IT Project Guidance - On Multiple Clouds

Version: 0.1

Purpose

This document provides strategic and technical guidance for organisations considering, or already operating, services across multiple cloud providers. It examines the motivations, architectural implications, and operational risks of multi-cloud environments, particularly within public sector and enterprise settings. Rather than advocating for or against the approach, this guidance aims to equip decision-makers, architects, and delivery teams with a balanced understanding of trade-offs and long-term impacts.

Synopsis

Multi-cloud operation is often positioned as a safeguard against vendor lock-in, an enabler of service redundancy, or a means of leveraging best-of-breed capabilities across platforms such as AWS and Azure. However, it introduces significant complexity across governance, security, networking, data movement, monitoring, identity management, and skill development. While developers may favour familiar platforms, enterprise operations must account for supportability, cross-cloud interoperability, and the hidden costs of fragmentation. This guidance explores when and how a multi-cloud approach might be justified, what architectural patterns and controls must be put in place, and where organisations risk undermining their strategic posture by overreaching.

Contents

Purpose and Audience

The purpose of this document is to clarify the rationale, risks, and technical requirements associated with operating a system across multiple public cloud platforms. It is intended to inform architectural decisions, procurement strategies, and operational planning where two or more cloud environments are being considered concurrently. The guidance applies equally to greenfield and brownfield projects, and to those proposing cloud-neutral patterns as well as those pursuing explicit multi-cloud delivery.

This document is written for Enterprise Architects, Technical Leads, Programme Managers, Security and Operations teams, and Governance stakeholders. It may also inform strategic investment decisions by senior leadership, particularly where cloud posture intersects with national capability, sector alignment, and service resilience planning.

Scope

his guidance focuses on the delivery, operation, and governance of systems spanning two or more major public cloud platforms—typically AWS and Azure—used in parallel. It addresses core infrastructure, application services, data handling, monitoring, identity, and cost management implications of multi-cloud design.

It does not cover multi-region strategies within a single provider, hybrid on-prem/cloud patterns, or legacy cloud migrations unless they intersect directly with multi-cloud concerns. The scope includes both shared service impacts (such as monitoring, IAM, and audit tooling) and system-specific challenges (such as cross-cloud database access or data replication).

Examples and considerations are framed in the context of public sector systems in Aotearoa New Zealand, but the principles and risks generalise to other jurisdictions and organisational types.

Background

Cloud adoption has matured to the point where most large systems rely on services from a major cloud vendor. Operating across two cloud platforms is now proposed more often, typically motivated by perceived risks of vendor lock-in, delivery team preferences, access to specialised services, or a desire for increased resilience. However, these rationales often obscure the significant costs, technical complexity, and governance overhead introduced by multi-cloud strategies.

Historically, systems were built within a single hosting environment—physical, virtualised, or single-cloud—enabling standardised tooling, security, and operations. Introducing a second cloud breaks this cohesion. Teams must duplicate patterns or adopt abstractions that can themselves become forms of lock-in. In most organisations, including public sector contexts, these challenges are amplified by fragmented capability, inconsistent practices, and a shortage of cloud-literate staff.

While developers may prefer a particular platform, enterprise systems must be supportable, observable, and governable across their full lifecycle. Multi-cloud introduces cross-platform dependencies, increased latency, reduced clarity, and higher costs—often undermining the very flexibility or resilience it was meant to provide.

Apparent benefits like improved availability or strategic independence are often better achieved within a well-architected single-cloud deployment. Unique services in one platform can create long-term risk if they form part of the critical path. Edge services such as Cloudflare may offer helpful intermediaries, but they mask—not remove—the complexities of multi-cloud operation.

This document outlines the issues clearly, to help organisations make deliberate, sustainable decisions about their cloud models.

Justifications for Multi-Cloud Operation

Multi-cloud architecture is often proposed with strong intentions but loosely tested assumptions. Common justifications include risk mitigation, price optimisation, service availability, team preference, and perceived strategic independence. Each has merit in concept, but each requires closer examination when applied to operational systems at scale.

Vendor Lock-in

Concern over over-reliance on a single provider is the most cited justification. However, the real lock-in rarely comes from infrastructure—it stems from application-level decisions: proprietary service dependencies, custom storage formats, or tightly bound identity models. These should be addressed through careful architecture, not a second cloud.

Resilience

Cross-cloud failover is often promoted as a resilience benefit. In practice, very few systems are truly engineered to switch providers during a failure. Core dependencies such as DNS, storage, or identity are platform-specific. High-availability is better achieved through multi-region, single-cloud designs.

Features Availability

Occasionally a specific service only exists in one provider. This may appear to justify adding a second platform, but such features often narrow rather than extend options. Their use should be carefully evaluated: if essential, contain their impact; if not, prefer alternatives. Finally, patience for parity is a valuable strategic objective.

Cost

The idea of shifting workloads for cost reasons is rarely practical. Real-time pricing differences are minor, and the ability to shift demand presumes a level of operational maturity few teams possess. The duplicated tooling and governance required for multi-cloud usually offsets any benefit.

Preference

Delivery teams may have experience with a specific provider. But preference is not strategy. Enterprise systems must consider support models, monitoring, security, and procurement. Organisations should invest in training rather than fragment their architecture to accommodate familiarity.

Each of these justifications contains a seed of logic. But none alone warrants multi-cloud operation. They must be tested not in isolation but as part of an integrated operating model, with full awareness of what they displace: simplicity, observability, cohesion, and institutional focus.

Platform Comparison Context

Context Both AWS and Azure provide comprehensive, mature cloud ecosystems. While some features differ in implementation, core capabilities—compute, storage, identity, monitoring, networking—are comparable.

The critical difference lies not in the capabilities themselves, but in how they integrate with the surrounding ecosystem.

Azure offers tighter integration across the broader Microsoft ecosystem—Active Directory, Office 365, GitHub, Sentinel, and compliance tooling—creating a more cohesive experience for teams beyond core development. AWS remains highly configurable and modular but assumes more bespoke integration, particularly across logging, monitoring, and identity boundaries. These differences become more important as system complexity grows and more stakeholders interact with the platform.

Most multi-cloud tooling claims neutrality—Terraform, Kubernetes, and OIDC-based identity among them—but provider-specific modules, naming conventions, and undocumented behaviours often undermine these claims. **Reusability across clouds aspiration, limited in practice unless the architecture is intentionally abstracted from the outset.**

In short, both platforms are capable, but they differ in their surrounding ecosystems and ease of organisational fit. Basing a strategy on perceived service gaps, rather than on team alignment and supportability, tends to overestimate short-term advantage and underestimate the cost of operational divergence.

Common Mistakes and False Assumptions

Multi-cloud proposals often carry implicit assumptions that appear sound in isolation but unravel when extended into operating models, organisational capacity, or real-world failure scenarios. The following are among the most frequent and consequential.

Service Parity Assumed

It is often assumed that because both AWS and Azure offer services named similarly—databases, queues, container hosts—they are functionally equivalent and interchangeable. In practice, they differ in behaviour, configuration defaults, resilience models, and how they integrate with surrounding services like IAM, monitoring, or VNet/VPC policies.

This parity myth is amplified by deployment tooling such as **Terraform**, which abstracts away platform-specific details behind a uniform configuration layer. While useful, this abstraction conceals the reality that modules must still be customised for each provider, and that resource behaviour may diverge in subtle but critical ways. The illusion of symmetry at deployment time often breaks during operations, monitoring, or security configuration—where the actual platform-specific behaviour becomes unavoidable.

Cloud Means Value

There is a persistent belief that simply moving to the cloud provides strategic benefit. In truth, lifting existing systems into IaaS layers of the cloud—especially with legacy patterns like active/active or mirrored environments—often increases costs and undermines agility. Cloud advantage is not found in virtual servers, but in the redesign of systems to leverage hyperscale capabilities: autoscaling containers, serverless compute, managed event streams, and consumption-priced services. Without rearchitecture, the cloud behaves more like expensive rented tin.

Overestimating Resilience

Multi-cloud is frequently promoted as a resilience measure. But this misjudges both the nature of outages and the effort required to achieve live failover. Most outages that affect a provider also disrupt core services such as DNS, identity brokers, or authentication paths. Running a service across two clouds doesn’t help if every user's browser, login, or dependency is also affected. True cross-cloud failover demands duplicated state, practice under load, and isolation from ecosystem-level failures—an uncommon and high-effort pattern that rarely justifies itself outside regulated or national-scale systems.

Cost Competition Overestimated

Organisations sometimes assume that using multiple cloud providers will create pricing leverage, or allow workloads to be shifted to the cheaper option in real time. In practice, cloud pricing converges rapidly—usually faster than internal teams can reconfigure pipelines, change tooling, or migrate services. Engineering for fluid workload migration is complex, and the marginal gains in unit price are often overwhelmed by the cost of making such migration possible.

Organisational Readiness Ignored

Developer preference often drives cloud selection, but supportability is a broader concern. Each platform brings its own deployment tooling, security model, incident response process, training needs, and cost telemetry. Running both doubles this footprint. Security, operations, monitoring, and procurement functions must be equipped for both environments—not just in theory, but in daily responsibility. Without this investment, multi-cloud becomes a series of blind spots, not a strategy.

Risk Framing Inverted

Sometimes the mere presence of a feature in one cloud—but not the other—is treated as justification for expanding footprint. This framing assumes that absence is a capability gap. But unique features are often immature, short-lived, or entangling. Their use should be evaluated first as a potential risk to portability, rather than a benefit. The opportunity cost of becoming anchored to a non-portable capability is rarely factored into early-stage decisions.

Scale of Problem Misjudged

Perhaps the most foundational misstep is misunderstanding what failure looks like at cloud scale. Architecting for intra-cloud region loss makes sense. Architecting for whole-of-provider failure, as though the rest of the internet or dependency stack will remain intact, usually does not. Outages of this scale disable large swaths of the digital ecosystem. Building for availability in that context is expensive, brittle, and unlikely to provide end-user continuity when it matters most.

Hidden Complexity and Cost

Multi-cloud strategies often appear cost-neutral when comparing headline infrastructure items. But true cost emerges in duplicated services, unaligned processes, degraded visibility, and the complexity of coordinating what surrounds the workload. These impacts are rarely visible in early diagrams but become persistent, systemic challenges over time.

Data Movement Is Cost Movement

Moving data between clouds introduces not just egress charges, but latency, coordination lag, and operational drag. Legacy systems in particular tend toward batch transfer, and few architectures are optimised for deltas or streaming updates. Once deployed, these patterns are rarely refactored—creating a long-term data handling burden that inflates both cost and fragility.

Cost Visibility Fragments

Each cloud platform provides its own billing tools, telemetry models, and usage reporting mechanisms. Without a third-party FinOps platform or dedicated internal tooling, multi-cloud spend becomes opaque. Comparing prices, allocating costs to services, and forecasting accurately across clouds is slow, error-prone, and frequently disputed. This undermines confidence in budgeting and optimisation efforts.

Shared Services Double

Monitoring, backup, alerting, IAM policies, deployment automation, and secret stores all need to be implemented twice or abstracted through additional tooling. Even where control planes are consolidated, the underlying implementations diverge. This increases training requirements, policy drift, and the likelihood of silent misconfiguration. The result is duplicated operational effort for diminished clarity.

Shared Ingress Infrastructure: Hidden Multipliers

Routing, caching, firewalling, and API brokering across clouds is rarely simple. To bridge cloud boundaries, organisations often introduce global edge services such as Cloudflare, Azure Front Door, AWS Global Accelerator, or Akamai. These promise centralised WAF, TLS termination, latency optimisation, and global failover.

But each layer introduces cost, state, and risk. Configurations include DNS-level failover, WAF rule synchronisation, TLS and certificate management, traffic steering, header rewriting, and gateway-level retries. Session handling, identity headers, and IP preservation are all additional concerns. These are not superficial tasks—they form a second layer of architecture.

While technically elegant, they concentrate failure at the entry point. Multi-cloud systems with shared ingress are exposed to the uptime of every element in the ingress stack, and debugging across vendor lines is often slow and unclear. Worse, these layers often exist only because earlier architectural decisions were made without unified ingress planning—making integration necessary rather than intentional.

Availability Compounds Downward

Each provider advertises 99.99% availability for individual services. But when systems depend on a chain of services—cloud platforms, edge layers, DNS, identity brokers, and regional endpoints—the practical availability is far lower. Multi-cloud systems multiply these dependencies. Outages affecting authentication, routing, or platform APIs cascade across environments, and triage across multiple support teams becomes a critical delay. High availability must be designed holistically, not presumed from the sum of its parts.

Elasticity Without Coordination Is Waste

Both AWS and Azure offer hyper scaling and autoscaling services, but when used independently, these can compete or overprovision. Without shared traffic intelligence or workload partitioning, each environment may scale independently in response to partial demand signals. The result is increased spend with limited actual gain in responsiveness or resilience. Elasticity is not free; it must be orchestrated.

Operational and Organisational Impact

While cloud architecture may appear service-led, what determines success is often the organisation’s ability to operate, support, and govern what is built. In a multi-cloud environment, every core function—provisioning, monitoring, alerting, incident response, identity, policy enforcement, backups, and deployment—must be established and maintained across both platforms. The overhead is rarely visible in architecture diagrams but emerges rapidly in practice.

Teams must be trained not only in how to deploy workloads, but how to interpret alerts, escalate incidents, perform root cause analysis, and restore services—twice. Backup policies, logging formats, IAM roles, and even terminology diverge. Security teams must duplicate effort across threat models, audit trails, and access reviews. Any shared service—API gateway, SIEM, certificate authority—must bridge both platforms or be split, increasing risk and cost.

Even when technically feasible, the process burden is substantial. Change management, role onboarding, policy testing, and breach containment all become more complex. This is not just about running services—it is about operating safely and predictably, under pressure, with clarity. In that context, multi-cloud often spreads resources too thin, leaving organisations with broad exposure but shallow control.

Recommendations and Strategic Guidance

1. Avoid Multi-Cloud Unless You Have a Clear, Justified Purpose

Multi-cloud introduces significant complexity with little automatic advantage. There is no guaranteed strategic benefit—only increased risk and cost. Stakeholder enthusiasm or abstract ideas of flexibility are not sufficient justification. Independent sources, including Gartner and industry reports, show that:

- Nearly **70% of organisations experience budget overruns on cloud**—averaging **15% excess spend**—when managing multi-vendor environments corestack.io+3en.wikipedia.org+3en.wikipedia.org+3.
- Governance-heavy multi-cloud—especially without mature cost control—can see operational costs increase by up to **40%** corestack.io.

Unless you have a concrete, measurable reason, the prudent default is to optimise within a single cloud environment.

2. If Proceeding with Considering Multi-Cloud

Start with Purpose, Not Platform

The objective is not to operate in two clouds, or even to “move to the cloud.” The goal is to access capabilities that are not feasible on-premises—elastic scaling, managed services, cost-aligned compute, integrated observability, and platform-supported automation. Achieving this requires rearchitecting. Simply replicating legacy IaaS patterns in the cloud—especially across two platforms—adds cost and complexity without delivering the expected benefits.

Accept That Divergence Is Expensive

Where multi-cloud becomes necessary, treat it as a divergence to be managed—not as a symmetry to be maintained. Choose one cloud as the operational anchor. The second may support a narrow, well-scoped purpose, but should not be allowed to drift into parity. Attempting to mirror deployments creates a dual burden of support, monitoring, and lifecycle management that few organisations can sustain effectively.

Partition by Function, Not Redundancy

If multi-cloud is required, partition by workload or functional boundary. Assign distinct responsibilities to each platform—such as analytics in one, and identity or transactional services in another. Avoid spreading a single system across clouds. Active/active or active/passive cross-cloud failover should only be attempted where explicitly justified and routinely tested.

Recognise Shared Services as Anchors

Shared services—identity, WAF, DNS, API gateways, telemetry—are not neutral. Their placement defines the system's centre of gravity. Once these are anchored in one cloud, the rest of the architecture is operationally tied to it. Attempting to virtualise or bridge them across clouds typically introduces fragility, latency, and support burden that outweighs any architectural elegance.

Base Decisions on Operational Capacity

Even where cross-cloud deployment is technically possible, sustainable operation depends on people, process, and institutional maturity. If an organisation cannot reliably monitor, alert, support, secure, and cost-manage systems across both platforms, then it is not ready for multi-cloud. Most strategic risk lies not in choosing the wrong platform—but in overestimating the organisation's readiness to run two.

Use Abstraction Judiciously

Tooling such as Terraform, Kubernetes, and OIDC can support abstraction, but they do not erase platform differences. These tools delay the moment of divergence—they do not prevent it. When used, they must be governed with clear domain boundaries, tested in both environments, and supported by teams who understand their limits.

Apply Strategic Restraint

Unless driven by clear external requirements—regulatory isolation, sovereign tenancy, vendor-specific capabilities not replicated elsewhere—multi-cloud should be deferred. Most perceived advantages, including cost flexibility, are better addressed through disciplined single-cloud architecture, mature contracting, or regional distribution. Multi-cloud is rarely the most effective starting point.

3. If Proceeding with Multi-Cloud: Define Conditions and Controls

Where multi-cloud is already in place—or where there is no practical path to avoid it—the following conditions should be applied rigorously to reduce long-term complexity and cost.

Establish Primary/Secondary Roles

Choose a lead cloud platform and assign it operational primacy. Any second cloud should support a limited scope, with explicitly defined ownership boundaries and service expectations.

Partition Workloads, Avoid Duplication

Separate workloads by domain. Assign one platform to analytics, another to transactional processing, for example. Avoid architectural symmetry unless systems are stateless and built for it. Cross-cloud failover should not be assumed—it must be tested.

Define Shared Service Placement Upfront

Place shared services with intent: API gateways, DNS, identity providers, WAF, and monitoring infrastructure should not float between clouds. Their placement determines support structure, network latency, and platform anchoring. Choose once and commit.

Plan for Data Locality and Movement

Design for known data gravity. Avoid bi-directional syncing or fragmented pipelines. Accept that duplicating storage, analytics, or backup across clouds is costly. Favour upstream/downstream separation or workload segmentation.

Enforce Cost Tracking and Governance

Implement FinOps discipline across clouds. Require cost tagging, enforce budget alarms, monitor egress, and perform cross-platform reporting as a first-order capability—not a post-implementation patch.

Rehearse Incidents, Not Just Deployments

Test failure paths between clouds. Ensure failover and incident escalation practices are documented, understood, and supported by real diagnostics. Assumed resilience without practiced response is unreliable.

Review Cloud-Native Alternatives First

Before defaulting to multi-cloud, assess whether a single provider already meets the need through newer services or regions. Cloud feature gaps often close within 12 months—usually faster than enterprise adoption cycles.

4. If having Proceeded

For organisations that have already adopted multi-cloud, the question is no longer *should we do it*, but *is it delivering discernible value?* Multi-cloud, once entered, tends to sprawl. Its complexity rarely unwinds itself. But most of its costs remain avoidable, if deliberately addressed.

Assess Current State of Value

Review what multi-cloud is enabling. Is it delivering capabilities otherwise unavailable? Have the expected benefits been realised? Are the workloads still clearly partitioned, or has ad-hoc drift occurred? If the answer is unclear, the structure is likely no longer strategic.

Roll Back Where Possible

Where duplication has occurred without a compelling benefit, consolidating services back to a single cloud often produces measurable improvements:

- Reduced tooling, IAM, logging, and alerting duplication
- Simpler cost tracking and FinOps oversight
- Fewer integration boundaries to secure, monitor, and govern
- Greater institutional fluency with one platform, not two

Rollback should not be seen as failure—it is evidence of maturity. Consolidation reduces long-term risk and realigns the system with sustainable delivery.

Contain Where Retained

If one cloud is still required for a legitimate domain—e.g. AI workloads, regulatory data hosting, or legacy third-party services—then restructure the architecture around bounded functional separation:

- Treat the retained cloud as a specialised domain, not a peer
- Avoid reintroducing shared ingress, WAF, identity, or telemetry layers
- Accept latency or cross-cloud access as trade-offs, not opportunities for re-coupling
- Ensure ownership is clearly allocated and visibility is maintained

Avoid Further Generalisation

Do not let retained partial multi-cloud presence become the basis for renewed platform symmetry. Re-expansion is easy to justify technically, but almost never pays for itself operationally. Treat any remaining dual-cloud architecture as an exception—not a direction of travel.

Closing Summary

Multi-cloud operation is not inherently flawed—but it is rarely the right default. It should be treated as a strategic exception, not a fashionable best practice. Without compelling external drivers and a high degree of organisational maturity, it introduces duplication, obscures cost, and creates coordination burdens that undermine delivery.

For most systems, architectural discipline within a single cloud remains the more effective and sustainable path.

Appendices

Appendix A - Document Information

Authors & Collaborators

- Sky Sigal, Solution Architect

Versions

0.1 Initial Draft

Images

Tables

References

There are no sources in the current document.

Review Distribution

The document was distributed for review as below:

Identity	Notes
Duncan Watson, Enterprise Architect	
Diane Simpson TPR Project and Product	

Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

Structure

Where possible, the document structure is guided by either ISO-* standards or best practice.

Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

Acronyms

API : Application Programming Interface.

DDD

: Domain Driven Design

GUI: Graphical User Interface. A form of UI.

ICT: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

IT : acronym for Information, using Technology to automate and facilitate its management.

UI : User Interface. Contrast with API.

Terms

Refer to the project’s Glossary.

Application Programming Interface : an Interface provided for other systems to invoke (as opposed to User Interfaces).

Capability : a capability is what an organisation or system must be able to achieve to meet its goals. Each capability belongs to a domain and is realised through one or more functions that, together, deliver the intended outcome within that area of concern.

Domain : a domain is a defined area of knowledge, responsibility, or activity within an organisation or system. It groups related capabilities, entities, and functions that collectively serve a common purpose. Each capability belongs to a domain, and each function operates within one.

Entity : an entity is a core object of interest within a domain, usually representing a person, place, thing, or event that holds information and can change over time, such as a Student, School, or Enrolment.

Function : a function is a specific task or operation performed by a system, process, or person. Functions work together to enable a capability to be carried out. Each function operates within a domain and supports the delivery of one or more capabilities.

Person : a physical person, who has one or more Personas. Not necessarily a system User.

Persona : a facet that a Person presents to a Group of some kind.

Quality : a quality is a measurable or observable attribute of a system or outcome that indicates how well it meets expectations. Examples include reliability, usability, and performance. Refer to the ISO-25000 SQuaRE series of standards.

User : a human user of a system via its UIs.

User Interface : a system interface intended for use by system users. Most computer system UIs are Graphics User Interfaces (GUI) or Text/Console User Interfaces (TUI).

Appendix B – Supporting Observations and Industry References

This appendix summarises key industry findings that reinforce the analysis presented in this guidance. These references provide external validation of the cost, complexity, and governance risks associated with multi-cloud strategies.

A1. Cost Overruns and Operational Impact

“Governance-heavy multi-cloud—especially without mature cost control—can see operational costs increase by up to 40%.”
— CoreStack (citing Gartner), Cloud Governance Trends Report
[Source: https://www.corestack.io/resources/cloud-governance-trends-report/]

“Nearly 70% of cloud projects exceed their budget, primarily due to underestimating ongoing operational costs and poor cost visibility across platforms.”
— Gartner (as cited in multiple analyst summaries)
[Secondary reference: https://en.wikipedia.org/wiki/Multicloud]

“Public cloud costs continue to exceed budget, with organisations reporting an average of 15% overspend.”
— Flexera State of the Cloud Report 2024
[Source: https://www.flexera.com/blog/cloud/cloud-computing-trends-2024-state-of-the-cloud-report]

A2. Multi-Cloud Adoption and Visibility Gaps

“89% of organisations have a multi-cloud strategy, yet managing cloud spend remains the top challenge, particularly in tagging, forecasting, and visibility across platforms.”

— Flexera 2024
[Source: see above]

“Tooling across providers differs significantly, even where abstraction layers such as Terraform are used. Provider-specific modules and behaviours make clean parity impractical.”
— HashiCorp Terraform Provider Documentation
[Source: https://registry.terraform.io/namespaces/hashicorp/providers]

A3. Platform Divergence and Abstraction Fallacies

“Kubernetes can run across clouds, but differences in service discovery, storage plugins, ingress, and IAM behaviour introduce operational divergence that abstraction layers do not eliminate.”
— Cloud Native Computing Foundation (CNCF) guidance

“Abstraction tools like Terraform and Kubernetes delay, rather than eliminate, the divergence between providers. Platform-specific behaviours still surface at runtime or under load.”
— Industry best practice commentary, including Gartner and Cloud Design Patterns
[Note: Based on compiled technical reviews rather than a single vendor statement.]

A4. Shared Edge Infrastructure and Latency Complexity

“Edge routing, WAF, CDN, and API gateway integration across clouds often adds more complexity than it removes. Cross-cloud ingress is a source of new failure paths, not resilience.”
— Cloudflare and Azure Front Door engineering documentation
[Cloudflare: https://developers.cloudflare.com/load-balancing/; Azure: https://learn.microsoft.com/en-us/azure/frontdoor/front-door-overview]