Google Cloud | vodafone

Self-Optimizing Autonomous Networks

An implementation guide

June 2026

TM Forum foreword

As telecommunications operators transition toward autonomous networks with increasing network complexity, traditional manual and imperative automation methods are no longer sufficient to maintain performance.

Communication service providers (CSPs) are increasingly constrained by an operational trilemma: scale, reliability, and efficiency. As networks scale, the industry has historically optimized reliability at the expense of efficiency, reinforcing rigid change controls that slow innovation.

A central cause is the 'fragility trap': change is the dominant source of outages, so operators reduce change to protect availability, but by doing so structural inefficiency is locked in and service agility is limited.

Best efforts on network reliability are no longer a differentiator. CSPs must move to productize the network as an investment for growth. Autonomous networks shift the model to intent-driven, closed-loop operations so change becomes safe, continuous, and scalable, unlocking efficiency and growth.

Many early 'AN Level 4' initiatives risk missing the point by applying AI to automate legacy processes ('intelligent automation'). This improves response speed but keeps the management plane dependent on human-authored workflows.

True autonomy requires a shift from imperative, script-driven 'how' to declarative, intent-driven 'whats' where decisions are supported by a knowledge plane (digital twin + policy/constraints) that can compute safe action sequences at runtime.

The evolution of the TM Forum Autonomous Networks Mission reframes operations around dynamic, autonomous management: intent establishes a desired outcome, and the network continuously re-optimizes (GPS-style) using telemetry, simulation, digital twins, and closed-loop controls to maintain that outcome.

Key message: AN Level 4+ is not 'more automation'. It is a change in operating model to intent-driven, simulation-backed, closed-loop control, so that change becomes inherently safe and routine rather than exceptional.

This paper, a collaboration between Google Cloud, Vodafone, and TM Forum, outlines a practical framework for self-optimizing autonomous networks.

George Glass
CTO, TM Forum

Authors and contributors

Google Cloud
Brian Naughton
Jen Hawes-Hewitt
Tom Curry
Piyush Mathur
Amrita Mukherjee

Vodafone
Steve Allen
Peter Cosimini
Andrea Campanella
Richard Kilmurray
Aitor Lorenzo

Contents

01

Document purpose

The purpose of this document is to provide a comprehensive framework for implementing **self-optimizing autonomous networks** using advanced **closed-loop** architectures. As network complexity increases with the rollout of new technologies and diverse service requirements, traditional manual and automated tuning is no longer viable.

At a time when the majority of major operators are committing to attaining Level 4+ across their networks by 2030, and when the advancement of AI-powered innovation is accelerating at an unprecedented pace, this paper builds on previous TMF architectural work and serves as a practical guide to support the TM Forum community to transition from reactive monitoring to **proactive, intent-based network autonomy.**

A central theme is the integration of **domain-specific automation** and **intelligent cross-domain reasoning.** Leveraging the complementary strengths of **local automation (e.g., rApps)** and **AI agents** to create a unified autonomous ecosystem as follows:

- **Local, domain-specific automation**
  AI capabilities (up to very specialized AI agents) deployed within the network domain; that is, SDN Controller or RAN Intelligent Controller (RIC), acting as specialized network automation and designed for high-speed, bounded, and more deterministic tasks, such as local radio resource optimization or congestion monitoring. While highly efficient, they often lack the visibility into broader business intents or cross-domain dependencies.
- **Cross-domain AI agents (global, cognitive reasoning)**
  Operating primarily in the cloud environment, AI agents can interpret complex, high-level business goals and reason across disparate data sources—such as social media feeds, weather reports, and cross-domain telemetry—that are not available to local automation tools.

Working together, AI agents with access to network and business knowledge can further orchestrate AI-native network controllers—to deliver better performance and/or business outcomes, **enabling a business-aware, service-level closed loop.**

Combining **low-latency specialized automation** with the **complex reasoning of AI agents** can bridge the gap between resource-level performance and business-level service guarantees.

02

Self-optimizing closed loops

Self-optimizing autonomous networks are essential to overcome the growing complexity of modern network infrastructure, which is characterized by thousands of configuration parameters and the continuous introduction of technologies like 5G SA diverse carriers, lossless DC fabrics, disaggregated transport networks, and varying quality of service requirements.

Leveraging the latest advancements in AI enables the network to move beyond traditional manual configuration and tuning. This allows for real-time, self-optimizing adjustments based on conditions such as traffic distribution and network load, maximizing overall network performance and efficiency.

Furthermore, AI and its associated sub-components, such as Graph neural networks, facilitate proactive anomaly detection and accelerated root cause analysis (RCA), transforming reactive monitoring into a predictive system, which significantly improves mean-time-to-recover (MTTR) and overall net promoter score (NPS).

2.1
Anatomy of a closed loop

Closed loops are **intent-based**; that is, instead of telling the network how to do something (imperative), the operator tells the closed loop what the desired outcome is (declarative). The closed loop then takes responsibility for maintaining that outcome dynamically as network conditions change.

A closed loop is an automated process that bridges the gap between network **assurance** (monitoring) and **fulfillment** (execution). Unlike traditional 'open-loop' systems where a human must interpret an alarm and then manually push a configuration change, a closed loop uses **AI and data analytics** to carry out the full AADE cycle:

01 **Awareness of** the network state

02 **Analysis to identify any deviations** from the goal

03 **Decision** on a solution

04 **Execution of a mediation to** resolve the issue

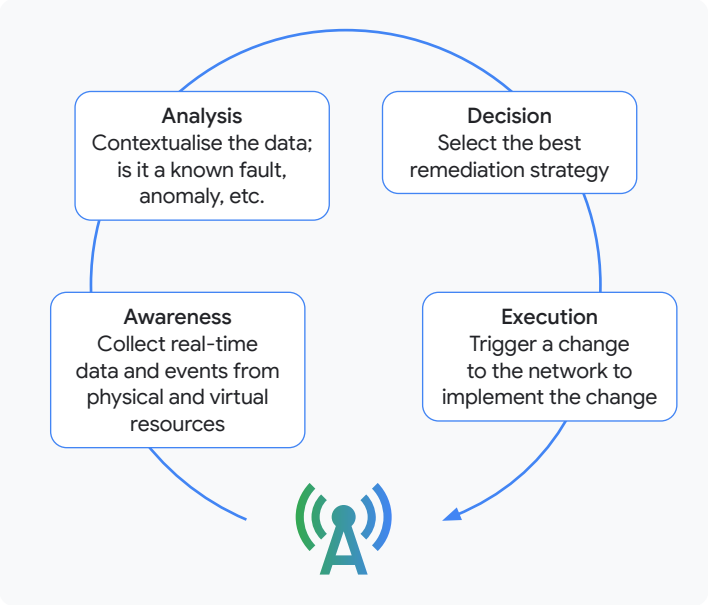


Figure 1: The four-step closed-loop cycle. Source: Google Cloud

Furthermore, the closed loops operate in a continuous learning pattern, where every operational outcome is stored, analyzed, and used to improve future decisions and automation behavior. Common closed-loop use cases that align well with the TM Forum's published Autonomous Network IG1339 High Value Scenarios include:

- **Automated service provisioning** to launch new offerings quickly
- **Real-time network optimization** to dynamically adjust network resources and configurations in response to customer or business intents, network load, or other network conditions
- **Proactive fault prediction and automated RCA** to predict failures weeks in advance and significantly reduce MTTR
- **Zero-touch remediation/ self-healing** to automatically trigger and execute corrective workflows
- **Energy efficiency** through AI-powered, near real-time augmentation of power-saving features on equipment
- **Capacity demand forecasting** for proactive, predictive planning, ensuring quality of service requirements are met with just-in-time network resources and avoiding capital waste

2.2
Hierarchy of closed loops

TM Forum defines a 'loop-of-loops' structure where closed loops operate at three distinct layers, interacting with one another to allow control to move up and down the layers:

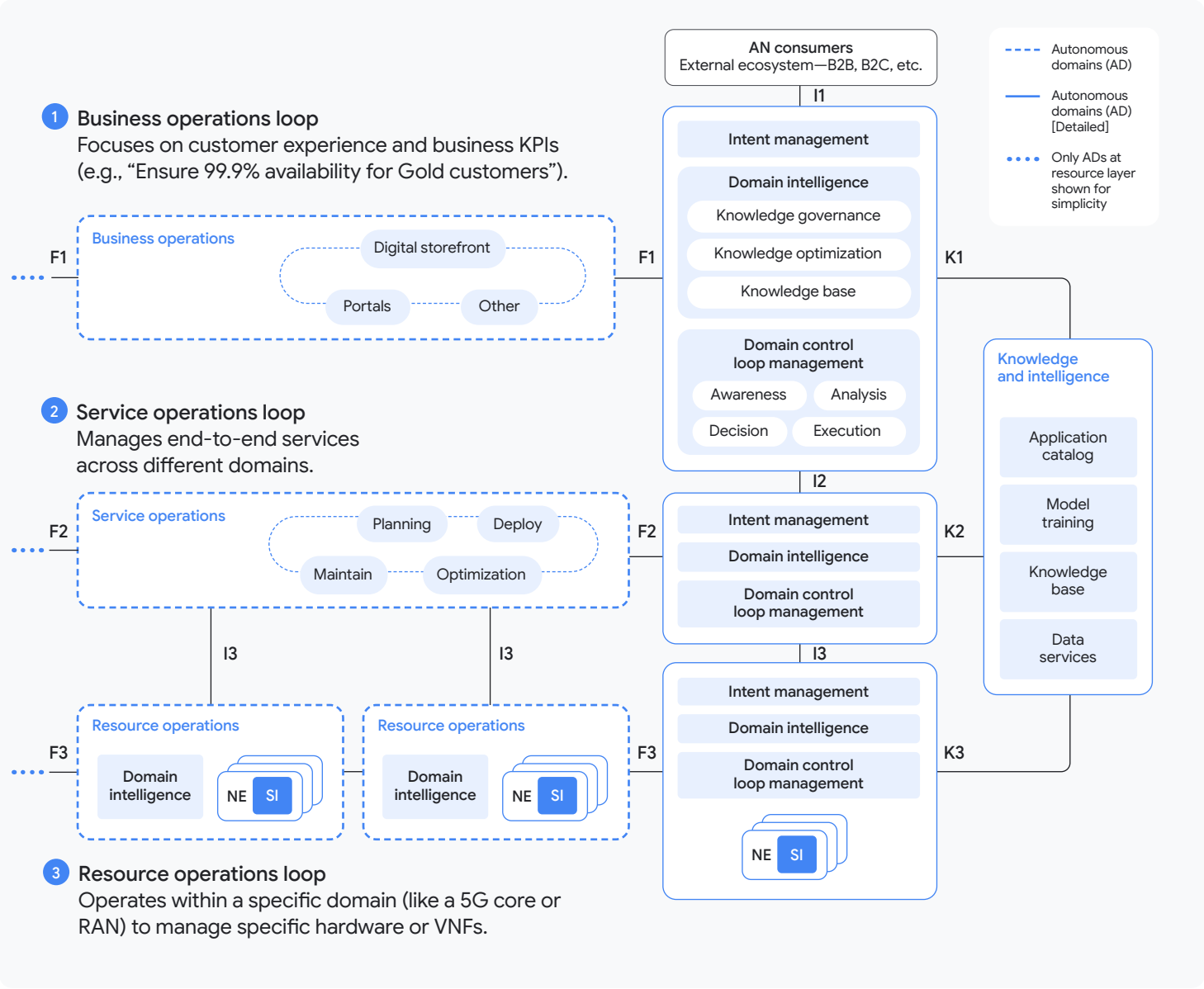


Figure 2: AN reference architecture (showing detailed ADs at all layers). Source: TM Forum

2.3 Interaction patterns

Closed-loop instances can interact with other instances to achieve their goal/objective. The figure below shows common closed-loop interaction patterns that must be accommodated by the implementation architecture.

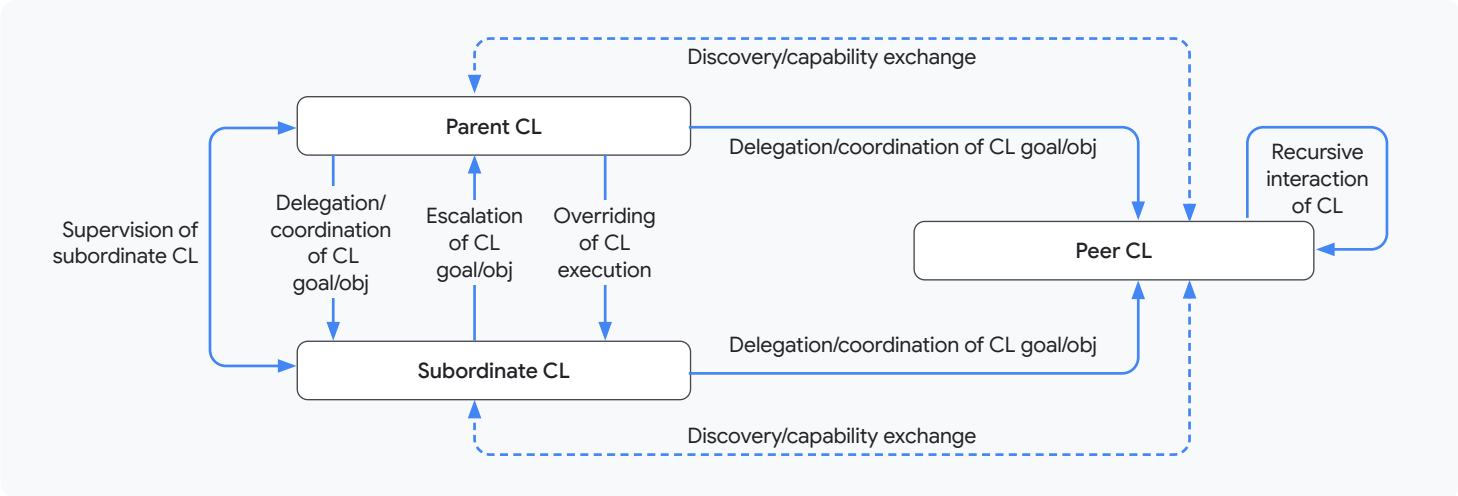


Figure 3: Closed loop capability—discovery, supervision, delegation, coordination. Source: TM Forum

Closed loops can interact in a hierarchical fashion utilizing an **'intent-driven'** delegation model where higher-level loops manage lower-level loops as 'black boxes'. Closed loops at lower levels can also escalate to those at a higher level, moving control to where it is needed for further decision-making.

Common patterns are as follows:

- **Delegation and coordination**
  A **parent CL** sends specific goals or objectives down to a **subordinate CL**. This follows the 'what vs. how' principle, where the parent defines the desired outcome and the subordinate manages the execution.
- **Escalation**
  If a subordinate CL cannot meet its assigned intent due to changing network conditions, it escalates the goal or objective back to the parent CL for further decision-making.
- **Supervision and overriding**
  The parent CL provides continuous supervision of the subordinate CL and maintains the authority to override its execution if the actions conflict with broader network goals.

Closed loops operating at the same functional layer also interact to ensure cross-domain harmony.

- **Discovery and capability exchange**
  Peer loops communicate to exchange information about their specific capabilities and discover how they can support one another.
- **Cross-domain delegation**
  Similar to hierarchical models, peers can delegate or coordinate specific goals to ensure that an action in one domain (e.g., core) does not negatively impact another (e.g., RAN).

A closed loop may also feature **recursive interaction**, where it continuously iterates on its own internal logic to refine its state and improve its decision-making accuracy over time. The same coordination, escalation, and coordination patterns happen within agents and automation systems used to deliver closed loops.

2.4
Network slice example

Network slicing is a common priority use case for operators looking to provide differentiated connectivity to meet specific business needs for their customers. This example shows how multiple closed loops cooperate horizontally and vertically to continuously optimize a 5G network slice.

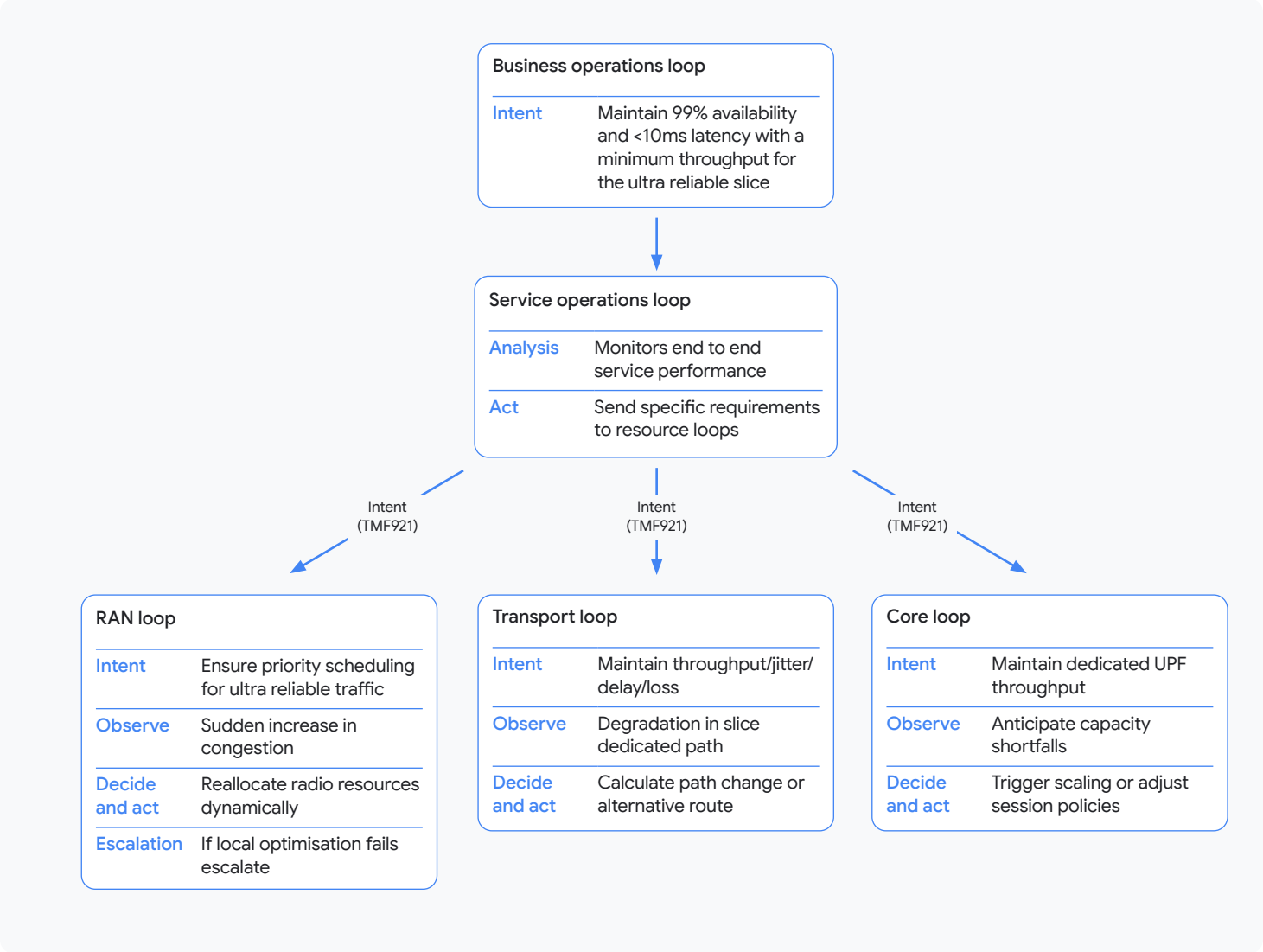


Figure 4: Implementing cooperating closed loops diagram. Source: Google Cloud

2.4
Network slice example

The following table details the task definitions of the key steps/loops:

Step	Loop	Observe/analyze task	Decide task	Act task
01	**Business operations loop**	N/A (focus is defining the high-level goal/intent)	Defines the high-level intent: "Maintain 99.99% availability and <10ms latency with a minimum throughput for the 'ultra-reliable' slice"	Delegates this defined intent to the service operations loop
02	**Service operations loop**	Monitors the end-to-end service performance	Determines specific performance requirements (delegated intent) for the domain-specific loops	Cascades the specific slice requirements to the RAN, transport, and core resource loops
03	**Ran resource loop**	Observes a sudden increase in local congestion at a specific cell site	Uses AI/ML reasoning (rApp on an RIC) to decide to dynamically reallocate radio resources to protect the slice	Reallocates radio resources dynamically, **OR** triggers an escalation to the service operations loop if optimization fails
04	**Transport resource loop**	Monitors real-time telemetry from IP/MPLS/ SDN infrastructure to detect performance degradation or congestion	Uses a knowledge graph of the network topology to calculate a path change to an alternative route	Executes the path change (traffic engineering and path optimization)
05	**Core resource loop**	Monitors core network function (CNF) resource utilization and uses predictive AI/ML models to anticipate capacity shortfalls	Determines the need for capacity management and resource scaling (scale-in/ scale-out) of CNF instances	Automatically triggers the scaling of CNF instances (e.g., UPFs) or adjusts session/bearer policies to guarantee quality of service (QoS)
06	**Conflict resolution and feedback**	Receives the escalation from a subordinate CL (e.g., RAN Loop) and analyzes the cross-domain impact	Determines a new coordination strategy that overrides the problematic local action	Overrides the original domain-specific local action to satisfy the end-to-end intent

03 Implementing closed loops

This section focuses on mapping the concepts described in the previous sections to high-level implementation choices that can inform an autonomous network architecture.

A closed-loop instance is not a single monolithic application, but rather a coordinated sequence of tasks designed to achieve a specific intent. These tasks are modular and can be implemented using a variety of technologies. Each technology choice makes sense for a task in hand and its requirements for real-time processing, etc.

This modularity allows tasks to be 'stitched together' to create complex, end-to-end automated workflows as in the diagram below.

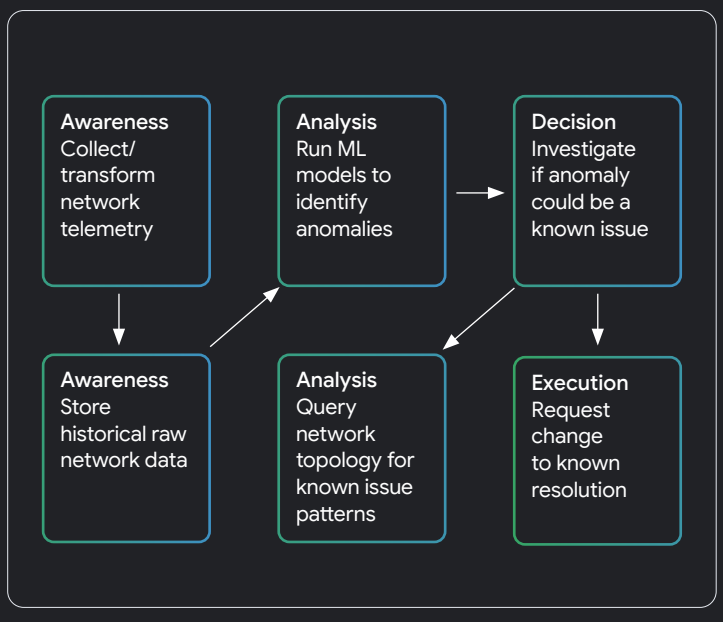


Figure 5: AI agent key components. Source: Google industry research

Closed loops include a sequence of tasks that perform the following type of roles:

- **Awareness and data persistence**
  Often starting with observing the network, where telemetry is collected and transformed. This data is then stored to maintain a historical record of raw network performance and state.
- **Intelligent analysis**
  The loop moves into the **analysis** phase, where AI/machine learning models can be executed to identify anomalies within the live data. Simultaneously, the system queries the network topology to see if these anomalies match known issue patterns.
- **Reasoning and decisioning**
  During the **decision** phase, the system investigates if an identified anomaly corresponds to a known issue with an established resolution.
- **Execution**
  Once a remediation strategy is selected, the loop performs the act task by requesting a specific change to the network to implement the resolution.

3.1
Task technology landscape

It is especially important given the fast-paced advancement of AI, and accompanying proliferation of technology choices, to have clarity over which technology is appropriate for a given closed-loop task. In this section each type of task/role introduced earlier is mapped to common implementation technology choices, including a rationale for each choice.

Closed-loop task	Technology choices	Rationale
Awareness	Network monitoring	Network faults and performance metrics representing the current state of network equipment
	Network data ingestion and transformation	Ensure real-time, clean, and normalized network data streams are available for analysis and decision-making systems
	Network data lake	Centralized, scalable storage for all historical and real-time data, crucial for AI/ML training and auditability
Analysis	Statistical ML	For pattern recognition, baseline establishment, anomaly detection, and capacity forecasting on structured data
	(Graph) neural networks	Required for deep learning on high-volume, complex data (e.g., streaming telemetry) and advanced predictive models
	Analytics	Provides human-readable insights, dashboards, and reporting for performance monitoring and validation of closed-loop actions
	Digital graph/digital twin	To model and understand the complex relationships between network assets, services, and policies, enabling sophisticated root cause analysis
	LLM/Agent	To process unstructured data (e.g., trouble tickets, user intent), correlate findings, and provide high-level, human-readable summaries and actionable insights
Decision	Workflow	To manage the sequence, coordination, and rollback of actions
	Rules/policy engine	For rapid, deterministic decision-making based on predefined policies; critical for compliance and urgent, non-ML-driven actions
	Imperative code	Used for simple, high-speed, direct actions often deployed closer to the network edge where low latency is paramount
	Digital twin	Virtual model of the network enabling simulation, decision assurance, changes, and actions validation before execution, ensuring safe and policy compliant autonomy
	LLM/Agent	For complex planning, reasoning across domains, and generating multi-step action sequences that require a high degree of adaptability
Execution	Workflow	To manage the sequence, coordination, and rollback of corrective actions across diverse network domains and systems.
	Domain controller	Provides the secure, authoritative interface for configuration and state changes within a specific network domain (e.g., RAN, core, transport)

3.1
Task technology landscape

The closed-loop tasks must be coordinated to deliver an end-to-end flow. This orchestration can be implemented with a variety of technologies as described in the table below.

Orchestration technology choices	Rationale
Code	For simple, high-speed, direct flow control, often deployed at the network edge where low-latency is critical
Workflow/DAG	To define and manage the sequence, dependencies, and rollback of multi-step, end-to-end flows across diverse network domains and systems
Agents	For complex, dynamic flow management, enabling reasoning across domains, generating multi-step action plans, and adapting the orchestration strategy in real-time

3.2
AI agents

Large language models (LLMs) enable complex reasoning by leveraging their vast training data. An LLM can interpret high-level business intents, perform logical inference to connect those intents to low-level network state, and even generate multi-step action plans to achieve the desired outcome.

This capability excels for dynamic, non-deterministic decision-making, where the system must reason across diverse data types, correlate findings from multiple domains, and adapt its strategy based on unforeseen network conditions.

AI agents are autonomous, goal-oriented software entities that leverage LLMs for complex reasoning, but they are also equipped with the memory, planning capabilities, and external tools (APIs, databases) needed to execute multi-step plans and drive the desired network outcome without human intervention.

The distinction between LLMs and agents is primarily characterized by the degree of autonomy and the capacity for action.

An **LLM** is a sophisticated language processor, primarily generating content and predictions, typically only outputting a single prediction or response limited to the data contained within its original training corpus.

In alignment with TM Forum's work defining an AN agent architecture (IG1251D), an AI agent is an autonomous, goal-oriented software system that makes decisions and takes actions to achieve a specific goal. Agents are defined by their capacity for reasoning, planning, memory, and a level of autonomy. Knowledge is extended through the connection to external systems via tools (e.g., APIs, databases, code)

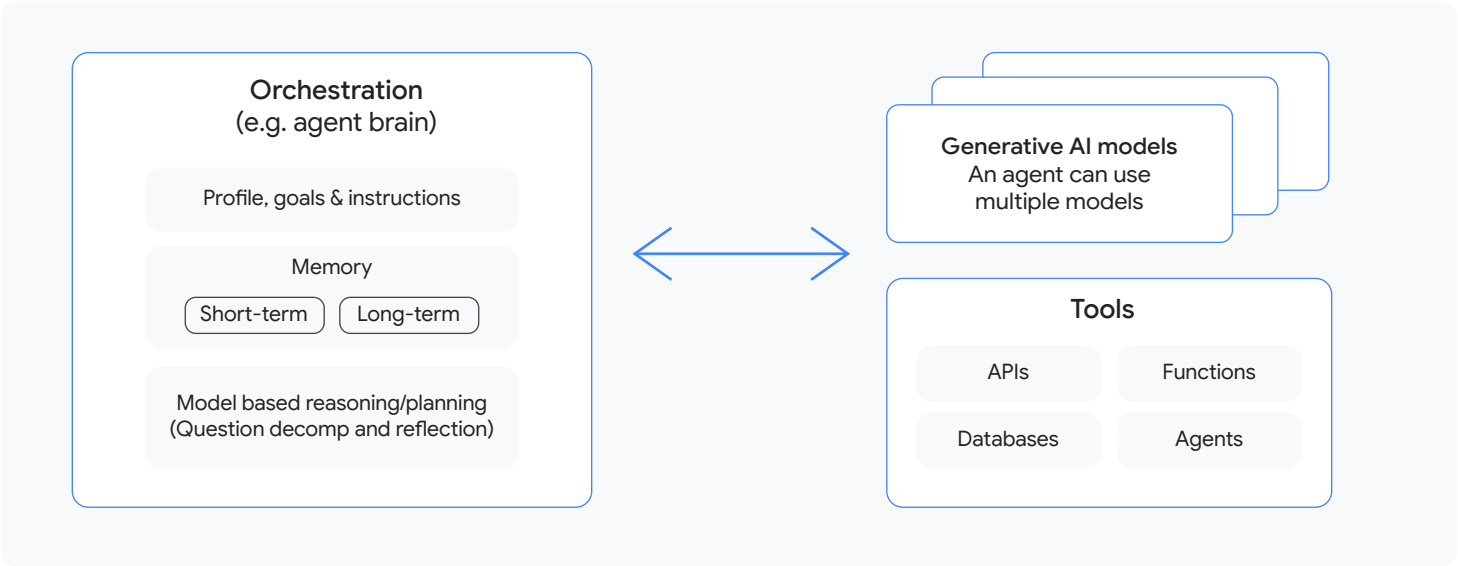


Figure 6: AI agent key components. Source: Google Cloud

3.2
AI agents

Complex objectives often require a progression beyond a single agent to a multi-agent/agentic system. This evolution follows a maturity roadmap shown in the figure below.

- **Single agents**
  Handle single-purpose or multi-action tasks with memory and state management.
- **Multi-agent/agentic systems**
  Systems composed of multiple specialized agents that interact to achieve complex, collective goals.

Key characteristics of a multi-agent/agentic system:

- **Specialized roles**
  Each agent operates with a defined role and unique context, allowing for specialization of function (e.g., a **planner/supervisor** agent and a **domain-expert** agent).
- **Problem decomposition**
  Complex tasks are broken down into smaller, manageable subtasks and distributed among the specialized agents.
- **Protocol-driven collaboration**
  Agents utilize established protocols, such as the Agent-to-Agent (A2A) protocol, to exchange information and coordinate actions dynamically.

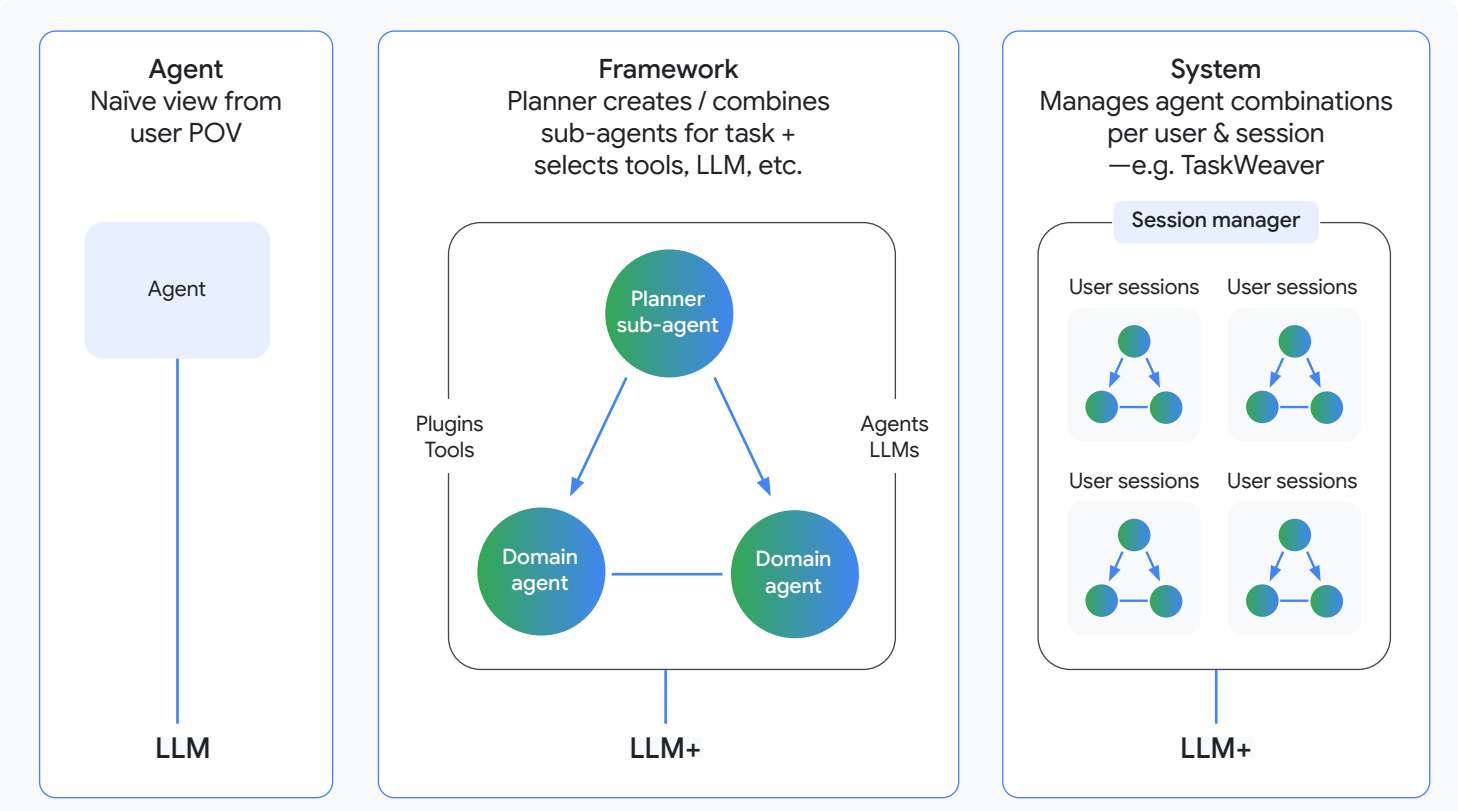


Figure 7: Developing multi-agent systems. Source: Google Cloud

3.2
AI agents

The figure below highlights the use of now established agentic protocols, specifically the **A2A protocol**, which allows agents to exchange information and coordinate their actions dynamically to achieve collective goals. The **Model Context Protocol (MCP)** acts as a standardized interface, providing them with the necessary tools, resources, and prompts from external systems.

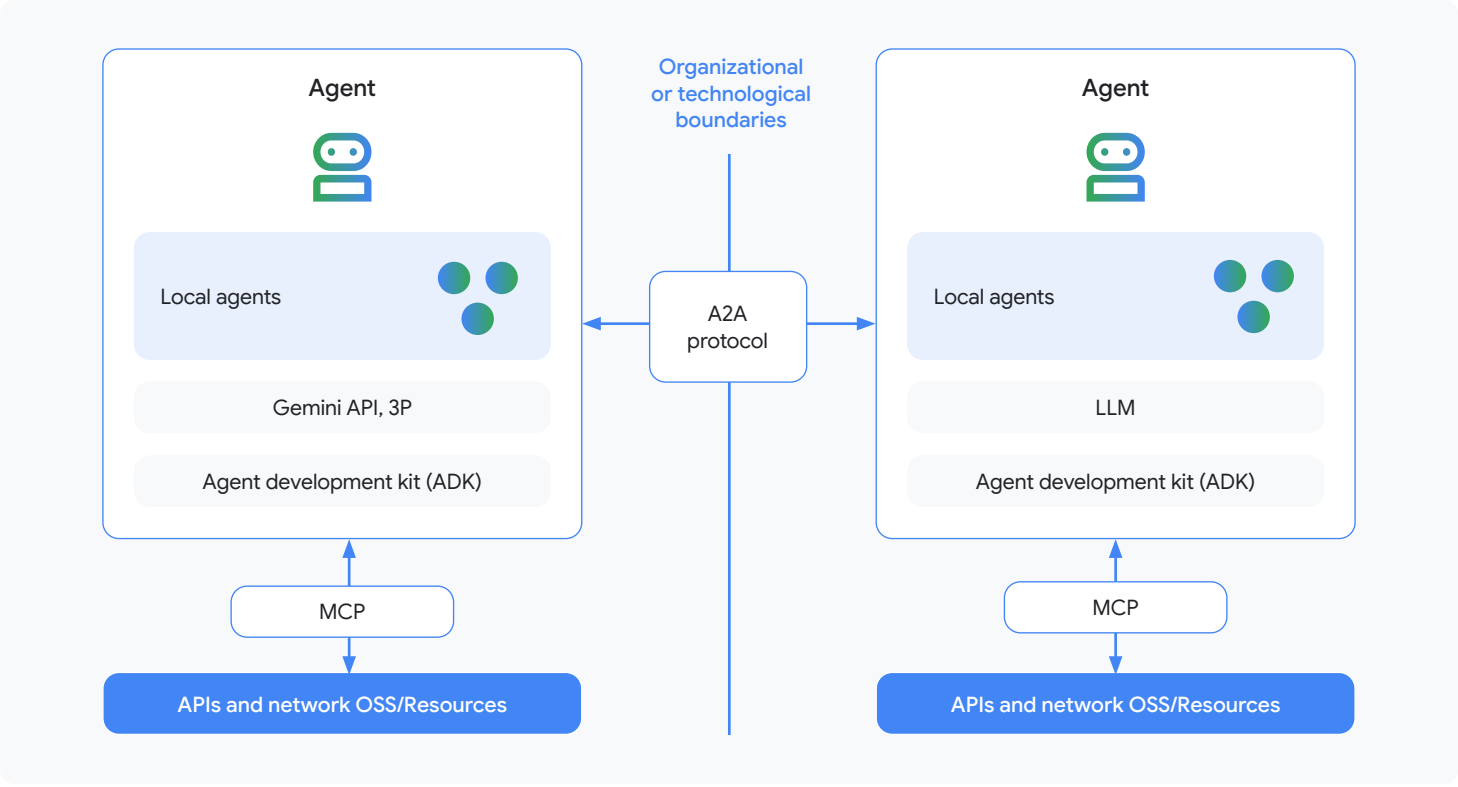


Figure 8: Agentic protocols within a multi agent system. Source: Google Cloud

3.3 Digital twin

As active contributors to TM Forum´s Digital Twin Classification and Functional Model workstream (IG1488), we see this virtual model of the network as the cornerstone to enabling, simulation, decision assurance, changes, and actions validation before execution—ensuring safe, effective, and policy-compliant autonomy.

The digital twin provides **simulation capabilities** that allow operators to perform sophisticated **'what-if' scenarios**, such as replaying traffic patterns or predicting the impact of routing changes to ensure accurate decision-making before any automated deployment is executed in the live network.

Integrating **digital twin** simulations into the **decide** phase is what transforms a network from being merely 'reactive' to truly 'proactive' and 'autonomous'. In a standard closed loop, the system must choose a remediation strategy; the digital twin acts as the safety net and validator for that choice.

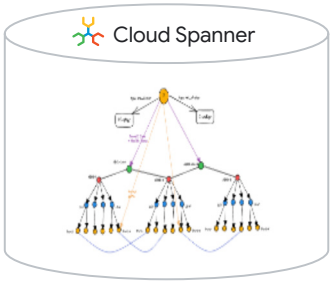
The transition from identifying a problem to taking action involves a high-speed validation step enabled by the digital twin:

- **Impact prediction**
  Once an anomaly is identified, the system proposes a fix. The digital twin uses a **Surrogate graph neural network (GNN)** (AI emulator) to predict the impact of that change on KPIs—such as latency, packet loss, and throughput—in milliseconds.
- **Conflict resolution**
  By simulating a proposed change (e.g., modifying a routing table), the digital twin ensures that an action intended to fix one area (like the RAN) doesn't inadvertently cause a bottleneck in another (like the transport or core).
- **Policy refinement**
  AI agents use the digital twin's feedback to refine multi-step action plans, choosing the strategy that best aligns with high-level business intents, such as "prioritize FWA users during a live sports event".

3.3 Digital twin

The journey to a fully capable digital twin is a progressive evolution of network modeling that requires gradually richer data, increased processing frequency, and advanced simulation capabilities. Digital twin development is typically broken down into three distinct stages:

Digital model (DM)
The foundation begins with a static representation of the network's physical or logical connectivity.

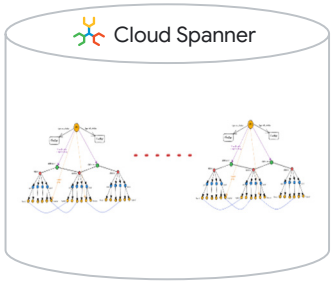


Attributes
It uses static attributes and is updated only occasionally (weekly or monthly), often through manual entry or partial automation from inventory databases.

Capabilities
Operators use this level to identify potential network paths between nodes and spot single points of failure in the topology.

Utility
It serves basic root cause analysis (RCA) by mapping error messages onto the network's known structure.

Digital shadow (DS)
The next step is a dynamic representation that enriches the static model with real-time operational data.



Attributes
It includes live metrics such as traffic levels, node status, log messages, and capacity parameters.

Processing
Updates are automated but typically performed in batch mode, occurring at a minute or hour pace.

Capabilities
The system can provide instant responses regarding network capacity (e.g., guaranteed bandwidth between two points) and identify over- or under-used oaths.

Digital twin (DT)
The final stage is a highly dynamic, virtual replica that achieves full synchronization with the live network

Attributes
It combines all digital shadow data with real-time status and deep context-related information.

Real-time streaming
Updates occur in a streaming mode at a near-instant pace (sub-minute) to preserve the exact sequencing of network events.

Sophisticated simulation
Its main feature is the ability to simulate the behavior of network equipment. This allows for advanced 'what-if' scenarios, such as replaying traffic patterns or predicting the millisecond-level impact of a routing change before it is deployed.

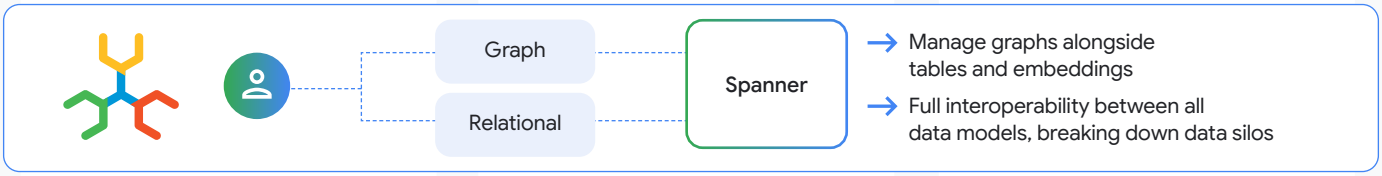


Figure 9: Digital twin development stages. Source: Google Cloud

3.4 Network data lake

A network data lake provides the high-fidelity, cross-domain data required to support AI-driven policy and conflict resolution by centralizing vast datasets that would otherwise remain siloed. This centralized intelligence is supported by three core pillars:

- **Curation of large-scale historical data**
  By maintaining a high-resolution historical record of network states across RAN, transport, and core domains, the data lake enables AI agents to distinguish between transient fluctuations and systemic issues. This deep history is essential for establishing the long-term context needed to resolve complex conflicts that span multiple operational layers.
- **Cross-domain trend analysis**
  The data lake facilitates the correlation of seemingly unrelated metrics, such as identifying how specific radio congestion patterns correlate with core network latency. Analysing these trends allow the 'cognitive brain' to identify the root cause of service degradation that a local rApp, limited to its own silo, would be unable to perceive.
- **Advanced ML model training**
  Access to high-fidelity, diverse datasets allows for the training of more accurate predictive models and digital twin emulators. These models learn the intricate dependencies of the network ecosystem, enabling proactive policy adjustments that anticipate and prevent potential conflicts before they impact the end-user experience.

A critical architectural decision is the split between the centralized data lake and on-prem data processing.

Data type and location	Rationale
High-frequency telemetry On-prem/edge	Processing raw network data or low-latency packet stats at the center can be cost-prohibitive and too slow for real-time loops
Cross-domain trends Data lake	To optimize an E2E slice (e.g., a 'gold' VR service), you need to correlate RAN signal strength with core latency. Only the lake sees both.
Historical baselines Data lake	AI requires 'normal' vs 'anomaly' training data. Storing 12 months of seasonal patterns on-prem is inefficient.
Real-time fault recovery On-prem	Local loops (e.g., sub-10ms protection switching) must happen near the hardware to meet strict latency service level agreements
Predictive maintenance Data lake	Analyzing slow degradation of hardware across 10,000 sites requires heavy compute and large-scale comparative analysis

3.5 Graph neural networks

Graph neural networks (GNNs) model a network as a graph where routers and interfaces are nodes and physical or logical connections are edges. By learning from historical telemetry, configuration state, and protocol events stored in the network's digital twin, GNNs can determine when the network deviates from a healthy state.

The main focus areas for applying GNNs are:

- **Failure pinpointing**
  uses GNNs to detect live or historical anomalies, isolate which layer of the stack (hardware, configuration, or protocol) is the root cause, and attribute the anomaly to a specific issue.
- **What-if simulation**
  uses GNNs to predict the downstream impact of a proposed change, topology modification, configuration update, traffic surge, or maintenance action before it is applied, giving operators a quantitative answer to "what would happen if...?" without touching the production network.

Failure pinpointing uses GNN-generated embeddings to detect when a device or interface has drifted from its learned healthy baseline and to explain precisely what caused the deviation.

Value of GNNs

The advantages of this approach include the ability to uncover latent misconfiguration errors and a more precise methodology for pinpointing faulty devices. This surpasses the capabilities of traditional tools, which often rely on manual investigation to correlate alarm storms.

What-if simulation uses GNNs to predict the impact of a proposed network change before it is applied.

Rather than detecting anomalies in current state, these models take a modified graph—a topology with a link removed, a config with an updated route policy, or a traffic matrix with a doubled load—and produce quantitative forecasts of service KPIs (reachability, latency, packet loss, throughput) under the hypothetical scenario.

GNN training

Training GNNs for these use cases requires a digital shadow with frequent snapshots of all network states. GNN development toolkits such as Google's Distributed Graphflow make it easy to build GNN models and access data with native integration to network data lakes, and graph databases such as Google's Spanner DB.

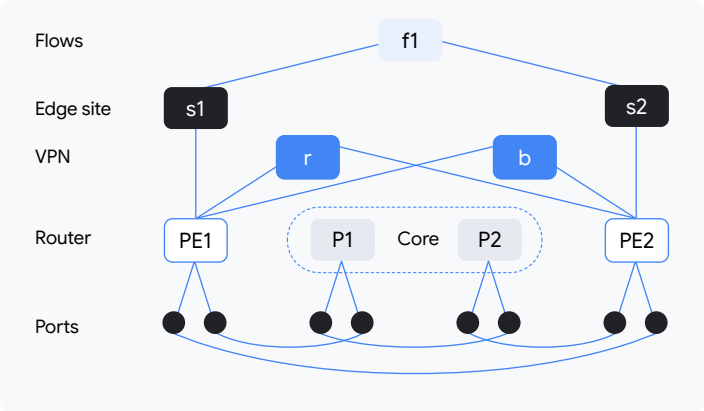


Figure 10: Example network graph. Source: Google Cloud

04

Deployment model

Closed-loop instances are deployed in a hybrid cloud manner, with entire closed-loop instances running within or alongside network domain controllers, cooperating with closed-loop instances running on a public cloud. A closed-loop instance can also run fully on a public cloud, such as Google Cloud, and leverage tasks that are running on network domain controllers.

4.1
Hybrid cloud deployment

Closed-loop deployment options are as follows:

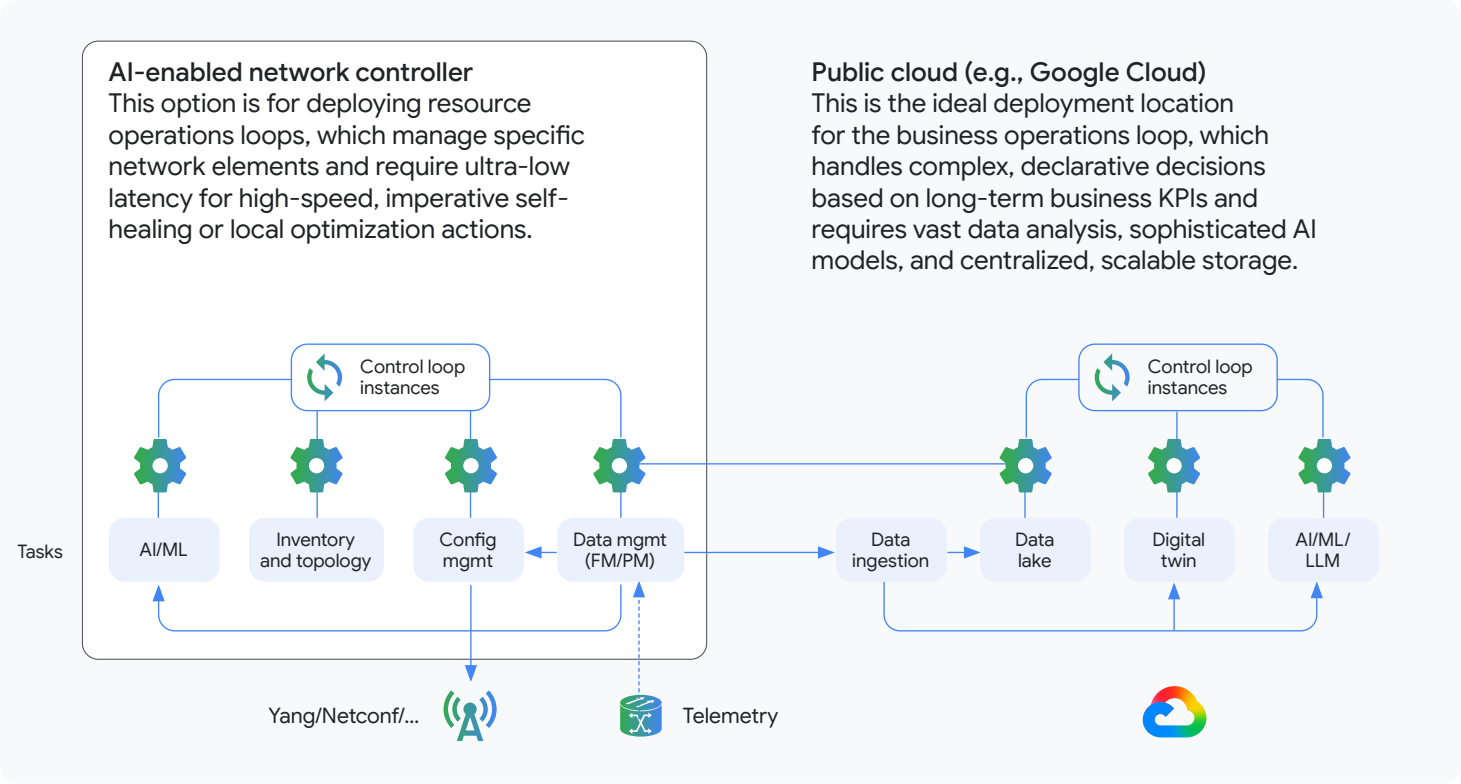


Figure 11(a): Deployment options (highlighted AI-enabled Network controller capabilities). Source: Google Cloud

Also, closed-loop instances can coordinate tasks that are running across multiple controllers and/or Google Cloud Platform. Network controller provides the following base capabilities:

Inventory
A complete, up-to-date catalog of all network assets, including hardware, software versions, and virtual resources. This provides the foundational knowledge base for all operational and planning tasks.

Data mgmt (data management)
The systematic collection, storage, processing, and governance of all network performance, fault, and configuration data. It ensures data is clean, accessible, and ready for analysis and decision-making systems.

Configuration
The authoritative function responsible for setting, changing, and verifying the parameters and state of network elements. It ensures the network operates according to design and intent.

AI/ML
The embedded platform and tools that:

- Run sophisticated models to enable network self-optimization, predictive fault analysis, and complex reasoning to drive autonomous actions.
- Deploy lightweight, pre-trained AI models or sub-agents that perform local analysis and decision-making for high-speed self-healing and optimization tasks.

4.1
Hybrid cloud deployment

Closed-loop deployment options are as follows:

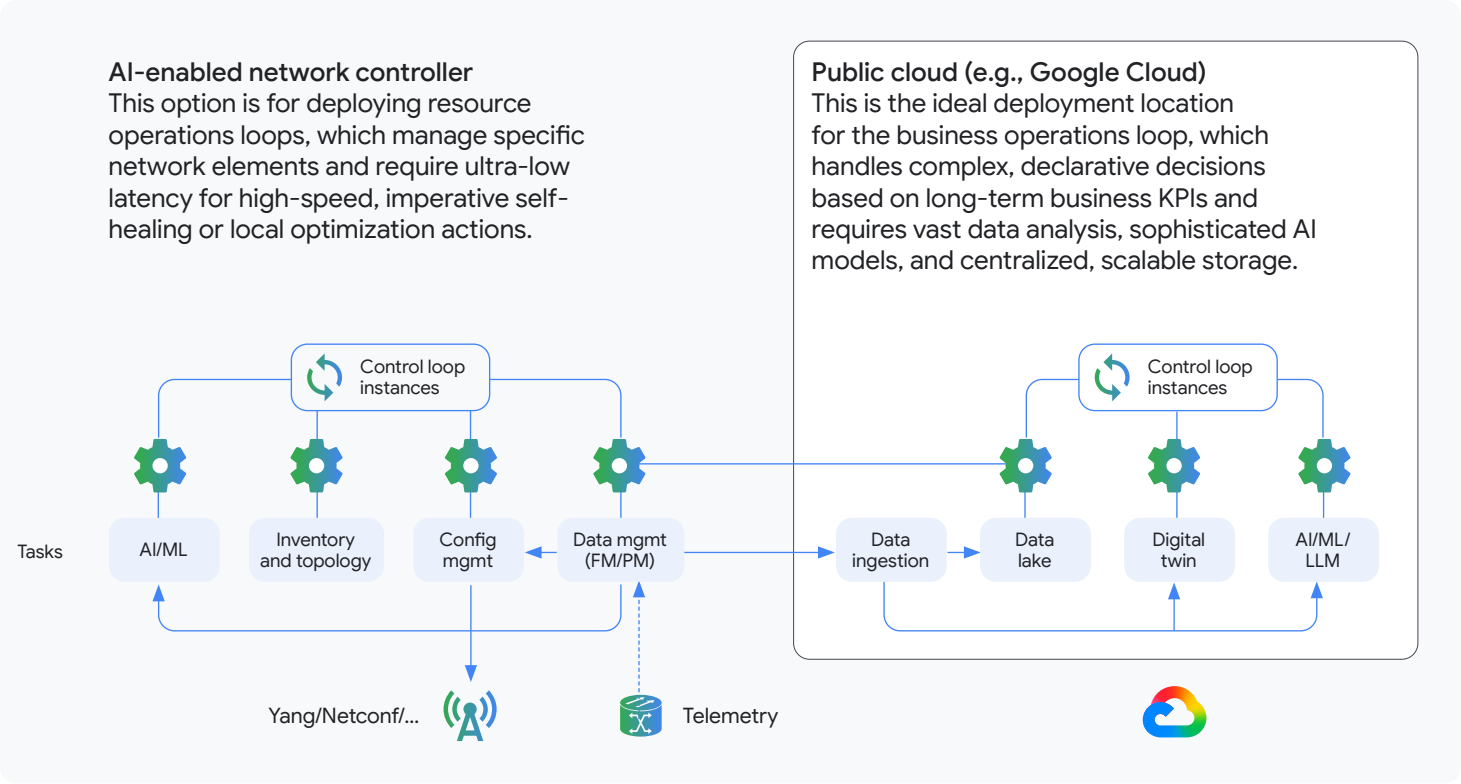


Figure 11(b): Deployment options (highlighting Public cloud capabilities). Source: Google Cloud

Public cloud capabilities in the context of autonomous networks:

Data ingestion
Services to capture and stream real-time, high-volume data (telemetry, logs, faults) from the network to the cloud platform, ensuring low-latency delivery and processing readiness.

Data lake
A massive, centralized, and scalable repository for storing all structured and unstructured network data (historical and real-time). It serves as the foundation for AI/ML training, long-term analytics, and auditability.

Digital twin
Advanced modeling and simulation capabilities to create a near real-time, dynamic virtual replica of the physical network. This enables complex 'what-if' scenario testing and accurate decision-making by closed loops.

AI/ML/agent
A suite of tools and platforms used to:

- Build, train, and deploy sophisticated models for predictive analysis, anomaly detection, capacity forecasting, and self-optimization within the autonomous network.
- Enable the creation, deployment, and management of AI agents. This provides the complex reasoning, planning, and tool-connection capabilities necessary for dynamic, multi-step autonomous actions.

4.2 Deployment rationale

The deployment location of a closed-loop instance, its tasks, and supporting technologies is a critical design decision based on a few key factors: **latency requirements, the scope of the decision, the technological capabilities of the platform, and data security and privacy.**

Deployment aligns with the **TM Forum loop-of-loops** structure, balancing response time with decision-making scope.

Closed loop layer	Deployment location	Rationale
Resource operations loop	**Network edge/ domain controller** (e.g., RAN, core)	These loops manage specific network elements (hardware/VNFs). They require **ultra-low latency** (near real-time) to perform imperative, high-speed self-healing or local optimization actions. The required data (telemetry, state) is local to the controller.
Service operations loop	**Hybrid/cloud-edge intermediary**	This layer coordinates services across multiple network domains. It requires medium latency and a broader view of the network state than a single domain controller. Decisions involve cross-domain coordination and delegation to lower-level loops.
Business operations loop	**Cloud (centralized)**	These loops focus on long-term business KPIs and customer experience (CX). Decisions are **declarative, complex, and non-deterministic**, requiring vast data analysis, sophisticated AI models, and long-term storage, which are best served by a centralized cloud data lake and AI/ML platform.

05

Security considerations

In this section we highlight some of the key considerations to ensure that autonomous networks are deployed in a way that is secure and compliant.

5.1
Infrastructure and data security

Securing the infrastructure and your data provides a foundational layer of trust.

Identity and access management
Implement least-privilege access for the autonomous network domain via granular identity and access management (IAM) for all closed-loop components, including rApps, AI agents, and orchestration engines. In Google Cloud, IAM is supported natively. Adopting agent identity enables AI agents to participate in IAM.

Data protection
All data should be encrypted at rest and in transit to ensure confidentiality and integrity of sensitive network data. Google Cloud implements encryption at rest and encryption in transit by default, offering native protection to the network and telemetry data in Cloud Storage, BigQuery, and Spanner.

Secure API gateways
Utilize robust API management to secure interactions between domain controllers and the cloud environment, preventing unauthorized command injection into the network. Google Cloud includes Apigee for native API management.

For more details about Google Cloud's infrastructure and data security, refer to:

- Google Cloud's infrastructure security design
- Google Cloud's enterprise foundation blueprint

5.2
AI and agentic security

Building on the secure foundation of infrastructure and data, the next challenge is to secure the AI models and AI agents in the cognitive layer.

This is another complex topic. The TM Forum is developing a detailed guide on this topic: GB1087: Agentic Interaction Security in Telecommunications

Some of the key considerations include:

Agent guardrails and human-in-the-loop
Implement explicit guardrails for AI agents to prevent unauthorized actions. Ensure high-stakes network reconfigurations require a 'human-in-the-loop' approval workflow (as established in the worked examples).

Model security
Protect AI models from prompt injection attacks, particularly when they ingest unstructured public data (e.g., social media feeds for event planning), and ensure that model output meets safety requirements. Tools such as Google Cloud's Model Armor play an important role here.

Tool usage security
Strictly scope the capabilities of the model context protocol (MCP) tools used by agents to limit their ability to execute commands outside of defined operational parameters. In Google Cloud, security policies for agents can be enforced by Agent Gateway.

For more details on AI and agentics security in Google Cloud, refer to:

- Google Cloud's secure AI framework
- Implementing SAIF in Google Cloud
- Building secure multi-agents systems in Google Cloud

5.3 Regulatory compliance and governance

Telecom providers are faced with a complex web of regulatory requirements, including:

- **Data protection laws**
  (e.g., GDPR and many similar regulations around the world)
- **Cyber security regulations**
  (e.g., EU NIS2, UK CSR)
- **Telecom-specific security regulations**
  (e.g., UK's Telecoms Security Act or India's Telecom Cyber Security Rules)
- **Global or regional security frameworks**
  (such as ISO 27001, NIST CSF, NIST 800-53), which may be voluntary or may be required under local laws

Some of the key themes that emerge from these regulations include:

Data sovereignty
Ensure network and telemetry data remains within specified regulatory boundaries during processing and storage (meeting data protection and local telecommunications data laws) and is managed by trusted partners in the ecosystem. Google Cloud's regional and multi-regional services enable autonomous networks to be implemented with support for data residency.

Observability and auditability
All actions within the autonomous networks solution, including all agentic actions, should be recorded to provide a robust audit trail for monitoring and post-incident analysis. This is a key requirement for many regulations. Google Cloud implements audit logs by default for all actions with cloud. For AI agents in Google Cloud, agent observability records all agent decisions, interactions, reasoning, and performance.

AI compliance
AI regulation is an evolving area. Google Cloud is certified against the ISO 42001 standard and has also been assessed as compliant with NIST AI RMF. Google has also committed to the EU AI Act Code of Practice.

Complying with these regulations and frameworks requires careful analysis, but is supported by Google Cloud's robust security posture and own compliance with many security standards.

For more details refer to:
Google Cloud's compliance resource center
Google Cloud's telecoms industry spotlight

5.4 Operational resilience

Security also requires resilience and operational safety; for example, specifying the availability criteria of the digital twin and ensuring it readily scales with ever-expanding network needs. These are important considerations, but also an important opportunity for autonomous network operations.

Fail-safe design
Design for 'fail-to-safe' states. If a closed-loop system (rApp or agent) loses connection or encounters anomalous telemetry, the network must revert to a pre-defined, stable baseline configuration rather than continuing with uncertain autonomous actions.

Simulation-first validation
Reinforce the role of the digital twin as a mandatory 'pre-flight' security check—ensuring proposed network changes are validated against safety policies before any live deployment.

Security benefits of digital twins
Implementing a digital twin of your operational network opens the door to additional security benefits, enabling cyber attacks to be simulated, security controls to be tested, BC/DR strategies to be evaluated, and incident response plans to be enhanced.

For more details, refer to:
Securing Telecom Networks with Digital Twins and Agentic AI

06

In summary

The capacity for **autonomous networks** to revolutionize the infrastructure of the global TM Forum community is immense. With STL Partners* estimating a per operator impact of circa $800M, the window to capture this trapped value is currently open.

Mastering the orchestration of **hierarchical closed loops** across disparate layers, while ensuring **cross-domain harmony** and strategic application of the 'right tool for the job' is vital. This approach facilitates the proactive execution required to realize the full potential of Level 4 autonomy and beyond at a global scale.

Get in touch

* Source: TELECOM Review "The Path to Autonomous Networks: AI and Automation"