requerem leitura atenta e tempo para interpretação. Essa barreira linguística e técnica aumentou o tempo necessário para projetar, ajustar e validar *pipelines* equivalentes nas duas plataformas.

A proposta inicial previa a utilização de múltiplos bancos, com volumes e naturezas variadas, de modo a ampliar o alcance da análise e avaliar o comportamento das ferramentas em diferentes contextos de carga. Entretanto, essa abordagem mostrou-se inviável diante das limitações de recursos computacionais disponíveis, sobretudo devido à alta demanda de CPU e memória exigida pelo Airbyte em execução local.

Outro ponto interessante de se mencionar é que, embora o Datadog seja uma

ferramenta de monitoramento extremamente completa e robusta, oferecendo recursos avançados para observabilidade de sistemas distribuídos, sua aplicação no contexto deste estudo foi severamente limitada pela arquitetura e pelo modelo de licenciamento do Airbyte. As versões mais recentes da plataforma restringem a exposição de métricas internas, permitindo sua coleta apenas por meio da edição Enterprise Self-Managed, cuja assinatura ultrapassa US$ 600, a depender de recursos contratados, conforme verificado na análise de viabilidade financeira. Essa limitação inviabilizou a integração plena entre o Airbyte e o Datadog, reduzindo a observabilidade a um conjunto mínimo de métricas de infraestrutura coletadas diretamente dos contêineres.

Em contraste, o processo Java correspondente à execução do KNIME Analytics Platform revelou-se muito mais acessível à análise detalhada, uma vez que o Datadog pôde explorar métricas internas da JVM (heap, garbage collector, threads, classes carregadas, etc.), permitindo uma visão granular do comportamento do sistema durante a execução dos pipelines. No entanto, por questões de tempo e da própria curva de aprendizado envolvida no domínio completo dessas métricas, optou-se por uniformizar as análises, limitando-as aos indicadores de CPU e memória equivalentes entre as duas ferramentas. Essa escolha metodológica assegurou paridade comparativa e reforçou a validade das observações, ainda que em níveis distintos de profundidade técnica.

6.2 Trabalhos futuros

A partir das contribuições e limitações identificadas neste estudo, destacam-se diversas oportunidades de trabalhos futuros capazes de ampliar o impacto e a aplicabilidade prática dos resultados obtidos.

Uma primeira e direta continuação seria a implantação piloto do KNIME e do Airbyte em um ambiente de produção ou em máquinas virtuais dedicadas na DPE/RN, com o objetivo de avaliar não apenas o desempenho técnico, mas também aspectos de integração operacional, governança de dados e aceitação pelas equipes de TI e análise. Essa abordagem permitiria mensurar ganhos reais em tempo de execução,

confiabilidade, manutenção e custo operacional, sob condições controladas e próximas da realidade institucional.

Outra linha relevante envolve a replicação dos experimentos em ambientes de nuvem escaláveis, comparando o comportamento das ferramentas em instâncias de diferentes configurações de CPU, memória e armazenamento. Esse tipo de avaliação poderia incluir métricas econômicas, como custo por gigabyte processado, possibilitando análises de custo-benefício entre execução local e cloud.

Além disso, recomenda-se o aprofundamento da observabilidade das execuções, especialmente por meio da utilização completa da ferramenta Datadog, explorando painéis avançados de APM, rastreamento distribuído, monitoramento de threads e análise de gargalos em tempo real. Mas vale destacar que a implementação dessa abordagem exigiria infraestrutura dedicada e versões licenciadas das ferramentas, conforme pode ser visto em capítulos anteriores.

Trabalhos futuros também poderiam incluir o uso de outros sistemas de monitoramento complementares, como Prometheus e OpenTelemetry, integrados a ferramentas de logging e alerting, permitindo a construção de *pipelines* de observabilidade completos, desde a coleta de métricas até o diagnóstico automatizado de falhas. Outra vertente promissora é o desenvolvimento de conectores customizados voltados às fontes de dados específicas da DPE/RN, ampliando a interoperabilidade das soluções testadas. De modo complementar, a integração das práticas de ETL/ELT com *pipelines* de CI/CD poderia ser avaliada, explorando automação, versionamento e implantação contínua de fluxos de dados.

Finalmente, como um último ponto que poderia ser avaliado, é enfocar a usabilidade e a produtividade das equipes, mediante estudos qualitativos, entrevistas e testes controlados com analistas de diferentes níveis de proficiência, buscando compreender a curva de aprendizado real, a adoção prática das plataformas low-code e o impacto dessas ferramentas na redução do esforço técnico e na democratização do acesso à engenharia de dados no setor público.

Referências

Afef Walha, Faiza Ghozzi & Faiez Gargouri, *Data integration from traditional to big data: main features and comparisons of ETL approaches*. The Journal of Supercomputing, vol. 80, 2024. Disponível em: https://link.springer.com/article/10.1007/s11227-024-06413-1. Acesso em: 15 setembro 2025.

ANDERSON, Robert et al. Magic Quadrant for Cloud ERP for Service-Centric Enterprises. Stamford: Gartner, 2024. Disponível em: https://www.gartner.com/doc/reprints?id=1-2LJIU4E6&ct=250728&st=sb. Acesso em: 13 ago. 2025.

Anthony Mbata, Yaji Sripada & Mingjun Zhong, *A Survey of Pipeline Tools for Data Engineering*. Department of Computing Science, University of Aberdeen, UK. 2024. Disponível em: https://arxiv.org/html/2406.08335v1. Acesso em: 15 outubro 2025.

Arvind Singh Kamlakar, *Low-code ETL vs. Traditional Code-based ETL: A Comprehensive Comparison*. 2023. Disponível em: https://medium.com/%40a.kamlakar/low-code-etl-vs-traditional-code-based-etl-a-comprehensive-comparison-d098e8f3457f. Acesso em 08 de setembro de 2025.

BELLI, Luca et al. Governança de dados no setor público: abertura de dados governamentais, proteção de dados pessoais e segurança da informação para uma transformação digital sustentável. Rio de Janeiro: Lumen Juris, 2024. 188 p. Disponível em: https://unu.edu/sites/default/files/2024-07/Governan%C3%A7a%20de%20dados%20no%20setor%20p%C3%BAblico_EBOOK_0.pdf. Acesso em: 02 out. 2025.

BERTHOLD, Michael R. et al. KNIME: The Konstanz Information Miner. University of Konstanz, Nycomed Chair for Bioinformatics and Information Mining, 2009. Disponível em: https://www.researchgate.net/publication/2009_KNIME_the_Konstanz_information_miner. Acesso em: 13 ago.

BODICHERLA, Balaji. The Rise of Low-Code/No-Code Development: Democratizing Application Development. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, v. 11, n. 2, p. 171-178, Mar./Apr. 2025. Disponível em: www.ijsrcseit.com. Acesso em: 13 ago. 2025. DOI: 10.32628/CSEIT251112398.

Documentação do Airbyte “Low-code framework enables you to build source connectors for REST APIs via a connector builder UI. Disponível em: https://docs.airbyte.com/platform/connector-development/config-based/low-code-cdk-overview. Acesso em 05 de agosto de 2025.

Documentação do: KNIME "is a single platform for end-to-end data science … Access any data type from any source with 300+ connectors. Disponível em: https://www.knime.com/knimepress. Acesso em 05 de agosto de 2025.
Guia de Componentes do KNIME, Disponível em: https://docs.knime.com/2020-07/analytics_platform_components_guide/index.html#introduction. Acesso em 15 de setembro de 2025.

Harald Foidl, Valentina Golendukhina, Rudolf Ramler, Michael Felderer, *Data Pipeline Quality: Influencing Factors, Root Causes of Data-related Issues, and Processing Problem Areas for Developers*. 2023. Disponível em: https://arxiv.org/abs/2309.07067. Acesso em 29 de setembro de 2025.

KIMBALL, Ralph; ROSS, Margy. The Data Warehouse Toolkit: The definitive guide to dimensional modeling. 3. ed. Hoboken: Wiley, 2013.

KLEINERMANN, F.; SOUSA, A.; LIMA, D. *Data Engineering for Beginners: ETL, Pipelines, and Data Warehousing*. São Paulo: TechPress, 2021.

MAR, Joseana. Acessibilidade digital como cultura: daltonismo e dislexia: o mundo já está na web e não podemos deixar ninguém de fora. 2020. Disponível em: https://brasil.uxdesign.cc/acessibilidade-digital-como-cultura-daltonismo-e-dislexia-16939161d517. Acesso em: 07 maio 2023.

MARR, Bernard. The Big Data & AI Guide: The essential guide to the future of data and artificial intelligence. [S. l.]: Bernard Marr & Co., 2022. 24 p. E-sampler. Disponível em: https://bernardmarr.com/wp-content/uploads/2022/05/Big-Data-Esampler-1.pdf. Acesso em: 02 out. 2025.

MENDES, J. P.; RODRIGUES, T. C. Adoção de Ferramentas Low-code em Projetos de ETL: Estudo Exploratório em Organizações Públicas. *Revista Brasileira de Computação Aplicada*, v. 14, n. 2, p. 89-105, 2022.

Moreira, Rogério de Freitas. Impactos econômicos da covid-19 no mercado odontológico privado no estado do RS: O desenvolvimento e avaliação de um app low code de data science no KNIME. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/270551/001193234.pdf?sequence=1&isAllowed=y. Acesso em: 07 setembro 2025.

QAISER, Asma et al. Comparative Analysis of ETL Tools in Big Data Analytics. Pakistan Journal of Engineering and Technology, Lahore, v. 6, n. 1, p. 7-12, jan. 2023. Disponível em: https://www.researchgate.net/publication/369094822_Comparative_Analysis_of_ETL_Tools_in_Big_Data_Analytics. Acesso em: 13 ago. 2025.
SAARINEN, Mikko. *Low-code platforms in software and data development*. 2024. Disponível em

https://trepo.tuni.fi/bitstream/handle/10024/157011/SaarinenMikko.pdf;jsessionid=8DFE3F5366322CEF362B130EADBE3F3E?sequence=2. Acesso em: 11 ago. 2025.

VASCONCELOS, Luciano. Data pipeline e ETL são a mesma coisa? [Vídeo]. Jornada de Dados - Luciano Vasconcelos, 22 ago. 2025. Disponível em: https://www.youtube.com/watch?v=AKjf74tS__Y. Acesso em: 3 set. 2025.

WELLER, A. *The Practical Guide to Data Pipelines: Building Efficient ETL Workflows*. New York: DataPub, 2020.