International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I1P133
Eureka Vision Publication | Volume 7, Issue 1, 218 -225, 2026

Original Article

Cloud-Native Connectivity Paradigms: A Comparative Study of Managed Kafka Networking on AWS and Azure

Girish Rameshbabu
Customer Success Technical Architect,

Received On: 14/01/2026 Revised On: 13/02/2026 Accepted On: 21/02/2026 Published On: 28/02/2026

Abstract - The paradigm of real-time data streaming has transitioned from the operational overhead of self-managed Apache Kafka clusters to the abstraction of Managed Service Provider (MSP) models, exemplified by the Confluent Cloud "Kora" engine. While this transition eliminates the infrastructure management of broker orchestration and stateful scaling, it introduces significant challenges in the secure orchestration of data ingress and egress within multi-cloud environments. In enterprise deployments on Amazon Web Services (AWS) and Microsoft Azure, standard public endpoints are frequently precluded by stringent security mandates and the necessity to prevent data exfiltration. This study classifies and evaluates the dominant connectivity paradigms utilized to facilitate private communication between consumer applications and managed Kafka clusters. By analyzing the architectural trade-offs between bi-directional peering models and modern, unidirectional endpoint-based solutions, this paper highlights the nuances in DNS resolution and routing logic. We focus on AWS PrivateLink and Azure Private Link as the primary mechanisms for achieving logical isolation. The objective of this research is to provide a formal framework for selecting networking topologies that mitigate CIDR overlap conflicts while maintaining robust, private-path communication for mission-critical event streams.

Keywords - AWS Privatelink, Azure Private Link, Cloud-Native Architecture, Confluent Cloud, Event Streaming, Managed Apache Kafka, Multi-Cloud Networking, Zero-Trust Security.

1. Introduction
The migration from self-managed Apache Kafka environments to Managed Service Provider (MSP) platforms has redefined the security boundaries of enterprise data architecture. In traditional on-premises or self-hosted cloud deployments, networking is typically contained within a static logical perimeter. However, the adoption of Event Streaming as a Service (ESaaS) necessitates a shift where the network fabric serves as the primary enforcement point for security and data integrity.

1.1. Network Architecture and the Challenge of Address Overlap
A primary obstacle in the integration of managed services into existing corporate networks is the fragmentation of address spaces. Large-scale enterprise infrastructures often possess disparate networks with overlapping Classless Inter-Domain Routing (CIDR) blocks. Standard Layer 3 connectivity, such as Virtual Private Cloud (VPC) or Virtual Network (VNet) peering, requires unique, non-overlapping IP ranges across all connected entities. When these requirements cannot be met, routing conflicts occur, necessitating more sophisticated abstraction layers that operate independently of the underlying IP topology.

1.2. Transition to Zero-Trust Connectivity
Cloud-native streaming demands a transition from perimeter-based security to a Zero-Trust model. This requirement stipulates that network paths must be restricted to specific service endpoints rather than entire network segments. By utilizing unidirectional gateways, organizations can prevent lateral movement within the cloud provider’s infrastructure. This architectural shift ensures that traffic is confined to a verifiable path between the consumer and the producer, effectively mitigating the risks associated with shared network environments.

1.3. Structural Isolation: Control Plane vs. Data Plane
To manage the complexities of cloud connectivity, it is essential to decouple management functions from data transmission paths:

- The Control Plane: Handles cluster orchestration, metadata management, and administrative configurations. This layer is logically separated from the actual data flow to prevent management interference.
- The Data Plane: Facilitates the high-volume production and consumption of messages. This layer requires high-availability, private connectivity to ensure data remains within the organizational boundary.

The scope of this study is focused exclusively on the architectural patterns used to secure the data plane within AWS and Azure environments. This analysis prioritizes the structural integrity of these designs, examining how logical isolation is maintained through private endpoint technologies while deliberately omitting performance-based metrics to focus on architectural reliability.

2. Literature Review & Background

The architectural maturity of managed event streaming platforms is deeply intertwined with the parallel evolution of cloud networking and the internal abstraction layers of the streaming engines themselves. This section explores the transition from traditional networking models to modern endpoint-based paradigms and the structural innovations within the Confluent "Kora" engine.

2.1. Evolution of Cloud Networking: From Peering to Private Links

Early iterations of cloud networking relied primarily on public internet endpoints and Basic Virtual Private Cloud (VPC) peering. As documented in early cloud computing frameworks, the initial focus was on establishing basic connectivity using standard Layer 3 routing protocols [1], [2]. VPC peering allowed for bi-directional communication between virtual networks, yet it introduced significant administrative overhead. Specifically, it required mutually exclusive IP address spaces a requirement that proved untenable for large-scale enterprise environments with complex, often overlapping, CIDR topologies [1].

The introduction of technologies such as AWS PrivateLink and Azure Private Link represented a shift toward service-oriented networking. Unlike peering, which connects entire network segments, these technologies project a specific service such as a Kafka cluster as a local IP address (Elastic Network Interface or Private Endpoint) within the consumer's VPC/VNet [4]. This unidirectional approach effectively abstracts the underlying network infrastructure of the provider from the consumer, mitigating the risks associated with transitive routing and IP address conflicts.

2.2. The Kora Engine: Network Layer Abstraction

The technical foundation of Confluent Cloud is the "Kora" engine, a cloud-native redesign of the Apache Kafka protocol. A central innovation of Kora is its decoupling of the logical Kafka interface from the physical infrastructure [6]. In traditional Kafka deployments, clients must maintain persistent connections to specific brokers, making the networking layer sensitive to broker failures and IP changes.

Kora introduces a multi-tier abstraction model:

- Logical Kafka Clusters (LKC): Users interact with a virtualized cluster that remains consistent regardless of the underlying physical hardware.
- Cellular Architecture: To ensure multi-tenant isolation, Kora utilizes a cellular design where brokers are organized into isolated units. This architecture minimizes the "noisy neighbor" effect and ensures that network-level disruptions in one cell do not propagate to others [6].
- Stateless Proxy Layer: A critical component of Kora's network abstraction is the proxy layer that intercepts client requests. This layer uses Server Name Identification (SNI) to route traffic to the appropriate brokers, allowing the network plane to scale independently of the storage and compute tiers.

By abstracting these low-level details, the Kora engine allows organizations to manage data streaming as a service rather than a collection of network endpoints, facilitating a more resilient and secure integration with AWS and Azure networking fabrics.

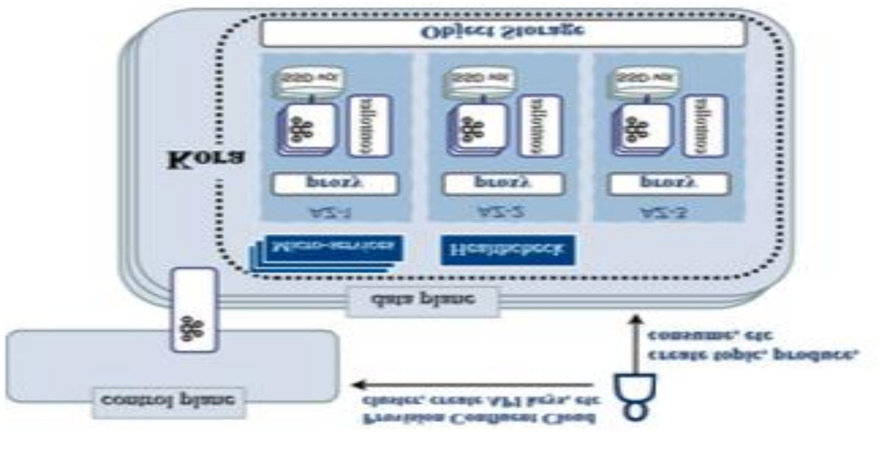

Figure 1. Control Plane and Data Planes in Confluent Cloud, Source [7]

3. Taxonomy of Cloud-Native Connectivity Paradigms

The integration of managed Kafka services into cloud-native architectures requires a classification of connectivity methods based on their logical isolation, routing requirements, and security enforcement. These paradigms are categorized according to their architectural impact on the data plane, progressing from simple public access to complex, centralized routing frameworks.

3.1. Public and Internet-Based Connectivity

Public connectivity is the most basic paradigm, utilizing the public internet for data transport. While packets traverse external networks, security is maintained through Transport

Layer Security (TLS 1.2+) and strict IP allowlisting. This model is typically characterized by a direct client-to-broker connection where the managed service provides a globally unique DNS endpoint. As noted in early cloud frameworks, while this provides ease of access, it lacks the private-path assurance required for high-sensitivity enterprise data [2], [1].

3.2. Peered Connectivity (Bi-directional Layer 3)

Virtual Private Cloud (VPC) and Virtual Network (VNet) peering represent a bi-directional Layer 3 connectivity model. In this paradigm, the managed Kafka network and the consumer network are logically linked to allow direct IP-to-IP communication [1]. While this eliminates internet exposure, it introduces significant "routing-state" complexity. As organizations scale, the requirement for non-overlapping Classless Inter-Domain Routing (CIDR) blocks becomes a primary constraint [5]. Peering remains a viable solution for single-region, low-complexity environments but lacks the abstraction required for modern multi-tenant isolation.

3.3. Endpoint-Based Connectivity (Private Link)

The endpoint-based paradigm, facilitated by AWS PrivateLink and Azure Private Link, shifts the connectivity model from network-level peering to service-level projection. In this model, the Kafka service is presented as a local network interface within the consumer's environment [6]. Connectivity is unidirectional (consumer-to-service), which inherently aligns with Zero-Trust principles by preventing the service provider from initiating connections into the consumer's network [8]. This paradigm effectively decouples the IP address space of the provider from the consumer, allowing for overlapping CIDRs and simplifying the security audit process.

3.4. Hub-and-Spoke (Centralized) Connectivity

For enterprise-scale environments with multiple AWS accounts or Azure subscriptions, a centralized routing model often termed Hub-and-Spoke is utilized. This paradigm leverages a central transit hub (AWS Transit Gateway or Azure Virtual WAN) to aggregate traffic from various "spoke" networks [6]. This architecture allows for a "write-once, connect-anywhere" approach, where the Kafka cluster is connected to the hub, and all spokes gain access via the central routing table. This model provides the highest level of scalability and centralized governance, though it requires advanced Software-Defined Networking (SDN) configurations to manage transitive routing paths [6], [7].

4. Implementation Paradigm: Amazon Web Services (AWS)

The AWS ecosystem offers several architectural patterns for integrating managed Kafka clusters, primarily categorized by their reliance on Layer 3 routing or endpoint-based abstraction. Each paradigm represents a trade-off between configuration complexity and the degree of logical isolation.

4.1. AWS PrivateLink: Interface VPC Endpoints

AWS PrivateLink is the primary mechanism for establishing a "Zero-Trust" data plane. This architecture utilizes Interface VPC Endpoints, which project an Elastic Network Interface (ENI) with a private IP address into the consumer’s VPC subnets [1]. Connectivity is strictly unidirectional: the consumer VPC can initiate requests to the Confluent cluster, but the provider cannot access the consumer's environment.

A critical component of this implementation is DNS resolution. Because Kafka clients require access to specific broker endpoints, AWS PrivateLink relies on Route 53 Private Hosted Zones (PHZ) to map service-specific DNS names (e.g., *.<region>.aws.private.confluent.cloud) to the ENIs [9]. This ensures that TLS-handshakes remain valid while keeping all traffic within the AWS global backbone.

4.2. VPC Peering and Routing Management

VPC Peering provides direct, bi-directional Layer 3 connectivity between the Confluent-managed VPC and the customer VPC. Unlike PrivateLink, peering enables full IP reachability across the network boundary [1]. The primary architectural constraint is the requirement for non-overlapping CIDR blocks.

Implementing peering requires explicit updates to VPC route tables on both sides of the connection. While this model is conceptually simple, it lacks the abstraction of endpoint-based services; any change in the provider's IP space requires a corresponding update in the consumer's routing logic. For this reason, peering is increasingly relegated to legacy or low-complexity environments where transitive routing is not required [5].

4.3. Transit Gateway (TGW) Integration

To facilitate connectivity across multiple AWS accounts and VPCs, the AWS Transit Gateway acts as a regional hub. In this hub-and-spoke model, the Confluent VPC is attached to the TGW as a single spoke, and the TGW manages the propagation of routes to all other attached VPCs [6].

This architecture centralizes network governance, allowing organizations to apply unified security policies through a central inspection VPC if necessary. However, Transit Gateway does not resolve CIDR overlap issues; it necessitates a coordinated IP management strategy across all participating accounts to ensure valid routing paths [10].

4.4. Private Network Interface (PNI): Multi-VPC Attachment

A more recent innovation in the AWS paradigm is the Private Network Interface (PNI). This model leverages the AWS Multi-VPC ENI attachment feature, which allows an ENI residing in the consumer's VPC to be directly attached to a virtual machine in the provider's (Confluent) environment [11].

PNI fundamentally changes the security posture by allowing the consumer to retain full ownership and control of the network interface, including its security groups and IP allocation. This eliminates the need for Network Load Balancers (NLBs) and PrivateLink endpoints, reducing the complexity of the data path while maintaining a private, high-throughput connection [11].

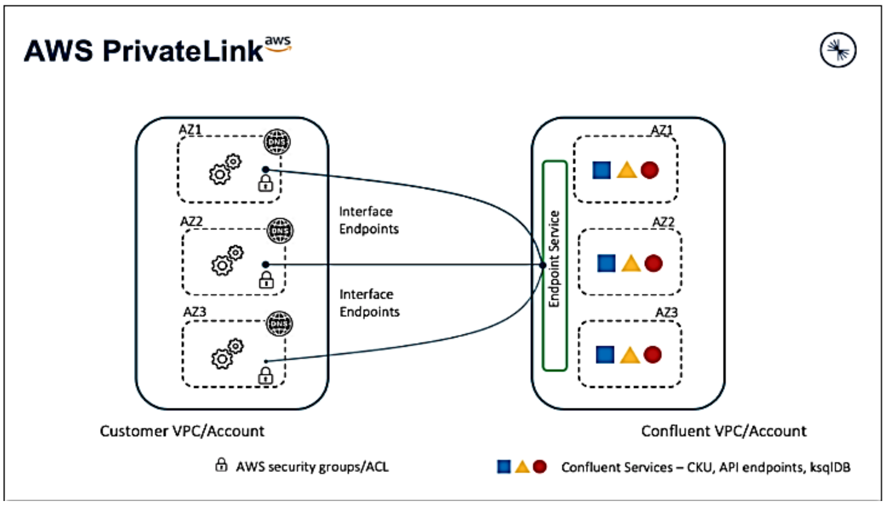

Figure 2. AWS Private Link overview, source: confluent documentation

5. Implementation Paradigm: Microsoft Azure

The integration of Confluent Cloud within the Microsoft Azure ecosystem utilizes a combination of Software-Defined Networking (SDN) and managed service projections. This section analyzes the architectural implementation of private connectivity on Azure, emphasizing the structural differences in resource projection and name resolution compared to other cloud providers.

5.1. Azure Private Link: Service Projection and Private Endpoints

Azure Private Link serves as the primary mechanism for establishing secure, unidirectional connectivity between a consumer's Virtual Network (VNet) and the Confluent-managed environment. Unlike traditional routing models, Private Link projects the Kafka service as a Private Endpoint, which is essentially a specific network interface (NIC) with a private IP address residing within the customer's VNet [12].

On the service provider side (Confluent), the architecture utilizes a Private Link Service associated with a Standard Load Balancer. This configuration ensures that traffic is encapsulated and transmitted over the Microsoft global backbone, completely bypassing the public internet. A distinct characteristic of the Azure implementation is its reliance on CNAME records within Azure Private DNS Zones. To facilitate proper routing, the customer must configure a DNS zone that aliases the cluster's bootstrap and broker endpoints to the local Private Endpoint IP addresses [12]. This approach provides a robust abstraction that prevents data exfiltration by ensuring that the Confluent environment cannot initiate outbound connections to the consumer.

5.2. Virtual Network (VNet) Peering: Regional and Global Logic

VNet Peering offers a Layer 3 connectivity model where the customer VNet and the Confluent VNet are logically fused at the backbone level [13]. Azure distinguishes between two types of peering:

- Regional VNet Peering: Connects VNets within the same Azure region, offering the lowest possible latency and highest bandwidth by leveraging local data center fabric.
- Global VNet Peering: Connects VNets across different Azure regions. While this facilitates cross-regional disaster recovery and data replication, it introduces slightly higher latency as traffic traverses the wider Azure global backbone [13].

Similar to AWS VPC peering, Azure VNet peering requires that participating networks do not have overlapping IP address spaces. However, Azure's peering is frequently used in simpler architectures where the customer requires bi-directional communication, such as when running self-managed connectors or complex monitoring agents that must poll the Kafka brokers directly via IP rather than through an abstracted endpoint [1], [13].

5.3. Azure Virtual WAN and Hub-Spoke Topologies

For enterprises requiring a centralized networking architecture, Azure Virtual WAN provides a managed hub-and-spoke framework. In this paradigm, the Virtual WAN hub acts as a regional or global transit point. The Confluent VNet is attached to the hub as a spoke, and the Virtual WAN service handles the transit routing between other spokes, on-premises branches (via ExpressRoute or VPN), and the Kafka cluster [14].

The integration of Confluent into a Virtual WAN topology simplifies the management of complex "many-to-one" connectivity. Rather than managing individual peering relationships for every application VNet, architects can define routing intent at the hub level. This centralized approach also allows for the insertion of Network Virtual Appliances (NVAs) or Azure Firewall for centralized traffic inspection, though this must be carefully configured to avoid disrupting the low-latency requirements of streaming protocols [6], [14].

5.4. Native Azure Integrations: SSO and Unified Billing Dynamics

The "Azure Native ISV Service" for Confluent introduces a deeper level of integration that extends beyond pure networking into the control plane. This integration utilizes the Microsoft.Confluent resource provider, allowing administrators to manage Confluent resources directly through the Azure Portal and CLI [15].

5.4.1. Identity and Access (SSO)

By integrating with Microsoft Entra ID (formerly Azure AD), the network-level security is augmented by robust identity-based perimeters. Single Sign-On (SSO) ensures that only authenticated principals from the customer's tenant can interact with the Confluent control plane, effectively merging network isolation with identity governance [15].

5.4.2. Networking Impact of Unified Billing

While primarily a commercial integration, unified billing through the Azure Marketplace simplifies the deployment of private networking. It allows for the pre-authorization of the Confluent service to create networking resources within the customer's subscription, reducing the manual steps required to establish Private Link handshakes and subscription approvals [15].

By combining these native services, Azure provides an environment where the Kafka data plane is not merely a "connected" resource but a seamless extension of the customer's own private network fabric.

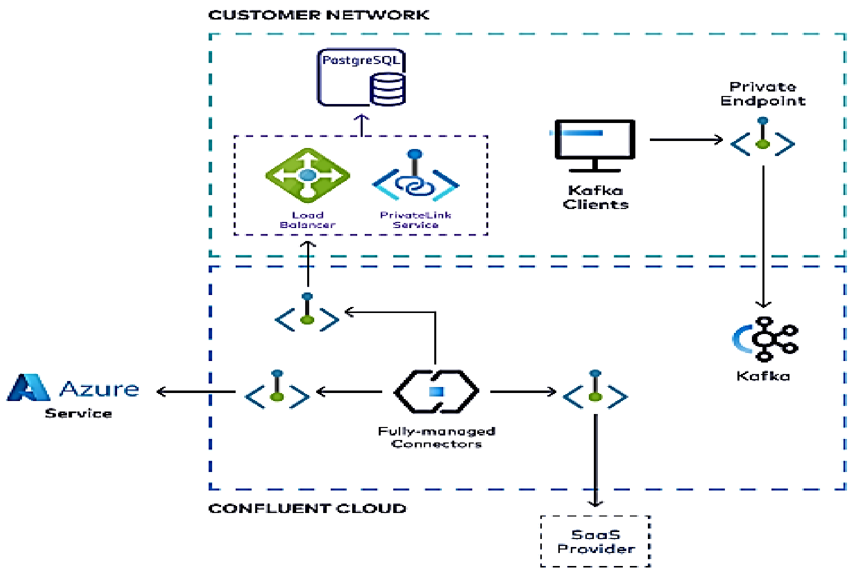

Figure 3. Azure Private link overview, source: Confluent documentation

6. Comparative Analysis

The selection of a networking paradigm for managed Kafka is not merely a choice of connectivity, but a strategic decision impacting security, operational scalability, and administrative overhead. This section provides a detailed comparative synthesis of the architectural implementations on AWS and Azure, evaluating how each CSP addresses the fundamental requirements of isolation and name resolution.

6.1. Security Posture: Attack Surface Reduction

The primary differentiator between the paradigms discussed is the degree of attack surface reduction.

6.1.1. Peering Models (VPC/VNet)

Bi-directional peering creates a Layer 3 "bridge" between the consumer and the provider. While traffic remains on the CSP backbone, this model exposes the entire CIDR range of the Kafka cluster to the consumer's network. From a security perspective, this increases the risk of lateral movement; if a consumer instance is compromised, an attacker could potentially attempt to scan or probe the IP addresses of the Kafka brokers directly [5].

6.1.2. Private Link Models

Both AWS and Azure Private Link represent a significant leap in security by adopting a "Service-Oriented" abstraction. By projecting only a specific interface (ENI or Private Endpoint) into the customer environment, the attack surface is reduced to a single IP address (or a small set of IPs). There is no route to the underlying management infrastructure of the MSP. Furthermore, because these connections are unidirectional, the risk of data exfiltration via the provider's network is mitigated, as the MSP environment cannot initiate an outbound connection into the customer's VPC/VNet [8], [12].

6.2. Connectivity Directionality: Traffic Flow Patterns

A critical distinction in the implementation of these paradigms lies in how traffic directionality is governed across the CSPs:

6.2.1. AWS Implementation

In AWS PrivateLink, the traffic is strictly Inbound-Only from the perspective of the service provider. The Interface VPC Endpoint acts as a gateway for the consumer. If a Kafka client needs to produce data to the cluster, the flow is Consumer $\rightarrow$ NLB $\rightarrow$ Kafka Broker. AWS manages the complexity of mapping these flows across multiple Availability Zones (AZs) using zonal DNS entries [9].

6.2.2. Azure Implementation

Azure follows a similar unidirectional flow but integrates more deeply with the Azure Load Balancer. The Azure Private Link Service (on the Confluent side) maps to the Private Endpoint (on the customer side).[3] Azure's implementation is often cited for its seamless handling of TCP state across regions, though it requires specific attention to "TCP Proxy" behavior which can occasionally obscure the original client IP unless Proxy Protocol is explicitly enabled [12], [15].

6.3. DNS Resolution Strategies and Name Management

DNS resolution is perhaps the most operationally complex aspect of managed Kafka networking, as Kafka clients rely on the advertised.listeners property to identify brokers.[4]

6.3.1. AWS Route 53 Private Hosted Zones (PHZ)

AWS requires the creation of a PHZ for each cluster. This zone must contain "A" records that point the cluster's wildcard DNS (e.g., *.abc.aws.confluent.cloud) to the VPC Endpoint IPs. The challenge in AWS is maintaining these records if the endpoint is recreated or if cross-region connectivity is required, necessitating the use of Route 53 Resolvers to bridge DNS queries between VPCs [9].

6.3.2. Azure Private DNS Zones

Azure simplifies this through a centralized Private DNS Zone that can be "linked" to multiple VNets.[5] When a Private Endpoint is created, Azure can automatically register the CNAME records in the linked DNS zone. However, if the customer is using an on-premises DNS server, Azure requires a Private DNS Resolver (formerly a Conditional Forwarder) to ensure that on-premises clients can resolve the internal Azure IP addresses rather than the public ones [12], [14].

6.4. Scalability and Management Complexity

The operational burden of these paradigms scales differently as the organization grows:

6.4.1. The "Peering Mesh" Problem

As the number of consumer VPCs/VNets increases, managing individual peering connections becomes a logarithmic difficulty. Each peer requires distinct route table entries and strict CIDR management to avoid overlaps. This "mesh" is notoriously difficult to audit and maintain [1].

6.4.2. Private Link Scalability

Private Link avoids the mesh problem by allowing multiple VPCs/VNets to connect to the same service without needing to know about each other. It effectively enables a "Consumer-Producer" model where the Kafka cluster is treated as a utility.

6.4.3. Hub-and-Spoke Governance

Both AWS and Azure offer centralized hubs (Transit Gateway and Virtual WAN) to mitigate complexity.[6] AWS TGW provides more granular control over route propagation but requires more manual configuration.[7] Azure Virtual WAN is a more "opinionated" service that automates much of the hub-spoke routing, making it easier to deploy but offering less flexibility for non-standard routing requirements [6], [14].

6.5. Synthesis of Implementation Criteria

The following table synthesizes the core findings of this comparative study, providing a reference for architectural selection based on CSP-specific technologies.

Table 1. Private Service Connectivity in Aws and Azure: Architectural Comparison

Criteria	AWS Implementation	Azure Implementation
Primary Private Technology	AWS PrivateLink (Interface VPC Endpoints)	Azure Private Link (Private Endpoints)
Connectivity Logic	Unidirectional; requires NLB on provider side	Unidirectional; utilizes Private Link Service
Primary Routing Hub	AWS Transit Gateway	Azure Virtual WAN
DNS Management	Route 53 / Private Hosted Zones (PHZ)	Azure Private DNS Zones (CNAME based)
IP Conflict Resolution	Inherently handled via Endpoint mapping	Inherently handled via Private Endpoint projection
Isolation Level	High: Endpoint-specific; no Layer 3 routing	High: VNet-local IP; no transitive peering risk
Cross-Account Access	Managed via Resource Shares (RAM)	Managed via Subscription Approvals
On-Premises Access	Via DX/VPN + Route 53 Resolver	Via ExpressRoute/VPN + DNS Resolver

6.6. Conclusion of Analysis

While both AWS and Azure offer robust private networking for Confluent Cloud, the choice often depends on the existing DNS architecture and the scale of the deployment. AWS PrivateLink is slightly more complex in its DNS requirements but offers highly granular control through Route 53. Azure Private Link provides a more integrated experience for organizations already utilizing Azure-native security services like Entra ID and Azure Firewall. In both cases, the move away from Layer 3 peering toward service-level endpoints represents the current gold standard for secure, cloud-native event streaming [8], [15].

7. Architectural Recommendations & Use Cases

In the deployment of managed Kafka services, the selection of a networking paradigm is often dictated by the existing enterprise topology rather than pure performance metrics. This section provides a detailed analysis of the criteria for prioritizing specific connectivity models and explores the architectural requirements for extending cloud-native event streams to on-premises environments.

7.1. Strategic Prioritization: Peering vs. Private Link

While Private Link is frequently marketed as the default choice for security-conscious organizations, Peering remains a vital tool for specific architectural requirements. The decision to prioritize one over the other is typically based on the following three criteria:

7.1.1. Bi-directional Traffic and Managed Connectors

A primary advantage of VPC/VNet Peering is its native support for bi-directional communication. In scenarios where Confluent Cloud hosts Fully Managed Connectors that must ingest data from sources (e.g., RDS, CosmosDB) or sink data into destinations residing in the customer's private network, peering provides the necessary Layer 3 reachability [9], [13]. Private Link, by contrast, is inherently unidirectional; it allows the consumer to reach the cluster but does not natively allow the cluster to reach out to the consumer’s resources without additional complex "Outbound Private Link" configurations [15].

7.1.2. Addressing IP Constraints

The defining technical constraint for peering is the requirement for mutually exclusive CIDR blocks. In large enterprises with highly fragmented IP spaces, Private Link is prioritized because it leverages Network Address Translation (NAT) via the provider's load balancer, effectively hiding the provider's IP space from the consumer and eliminating the risk of routing conflicts [12], [16].

7.1.3. Cost and Administrative Simplicity

Peering is often prioritized for high-throughput, single-region workloads where administrative simplicity is preferred over maximum isolation. Because peering does not incur the data processing fees associated with AWS PrivateLink endpoints or Azure Private Link services, it offers a more predictable cost structure for organizations that can manage the security requirements through fine-grained Security Group and Network ACL (NACL) rules [2], [17].

7.2. Designing for Hybrid-Cloud: The Extension of the Data Plane

The integration of on-premises data centers with Confluent Cloud requires a multi-layered connectivity strategy that bridges traditional hardware-based networking with cloud-native Software Defined Networking (SDN).

Physical Transport: DirectConnect and ExpressRoute: The foundation of hybrid connectivity is the physical circuit. For AWS environments, DirectConnect (DX) provides a dedicated network link, while Azure utilizes ExpressRoute. These circuits terminate at a virtual gateway (VGW) or an ExpressRoute Gateway within the customer’s cloud hub. The primary architectural challenge is that cloud-native peering and endpoint services are non-transitive by default [1], [14]. For example, a client in an on-premises data center cannot automatically "hop" through a customer VPC to reach a peered Confluent VPC.

The "Transit Hub" Pattern To overcome non-transitivity, organizations utilize a Hub-and-Spoke architecture.

- In AWS: The on-premises network connects to an AWS Transit Gateway (TGW). The Confluent VPC is attached to this TGW. By managing the TGW route tables, the on-premises CIDRs are advertised to Confluent, and the Confluent CIDRs are advertised to the on-premises routers via BGP [6].
- In Azure: The Azure Virtual WAN hub acts as the transit point. ExpressRoute circuits and VNet peerings are centralized at the hub, allowing the Virtual WAN to manage the routing propagation across the entire hybrid fabric [14].

Hybrid DNS Resolution and Forwarding The most common point of failure in hybrid Kafka deployments is DNS resolution. On-premises clients must be able to resolve the private IP addresses of the Confluent brokers.

- AWS Route 53 Resolvers: Organizations deploy Inbound Endpoints in their VPC. The on-premises DNS server is configured with a conditional forwarder that sends all queries for *.confluent.cloud to these endpoint IPs [9].
- Azure DNS Private Resolvers: Similar to AWS, Azure provides a managed service that accepts DNS queries from on-premises sources and resolves them against the Azure Private DNS Zones associated with the Private Link endpoints [12].

The Proxy/Bridge Architecture (The "Double Hop") In environments where Transit Gateways or Virtual WANs are not feasible due to policy constraints, a TCP Proxy (e.g., NGINX or HAProxy) is deployed within a "DMZ" VPC/VNet. This proxy has one interface connected to the on-premises circuit and another interface connected to the Confluent cluster (via Peering or Private Link). The on-premises clients point to the proxy, which then forwards the Kafka protocol traffic to the managed service [18]. This approach provides a clear demarcation point for security auditing but introduces an additional management layer for the proxy fleet.

8. Conclusion

The transition to managed event streaming via Confluent Cloud on AWS and Azure has fundamentally altered the networking responsibilities of the cloud architect. This study has demonstrated that while the operational burden of managing Kafka brokers has decreased, the complexity of securing the data plane has migrated to the network layer.

By classifying the taxonomy of cloud-native connectivity, we have identified a clear evolution from Layer 3 peering models toward service-level endpoint projections. AWS PrivateLink and Azure Private Link represent the current zenith of secure connectivity, offering logical isolation that aligns with Zero-Trust mandates. However, our analysis reveals that these benefits must be balanced against the administrative overhead of DNS management and the non-transitive nature of cloud routing, particularly in hybrid-cloud scenarios.

Ultimately, the choice of a networking paradigm whether it be a hub-and-spoke TGW implementation on AWS or a Private Link integration on Azure must be informed by the organization's existing IP management strategy and its requirements for bi-directional data flow. As managed services continue to abstract deeper layers of the infrastructure, the ability to orchestrate secure, private, and scalable network paths will remain a cornerstone of resilient data-in-motion architectures.

References

[1] J. Krepset al., "Kafka: A Distributed Messaging System for Log Processing," *Proceedings of the NetDB*, vol. 11, pp. 1-7, 2011.

[2] M. Armbrust et al., "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[3] B. Furht and A. Escalante, *Handbook of Cloud Computing*. Springer Science & Business Media, 2010.

[4] R. Buyyaet al., *Mastering Cloud Computing: Foundations and Applications Programming*. McGraw-Hill Education, 2013.

[5] C. J. Anderson, "The Evolution of Cloud Networking Architectures," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 115-128, 2015.

[6] S. D. A. Shah et al., "Cloud-Native Network Slicing using Software Defined Networking based Multi-Access Edge Computing: A Survey," *IEEE Access*, vol. 9, pp. 29846-29871, 2021.

[7] A. Povzner et al., "Kora: A Cloud-Native Event Streaming Platform for Kafka," *Proceedings of the VLDB Endowment*, vol. 16, no. 12, pp. 3822-3835, 2023.

[8] A. Tundo et al., "Monitoring Probe Deployment Patterns for Cloud-Native Applications: Definition and Empirical Assessment," *IEEE Transactions on Services Computing*, vol. 17, no. 4, pp. 1636-1650, July/Aug. 2024.

[9] AWS Whitepaper, "Building a Scalable and Secure Multi-VPC AWS Network Infrastructure," *Amazon Web Services Documentation*, 2025. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/building-scalable-secure-multi-vpc-network-infrastructure/welcome.html [Accessed: Feb 6, 2024].

[10] AWS Documentation, "AWS Transit Gateway design best practices," *Amazon VPC Guide*, 2025. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/tgw/tgw-best-design-practices.html [Accessed: Feb 5, 2026]

[11] Confluent Engineering, "Introducing Private Network Interface (PNI) on AWS," *Confluent Technical Publications*, 2025. [Online]. Available: https://www.confluent.io/blog/introducing-private-network-interface/ [Accessed: Feb 5, 2026]

[12] Microsoft, "Azure Private Link: Private Connectivity for Azure Services," Azure documentation , 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/private-link/private-link-overview [Accessed: Feb 5, 2026]

[13] Microsoft, "Global Virtual Network Peering and Regional Backbone Performance," *Azure Networking Technical Standards*, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview [Accessed: Feb 5, 2026]

[14] Microsoft, "Azure Virtual WAN: A Unified Global Transit Architecture for Cloud-Native Workloads," *Cloud Design Patterns*, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-wan/virtual-wan-global-transit-network-architecture [Accessed: Feb 5, 2026]

[15] Microsoft and Confluent, "Azure Native ISV Service for Apache Kafka: Technical Integration and Management," *Joint Technical Overview*, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/partner-solutions/apache-kafka-confluent-cloud/overview [Accessed: Feb 5, 2026]

[16] G. Pallis, "Cloud Computing: The New Frontier of Internet Computing," *IEEE Internet Computing*, vol. 14, no. 5, pp. 70-73, Sept.-Oct. 2010.

[17] K. Hwang and G. C. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Morgan Kaufmann, 2013.

[18] N. G. S. Dharmasiri et al., "Software-Defined Networking for Hybrid Cloud Architectures: A Review," *IEEE Access*, vol. 12, pp. 4500-4525, 2024.

[19] Vamshidhar Reddy Vemula.(2023).Multi-Cloud Security Orchestration Using Deep Reinforcement Learning.