Canonical

WHITEPAPER

AI infrastructure for telecom

How telecommunication companies can evolve their infrastructure, from network operations to AI-powered services

Contents

Contents.....2
Making AI work where the network already runs.....3
The challenge and the opportunity for telco infrastructure.....4
How to build an AI-ready telco architecture.....8
Turning AI into value.....15
Operating model and next steps.....20
Conclusion.....26
Citations.....27

Making AI work where the network already runs

Telcos don’t get a quiet sandbox for AI. The environments they work with every day are complex and do not sleep: there’s traffic on the network already, complex infrastructure spread across the country (or entire world), legacy systems that cannot be disturbed, and change windows that close fast. This means the traditional route to experimental, fast AI is too risky: after all, having to fix a small bug in a remote location can quickly become half a day of logistics. And people always remember the upgrade that looked harmless until it wasn’t.

Getting AI right in telco environments, however, carries significant benefits that make these hurdles worth overcoming: operators already sit close to users, devices, enterprises, and national-scale data flows. That position gives them a chance to become the first port of call for AI services that need more than remote compute: local infrastructure, trusted operations, network awareness, and integration from the device through to the platform. The next question is how to turn that position into AI services that can be run, trusted, repeated, and sold.

This whitepaper looks at that question from the infrastructure up. It begins where operators already live: promising AI trials on one side, live services on the other, and a familiar set of constraints in between. As mentioned, the challenge is considerable: in this world, change windows are tight, service assurance comes first. Data needs to be governed, and costs need to be understood. What’s more, when something breaks at the edge, recovery is rarely as simple as restarting a service.

From there, the paper follows the path from AI trial to production service: how data is found, moved, protected, and used; how platforms become visible and auditable enough to trust; how AI capabilities reach central and edge locations; and how ownership is handed to the teams that have to run the service after launch.

The paper also looks at value. AI can help operators improve planning, fault handling, energy management, maintenance, customer experience, and fraud detection. The same foundation can support new customer offers too, especially for organizations that need intelligence close to their sites, under local control, and run by a provider they already trust. That connection matters; internal efficiency and external revenue should not require two separate platforms, two operating models, and two sets of exceptions.

And finally, this paper keeps the spotlight on the less glamorous work behind production AI, and gives real answers to the hard questions that telcos face in building effective AI infra. Which parts of the platform really need to change? Where does open source give operators more breathing room? How should telcos operate AI infrastructure at scale?

By the end of this white paper, readers should come away with a clearer way to frame AI readiness in telecom, and build systems that capture the great opportunities offered by their unique position in the mass-market AI pipeline.

1 The challenge and the opportunity for telco infrastructure

Telco infrastructure is built for AI, burdened by history

When it comes to building AI infrastructure, Telcos have a unique challenge: they aren't starting from scratch. This means that whatever they want to build with this industry-changing technology, they need to do in a way that fits perfectly into their years of accumulated architecture already in motion: legacy network functions, VM-based VNFs, CNFs, OSS/BSS platforms, and IT systems that already span private and public clouds. That coexistence is not temporary clutter. It is the normal state of the modern telecom estate, and it persists because many network functions still need exceptional control, observability, fault tolerance, availability, and carrier-grade reliability.

At the same time, AI is no longer just a lab exercise or a promise in a slide deck. Across industries, organizations are using AI widely, yet most are still struggling to scale it across the business. Recent research[1][2] shows that nearly two-thirds of organizations have not begun scaling AI at the enterprise level, and that the main barrier is usually not model quality but fragmented data, governance, and the hard work of fitting AI into live workflows. Telecom inherits that same move from pilot to production, only under harsher technical conditions.

So the pressure is clear. The ideas are here. The pilots are here. The business demand is getting louder. But the limiting factor is now the foundation under the workload: data, compute, network paths, storage, and operations. What happens when a bursty, accelerator-heavy workload lands on an estate that is built for deterministic timing and five-nines service, and which is already doing several jobs at once? The risk is not limited to the AI service itself. Failures can spread into services that were never designed to fail together. To reduce that risk, telco operators need to understand three layers of complexity before they design AI-ready infrastructure: the infrastructure estate, the service model, and the organization that runs both.

Complexity is already built into telco infrastructure

That complexity shows up in three connected ways: in the infrastructure estate, in the service expectations placed on that estate, and in the teams responsible for running it. Each one matters on its own. Together, they explain why AI readiness in telecom is not just a question of adding more compute.

First, most telco infrastructure estates are hybrid already by necessity. Legacy functions have not vanished. A recent industry analysis[3] confirms that VM-based VNFs are being complemented by container-based CNFs, not cleanly replaced by them, and the broader telco cloud now also carries OSS/BSS and IT workloads that move between private and public cloud. In practice, the result is an environment where old and new systems must keep talking to each other while the architecture changes under load.

Second, the service model is strict. Telecom compute choices still have to account for deterministic performance, ultra-low latency, hardware acceleration, and specialized telecom protocols. In parts of the RAN, that gets severe fast: some lower layers work within timing windows measured in tens of microseconds, and carrier-grade uptime targets still shape where workloads can run and how much variation the platform can tolerate.

That discipline is a feature, not a bug. It is also why ordinary cloud assumptions break so quickly in telecom.

Third, the organization tends to mirror the stack. Network administrators, IT administrators, storage teams, security teams, AI developers, and production operations teams all touch different parts of the platform. Recent reference architectures for production AI[4] spell out those separate roles in detail, and GSMA Intelligence[5] points to the same weakness beside them: very few operators have a data layer that spans the whole business. The result is familiar to anyone who has worked inside an operator. The network team sees one truth, the IT team sees another, and the data needed for AI often sits in between somewhere between these two teams, half-visible and hard to trust.

That complexity is manageable while AI remains small. It becomes a production risk when AI workloads begin to compete with live services for data, compute, network capacity, and operational attention.

AI changes the physics of telco infrastructure

AI is even harder for telcos to implement because of its design and architecture: it does not arrive as one more application image to schedule on spare capacity. Instead, modern AI production environments are built from dense accelerator nodes, low-latency fabrics, high-speed east-west networking, scale-out storage, power, cooling, drivers, operators, and management software. That is why recent design guides[6] describe AI infrastructure as a co-designed hardware-and-software system rather than a standard data center extension. It behaves more like a supply chain than a single server: data has to arrive clean, move fast, and reach the right compute at the right moment, or the whole line backs up.

That stack pushes directly against classic telecom assumptions. Generative models produce outputs by predicting the next token from context, while telecom network functions are usually engineered around bounded, repeatable behavior and tightly controlled timing. At the infrastructure layer, distributed training and multi-node inference also create heavy east-west traffic for parameter synchronization, model sharding, and request routing. Current AI reference designs[7] isolate those GPU east-west fabrics for a reason, and modern inference platforms assume multi-node deployment once models outgrow a single GPU or node.

The data problem is just as sharp. Operators already generate a huge amount of service, network, and operational data, but that does not mean the data is AI-ready. Recent telecom analysis from McKinsey[8] indicates that a useful telco decisioning engine depends on four connected layers: data, analytics, decisioning, and channel execution, with commercial and service data combined into a single customer view.

Many telcos struggle to connect those layers in practice. Service data and commercial data often live in separate systems, data governance is patchy, and useful signals still need too much manual preparation before models can act on them. In the network domain, the challenge gets even more specific: telemetry that is fine for trend analysis may be too coarse for AI models that need micro- to millisecond context in live RAN environments.

The picture is more nuanced for generative AI. Production deployment in telecom network operations is still limited, but the issue is not shortage of useful data or credible use cases. GenAI fits the messy middle of OSS/BSS particularly well, where customer context, tickets, policies, inventory records, channel history, and service state all have to be coordinated. Its strength is adaptive orchestration: helping teams move from signal to decision to action without forcing every workflow into a fixed script.

Then there is the topic of accelerator operations. GPU support in Kubernetes still begins with node drivers and vendor device plugins. DPU orchestration adds another layer, with cluster-wide provisioning, firmware control, operating system consistency, runtime stack management, and service life-cycle management across the estate.

As a result, adding GPUs and DPUs is not just a hardware purchase: it is a platform change, with consequences for tooling, patching, observability, security, and day-two operations.

Still, the same friction that makes AI hard for telcos also points to where they can win.

The opportunity is already within reach for telcos

The inherent complexity of AI is a real opportunity for the unique systems and architecture of telecom enterprises. As mentioned above, the same qualities that make telecom hard also make it unusually relevant to AI. To name just a few:

- Operators already own distributed infrastructure near users
- Operators already run environments built for ultra-low latency, high bandwidth, and real-time access to radio and network information
- Operators already manage large, high-availability systems across broad footprints, with fiber, edge presence, space, and power in places that matter

If operators already sit on those assets, why should they settle for being only the transport layer in the AI economy? Running AI as part of the telco platform, rather than treating it only as a third-party service consumed over the internet, creates two clear paths to value: internally, it can improve how operators run networks and support customers; externally, it can help them meet growing demand for local, trusted, low-latency AI services.

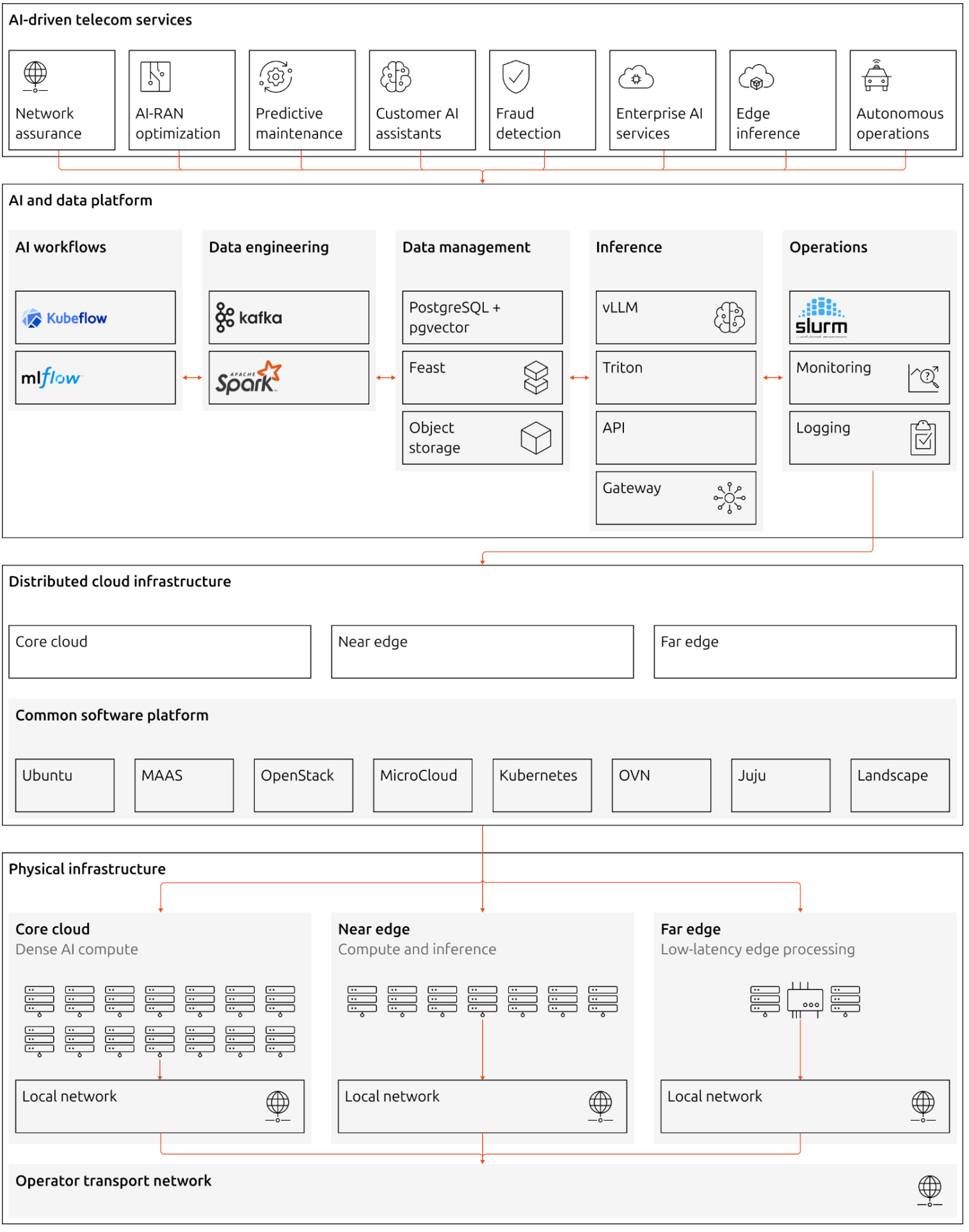
The internal value is straightforward. AI can improve network optimization, energy management, issue handling, and maintenance. It can push field work and maintenance away from reactive repair and toward earlier intervention. It can also strengthen trust services, where real-time fraud and scam signals matter more every quarter. Those are not side projects. They directly impact network quality, cost, resilience, and customer trust.

The external value is just as clear. Edge AI is moving from theory into service design, with operators using distributed inference and edge compute to reduce bandwidth strain, improve response times, support sovereignty needs, and create new enterprise offerings. That includes video analytics, industrial automation, robotics, AR/VR, and API-based fraud protection. Current telecom surveys[13] show that 60% of respondents are already using or assessing generative AI, 65% say AI is driving network automation, and operators are planning for growing AI traffic and higher-value distributed AI services. In other words, AI will clearly deliver significant financial, market, and operational advantages. It is taking shape. However, that opportunity will not materialize on ambition alone. It needs a foundation that can make AI repeatable.

But the telco infrastructure foundation has to change

That brings us back to the contradiction at the heart of this white paper: telcos are unusually well positioned to win in AI, but their current estates can also hold them back.When AI grows through isolated pilots, separate toolchains, and scattered data flows, the result is familiar: higher infrastructure and inference costs, more technical debt, weaker governance, and systems that are hard to trust in production. The issue is not lack of ambition, but instead structural friction.

So the answer is not to keep layering AI tools onto yesterday's estate and hope they cooperate, but instead to rethink infrastructure as a platform: shared, reusable, and built for production across data, networking, storage, accelerators, security, and operations. That is the real shift. AI readiness is less about adding intelligence at the edges of the system and more about rebuilding the system beneath it so models, agents, and network functions can run with the same rigor as any other critical telecom workload. The next section looks at what that means for architecture itself.


AI-driven telecom services
Network assurance
AI-RAN optimization
Predictive maintenance
Customer AI assistants
Fraud detection
Enterprise AI services
Edge inference
Autonomous operations
AI and data platform
AI workflows
Kubeflow
mlflow
Data engineering
kafka
Spark
Data management
PostgreSQL + pgvector
Feast
Object storage
Inference
vLLM
Triton
API
Gateway
Operations
slurm
Monitoring
Logging
Distributed cloud infrastructure
Core cloud
Near edge
Far edge
Common software platform
Ubuntu
MAAS
OpenStack
MicroCloud
Kubernetes
OVN
Juju
Landscape
Physical infrastructure
Core cloud
Dense AI compute
Local network
Near edge
Compute and inference
Local network
Far edge
Low-latency edge processing
Local network
Operator transport network

2 How to build an AI-ready telco architecture

Telco infrastructure was built around predictability: calls have to connect, packets have to arrive within tight service windows, and network functions have to behave in known ways, even when demand spikes or hardware fails. For decades, that shaped the architecture: isolation, deterministic behavior, strict operational control, and careful traffic engineering.

AI brings a different kind of workload into that environment. Training jobs, model serving, vector search, real-time analytics, and inference pipelines all behave differently from classic network functions. They are data-hungry, compute-heavy, and often probabilistic by design. A model may return a slightly different answer to the same prompt. A distributed training run may generate intense east-west traffic between accelerators. An edge inference service may need to sit close to the radio access network, the enterprise customer, or the data source[9].

That's where the friction starts. AI cannot simply be dropped onto an existing telco cloud as another workload class. The infrastructure may run it, at least in a lab. But running it reliably, across central sites and edge locations, with clear operational ownership and cost control? That takes a different architectural view.

The practical path is to evolve the telco cloud, rather than replace it. That keeps the architecture grounded in infrastructure operators already run, while extending it into a platform that can carry deterministic telecom workloads and AI services side by side.

Beyond the GPU cluster

A common mistake is to treat AI infrastructure as a GPU cluster. GPUs matter, of course: they provide the parallel compute needed for training, fine-tuning, and high-volume inference. However, while GPUs are often the most visible and expensive part of the design, they are only one layer in a larger system.

A production AI platform spans compute, networking, storage, cloud orchestration, data services, MLOps tooling, automation, and lifecycle management. Our related whitepaper, "Productizing AI: An Enterprise Guide to Private AI Infrastructure" [10], makes this point clearly: effective AI infrastructure is an integrated stack, not a collection of isolated components. It includes silicon compute, networking, container orchestration, data solutions, MLOps software, and the operating system that binds the hardware to the software layer.

In a telco environment, that stack stretches further. It has to reach data centers, regional facilities, aggregation sites, and edge locations. It has to support virtual machines, containers, physical servers, accelerators, and sometimes very constrained hardware. It also has to fit into operational models that already carry regulated, revenue-generating network services.

So the architecture question becomes simple: how do these layers work together without creating another silo?

A useful AI-ready telco architecture starts with a consistent substrate:

- Physical servers are provisioned through automation
- The host operating system supports the required silicon and drivers
- Virtualization and containers provide the right abstraction for each workload
- Kubernetes schedules cloud-native services and AI pipelines where it makes sense
- OpenStack continues to serve VM-based workloads and mature telco cloud patterns
- Lightweight edge cloud platforms support smaller, repeatable sites

That stack is useful only when each layer carries its weight. The first pressure point is compute, because AI changes what operators have to place, power, cool, and maintain across the estate.

Compute: central power, edge discipline for AI

AI-ready telco compute will not be uniform. That's uncomfortable, but it's also unavoidable.

Central AI clusters may use dense GPU nodes for training, fine-tuning, large inference services, and shared model platforms. These environments need high power density, careful cooling, fast storage, and very high-throughput networking. NVIDIA's DGX SuperPOD reference architecture, for example, combines DGX systems, Ethernet networking, InfiniBand networking, management nodes, and storage, with rack-level power consumption reaching about 56kW in one B300 design[10]. These clusters will often sit in national or regional data centers where telcos already operate large cloud estates.

At the edge, the design looks different. The constraint is not only compute power. It is space, energy, maintainability, and site access. A small edge site may need to run a radio function, an enterprise application, local analytics, and a compact inference service on the same footprint[12]. That changes the economics. Every watt and every operational touch matters.

CPUs still matter in this design. Many AI services need general-purpose compute for preprocessing, orchestration, API services, databases, observability, and control plane functions. GPUs and other accelerators handle the heavy parallel work. Neural processing units may appear in edge devices. DPUs and SmartNICs become relevant where networking, storage, security, or virtualization overheads need to move away from the CPU.

This is where hardware flexibility becomes more than a procurement preference. Telcos will need to mix silicon generations, vendors, and acceleration profiles across many locations. A single-site design might look elegant on a slide, but real networks are rarely that neat.

The architecture has to assume heterogeneity from the start. Once that compute is distributed across central sites and edges, the next constraint is clear: the fabric that has to connect it all.

Networking: AI stresses the fabric differently

Networking is the layer that often decides whether an AI platform operates efficiently or becomes painfully expensive to operate.

Cloud-native network functions are already sensitive to latency, jitter, throughput, and packet loss. Telco teams know this terrain. But AI workloads stress the network in different patterns. Distributed training can create intense east-west traffic between nodes. Inference services may need fast access to feature stores, vector databases, object storage, and streaming data. Model updates can move large artifacts through the platform. Real-time analytics can add constant pressure on transport and storage networks.

Our previous whitepaper[10] highlights a key issue in distributed AI training: collective operations such as all-reduce and all-gather can make the whole job wait for the slowest path. Tail latency matters because one delayed packet can reduce the efficiency of many GPUs.

That lesson applies directly to telco AI architecture.

A telco fabric built mainly for north-south application traffic may not handle AI clusters well. AI training and high-volume inference need low-latency, high-throughput east-west paths. Remote direct memory access (RDMA), RDMA over Converged Ethernet (RoCE), tuned Ethernet fabrics, or InfiniBand may be relevant depending on the cluster design. Congestion management becomes more than a nice tuning exercise: it becomes a cost issue because idle accelerators waste money quickly.

At the same time, telcos cannot design the AI fabric in isolation. The network still has to support existing traffic engineering policies, service separation, observability, and operational controls. AI traffic should not become the noisy neighbor that harms network functions, and network functions should not starve AI jobs either.

The practical answer is a clear architecture for workload placement, traffic separation, capacity planning, and lifecycle operations. Some AI workloads belong in central clusters. Some belong at the edge. Some should stay in the public cloud. The network design needs to make those choices explicit.

Those choices are easier to keep open when the platform itself is open. That is where architecture starts to become a question of control, not just performance.

Open source: real control over your AI infrastructure

Open source is sometimes framed as a philosophical stance. For telcos, it is often more practical than that. It gives operators a way to keep control over architecture choices while the AI market is still moving fast. New accelerators will appear. Model-serving patterns will change. Some workloads will fit private clouds, some will sit at the edge, and some will still belong in public clouds. An open foundation helps telcos adapt to those changes without tying the whole platform to one vendor roadmap, one silicon stack, or one deployment model.

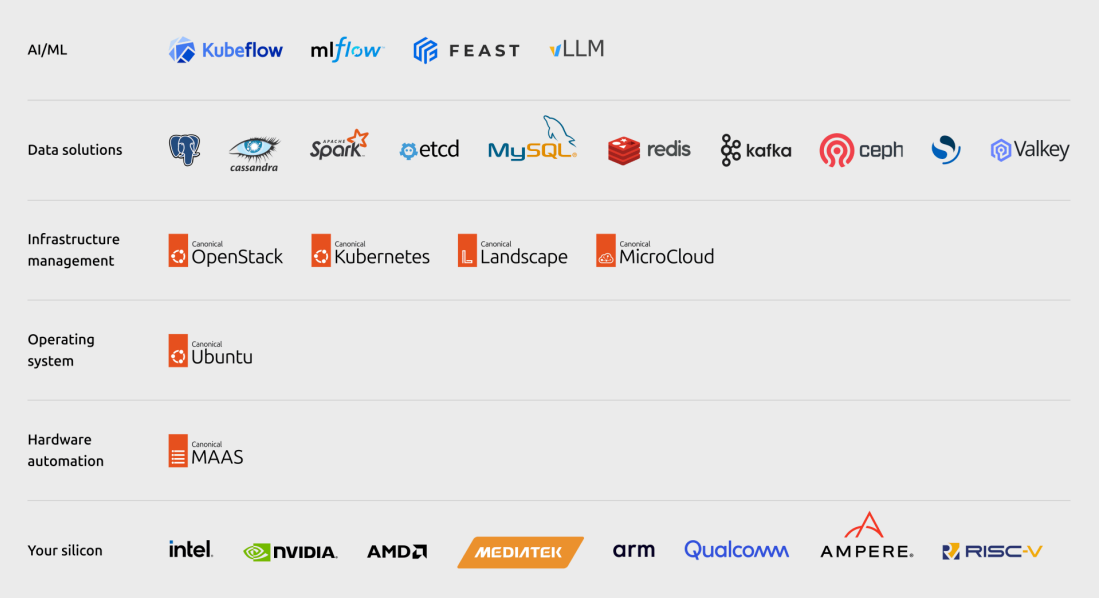
One can see why they adopt a more practical lens. After all, AI infrastructure will evolve quickly: sIlicon, model serving patterns, and regulatory expectations will change. The knock-on effect of this evolution is that edge requirements will change too, because enterprise customers will keep asking for more local processing, lower latency, and clearer data control. AI hardware and infrastructure are expensive, and they take time and care to build. Telcos therefore need to design for more than the first wave of AI workloads. They need a foundation that can adapt as the market changes, without turning every new model, accelerator, or deployment pattern into downtime, scope creep, or another budget shock.

In that context, open source gives telcos room to maneuver, due to its two unique and powerful value propositions.

The first value is transparency. Operators can inspect how the platform is built, how components behave, and how the software is maintained. That matters when AI infrastructure becomes part of regulated services, enterprise contracts, and critical network operations. It also fits the way Canonical develops: in the open, with source code, packaging, and engineering work available for scrutiny rather than hidden behind a paywall.

The second value is choice. Telcos rarely standardize on one workload type, one site profile, or one silicon vendor. They need room to run the right open source components for their architecture and the right hardware for each workload. That pragmatism shows up in the market too: in NVIDIA’s

2025 telecom survey[13], 40 percent of respondents said they planned to use open source tools for AI implementation. Canonical's AI infrastructure portfolio[14] is relevant here because it is designed around vendor neutrality: Ubuntu, MAAS, OpenStack, Kubernetes, MicroCloud, Charmed data solutions, and MLOps components can support different deployment patterns without forcing operators into a single proprietary stack.



However, hardware choice only helps if the software stack can use it well. Canonical works closely with silicon and hardware partners, including Intel, AMD, Arm, NVIDIA, and Qualcomm, to help operators adopt new AI infrastructure with fewer integration surprises. That work shows up in the details that make or break production: accelerator drivers, kernel support, firmware handling, GPU operators, and high-performance network fabrics such as RoCE and InfiniBand. The goal is simple enough to state, even if the engineering is not: help telcos extract reliable performance from new AI hardware while keeping the platform maintainable.

The sovereignty angle is important, but it should not be overstated. Open source does not make infrastructure sovereign by itself. It does, however, give operators more control over where software runs, how it is maintained, who has access, and how the stack is audited. For telcos building services for governments, regulated industries, and enterprises with strict data policies, that control can become part of the commercial offer.

The same logic applies to security. AI platforms need to be designed with security in mind from the host operating system through the orchestration, data, and MLOps layers. Ubuntu can provide the host foundation. MAAS can automate physical infrastructure. OpenStack, Kubernetes, and MicroCloud can support different cloud patterns, from central cloud to smaller edge sites. Charmed data and AI components can help operate databases, streaming platforms, and MLOps tools with less bespoke integration work. Ubuntu Pro can add commercial support and security maintenance across the stack.

That common foundation matters most where AI projects usually get messy: data, models, and the lifecycle between them.

Data and MLOps: the quiet center of the design

AI is often discussed as a compute problem. In production, it quickly becomes a data problem.

Telcos already hold rich operational data: network telemetry, performance counters, alarm streams, subscriber experience signals, inventory data, service assurance records, and customer interaction data. That data is valuable, but it is also fragmented. It often lives across network domains, IT

systems, data lakes, monitoring tools, and vendor platforms.

AI-ready architecture has to move data safely and usefully across those boundaries. Streaming platforms such as Kafka can feed near-real-time analytics and inference pipelines. Spark can process large datasets for training, feature generation, and reporting. PostgreSQL and other databases can support application state, metadata, and structured data. Object storage is essential for datasets, model artifacts, logs, and pipeline outputs. Vector search may become important for retrieval-augmented generation and knowledge systems.

Then MLOps adds the missing discipline. A model in a notebook is not a service. A model in production needs versioning, testing, approval, deployment, rollback, monitoring, and drift detection. Kubeflow can help orchestrate machine learning workflows. MLflow can track experiments and manage model lifecycle metadata. Slurm may remain relevant for large training environments, especially where batch scheduling and high-performance computing patterns are already familiar. Canonical's Charmed HPC supports Slurm-based clusters, including GPU-enabled workloads[15] with driver and workload-manager configuration handled through charms.

The data and MLOps layer also brings governance into the architecture. Which data trained the model? Which model version served the customer? Which team approved the deployment? How is model behavior monitored across regions? These are the kinds of questions AI governance frameworks, such as the NIST AI Risk Management Framework (AI RMF 1.0)[16], are designed to make operational. In telco, they can affect regulated services, enterprise contracts, and trust.

Model serving is where many GenAI projects become real infrastructure projects. A telco platform needs a repeatable way to serve large language models, route requests, manage accelerator capacity, and keep inference costs under control. Tools such as vLLM and llm-d fit into this layer: vLLM provides high-performance LLM serving, while llm-d brings a Kubernetes-native approach to distributed inference, including smarter routing and cache-aware scheduling. For telcos, that matters because OSS/BSS copilots, customer service agents, network assistants, and edge inference services all need the same basic discipline: predictable deployment, efficient serving, clear ownership, and day-two operations.

Once data and model lifecycle are part of the platform, the cloud layer has to carry more responsibility. It is no longer just a place to run workloads. It becomes the control surface for where AI runs, how it moves, and who can operate it.

Cloud and orchestration: keep the hybrid reality

Telco cloud is hybrid by nature. VM-based virtual network functions are still present. Containerized network functions continue to grow. OSS, BSS, IT platforms, data services, and customer-facing applications often share the broader infrastructure estate. AI adds another set of workload patterns to that already mixed environment.

That means virtualization still matters. OpenStack remains relevant where telcos need mature VM operations, tenant separation, and integration with existing cloud patterns. Kubernetes becomes central for cloud-native services, AI pipelines, model serving, and many MLOps components. Lightweight edge cloud approaches can help where a full cloud stack is too heavy for small sites.

The important point is balance. Kubernetes should not be treated as a cure-all. It is a powerful control plane for containers and AI tooling, but many telco workloads still need virtual machines or direct access to physical resources. Some AI workloads may also need physical server deployment for performance, accelerator access, or predictable scheduling.

Public cloud still has a role in this architecture. It may be the right place for early experimentation, occasional training bursts, access to specific managed AI services, or collaboration with partners already building in hyperscaler environments. The question is not whether telcos use public clouds. Public clouds are already part of the telco estate, especially in hybrid BSS/OSS hosting patterns. Omdia's 2025 survey[27] found that 42% of respondents preferred telco data center hosting with

public cloud available for bursting and disaster recovery, while another 21% preferred exclusively public cloud. The harder question is how to keep workload placement intentional, data movement controlled, and operations consistent across public cloud, private cloud, and edge sites. Our "Productizing AI" whitepaper[10] makes the same point in enterprise terms: hybrid deployment works when organizations choose the right platform for the right workload.

Operating across public cloud, private cloud, and edge sites makes interoperability a design requirement. AI platforms need common images, APIs, identity patterns, observability, policy controls, and data governance across environments. Kubernetes can help with portability, but it does not remove the hard parts: accelerator differences, network paths, egress costs, model provenance, and service ownership. For telcos, hybrid cloud only works when the platform team can explain where a workload runs, why it runs there, what data it touches, and how it is supported.

AI adds practical pressure at the orchestration layer. A platform that was already running VMs, containers, and edge services now has to place accelerator-heavy workloads, manage model-serving runtimes, keep data close enough to the workload, and still respect the change controls that protect live telecom services. The work is less about choosing one control plane and more about making several control planes behave predictably together.

In a private cloud, that starts with the physical estate. GPU nodes, DPU-enabled servers, storage systems, and compact edge machines need to be discovered, provisioned, rebuilt, and retired through repeatable workflows. MAAS is relevant in that private infrastructure context because it treats physical servers as programmable resources. That matters when a telco needs to add a new accelerator pool, rebuild a failed edge node, or keep firmware and host deployment patterns consistent across sites.

Above the infrastructure layer, the problem becomes application operations across environments. A model-serving stack may run on a private Kubernetes cluster today, move to a public cloud region for a short training run tomorrow, and later land on an edge platform for inference. Juju and charms help here because they encode deployment, integration, scaling, upgrade, and day-two operational logic in a way that can span private and public substrates. The aim is to build a process that results in fewer hand-built exceptions when the same AI service has to run in more than one place.

Canonical Landscape then fits the estate management side of the story: it delivers patch visibility, policy, audit reporting, and administration for Ubuntu systems. For telcos, those details are not housekeeping. They are what keeps an AI platform supportable once it spreads from a central lab into regional clouds and edge sites.

The product names are less important than the operating model. Telcos need infrastructure that can be rebuilt, patched, expanded, and audited without fragile manual work. That is the thread running through the whole architecture: compute, network, open source, data, and cloud choices only matter if teams can run them together. So the final design question is operational.

From architecture to operations

By this point, the architecture story has moved from hardware to networks, from open source control to data, and from cloud placement to automation. The common thread is clear: AI-ready infrastructure is platform discipline, rather than simply a single technical choice.

That discipline changes how teams work:

- NetOps teams bring knowledge of latency, traffic engineering, service assurance, and failure domains
- Platform engineering teams bring cloud operations, automation, lifecycle management, and workload placement
- Data and MLOps teams bring pipelines, experiments, model lifecycle, and governance

Production AI needs these teams to share the same operating surface, rather than hand work across organizational seams. O-RAN's SMO model points in a similar direction[17], bringing network functions, resource allocation, service provisioning, data collection, model training, deployment, and refinement into the management and orchestration layer.

That is easier said than done. The network fabric has to account for AI traffic without harming live network functions. The cloud platform has to expose accelerators without creating operational shortcuts. The data platform has to serve model teams without weakening controls over sensitive information. Edge sites have to follow repeatable patterns, even when they are small, remote, or intermittently connected.

Telcos already understand distributed systems, critical infrastructure, change control, and lifecycle discipline. They know what happens when maintenance is manual, when ownership is unclear, or when a site becomes a special case. AI makes that experience more valuable, because it adds new moving parts while keeping the existing service expectations firmly in place.

The architectural task, then, is to bring those capabilities together into one coherent platform: automated from the physical server up, open enough to support hardware choice, consistent enough to span central and edge sites, and governed enough to carry production AI services.

That is where the business case starts to sharpen. The same platform that supports reliable internal AI operations can also support customer-facing services: private inference, edge analytics, sovereign AI platforms, AI-enabled network APIs, and industry-specific applications that run close to where data is created.

So architecture becomes the hinge between operations and revenue. It decides what telcos can run reliably, what they can repeat across sites, and what they can sell with confidence.

Before we get to the operating model, though, there is one more question to answer: where does the return come from? AI-ready infrastructure only matters if it helps operators reduce cost, improve service quality, or create offers customers are willing to buy. The next section connects the architecture to value: internal gains from better operations, external opportunities around edge and private AI services, and the practical limits that decide whether AI becomes a platform business or another expensive pilot.

3 Turning AI into value

AI starts to matter when it stops being a lab project.

That sounds obvious, and yet it's where many telecom AI programs get stuck. A model can detect an anomaly in a notebook. A demo can show a field engineer a likely root cause. A chatbot can answer questions from a product catalog. Useful? Yes. Valuable? Maybe. The value appears only when the same capability works every day, across messy systems, distributed sites, shared teams, and real operational risk.

That is the telco version of the AI problem. It isn't a shortage of ideas. It's the hard work of making AI repeatable.

From AI ambitions to AI returns for Telcos

Telcos are under pressure from several directions at once. Connectivity remains essential, but it is hard to price as a premium service. 5G investment has created better networks, and yet monetization has been uneven. Energy costs keep rising and affecting margins. Operational complexity keeps growing. Meanwhile, enterprise customers expect low latency, sovereign services, and cloud-like consumption without cloud-like loss of control. GSMA Intelligence frames this AI discussion in practical terms[18]: operators are trying to link AI programs to commercial activity, return on investment, and strategy rather than broad innovation theater.

So AI has two jobs in telecom.

Inside the network, AI can do the unglamorous work that matters: spot faults earlier, shorten diagnosis, plan capacity with fewer surprises, trim energy waste, and make customer experience less dependent on heroic manual effort.

AI can also become part of what telcos sell. Not generic AI in the abstract, but localized AI infrastructure, inference services, private AI platforms, and industrial services tied to network reach and operational trust.

Here's the thing: those two jobs share the same foundation. The platform that runs AI for internal operations can also support external services, provided it is built with the right controls, economics, and lifecycle model.

Where AI earns its keep

For telcos, the first source of AI value is operational efficiency. This is where they already have a strong starting point. They understand telemetry, control loops, service assurance, field operations, and regulated uptime. AI can improve many of those functions, especially when it is connected to reliable data pipelines and clear automation boundaries.

Predictive maintenance is a good example. A model that spots early signs of equipment degradation can reduce truck rolls and service interruptions. But the model is only a small part of the system. The useful platform must collect telemetry from network functions, radios, transport, servers, power systems, and cooling. It must retain enough history to learn patterns. It must run inference close enough to the event to be useful. And it must integrate with operations support systems (OSS) without creating another island.

Anomaly detection is a little messier. Static thresholds still have their place, but they struggle when

“normal” changes by cell site, enterprise customer, hour, weather event, and whatever happened after the last maintenance window. TM Forum’s autonomous networks work[19] describes how operators are moving from manual automation toward closed-loop operations, with AI supporting self-healing, predictive maintenance, and service reliability improvements. The catch is usually upstream of the model. Without clean telemetry, enforceable policy, and orchestration that can act safely, anomaly detection becomes one more screen for an operator to watch.

Automated fault remediation needs even more discipline. A recommendation is one thing. A system that changes network state is another. Telcos will need staged control: human review at first, limited automation for low-risk actions, and stronger autonomy only where the evidence, rollback, and audit trail are mature. That puts lifecycle management right in the middle of the AI design. Models, prompts, and policies need release discipline. Every action needs a record. Every record needs to help answer the awkward question later: did this actually improve the service?

RAN optimization adds another layer. The O-RAN alliance’s work on the non-real-time RAN Intelligent Controller (Non-RT RIC), near-real-time RIC, and AI/ML workflows[20] points toward a radio environment where models support policy, optimization, and closed-loop control across multiple time scales. Academic work on AI/ML-driven SMO[17] also highlights the role of the Non-RT RIC in data collection, model training, deployment, and refinement. For infrastructure teams, this means AI can’t sit in a distant central cloud for every decision. Some inference belongs near the edge. Some training can run centrally. Some policy logic will live in between.

When AI becomes a control loop

When telcos talk about AI, the conversation often drifts toward copilots, chatbots, and generative AI. Fair enough. They’re visible, useful, and easy to demo.

But some of the most interesting telco AI will never write a sentence.

In the network, AI may be asked to make small, fast, high-stakes decisions: how to share radio resources between slices, when to move traffic, where to place workloads, or how to reduce congestion before users notice. This is where deep reinforcement learning (DRL) becomes relevant. One recent Open RAN paper[21] describes DORA as “a deep reinforcement learning (DRL) framework for dynamic slice-level Physical Resource Block (PRB) allocation in Open RAN”. It points to a future where AI helps tune the network after analyzing it.

There’s a catch, of course. Telco networks are not neat little lab environments. They’re distributed, noisy, and only partly visible from any one place. A RAN agent, a transport controller, and an edge placement system may each have a useful view, but not the whole picture. Groaner and Diepold[22] note that agents cannot observe the full environment or the internal knowledge of other agents.

That’s why multi-agent DRL (MA-DRL) is worth watching. It matches the shape of the problem: many domains, many local decisions, one shared service experience. It also raises the bar for infrastructure. MA-DRL doesn’t ask telcos to build a second AI platform from scratch. It asks them to extend the same AI-ready foundation toward closed-loop control: richer telemetry, safer experimentation, simulation-first testing, and stricter operational guardrails.

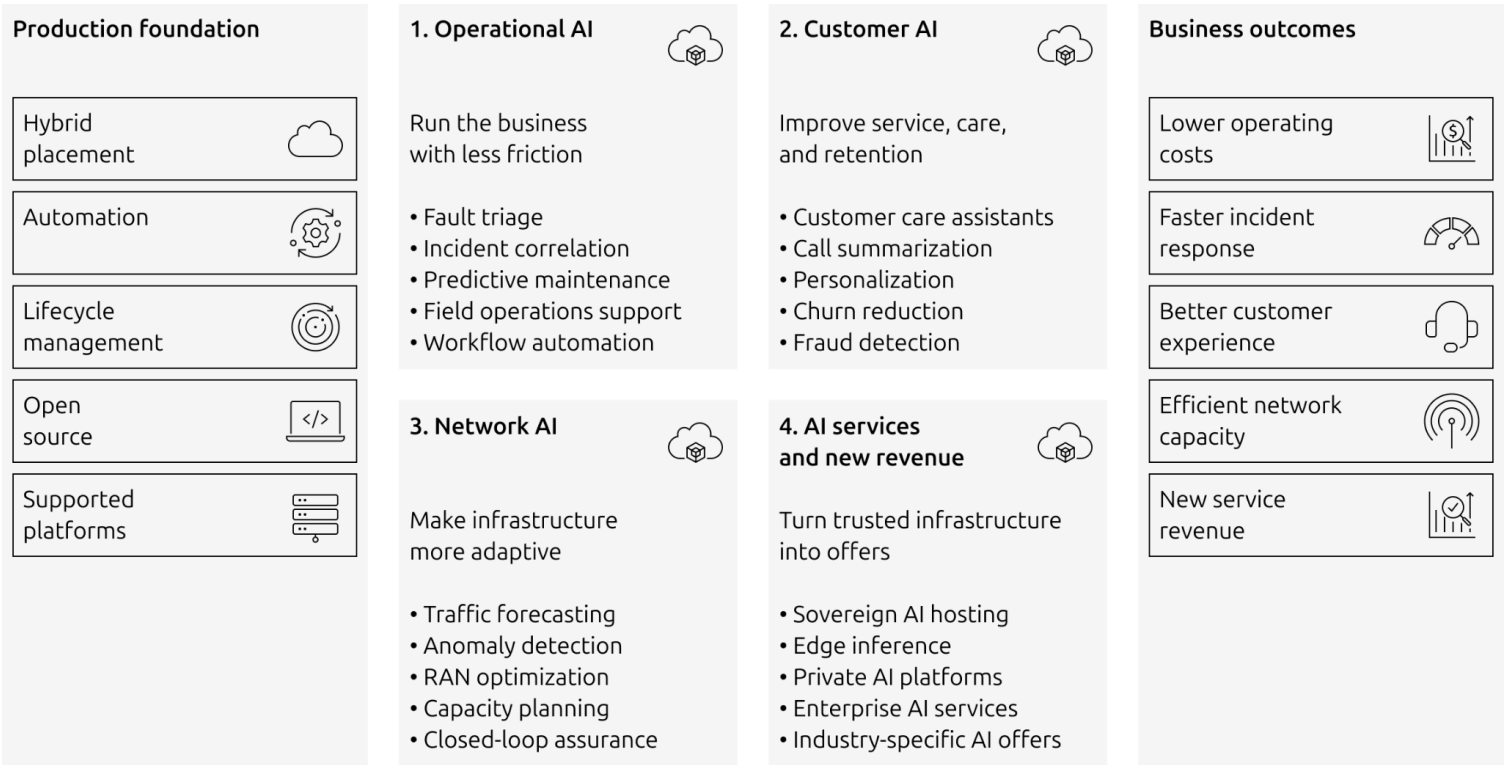
So, no, not all AI is generative. Some of it may look more like a very patient network engineer, testing millions of tiny decisions before making one that counts.

Energy is a good place to get practical. The bill runs through the RAN, transport, data centers, and edge sites, so even small gains add up. AI can forecast demand, nudge resources into better states, and find waste that static rules tend to miss. GSMA Intelligence has treated AI and energy efficiency as a dedicated area of research[23], which says something about where the pressure now sits: cost, sustainability, and operations are no longer separate conversations. For that to work, the AI system

has to know more than traffic demand. It needs power data, workload placement, live service context, and enough operational restraint to avoid saving watts at the expense of an SLA.

Customer experience analytics and capacity forecasting round out the picture. Both depend on connecting data that often lives in different places: network telemetry, customer tickets, service inventory, billing events, probe data, and application performance signals. AI can help find patterns across these domains. But it can't fix fragmented data ownership by itself. The platform needs shared data services, access controls, and a way to move from analysis to action.

In plain terms: internal AI value comes from closing the loop between data, inference, decision, and operation. The loop is the product.



Where AI becomes a service

The second source of value from AI is revenue. This is where the conversation gets more delicate.

Telcos should not try to become hyperscalers. That road is expensive, crowded, and shaped by economics that favor a small number of global platforms. Telcos have different assets. They have metro and edge sites, spectrum, transport networks, enterprise relationships, local operations, and experience running critical infrastructure. Those assets point to a more focused AI business: localized infrastructure that customers can trust.

Distributed inference is the clearest starting point. GSMA Intelligence has argued[9] that AI traffic is pushing operators to rethink the network edge, with distributed inference creating new demand for compute closer to users, devices, and data sources. For telcos, that creates a natural service pattern: place inference where latency, data gravity, cost, or sovereignty make central cloud less attractive.

Manufacturing is an easy example. A factory may need visual quality inspection, worker safety analytics, robotics support, or predictive maintenance on industrial equipment. Sending high-volume video to a distant region can be costly and slow. Running inference on a local edge platform tied to private wireless and local data policy is cleaner. The customer buys an outcome, but the telco provides the infrastructure wrapper: connectivity, edge compute, storage, lifecycle

operations, and support.

Logistics has similar needs. Ports, warehouses, rail yards, and distribution centers generate video, sensor data, and operational events. AI can help with routing, asset tracking, hazard detection, and congestion prediction. The useful service is rarely “AI” as a standalone SKU. It is a managed environment where applications can run near the operation with predictable latency and clear data boundaries.

Smart cities, healthcare, and retail analytics follow the same logic. Each has constraints around privacy, data location, service uptime, and local integration. A healthcare provider may want AI-assisted imaging or patient flow analytics inside a controlled environment. A city may want traffic analytics without shipping raw video out of jurisdiction. A retailer may want demand forecasting and computer vision close to stores. Telcos already sell to these customers. AI gives them a reason to have a larger conversation.

Sovereign AI is another strong fit. Enterprises and public sector organizations increasingly want control over where data is stored, processed, and audited. Canonical’s earlier private AI infrastructure whitepaper[10] makes a similar point for enterprises: cost, regulatory compliance, sovereignty, and the rise of AI factories are pushing many AI workloads back toward private and hybrid environments. Telcos can turn that same pattern into a service, especially where national or regional trust matters.

This is where the platform choice becomes commercial, not just technical. Enterprise customers will ask who controls the stack, where their data runs, how software is maintained, and whether the service can move with them as requirements change. An open source foundation helps answer those questions without forcing every offer into a single proprietary lane. Canonical’s AI infrastructure portfolio reflects that approach[14]: a supported stack spanning hardware automation, Ubuntu, cloud infrastructure, data services, and MLOps.

The reason to mention the stack here is to show what makes the business pattern repeatable: the same open operating foundation can support internal AI, private enterprise AI, and edge inference without rebuilding the service model each time. A service is easier to sell when the operator can repeat it, support it, and run it consistently from core data centers to edge sites.

Selling AI services means more than packaging compute and connectivity. Operators also have to prove that pilots can become services, and that services can survive change, drift, faults, audits, and the usual rough edges of distributed infrastructure.

Where AI has to grow up

Many telco AI pilots work. That’s the good news. The harder truth is that pilots often skip the parts that production cannot avoid.

Production AI needs lifecycle management. Models change. Data drifts. Software dependencies age. GPUs need scheduling. Drivers and firmware matter. Security maintenance has to cover the whole stack, not just the application layer. Network paths change. Edge sites fail in boring, inconvenient ways. Someone has to patch, audit, back up, restore, and explain the system when it behaves oddly at 02:00.

AI infrastructure behaves more like an industrial platform than a traditional IT environment. A factory doesn’t create value because it has one excellent machine; It creates value because materials, tooling, quality checks, maintenance, and logistics work together. AI is similar: data pipelines, model serving, accelerators, storage, networking, orchestration, observability, and governance all need to work as one operating model.

This is also where GPU economics become real. A telco can buy accelerators and still fail to create value if jobs are poorly scheduled, data movement is slow, or inference services sit idle. NVIDIA’s telecom survey[16] reports that many operators are already seeing AI affect revenue and cost, but

it also points to skills, implementation approach, and open source adoption as practical issues for the sector. That fits what infrastructure teams already know: expensive hardware needs disciplined operations.

Distributed AI also changes traffic patterns. Training may create intense east-west traffic inside a cluster. Inference may create many smaller flows across metro, edge, and central sites. Retrieval-augmented generation (RAG) may pull from vector databases, object stores, and enterprise systems. Real-time use cases may care less about average latency and more about the worst tail of delay. The network becomes part of the AI system, not just a pipe underneath it.

Governance becomes a production requirement. Telcos serve regulated customers and often operate under national critical infrastructure rules. AI services need access control, audit logs, data lineage, model provenance, and clear separation between tenants. Recent legal research on AI in telecommunications[24] warns that AI risk is often split across telecom, cybersecurity, data protection, and AI regulation, leaving governance gaps as AI moves into critical infrastructure. That is not a reason to slow down, but instead a reason to build the platform properly.

Make it repeatable, then make it profitable

The pattern across these use cases is simple: AI creates value only when the operating model can absorb it. A fault model, an edge inference service, and a sovereign AI offer may look different to the customer, but the platform questions are much the same. Can the service be deployed twice? Can it be patched? Can it be audited? Can it fail safely? Can the cost be explained?

That is the bridge between efficiency and revenue. Internal AI helps telcos learn how to run these systems under real network pressure. External AI turns that same discipline into something customers can buy: locality, trust, network integration, and controlled infrastructure.

The operators that win here will not be the ones with the loudest AI story. They will be the ones that turn scattered pilots into services with owners, economics, and repeatable operations. AI monetization depends on infrastructure discipline as much as model quality. Telcos already own many of the assets. The next step is to make those assets work as a platform.

4 Operating model and next steps

Most telcos already know how to run critical systems under pressure. They live with maintenance windows, incident bridges, field constraints, vendor handoffs, regulatory checks, capacity forecasts, and the slow archaeology of finding out what changed before the alarm storm began.

AI puts that operational muscle under a new kind of strain. The work now has to cover model versions, feature pipelines, embeddings, accelerator queues, prompt templates, data retention rules, and inference policies. The old disciplines still matter;They just have to reach further now.

That can sound like a technology list. In practice, it is a handover problem.

A lab team can keep a prototype alive through attention and goodwill. Production needs plainer things: named owners, repeatable deployment paths, rollback, audit evidence, support boundaries, failure drills, and a way for the night-shift operator to work out whether the fault sits in the platform, the data feed, the model, or the network path feeding it.

ETSI's work on zero-touch network and service management[25] describes comprehensive automation as both "vertically end-to-end," across layers from service to physical infrastructure, and "horizontally end-to-end," across technologies or administrative domains. That is a useful operational lens for AI. It says, in effect, that automation has to survive the places where teams, tools, and layers meet.

The real test begins when the first success has to be repeated.

The second deployment tells the truth

The early AI service often works because people quietly bend the organization around it. Someone knows which dataset was safe to use. Someone else remembers the GPU driver that worked. A third person knows which telemetry feed has the odd timestamp behavior. The service is successful, but only because it has a small human scaffolding around it.

That is fine for learning. As a pattern, it becomes risky.

The next service copies half of the setup. The third service uses a different runtime. The edge version has a local exception. The regulated customer version adds a manual approval step. Six months later, operations has a portfolio of almost-similar services and no clean way to answer basic questions. For example: Which ones are patched? Which ones can be rolled back? Which ones process sensitive data? Which ones depend on a deprecated stream? Which ones still have a named owner?

Operational debt tends to arrive quietly. It looks like a slower change review, a longer incident call, a nervous audit, or an engineer saying, "Let's leave that one alone before the weekend."

The practical move is to treat the *second* deployment as the test. When the second deployment needs the same people in the room, the same manual notes, and the same one-off fixes, the operating model needs more work.

That same need for repeatability leads to a sharper question: who owns each decision when AI services enter production?

Put ownership in the service path

Automation can make a clean process fast; but it can also make a vague process risky.

AI services create decisions that need an owner. Who can approve a model update that changes service behavior? Who can pause an automated recommendation loop? Who owns a feature pipeline that crosses OSS, BSS, radio, and transport data? Who decides whether an edge site may serve an older model while disconnected from the control plane?

These questions sound organizational, but they are operational control points. TM Forum's Autonomous Networks Implementation Guide[19] links higher autonomy to structured operating models for communications service providers. The same principle applies here: before autonomy expands, authority must be legible.

Every decision does not need a committee. Very frequently, too much committee becomes its own failure mode. A considered approach is recommended: define which decisions are local, which require platform approval, which require data governance review, and which are safe to automate because rollback and blast radius are understood.

A useful pattern is to write ownership into the service contract, rather than into a separate governance document nobody opens during an incident. The contract should state who owns the model, the runtime, the data feeds, the infrastructure pattern, the tenant policy, and the incident path. It should also state what happens when those owners disagree, given how production stress exposes ambiguity quickly.

Once ownership is clear, the next question is what operations teams need to know when the service behaves strangely.

Give runbooks model context

Traditional runbooks assume a fairly direct relationship between symptoms and systems. Packet loss points to a path. CPU pressure points to a host or workload. A failed process points to a service. AI makes some incidents less direct.

The service may be up, but the answer quality drops. Latency may be acceptable, but retrieval returns stale context. A model may behave differently when the pattern of its input features changes, such as the range, frequency, or mix of values it receives from live data. A rollback may restore the container image while leaving the model artifact untouched. Everything looks "green," and yet users get a worse service.

NIST's AI Risk Management Framework[15] treats measurement and management as continuous activities for trustworthy AI. For telco operations, that means the runbook must include model and data signals, as well as infrastructure signals. Accuracy drift, retrieval quality, feature freshness, model version, prompt version, and policy decisions belong near the same operational view as cluster health and network reachability.

The human side still matters, but the fix should stay practical. The on-call engineer should not have to become a data scientist at 02:00. They need enough context to triage: is this a platform incident, a data incident, a model incident, or a dependency incident? Then they need the escalation path to match that answer.

A small amount of shared language helps. "Model serving," "feature freshness," "data drift," "rollback," and "safe mode" should mean the same thing to the platform team, network operations, and MLOps team. Glossaries are rarely glamorous. They prevent long calls.

Term	Working definition for operations
Model serving	The runtime path that exposes a trained model to applications or users, usually through an API or service endpoint
Feature freshness	How current the input data is when it reaches the model, especially for telemetry, customer events, or service assurance signals
Data drift	A change in live input data patterns (statistical properties) that means the model is seeing different conditions from those it was trained or tested against
Rollback	A controlled return to a previous known-good model, runtime, configuration, or deployment state
Safe mode	A constrained operating state that keeps the service running with approved behavior while automation, connectivity, or data quality is degraded

That shared context is useful, but it still needs boundaries that teams can apply across uneven sites and platforms.

Define the operating envelope

Telco estates are uneven by design. Central facilities, regional sites, aggregation locations, and small edge footprints cannot all look the same. The mistake is to confuse unevenness with freedom.

An AI operating model needs a defined envelope. Your organization should be able to answer a few vital operational questions without turning every review into an investigation. Which hardware profiles are supported? Which accelerator generations are accepted for each workload class? Which runtimes are maintained? Which deployment patterns can be used without exception review? Which observability signals are mandatory? Which data classes may be processed at which locations?

The 2025 annual CNCF survey[26] reported that 82% of container users run Kubernetes in production, and that 66% of organizations hosting generative AI models use Kubernetes for some or all inference workloads. That makes Kubernetes a common tool in the operating envelope. The envelope itself is broader: firmware, drivers, host OS maintenance, storage behavior, network policy, identity, logging, and support expectations all belong there too.

Declarative operations help when they encode real decisions. GitOps workflows, infrastructure as code, and software operators can preserve intent, review history, and rollback paths. They can also preserve bad patterns when every team writes its own version. The benefit comes when teams agree on a few supported paths and make exceptions visible.

Canonical's portfolio can fit into that kind of model without forcing a single abstraction everywhere. MAAS can standardize server provisioning. Ubuntu provides the maintained host foundation. Landscape helps with estate administration and policy. Juju can operate complex software across substrates. Kubernetes, OpenStack, and MicroCloud can then serve different workload shapes. The useful idea is a common operating envelope with fewer surprises, not one tool for every problem.

Those boundaries become most important at the edge, where consistency has to survive distance, latency, and imperfect connectivity.

Edge operations need local continuity

Edge AI will fail in boring ways: A link drops. A site misses an update. A local data source changes format after a maintenance window.

That is normal field behavior, not an exotic edge case.

ETSI MEC requirements[12] emphasize local processing, low latency, and proximity to users or devices. Those requirements fit many telco AI services, but they also imply a tougher operational

rule: the edge site must keep behaving sensibly when the center is not immediately available.

“Local continuity” is a good way to think about it. The site should know what it can do safely while disconnected. It should know which model version is approved for local service, which logs to retain, which data cannot leave, which actions are paused without fresh policy, and how to reconcile state when connectivity returns.

Local continuity does not mean every edge site becomes autonomous. Too much local freedom creates drift. Too little local freedom creates fragility. The operating model has to define the middle ground.

A practical edge readiness test is blunt: disconnect the site, then watch what happens. Does inference continue? Does the service degrade safely? Do logs survive? Do queued events replay cleanly? Does the central view show that the site is out of sync? Can the site catch up without a manual rescue?

Mundane questions like these decide whether edge operations hold up.

Once edge behavior is defined, the operating model still has to prove what happened, where it happened, and who changed what.

Let the system produce the evidence

Maturity in AI operations should come from evidence generated during normal work. The platform should not rely on someone assembling proof after the fact.

A release should record which model moved, who approved it, which datasets or features it depends on, which sites received it, and what rollback target exists. A runtime should show which model served a request and under which policy. A data pipeline should expose freshness, lineage, and consumers. A fleet view should show patch status, accelerator availability, and configuration drift.

The Linux Foundation and CNCF survey report[26] connects higher cloud native maturity with standardized platforms and GitOps-style operations. For telco AI, the lesson is direct: evidence becomes cheaper when deployment paths are standard. It becomes expensive when every service is handcrafted.

Sovereignty makes this even more important. A telco serving regulated sectors may need to prove where data was processed, where inference ran, who changed a policy, and which software components were maintained. Those answers should come from the platform, not a heroic spreadsheet built during an audit week.

The same evidence helps operations too. Auditability and debuggability are cousins. Both need timestamps, ownership, change history, and a trusted source of truth.

With evidence in place, the final question becomes practical: what should teams change first?

Next steps that reduce future mess

The right starting point is the one that the organization can operate after the project team leaves.

Pick a real service with real data and a bounded blast radius.

The showcase use case may be the wrong choice when it requires too many special paths. A smaller service that exercises deployment, rollback, observability, data access, and incident ownership will teach more than a spectacular demo that nobody can run twice.

Define the production contract early.

It should cover owners, supported environments, data classes, approval gates, rollback, retention, service levels, and escalation. Keep it short enough that teams use it.

Create one standard inference pattern and one standard edge pattern before building a catalog.

The inference pattern should cover model packaging, serving, observability, policy, versioning, and rollback. The edge pattern should cover disconnection, local logging, policy sync, update windows, and recovery.

Shift observability left.

Instrument the service before rollout pressure begins. Include model behavior, data freshness, infrastructure health, accelerator state, and network dependencies. Waiting until the third incident is a familiar mistake.

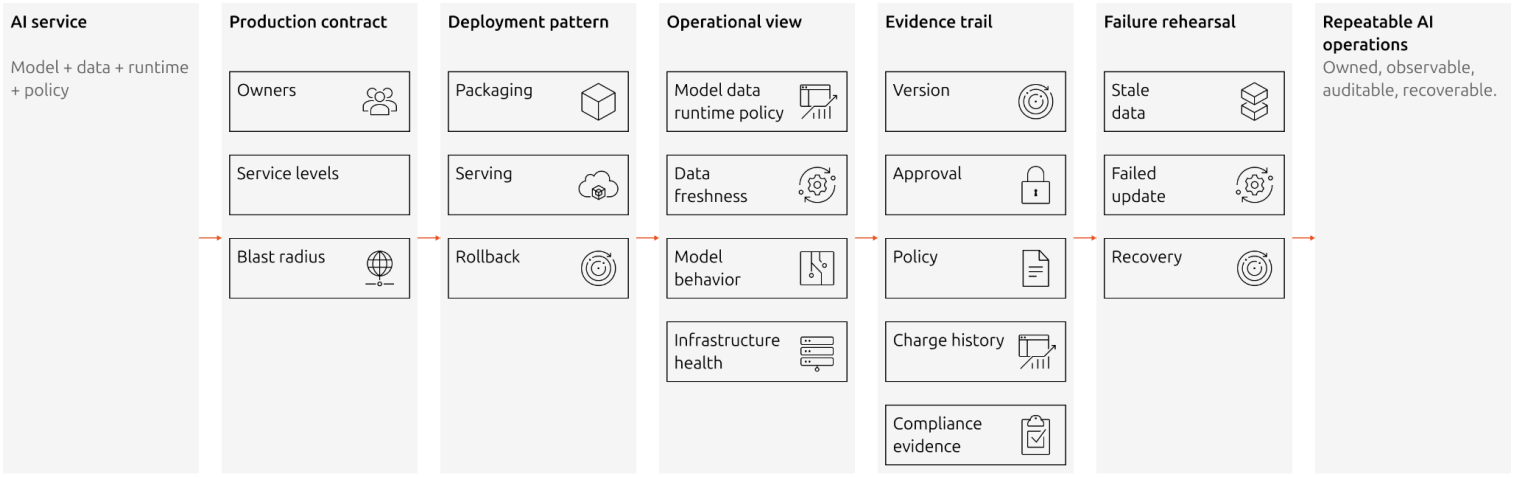
Don’t limit the scope to the happy path.

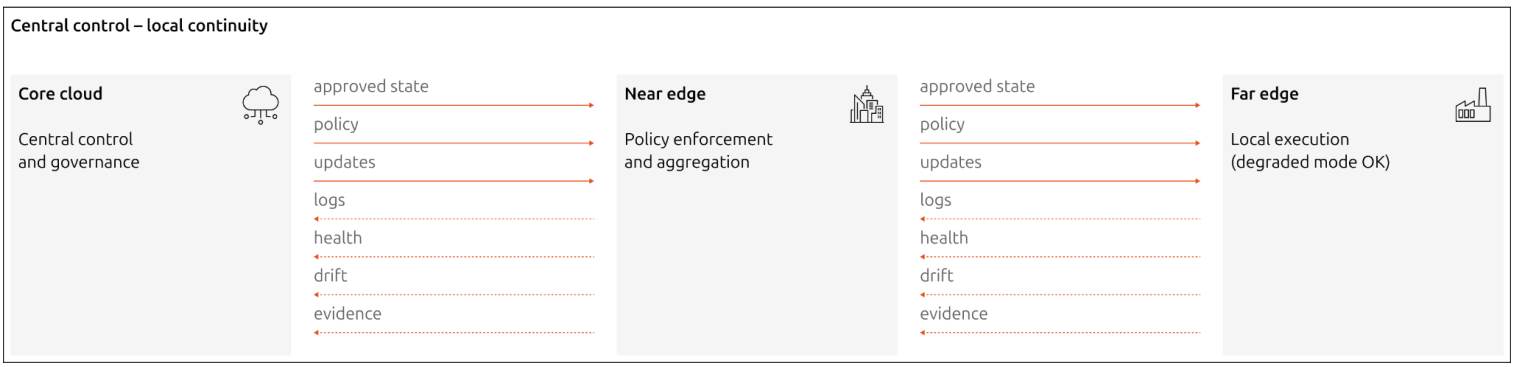
Test for awkward failures. Rotate certificates. Break a telemetry feed. Patch a GPU node. Delay a model update. Fill a queue. Disconnect an edge site. Then ask two questions: did the service fail safely, and did the right team know what to do?

Each operator will move at its own pace.

That is fine. The goal is not to look mature in a diagram. The goal is to make AI services easier to own, easier to change, easier to audit, and easier to recover.

That is what turns AI infrastructure into an operating capability. Quietly, steadily, and without adding another fragile exception to the network.


AI service
Model + data + runtime + policy
Production contract
Owners
Service levels
Blast radius
Deployment pattern
Packaging
Serving
Rollback
Operational view
Model data runtime policy
Data freshness
Model behavior
Infrastructure health
Evidence trail
Version
Approval
Policy
Charge history
Compliance evidence
Failure rehearsal
Stale data
Failed update
Recovery
Repeatable AI operations
Owned, observable, auditable, recoverable.


Central control – local continuity
Core cloud
Central control and governance
approved state
policy
updates
logs
health
drift
evidence
Near edge
Policy enforcement and aggregation
approved state
policy
updates
logs
health
drift
evidence
Far edge
Local execution (degraded mode OK)

Conclusion

AI is moving into the working fabric of telecoms: network planning, fault detection, customer care, enterprise services, and edge platforms. That puts new pressure on the infrastructure beneath it. The platform has to be open enough for innovation, steady enough for operations, and efficient enough to make the business case hold.

The strongest telco AI strategies will turn promising models into dependable services, close to the data and close to the customer. That takes more than GPUs. It takes automation, lifecycle management, open platforms, strong data foundations, and an operating model that engineering teams can trust on a Tuesday night when something breaks.

Canonical helps telcos build that foundation with trusted open source, from Ubuntu and Kubernetes to OpenStack, MicroCloud, MAAS, Juju, and the Data & AI portfolio. The goal is simple to state, but demanding to deliver: make AI practical for real telco environments, from central cloud to far edge.

Visit our website to learn more about Canonical's Data & AI and Telco solutions.

Further reading

Report: Gartner® Predicts 2026: AI Sovereignty
Whitepaper: Sovereign cloud: the essential guide for enterprises
Whitepaper: AI-Powered Edge Networking with Dell, Intel, and Canonical
AI Factories: How to architect bare metal for peak performance

Citations

[1] *The State of AI in 2025: Agents, Innovation, and Transformation*, McKinsey & Company, 5 Nov. 2025, www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai.

[2] Beharry, Ray. *Why Most Enterprise AI Projects Stall before They Scale*, IBM, 9 Apr. 2026, www.ibm.com/think/insights/why-most-enterprise-ai-projects-stall-before-scale.

[3] Attwood, Joseph, and Gorkem Yigit. "Cloud-Native Networks: Telecoms Operator Readiness and Strategies for Deployment and Operations." Analysis Mason, Dec. 2025, www.analysysmason.com/research/content/perspectives/cloud-native-deployment-strategies-rma16/.

[4] "Agentic AI in the Factory." *NVIDIA Enterprise AI Factory Design Guide White Paper*, NVIDIA, 16 Mar. 2026, docs.nvidia.com/ai-enterprise/planning-resource/ai-factory-white-paper/latest/agentic-ai-in-the-factory.html.

[5] Putcha, Shiv. *The Long and Winding Road to Agentic AI for Telcos*, GSMA Intelligence, 25 Nov. 2025, www.gsmaintelligence.com/blogs/the-long-and-winding-road-to-agentic-ai-for-telcos.

[6] "Enterprise AI Factory Overview." *NVIDIA Enterprise AI Factory Design Guide White Paper*, NVIDIA, 16 Mar. 2026, docs.nvidia.com/ai-enterprise/planning-resource/ai-factory-white-paper/latest/ai-factory-overview.html.

[7] "Networking Physical Topologies." *NVIDIA HGX AI Factory*, NVIDIA, 19 Feb. 2026, docs.nvidia.com/enterprise-reference-architectures/hgx-ai-factory/latest/networking-physical-topologies.html.

[8] *Why Ai-Enabled Customer Service Is Key to Scaling Telco Personalization*, McKinsey & Company, 24 Oct. 2023, www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/why-ai-enabled-customer-service-is-key-to-scaling-telco-personalization.

[9] Hatt, Tim, and Jarich, Peter. *Distributed inference: how AI can turbocharge the edge*, GSMA Intelligence, Mar. 2025, www.gsmaintelligence.com/research/distributed-inference-how-ai-can-turbocharge-the-edge.

[10] Cooper, Freyja. *Productizing AI: An Enterprise Guide to Private AI Infrastructure*, Canonical Ltd., 24 Feb. 2026, ubuntu.com/engage/open-source-ai-infrastructure.

[11] "DGX SuperPOD Architecture." *NVIDIA Docs*, NVIDIA, 2025, docs.nvidia.com/dgx-superpod/reference-architecture/scalable-infrastructure-b300-xdr/latest/dgx-superpod-architecture.html.

[12] "Multi-access Edge Computing (MEC); Use Cases and Requirements." *ETSI GS MEC 002 V4.2.1*, European Telecommunications Standards Institute, May 2026, www.etsi.org/deliver/etsi_gs/MEC/001_099/002/04.02.01_60/gs_MEC002v040201p.pdf.

[13] *State of AI in Telecommunications: 2025 Survey Report*. NVIDIA, 2025, www.nvidia.com/en-us/lp/industries/telecommunications/state-of-ai-in-telecom-survey-report/.

[14] *Canonical AI Infrastructure Solutions*, Canonical, Ltd. canonical.com/solutions/ai/infrastructure.

[15] "GPUs." *Charmed HPC Documentation*, Canonical, Ltd. documentation.ubuntu.com/charmed-hpc/latest/explanation/gpus/.

[16] Tabassi, Elham. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. National Institute of Standards and Technology, 26 Jan. 2023, doi.org/10.6028/NIST.AI.100-1.

[17] Habibi, Mohammad Asif, et al. *Towards an AI/ML-Driven SMO Framework in O-RAN: Scenarios, Solutions, and Challenges*. arXiv, 8 Sept. 2024, arxiv.org/abs/2409.05092.

[18] *Telco AI: State of the Market, Q4 2024*, GSMA Intelligence, Jan. 2025, www.gsmaintelligence.com/research/telco-ai-state-of-the-market-q4-2024.

[19] *IG1443 Autonomous Networks Implementation Guide v7.0.0*, TM Forum, 1 Jul. 2025, www.tmforum.org/resources/how-to-guide/ig1443-autonomous-networks-implementation-guide-v7-0-0/.

[20] *O-RAN Alliance Introduces 48 New Specifications Released since July 2021*, O-RAN Alliance, 22 Oct. 2021, www.o-ran.org/blog/o-ran-alliance-introduces-48-new-specifications-released-since-july-2021.

[21] Dorcheh, Alireza Ebrahimi, Tolunay Seyfi, and Fatemeh Afghah, *DORA: Dynamic O-RAN Resource Allocation for Multi-Slice 5G Networks*, arXiv, 8 Sept. 2025, doi.org/10.48550/arXiv.2509.07242.

[22] Gronauer, Sven, and Klaus Diepold, *Multi-Agent Deep Reinforcement Learning: A Survey*, Artificial Intelligence Review, vol. 55, 2022, pp. 895-943. Springer, doi.org/10.1007/s10462-021-09996-w

[23] *Telco AI: State of the Market, Q3 2024*, GSMA Intelligence, 9 Oct. 2024, www.gsmaintelligence.com/research/telco-ai-state-of-the-market-q3-2024.

[24] Agarwal, Avinash, Peeyush Agarwal, and Manisha J. Nene, *AI Regulation in Telecommunications: A Cross-Jurisdictional Legal Study*, arXiv, 3 Dec. 2025, arxiv.org/abs/2511.22211.

[25] "Zero-touch Network and Service Management (ZSM); Means of Automation." *ETSI GR ZSM 005 V1.1.1*, European Telecommunications Standards Institute, May 2020, www.etsi.org/deliver/etsi_gr/ZSM/001_099/005/01.01.01_60/gr_zsm005v010101p.pdf

[26] *The CNCF Annual Cloud Native Survey: The Infrastructure of AI's Future*, Cloud Native Computing Foundation, 20 Jan. 2026, www.cncf.io/reports/the-cncf-annual-cloud-native-survey/.

[27] Crawshaw, James, *Telco Software Evolution Survey*, Omdia, 13 June 2025, omdia.tech.informa.com/om127723/telco-software-evolution-survey--2025.

© 2026 Canonical Ltd.

Ubuntu, Kubuntu, Canonical and their associated logos are the registered trademarks of Canonical Ltd. All other trademarks are the properties of their respective owners. Any information referred to in this document may change without notice and Canonical will not be held responsible for any such changes.