International Journal of Academic Research and Development
www.allstudiesjournal.com
ISSN: 2455-4197
Received: 08-09-2024, Accepted: 09-10-2024, Published: 24-10-2024
Volume 9, Issue 5, 2024, Page No. 42-47

Iterative build approach for realtime data streaming infrastructure in retail data organization

Ramla Suhra
Department Of Data Solutions, H-E-B Digital, Texas, USA

Abstract
When data-driven decisions become a strategic priority for retail organizations, real-time analytics often emerges as an indispensable tool. The initial step is typically to invest in real-time infrastructure, with streaming platforms as a core component. This case study examines the journey of a retail organization as they built and matured their real-time data streaming infrastructure which is cloud native. We will discuss how they continuously assessed and adjusted the platform to meet evolving requirements and leverage emerging technologies. The whitepaper provides a detailed examination of the iterative approach used to rapidly develop a robust streaming infrastructure capable of meeting the organization's future demands. The platform's lifecycle is broken down into two sections: a high-level overview and in-depth technical analysis of various phases, including challenges and advantages. In conclusion, the paper summarizes key learnings from each iteration and improvements made during each and outlines a proposed roadmap for the future development of the streaming platform. By following this roadmap, the organization can continue to harness the power of real-time analytics to drive innovation and achieve business success.

Keywords: Data analytics, data ingestion, big data, real time data streaming

Introduction

The origin

Many retail data organizations migrating their legacy systems to the cloud often opt for batch ingestion as an initial strategy to load data. While this approach suits batch analytics, below key factors drive the shift to real-time analytics [3, 4, 5, 10].

Diminishing value of data with time: Organizations have more streaming data every year from internal sources, such as corporate websites, sensors, machines, mobile devices and business applications; and from external sources, such as social media platforms, data brokers and business partners. This information is most valuable when it is used as soon as it arrives to improve real-time or near-real-time business decisions.

- **Increased competition:** The need to respond quickly to market changes and customer demands.
- **Evolving customer expectations:** Customers expect personalized experiences and immediate gratification.
- **Technological advancements:** The availability of scalable and affordable real-time data processing solutions.
- **Growing complexity of data:** The volume, variety, and velocity of data require real-time processing capabilities.

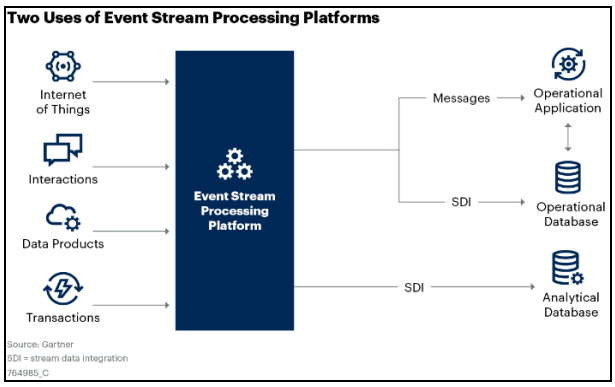

Fig 1: Event stream platform usages [11]

"Gartner Predicts Top 10 Global Retailers will user Real-Time In-Store Pricing by 2025" [1]. It becomes imperative to invest and build a dedicated streaming platform to meet this requirement. In most cases, a rapid development or deployment of an initial streaming platform will be preferred by the organization than waiting longer a fully blown infrastructure. It hence become crucial to build a platform which meets the minimum viable product requirements while leaving room for further expansion. Following such an approach helps both the platform engineering as well the business to meet on a most suitable middle ground to build a mutually beneficial solution.

Agile evolution

To expedite the setting up streaming infrastructure, it is always recommended to have a focused development approach. In this approach, key features and functionalities will be marked as minimum viable product (MVP). Once the MVP is deployed, a series of iterative development cycles can be implemented to enhance functionality, address feedback, and optimize performance. Through this continuous improvement process, the platform will evolve into a robust solution capable of meeting both current and future needs.

In the MVP phase, the platform was required to designed for ingesting, transforming and storing streaming data for subsequent analytics applications. Thus, the infrastructure components were also identified based on this limited scope. Due to the nature of the requirement, pre-built streaming connectors [2] were preferred components in the infrastructure for the MVP as they are configuration-driven solutions that simplify the integration of streaming platforms with other systems. They offer a low-code or no-code approach, making it easier to onboard producers and consumers without extensive development effort.

Schema evolution is another aspect that had to handle and schema registry was also added as a component in the MVP. Source system types were identified, and their connectors

were planned to be pre-configured for use. Infrastructure as code was mandated in the MVP and Terraform was chosen as the tool for this build.

Advantages of the approach

- While real-time data streaming enables data-driven decision-making based on current trends and customer behavior following an MVP approach for infrastructure build will increase the time-to-market.
- Focuses on core functionalities initially, minimizing upfront costs.
- Allows platform engineers for gradual feature additions based on the platform usage and customer experience and feedback.

Process workflow

The workflow shown in Figure II captures various phases involved in the approach followed by the organization.

Step 1: Gather Initial Requirements for Streaming Platform

In this phase focus is primarily on finding all the specific goals and needs for the organization and below are some of the key areas identified.

- Support data import from multiple sources.
- Support integration with multiple storage solutions.
- Must support various data formats and protocols.
- Reusable platform/tool with Multi cloud support.
- High throughput, Low latency and Fault tolerant.
- Highly scalable and reliable with disaster recovery options.
- Strong security features like RBAC, Monitoring and Alerting capabilities.
- Ease of onboarding with pre-built connectors for common use cases.
- Cost efficiency.
- Ease of maintenance and operation.
- Community and support for the platform.

Step 2: Platform/Tool Selection

Various platforms were evaluated based on the key items from above requirements and results were compiled as per Table I.

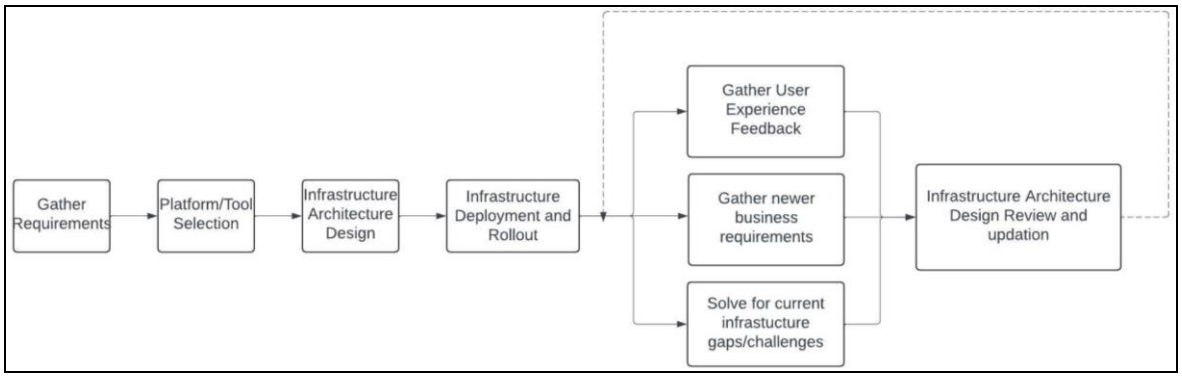

Fig 2: High-level workflow of iterative build approach

Table 1: Platform comparison matrix

Capabilities	Open Source	Commercial–Self-managed	Commercial-SaaS	Commercial–Cloud Specific
Support for multiple sources	Yes	Yes, Better support than open source		
Support for various data formats and protocols	Yes	Yes	Yes	Yes, needs some extra logic
High throughput, Low latency and Fault tolerant	Yes	Yes	Yes	Yes, specific solution for the cloud
Security Features	Yes	Yes, more tailored for enterprises	Yes, more tailored for enterprises	Yes. However, needs more infrastructure setup while working with other cloud.
Out of the box metrics for monitoring	Yes, slightly more granular than cloud specific	Yes, more tailored for enterprises	Yes, more tailored for enterprises	Yes
Ease of onboarding	No, Needs more technical knowledge	Yes, Lesser learning curve than Open Source	Yes, Much Lesser learning curve	Yes, Lesser learning curve than Open Source
Custom pre-built connectors	No	Yes	Yes	No
Cloud Agnostic	Yes	Yes	Yes	Nil
Simplified operations	No, needs more administration	Yes, however still needs more administration effort	Yes, great	Yes, however still needs more administration effort
Community support	Yes	Yes	Yes	No, Lesser compared to others
Vendor Support	No	Yes	Yes, greater support since it is a managed service	Yes
Cost efficiency	Yes	No	Yes, with vendor discounts.	No

Table 2: Platform capabilities

Capabilities	Required for MVP?	Comments
IaC	Yes	Infrastructure as Code for faster and automated deployment
Networking setup for legacy data stores	Yes	
Networking setup for additional data stores	No	Will be setup based on future demand
Custom source connectors readiness	Yes	All the vendor provided ones to be made available.
Custom sink connectors readiness	No	For MVP, this is going to be used only for cloud data lake hydration.
Highly secure with features like RBAC	No	Default ACL based security features for MVP. This will be implemented as a fast follower
Monitoring and alerting	Yes	Default ones will be enabled. More tightening will be done in future.
Self Service Model for client connector deployments	Partial	Centrally deployed by platform engineering managed Continuous Integration and Continuous Deployment (CICD)
Advanced features like real-time transformations	No	Future requirement
Distributed Deployment Architecture	Partial	Advanced deployment architecture will be implemented in the later phases.

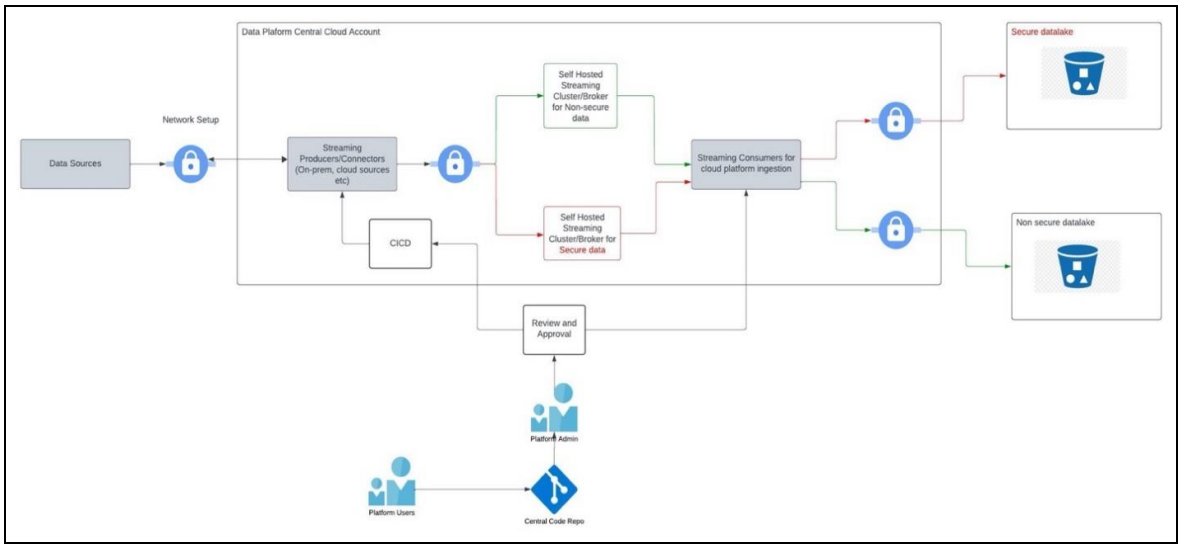

Fig 3: Build iteration #1

Step 3: Infrastructure architecture and design

After thoroughly reviewing the pros and cons, Commercial self-managed seemed to strike a balance and this is a good approach for MVP solution. Since there is a potential to switch to SaaS later down the lane, self-managed is always a good initial choice.

Infrastructure design is the next phase in the workflow. Key areas shown in Table II were reviewed during this phase. Some of the features were categorized as phase 2 items so that MVP could be finished quickly.

Figure III illustrates the finalized core architecture of the minimum viable product (MVP), which was designed by carefully selecting essential features.

Step 4: Infrastructure Deployment and Rollout

Now that the infrastructure design, finalized, next step is deploying the infrastructure. Various steps involved in this phase are as below.

Determine the specific needs of the organization, including data volume, latency requirements, scalability needs, and security considerations.

Select a suitable architecture for the streaming platform.

Provision virtual machines or containers on the cloud platform to run the streaming platform components.

Set up network infrastructure, including virtual private clouds (VPCs) and security groups.

Connect the streaming platform to cloud-based data sources, such as data lakes, data warehouses, and databases.

Setup IaC and CICD to help automate the provisioning, configuration, and management of infrastructure, reducing the risk of human error and improving efficiency.

The streaming platform was deployed on a cloud-based infrastructure managed by a central team. Private network connections were established to integrate with both legacy and cloud data sources. The platform's components were installed on virtual machines or containers in the cloud, including streaming brokers and connector nodes. Two separate broker instances were created to enhance data security and prevent unauthorized access of highly secured data. CICD based user code deployment was setup on the central code repository maintained for connector deployments.

Testing

The infrastructure was thoroughly tested for all its key features. This rigorous testing process was required in identifying and addressing most of the existing issues before the platform's official release. The use of Infrastructure as Code (IaC) greatly helped to the efficiency of this testing phase, as it allowed for quick and repeated redeployment of the platform. This agility was crucial in ensuring the platform's readiness for production.

Rollout and communication

The Platform Admin Team developed a comprehensive communication plan and documentation to effectively

inform stakeholders about the upcoming platform rollout. This proactive approach ensured a smooth transition and minimized any potential challenges that users might face.
The documentation provided exhaustive details, including instructions on connectivity to producer and consumer connector instances, monitoring, and troubleshooting. Platform administration team also conducted training sessions and piloted the release for some teams for initial beta testing.

Step 5: Gather Feedback and Build next iteration
After launching the initial MVP, the platform remained in its minimum viable product state for a period. During this phase, the platform engineering team focused on handling and resolving user support requests. The use of pre-built connectors significantly accelerated time-to-market for many use cases.
To enhance the platform, the team conducted a thorough review of frequently requested support activities and features and identified areas for improvement. In parallel, they also held feedback sessions with users to gather additional insights and incorporate their suggestions.

Major Learnings
Since the deployment is handled centrally, user teams found this as bottleneck sometimes as they had to wait for platform engineer's approval before deployment.
Most of the connector failures were due to incorrect values in connector configurations. Users found it difficult to troubleshoot many such instances.
Central team started becoming a bottleneck to deployment of connectors in production as they were the assigned reviewers for the code deployments.
User team didn't have the ability to ability their connectors/producer/consumer apps in the case of failures.
Connector logs were not accessible to users in some cases.
User team had to share the source system connection credentials with Platform team, and they didn't have a way to share them securely to the connectors directly.
Platform team identified need for timely clean-up of streaming storage to avoid cost.
It was found that advanced streaming transformation use cases are not required.

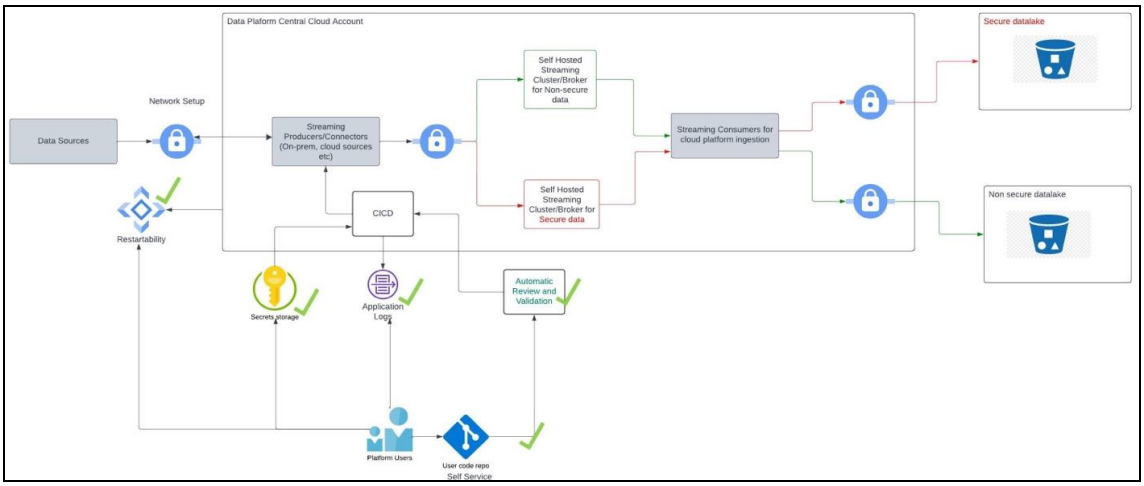

Fig 4: Build iteration #2

Step 6: Design and roll out second iteration
By the next iteration the platform was revamped to accommodate most of the user requirements as shown in Figure IV. Some of the major improvements are as detailed below.
User self-service connector deployment model: Decoupled the dependency on central code repository for connector deployments. Users could deploy connectors from their own code repository by cloning the base template from central team's repository.

Automatic connector pre-deployment validation: Introduced a validation framework to validate the connector configuration files automatically before deployment. This proactive validation reduced connector failures to a great extend and improved user experience.

An open source json validation framework, jsonnet [9] was integrated for this purpose.
After introducing validation framework, the connector code deployment approval process from platform team could also be automated. Teams could follow their peer review and approval process as usual there by ensuring the deployment process is much smoother than before.
Introduced restart ability for connectors through API end points accessible by the users. This helped the users during failures to restore the connectors as quickly as possible.
Connector and broker logs were reviewed and enhanced to include as much detail as possible. They were also routed to a central enterprise logging which is available for all teams.
All the key metrics were identified, and Platform administration team ensured that they were included and broadcasted to the wider audience. Some examples are message throughput, consumer lag, and broker resource utilization.
The most crucial issue related to lack of secure credentials sharing was solved by integrating with secret vault so that the connectors can read from the vault directly and deployed them into the connector nodes. This greatly improved the security of the platform and increased trust from stakeholders.
The new release was well received by user teams for significantly enhancing user experience and platform stability. Additionally, the release included improved monitoring and automated housekeeping features.

Major learnings

After rolling out the next iteration and keeping this running for a period, another round of feedback was gathered. Some learnings are included here.

Scalability Concerns: Although the MVP model was sufficient to cater to the needs for initial user base, we found that the infrastructure could not be scaled down the lane. Each streaming broker was deployed on its own dedicated virtual machine node. This system was managed by a variety of tools, such as Chef, Terraform, as well as third-party services, which were located under different git project locations.

Complicated operations and management: Each component in the platform required careful consideration, testing, and approval–even for seemingly minor tasks, such as upgrades. These complications resulted in a much bigger workload and complexity for platform administration team.

Reduce manual housekeeping/administration: We identified some areas where automation can be introduced to eliminate manual administration tasks. Topic management, connector clean-up are some frequent examples.

Lack of role-based access control: This will increase security risk as the data access can be compromised especially when onboarding health and wellness or any other highly confidential information.

Retail industries due to its nature of products and services often get to deal with variety of data like highly sensitive health and wellness data, customer credit card information etc. The subsequent iterations had to review such scenarios and build custom solutions for them including private link connectivity for data transfers, secured storage of highly secured data and so on.

Major Milestones

Private connectivity

There were some use cases where customers needed to move data between different parts of the organization through streaming where direct connect was restricted due to internal security restrictions setup within the enterprise. This situation had hindered our ability to achieve our business goals, leading us to invest resources in creating inefficient workarounds. After extensive research and development, streaming administration team has released private endpoints in streaming infrastructure to access services directly from their virtual cloud premise, eliminating the need for network traffic to travel over the public internet (shown in Figure V).

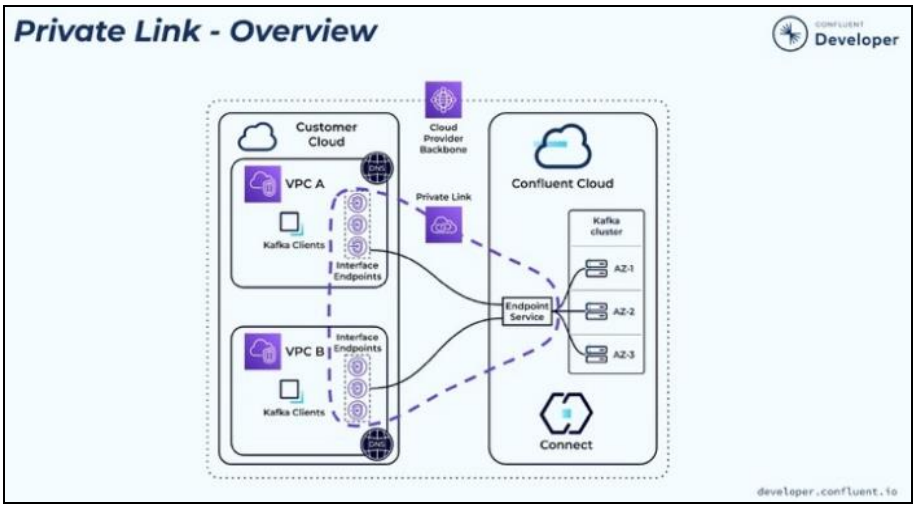

Fig 5: Confluent cloud private link connectivity [8]

Future steps for a lasting impact

There is a need to migrate the clusters to a more advanced, automated, high-performing, and easy-to-manage infrastructure–benefiting both our clients and the team's daily operations. Kubernetes architecture is the chosen approach for this revamp.

Upon research, we found some benchmarking done by AppsFlyer which is a global leader in mobile attribution and marketing analytics who transitioned from a similar model to K8s based solution.

"Between a Kafka cluster running on our original i3.2xlarge instance and a Kafka cluster running on the newer, improved im4gn.2xlarge Graviton instance, we saw amazing results, such as a 75% increase in throughput, a 10% lower CPU consumption, and most notably, a remarkable 58% increase in write I/O performance and a 92% increase in read I/O performance.

Fast-forward to now. As we look at our Kafka running on our new Kubernetes architecture on im4gn Graviton instances, we can confidently say that we've halved our CPU cores count, reached a 50% reduction in costs, and have the additional advantage of lowering our carbon footprint"[7].

Implementing an autoscaler for automation of scaling per usage is another enhancement that can be added to this mode. Karpenter [7] is an open-source project that serves as a Kubernetes node autoscaling solution. It is developed to optimize cost and resource efficiency considering Kubernetes pods' requirements and constraints, which aims to be more responsible and flexible.

In the future, AI and ML can be utilized to enhance recommendations for optimizing platform infrastructure. This could involve predicting future resource needs based on usage patterns and trends.

As a final step, cost analysis studies can be conducted to evaluate the platform's operational and maintenance costs. This will help determine if transitioning to a SaaS platform is more suited economically, especially as the platform gains wider adoption and matures.

However, with the growth of data and streaming needs, the current model can take increasing engineering effort to provision, configure, scale, and orchestrate the brokers and Zookeeper. To reduce infrastructure management and focus more on our growing business, we can analyze and migrate from our self-managed clusters to Commercial Managed Streaming from the Vendor.

With stronger vendor support and accountability organization can scale up faster with lesser operational overheads.

Conclusion

While this article explored the benefits of using rapid development approach to quickly deploy infrastructure and evolve subsequently, we would stress that identification of the requirements is the key aspect that helps one succeed while following this model. More time and effort spent on the analysis and design phase has helped us deliver an infrastructure with less pain points even from the initial version.

References

1. G Omale. “Gartner predicts top 10 global retailers will use real-time in-store pricing by 2025,” Gartner, https://www.gartner.com/en/newsroom/press-releases/2019-03-27-gartner-predicts-top-10-global-retailers-will-use-rea (accessed Oct. 15, 2024). R. W. Schulte, P. den Hamer, and E. Zaidi, “Market Guide for Event Stream Processing,” Gartner, https://www.gartner.com/doc/reprints?id=1-2E6D79FI&ct=230619&st=sb (accessed Oct. 15, 2024).
2. Apache Software Foundation. “Apache Kafka,” *Apache Kafka*, 2020. https://kafka.apache.org/documentation/#connect
3. H Isah, F Zulkernine. “A scalable and robust framework for data stream ingestion,” *2018 IEEE* International Conference on Big Data (Big Data), 2018, 2900–2905. doi:10.1109/bigdata.2018.8622360
4. P Cappellari, M Roantree, SA Chun. “A scalable platform for low-latency real-time analytics of streaming data,” *Communications in Computer and Information Science*, 2017, 1–24. doi:10.1007/978-3-319-62911-7_1
5. Y Fu, C Soman. “Real-time data infrastructure at uber,” *Proceedings of the 2021 International Conference on Management of Data*, 2021. doi:10.1145/3448016.3457552
6. G Omale. “Gartner predicts top 10 global retailers will use real-time in-store pricing by 2025,” Gartner, https://www.gartner.com/en/newsroom/press-releases/2019-03-27-gartner-predicts-top-10-global-retailers-will-use-rea (accessed Oct. 15, 2024).
7. A Plotnikov, E Arditt, “Harnessing karpenter: Transitioning Kafka to Amazon Eks with AWS Solutions | Amazon Web Services,” AWS, https://aws.amazon.com/blogs/containers/harnessing-karpenter-transitioning-kafka-to-amazon-eks-with-aws-solutions/ (accessed Oct. 15, 2024).
8. J Lee. “Connect to confluent cloud via AWS/Azure Private Link,” Confluent, https://developer.confluent.io/courses/confluent-cloud-networking/private-link/ (accessed Oct. 15, 2024).
9. Jsonnet. “Jsonnet configuration language,” Jsonnet, https://jsonnet.org/ (accessed Oct. 15, 2024).
10. C Lekkala. “Advancements in Data Ingestion: Building High-Throughput Pipelines with Kafka and Spark Streaming,” *ResearchGate,* Jul. 2020, doi: 10.5281/zenodo.11489050.
11. A Woodie. “Yes, Real-Time streaming data is still growing,” *BigDATAwire*, 2023. https://www.datanami.com/2023/07/12/yes-real-time-streaming-data-is-still-growing/