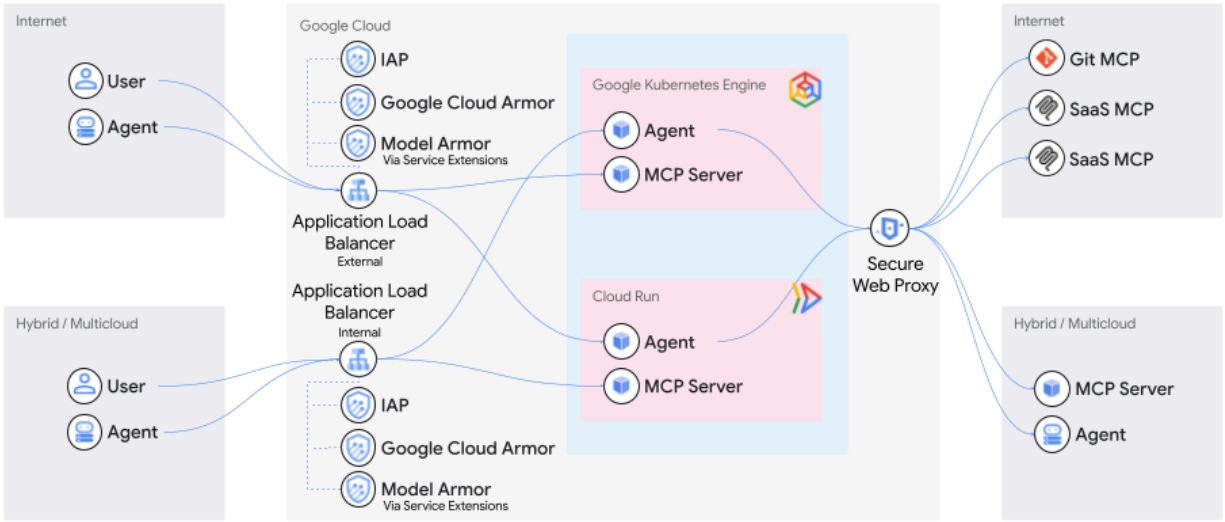
Google Cloud

Networking Patterns for Self-Hosted Agents

Solution Overview



For more information visit **cloud.google.com**

Solution Overview

Networking patterns for governing and securing AI agents and MCP servers in Google Cloud and hybrid environments

Autonomous agents utilizing the Model Context Protocol (MCP) require dynamic, many-to-many connections across heterogeneous environments. This introduces new demands in governance, observability, and security as organizations transition from generative AI prototypes to agentic applications in production.

Google Cloud's Cross-Cloud Network provides capabilities to secure and govern agent interactions with users, MCP servers and other agents. By leveraging a platform-centric approach, decoupling agent developer experience from the agent governance policy decisions from policy enforcement, platform engineers can maintain strict security postures without inhibiting the agility of AI development.

The Agentic Networking Challenge

Traditional microservices architectures rely on predictable, static call graphs where service dependencies are mapped and firewall rules are defined at deployment time. In contrast, AI agents operate autonomously, determining their own execution paths, tool usage, and data access patterns in real-time based on user prompts and reasoning models. This introduces a fundamental shift from deterministic to probabilistic network behavior.

01

Dynamic Connectivity

Agents interact with a diverse ecosystem of MCP servers, SaaS APIs, and internal databases, often traversing public and private networks simultaneously.

02

The Identity Imperative

Agents act as the new knowledge workers, making autonomous decisions that require distinct identities similar to human users to establish accountability. Furthermore, security models must now handle compound authorizations, enforcing policies that validate both the agent's Identity (is the code trusted?) and the user's identity (does the person have permission?) whenever agents act on a user's behalf.

03

The Governance Gap

Standard firewall rules are insufficient for traffic where the intent and payload content (prompts/responses) matter as much as the source and destination IP. Organizations must enforce Zero Trust principles that account for the semantic context of every interaction.

Architectural Philosophy: The "One Network" Approach

Organizations need an architecture that allows them to **consistently** enforce governance and access policies across all possible agent interactions & deployment patterns. Google's "one network" approach provides a framework to establish **consistent**, robust, centralized control over all agent and MCP server communications.

Further, this architecture adopts a "shift down" philosophy, moving the security, compliance, and observability functions from individual application developers down to the network infrastructure itself. This not only simplifies the agent developer experience, but also allows your organization's infrastructure platform and security teams to **consistently** secure and govern agents developed in any framework (e.g. ADK, AG2, Langgraph, etc.) by any agent provider.

A Universal Control Plane

Underpinning these enforcement points is Google Cloud's unified networking stack, composed of two core pillars:

1. **One control plane:** Traffic Director serves as the global brain, distributing policies via the open-source xDS API. This allows administrators to define routing and security rules once and enforce them consistently everywhere.

2. **One proxy:** The open-source Envoy proxy acts as the universal agent gateway data plane, providing a consistent, programmable enforcement point for ingress, egress, and service-to-service communication.

   Envoy is evolving to be an agent gateway with support for AI-first protocols like MCP, A2, and OpenAI. This allows Envoy policy capabilities, extensibility, and observability to be natively extended for agentic AI workloads. Envoy's extensibility architecture enables the addition of these protocols while allowing organizations to leverage the production hardened features of Envoy that are critical to running AI workloads reliably at scale.

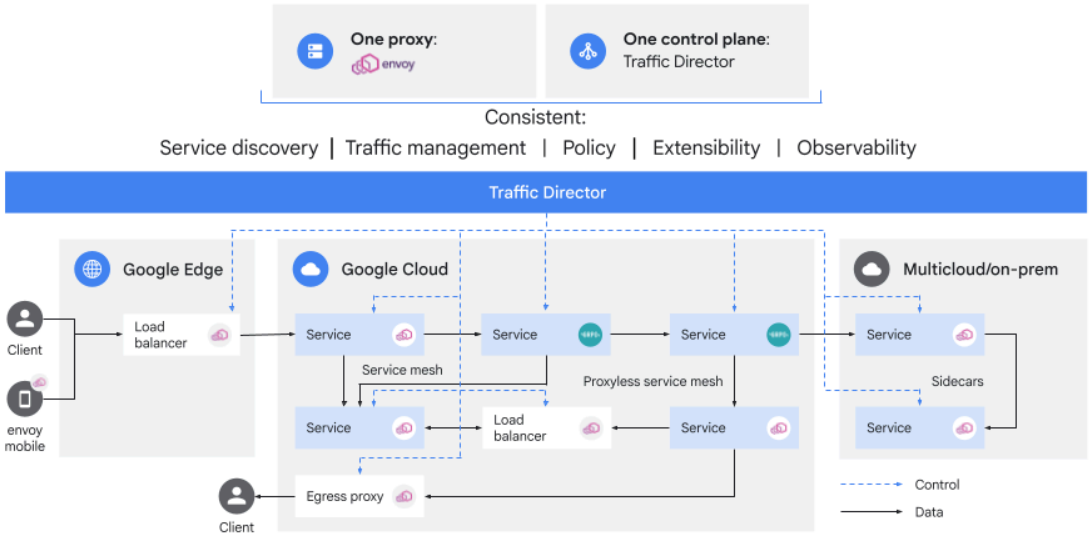


Figure 1: "one network" reference architecture

Programmable Intelligence with Service Extensions

A key differentiator of this network architecture is its programmability. Through Service Extensions, the network becomes an active participant in AI security. This capability allows for the insertion of custom logic, via plugins or callouts such as the Model Armor callout directly into the data path. This enables real-time, semantic inspection of agent payloads to detect and block threats like prompt injection without altering application code.

- **Callouts**: These enable Cloud Load Balancing to make gRPC calls to both Google-managed and user-managed services during data processing. Their primary benefit is flexibility, as they have no runtime restrictions and allow you to reuse existing software to customize fully managed services for specific workload needs.
- **Plugins**: These allow you to insert custom code directly inline into the networking data path using WebAssembly (Wasm) and the Proxy-Wasm ABI. They run on a Google-managed infrastructure close to the data plane, providing the benefit of managed latency optimization, though they have stricter runtime requirements compared to callouts.

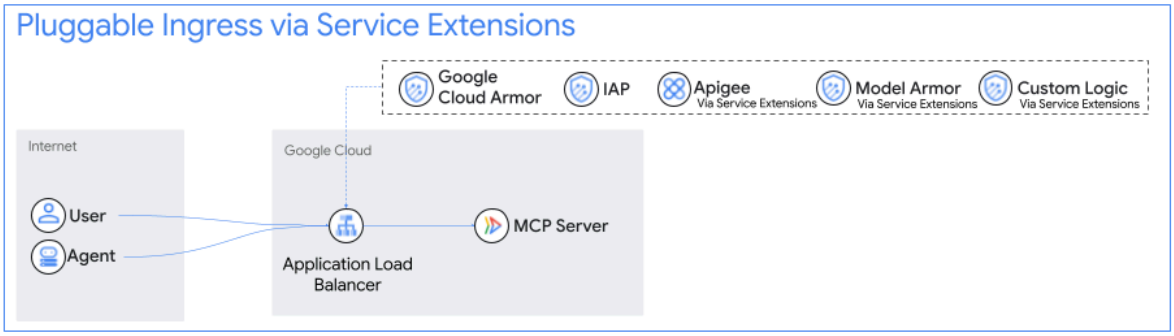


Figure 2: Pluggage ingress via Service Extensions

Extending the Network for Agentic Governance

This architecture leverages Google Cloud's proven networking stack of load balancers, proxies, and Traffic Director and extends them to natively understand and govern agentic traffic. By embedding governance directly into these existing components, organizations can enforce policies at every boundary without deploying separate security appliances

- **Gateway patterns (enforcement layers):** The architecture establishes dedicated ingress and egress gateways using standard Cloud Load Balancing and Secure Web Proxy components. These gateways act as critical layers of separation, validating identity and sanitizing content before it reaches the application logic.
- **Decision engines (policy logic):** Instead of static rules, the network integrates with dynamic policy engines. It evaluates request context such as identity (IAP), permissions (IAM), and payload data (Model Armor) to authorize or deny traffic in real-time

Reference Architecture Patterns

The following patterns illustrate how to compose Google Cloud networking services to secure specific agentic interaction flows.

Pattern 1: Ingress Access Control & Security for MCP Servers and Agents

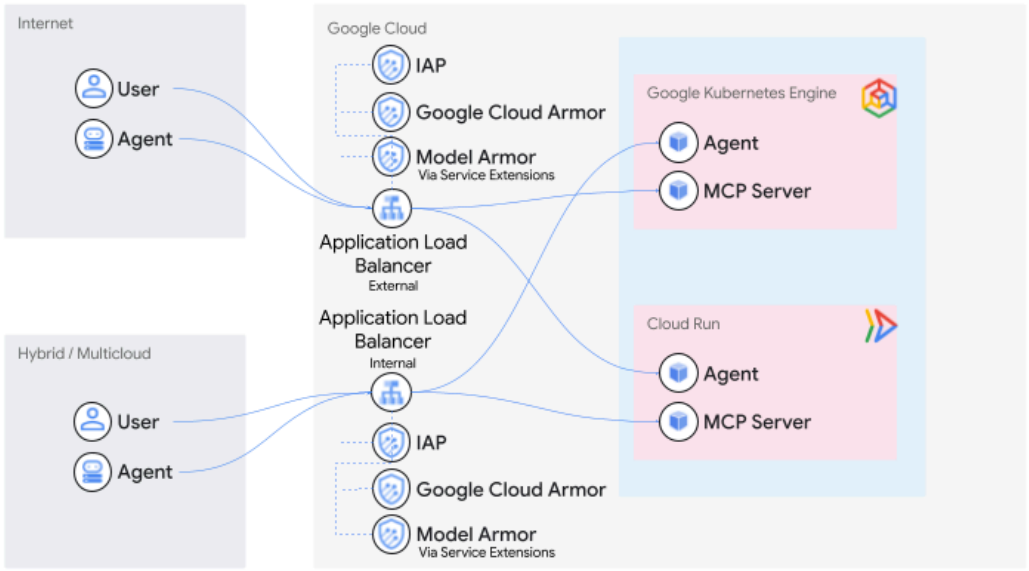


Figure 3: Ingress security for external users and agents

Use case: A partner's agent or a developer's CLI accessing an internal MCP tool.

Organizations creating agentic platforms often need to expose internal tools to external partners or developers while maintaining a Zero Trust posture. This requires a unified ingress endpoint that shifts the attack surface from the application server to the global network edge, ensuring consistent authentication and threat protection.

Identity-aware edge protection: The solution employs an Application Load Balancer integrated with Identity-Aware Proxy (IAP) to act as the primary access control engine. This configuration supports diverse authentication flows including user-delegated OAuth for human-driven CLI tools and service account keys for programmatic access normalizing the entry point for all consumers.

Edge defense and observability: Before requests reach the backend, Cloud Armor filters traffic at the edge to mitigate DDoS attacks and enforce WAF rules. By terminating connections at the global load balancer, the architecture ensures that only authenticated, authorized, and safe traffic reaches the internal MCP servers, providing a consolidated surface for logging and access control.

Pattern 2: Governed Agent Egress and Service-to-Service Connectivity

To ensure the safe deployment of autonomous agents, organizations require a consolidated strategy that normalizes security policies across all traffic flows, whether they are destined for external SaaS APIs, other clouds, or internal services.

Pattern 2a: Governed Egress and Content Inspection

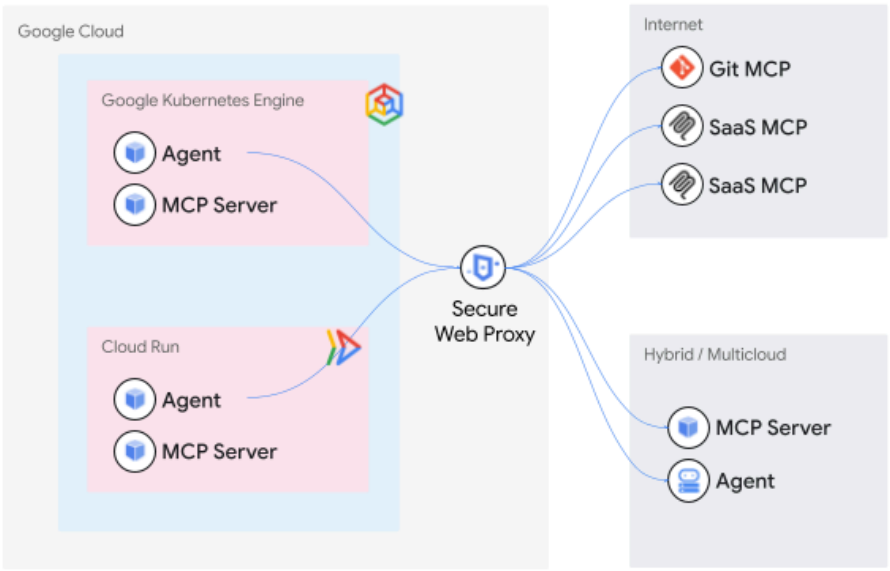


Figure 4: Governed egress and content inspection

Use case: An agent querying external SaaS APIs or public knowledge bases.

To ensure the safe deployment of autonomous agents that interact with external ecosystems, organizations require a consolidated egress strategy that normalizes security policies across all outbound traffic. The "one network" architecture leverages a managed egress gateway to enforce governance without inhibiting agent performance.

Consolidated egress policy: The Secure Web Proxy (SWP) serves as the centralized inspection and control point for all agent-initiated outbound communications. Unlike traditional firewalls, SWP enables transparent egress policy enforcement, allowing administrators to maintain strict allow-lists of authorized external MCP servers (e.g. api.github.com) without requiring code changes in the agent

Advanced content guardrails: To mitigate exfiltration risks, the architecture supports the insertion of AI-specific guardrails directly into the data path. Through Service Extensions, the proxy can trigger callouts to services like Model Armor to scan payloads for sensitive data or prompt injection attacks before traffic leaves the VPC.

Pattern 2b: Cross-Cloud Security Hub

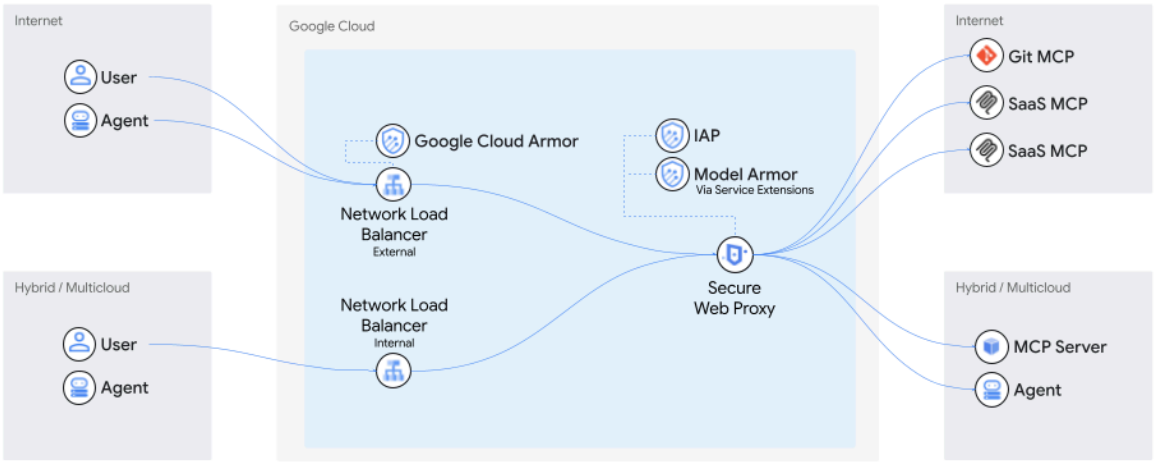


Figure 5: Cross-cloud security hub

Use case: Securing traffic between an external agent (e.g. on another cloud provider) and an external SaaS, using Google Cloud as the intermediary.

By utilizing Google's global private backbone as your transit fabric fronted by global and regional external Application Load Balancers you can decouple security from physical location. These load balancers serve as the unified ingress gates, leveraging anycast IPs to ingest traffic at the point of presence nearest to the source, regardless of which cloud or region the agent resides in. This enables you to terminate SSL/TLS sessions at the edge and apply immediate DDoS protection before the traffic ever reaches the inspection layer.

This architecture allows you to achieve a "single pane of glass" for policy enforcement, applying the same rigorous inspection standards to external-to-external traffic as you do for internal workloads.

Global connectivity backbone: Cloud WAN uses Google's global infrastructure as an enterprise backbone, avoiding public internet risks. This unified network connects all workloads, including branches and edge locations, via managed solutions.

- **Cross-Cloud Interconnect:** establishes direct, high-bandwidth links to other cloud providers (AWS, Azure, OCI), ensuring secure, low-latency ingress for agents hosted in other clouds.
- **Cross-Site Interconnect:** connects on-premises data centers via wire-speed Layer 2 connectivity
- **SD-WAN & Cloud VPN:** Integrates branch offices and edge locations via Network Connectivity Center. This enables agents deployed at the edge to connect back to MCP servers hosted in Google Cloud (or external SaaS) over Google Cloud's Premium Tier network, ensuring low latency and consistent security enforcement

Together, these technologies funnel traffic securely into the Google Cloud network, where it is optimally routed through an enforcement point. This pattern validates the external entity's identity and authorization context, enforcing consistent data loss prevention (DLP), threat detection, and logging standards across your entire global footprint.

Pattern 2c: Governed Service to Service Connectivity

Use case: An agent hosted on GKE or Cloud Run accessing an internal MCP server or data service within Google Cloud.

Organizations deploying autonomous agents seek the normalization of security postures across diverse internal services while maximizing agility. The network infrastructure plays a critical role in enforcing Zero rust principles for agent-to-MCP server communications. To achieve this, the solution offers flexible architectural patterns that decouple policy enforcement from application logic, leveraging Google Cloud's "one network" foundation.

To normalize security across internal services, the architecture provides two distinct implementations for enforcing Zero Trust principles. Both approaches decouple policy from code, but they differ in where the enforcement occurs: at the network boundary (gateways) or directly alongside the workload (sidecars).

Implementation Option 1: Gateway-Based Enforcement (Application Load Balancer & Secure Web Proxy)

This pattern relies on centralized infrastructure components to intercept and govern traffic at the boundaries of the service.

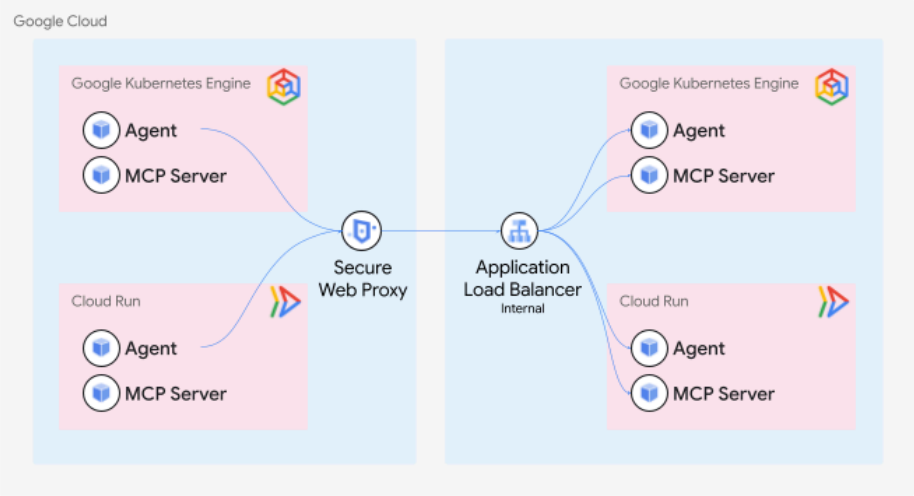


Figure 6: Intra-cloud service-to-service communication via Application Load Balancer and Secure Web Proxy

Mechanism:

- **Ingress (single-layer enforcement):** An internal load balancer fronts the MCP server, acting as the primary gatekeeper. It evaluates authorization policies to verify the calling agent's identity before routing traffic to the backend
- **Egress (defense-in-depth)**: For additional control - such as traffic crossing trust boundaries (e.g. between VPCs or to the internet) a Secure Web Proxy (SWP) is added to the path. The

SWP governs the agent's egress, validating the intent (what it calls), while the load balancer or the third-party remote MCP server enforces ingress policies on the destination.

When to choose this:

- Simplicity: Ideal for environments that want to enforce policies without the operational complexity of managing a full service mesh.
- Boundary control: Best for "defense-in-depth" scenarios where you need independent control points for the team managing the agent (egress) and the team managing the MCP server , including remote MCP servers that may be outside your organization. (ingress).

Implementation Option 2: Sidecar-Based Enforcement (Cloud Service Mesh)

This pattern distributes enforcement logic to every individual workload, creating a unified "mesh" of secure connectivity.

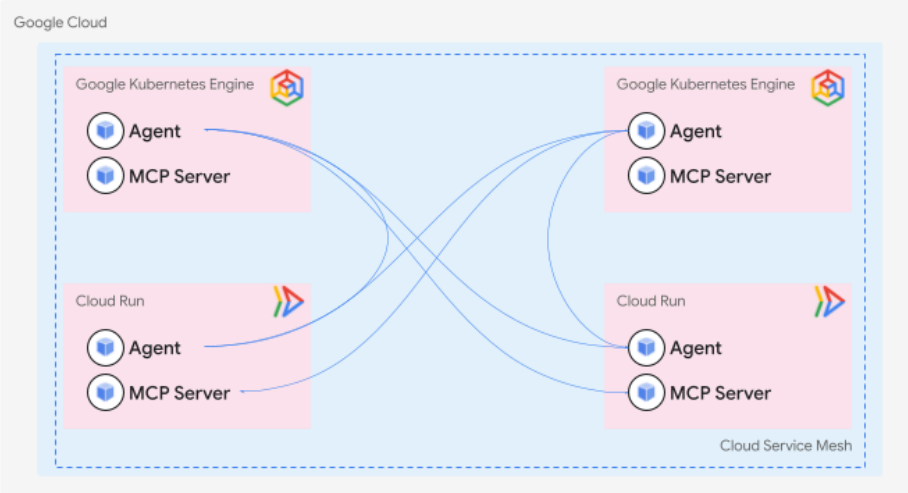


Figure 7: Intra-cloud service-to-service communication via Cloud Service Mesh

Mechanism:

- **Distributed enforcement:** Envoy sidecar proxies are deployed alongside every agent and MCP server. These proxies intercept all traffic at the source (egress) and destination (ingress), enforcing policies locally before packets ever hit the wire
- **Zero Trust fabric:** Cloud Service Mesh manages these proxies, automatically handling mutual TLS (mTLS) for transport security and identity assertion, abstracting the network completely.

When to Choose This:

- **Granularity:** Best for organizations requiring the most granular, identity-based policies for every single interaction, regardless of network topology.
- **Unified fabric:** Ideal if you are already adopting a service mesh strategy to standardize observability, traffic management, and security across a large fleet of microservices.

Identity Federation for Agents outside Google Cloud

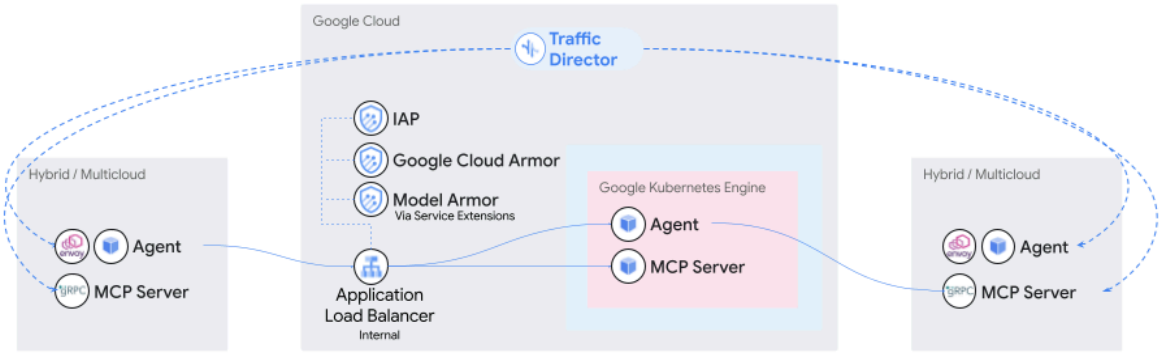


Figure 8: Hybrid connectivity with identity federation

Use case: A cloud-native agent accessing legacy data on-premises

Extending the agentic mesh to on-premises environments demands a universal control plane that can manage policies consistently across physical and cloud boundaries. This solution normalizes the hybrid environment by extending the Google Cloud service mesh to on-premises infrastructure.

Agent identity: To address the "Identity Imperative," Google Cloud introduces agent identity, a platform-managed principal based on the SPIFFE standard. This identity serves as the foundation for Zero Trust enforcement within Cloud Networking products:

- **mTLS for known agents:** Cloud Networking infrastructure (such as Cloud Service Mesh and Application Load Balancer) leverages agent identity to enforce mutual TLS (mTLS). This ensures that only known, cryptographically verified agents and clients can establish connections.
- **User authentication:** For scenarios where agents act on behalf of a person, Identity-Aware Proxy (IAP) validates the user's identity (e.g. via OIDC tokens). This allows the network to enforce compound policies that authenticate who is driving the agent, ensuring that the agent cannot exceed the permissions of the user invoking it

Identity federation for external agents: To integrate agents running outside Google Cloud (on-premises or in other clouds), the architecture relies on workload identity federation.

- **Eliminating long-lived keys:** Instead of managing risky, long-lived service account keys, external agents exchange their verifiable local credentials (e.g., AWS AWS IAM tokens or OIDC tokens) for short-lived Google Cloud access tokens.
- **Unified control plane:** Once authenticated, these external agents are fully integrated into the mesh by Traffic Director. This enables platform engineers to apply the same consistent authorization policies and mTLS encryption to external traffic as they do for internal workloads.

This integral integration enables platform engineers to implement uniform authorization policies and resilient mTLS encryption throughout the hybrid infrastructure, thereby substantially optimizing the deployment of autonomous agents across both cloud and data center environments.

Summary

As organizations transition from experimental generative AI to production grade agentic applications, they face a fundamental shift in networking requirements. Unlike traditional microservices with static dependencies, autonomous agents exhibit probabilistic behavior, dynamically determining their own execution paths and connecting to a diverse ecosystem of tools, data, and other agents. This dynamic nature creates a governance gap where traditional, perimeter-based security controls are no longer sufficient.

Google Cloud's agentic networking solution addresses this challenge by decoupling security and governance from application logic, leveraging the "one network" architecture to establish a unified, identity-centric control plane. By shifting the burden of policy enforcement down to the infrastructure, organizations can secure agentic workflows without inhibiting developer agility.

Key architectural pillars:

- **Identity-driven Zero Trust:** The solution moves beyond IP-based allow-lists, enforcing a strict Zero Trust model based on agent identity and workload identity federation. This ensures that every interaction whether internal, cross-cloud, or on-premises is authenticated, encrypted via mTLS, and authorized based on cryptographic identity rather than network location.
- **Unified governance plane:** Through a consolidated set of gateway patterns, the architecture normalizes policy enforcement across all traffic flows:
    - **Ingress security:** Protects agents and tools from unauthorized access using Identity-Aware Proxy (IAP) and Cloud Armor.
    - **Governed connectivity:** A centralized Secure Web Proxy (SWP) and Internal Load Balancers enforce granular egress and service-to-service policies, preventing data exfiltration and ensuring agents connect only to approved MCP servers.
    - **Hybrid reach:** Leveraging Cloud WAN and Traffic Director, the solution extends this governance fabric to the edge and other clouds, treating external workloads as first-class citizens of the mesh.
- **Programmable intelligence:** A key differentiator is the use of Service Extensions, which transforms the network into an active security participant. By inserting custom logic (like Model Armor) directly into the data path, the infrastructure can perform real-time, semantic inspection of agent payloads to detect and block AI-specific threats such as prompt injection.

By consolidating these capabilities under a universal control plane and data plane (Envoy), Google Cloud provides a scalable, secure foundation that empowers platform engineers to govern autonomous agents with the same rigor as traditional workloads.

Google Cloud