International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.3 (2026) pp. 194-204
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

GOVERN-CTRL: A Governance Control Plane Architecture for Production-Scale Data and Machine Learning Pipelines

Sunil Kumar Vytla*

Independent Researcher, USA
* **Corresponding Author Email:** reachskvy@gmail.com- **ORCID:** 0000-0002-5247-7722

Article Info:

DOI: 10.22399/ijcesen.5394
Received : 26 May 2026
Revised : 05 July 2026
Accepted : 06 July 2026

Keywords

Governance Control Plane;
Machine Learning Pipelines;
Policy Checkpoint Coordination;
Compliance Evidence Collection;
Lineage Validation

Abstract:

Governance over enterprise AI delivery pipelines has long suffered from a fragmentation problem that most organizations recognize but struggle to fix. Approval processes are owned by different teams, audit trails live in incompatible systems, and compliance checks are embedded into individual pipelines without any shared standard to anchor them. GOVERN-CTRL is a governance framework designed specifically to break this pattern. Its core idea is straightforward: move governance orchestration out of individual pipelines and into a dedicated control layer that all pipelines report to. Policy checkpoints, lineage validation, approval routing, and compliance evidence collection all happen through this central layer, which means the same standards apply everywhere regardless of which pipeline framework is running underneath. Centralized governance orchestration of this kind has measurable consequences—approval fragmentation decreases, audit trails become coherent end-to-end, and compliance enforcement no longer depends on every team independently maintaining the same standards. Organizations previously requiring four to six weeks to assemble audit packages from fragmented pipeline logs can reduce that process to hours through unified evidence indexing; approval cycle times similarly contract when escalation is managed centrally rather than left to individual pipeline conventions. In representative deployments spanning heterogeneous pipeline environments, structured approval routing reduces median approval cycle times by more than 60 percent relative to pipeline-native implementations; policy updates that previously required multi-team coordination to propagate take effect across all connected pipeline stages in under five minutes through a single Policy Registry update. The deeper architectural value is that governance logic becomes genuinely separable from execution logic, which lets organizations update compliance requirements without touching the pipeline code.

1. Introduction

Getting machine learning into production is not primarily a modeling challenge—it is an operational and governance challenge. Data has to be traced from source to artifact. Model deployments have to clear approval gates. Compliance evidence has to be collected in formats that mean something to regulators. When AI delivery systems span feature stores, training clusters, validation frameworks, and deployment orchestrators owned by different teams, each of those teams tends to govern its pipeline in its own way [1].

The result is familiar: fragmented approvals, lineage records that stop at team boundaries, and compliance documentation that has to be assembled manually whenever an audit arrives. The problem is not that individual teams are governing wrong. The problem is that there is no shared governance layer—each pipeline handles compliance according to its conventions, and there is nothing to enforce consistency across the boundaries between them [2].

This is the fragmented pipeline silos problem: each team governs locally, and enterprise-wide coherence is a fiction. GOVERN-CTRL replaces this with a single pane of governance, one control layer through which all pipeline environments surface compliance decisions, approval states, and audit evidence. Regulatory frameworks like the NIST AI Risk Management Framework, ISO/IEC 42001, and the EU AI Act have started making this problem consequential in new ways. They require

auditable, end-to-end governance coverage across the complete AI lifecycle [3]. That is a requirement that pipeline-embedded compliance controls, however well designed individually, cannot satisfy collectively.

GOVERN-CTRL addresses this challenge by drawing on a design principle borrowed from distributed systems: separate the control layer from the execution layer. This separation in networking led to the development of software-defined architectures, allowing for centralized policy management without affecting individual forwarding devices [4]. In container orchestration, this separation resulted in systems that independently manage scheduling and health, regardless of the workloads they oversee. GOVERN-CTRL applies the same logic to AI governance—pipeline stages invoke standardized interfaces to request governance decisions, and a dedicated control plane provides those decisions based on centrally managed policy, without any governance logic living inside the pipelines themselves.

The remainder of this article proceeds as follows. Section 2 reviews the background and related work. Section 3 presents the architecture. Sections 4 and 5 describe governance orchestration and compliance evidence components. Section 6 covers enterprise integration and deployment. Section 7 evaluates the architecture. Section 8 discusses limitations and implications, and Section 9 concludes.

2. Background and Related Work

2.1 Pipeline-Native Governance Limitations

The most common approach to AI pipeline governance today is also the most brittle at scale: compliance logic is written directly into pipeline definitions. MLOps platforms provide the hooks—logging, versioning, and checkpoint callbacks—and individual teams use them according to local conventions [5]. For a single team maintaining a single pipeline, this approach is workable. Across a large enterprise running dozens of pipelines in multiple frameworks, it produces a governance environment that is difficult to reason about and nearly impossible to audit holistically. Part of what makes the process hard is a provenance problem. Lineage records generated by a feature store use a different schema than lineage records from a model registry, which differ again from those produced by a training orchestrator [6]. Connecting these records into an end-to-end provenance trace requires knowing where each system stores its data, how the schemas relate, and which records correspond to which pipeline run—none of which is standardized. When a compliance team needs to demonstrate provenance coverage for a regulatory audit, they are typically reconstructing a picture from fragments rather than querying a coherent record. The organizational dimension compounds this. Teams building data pipelines, teams building model training infrastructure, and teams building deployment tooling operate with different toolchains, different governance conventions, and often different interpretations of enterprise compliance requirements [7]. Pushing a policy update out to all of them simultaneously—so that a new data classification rule or an updated approval threshold takes effect consistently—requires coordination that rarely happens cleanly. The practical result is policy drift: the enterprise has one compliance standard on paper and several in practice.

2.2 Control Plane Separation in Distributed Computing

Software-defined networking demonstrated that moving control logic out of individual network devices and into a centralized controller does not just simplify management—it changes what is governable [8]. Policies that would require touching every device individually can instead be applied once, centrally, and propagated. The same principle appeared later in cloud application modeling and —execution frameworks, which separated resource lifecycle management from application workload logic so that infrastructure policies could be enforced consistently without being embedded in each application [9].

Data governance work has explored a version of this idea through intelligent data lake systems, which centralize metadata management, schema registries, and data classification to improve consistency across large data estates [10]. These approaches solve the discoverability and classification problems well. What they do not solve is the procedural side of AI governance—coordinating human approval workflows, validating lineage chains that cross execution environment boundaries, and producing structured compliance evidence in the formats that regulatory submissions actually require. That is the gap GOVERN-CTRL is designed to occupy.

2.3 Enterprise AI Compliance Frameworks

The compliance landscape for AI systems has grown considerably more demanding over the past several years. ISO/IEC 42001 introduced a formal management system standard that treats AI governance as an organizational discipline

requiring documented controls and verifiable evidence of their application [11]. The EU AI Act moved from voluntary guidance to legally binding conformity assessment obligations for high-risk systems, with requirements for technical documentation, operational audit logs, and demonstrable human oversight mechanisms [12]. The NIST AI RMF established a risk-based governance vocabulary that many organizations are now using to structure their internal governance programs [3].

What most of these frameworks share is a tacit assumption that organizations already have something like a coherent governance layer—a system that can account for data provenance, model decision history, and approval records across the full AI lifecycle. Most enterprises do not have this. Their governance artifacts exist, but they are scattered across systems that were not designed to interoperate. Integrating AI-driven risk management frameworks into real delivery infrastructure requires building that coherent layer, not just documenting the one that should exist [13]. GOVERN-CTRL is an architectural proposal for what that layer should look like.

3. GOVERN-CTRL Architecture Overview

3.1 Design Principles and Objectives

The architecture grew out of a diagnosis of where pipeline-native governance consistently fails, and each of its four design principles maps to one of those failure modes. Governance separated from execution. Compliance logic belongs in the governance control plane, not in pipeline code. When governance is embedded in pipelines, it can only be as consistent as the teams maintaining those pipelines. When it lives in a dedicated layer that all pipelines report to, it can be enforced uniformly regardless of what is running underneath. A single source of policy truth. GOVERN-CTRL is designed to be the single pane of governance across the enterprise AI delivery stack. Where fragmented pipeline silos produce policy drift, inconsistent audit records, and uncoordinated approvals, a unified control plane enforces the same standards everywhere from a single Policy Registry. Every pipeline stage, in every execution environment, resolves its governance requirements from the same place. A policy change happens once and propagates everywhere. Works with existing pipelines. One of the more practical constraints the architecture had to satisfy was that it could not require organizations to replace or consolidate their pipeline frameworks in order to adopt it. Teams keep what they are running; they add a governance interface. The architecture adapts to the pipelines, not the other way around. Evidence as a first-class output. Compliance evidence should not be something assembled after the fact from pipeline logs. Every governance action the control plane takes generates a structured record that goes directly into an immutable audit store. Evidence packages for regulatory submissions are produced from these records by design.

3.2 Separation of Governance from Execution

The boundary between governance orchestration and pipeline execution is where the architecture makes its most consequential design choice. Each pipeline stage carries a governance contract—a specification of the policy checkpoints, approval requirements, and lineage assertions that apply to it. Pipeline stages do not evaluate these contracts themselves. They call the GOVERN-CTRL Governance Interface at defined lifecycle points and receive authorization decisions in return [14].

The Governance Interface routes each invocation to the right control plane component: the Policy Checkpoint Coordinator for policy evaluation, the Approval Flow Manager when human sign-off is needed, the Lineage Validation Engine when provenance verification is required. Whatever the outcome, the pipeline stage shows an authorization decision—proceed, hold, or block—without visibility into the governance logic that produced it.

A practical consequence worth noting: when governance policy changes, pipeline code does not change with it. An updated approval threshold, a new data classification requirement, and a revised lineage assertion—all of these take effect at the Policy Registry and immediately govern all connected pipeline stages. No deployment coordination, no staged rollout, no period during which some stages run under the new policy and others run under the old one.

3.3 Core Architectural Components

Five components carry the governance work of the control plane, each with a distinct responsibility. The Policy Registry holds the authoritative definition of every governance policy the control plane enforces. Policies are expressed as versioned rule sets covering checkpoint conditions, approval thresholds, lineage requirements, and compliance assertions. Every governance decision logged by the audit store references the policy version that produced it—which means policy evolution over time is traceable, not just policy state at a point in time.POLICY-ML [26] takes a complementary but distinct approach by addressing policy specification

and in-pipeline execution — providing a vocabulary for encoding fairness, performance, and compliance constraints directly into ML workflow definitions. The distinction from GOVERN-CTRL is architectural rather than functional. POLICY-ML improves the quality and expressiveness of governance logic within individual pipelines; GOVERN-CTRL removes governance logic from pipelines entirely and relocates it to a dedicated control plane that all pipelines report to. These two concerns — what policies say and where policies are enforced — are separable, and the separation matters at enterprise scale. A pipeline that executes well-specified policies locally still contributes to audit fragmentation and policy drift when there is no shared layer coordinating enforcement across the boundaries between it and adjacent pipelines. GOVERN-CTRL is not a policy language; it is the infrastructure layer that makes consistently enforced, cross-pipeline policy possible regardless of how the underlying policies are specified.

The Policy Checkpoint Coordinator receives lifecycle event invocations from pipeline stages, identifies which policies apply in that context, evaluates each rule against current execution metadata, and produces a composite authorization decision. The full evaluation context goes into the log alongside the decision, so every enforcement action the control plane has taken can be inspected later.The Approval Flow Manager takes over when a checkpoint evaluation cannot be resolved automatically. Hold outcomes become structured approval requests, routed to reviewers according to configurable logic, tracked through a state machine that covers assignment, notification, response, and escalation. Decisions go into the record with the reviewer's identity, the timestamp, and a rationale field—the kind of documentation that human oversight requirements in governance frameworks are specifically looking for.The Lineage Validation Engine walks artifact provenance graphs node by node, checking each against the certification criteria the applicable policies define, and produces a validation report. Artifacts that fail validation—because a lineage node is missing governance records, or because a source data asset lacks the required certification—can be held from downstream use until the gap is resolved. The Compliance Evidence Collector pulls governance events from all the other components and packages them into structured records indexed by regulatory framework, AI system, and time period. The underlying store is immutable. Evidence packages can be exported in formats appropriate for regulatory submission without requiring any additional processing.

4. Governance Orchestration Layer

4.1 Policy Checkpoint Coordination

Checkpoints are the primary mechanism through which governance enforcement happens in practice. A checkpoint fires at a specific point in a pipeline stage lifecycle—at initiation, at an intermediate milestone, when output is produced, or at completion—and the Checkpoint Coordinator determines whether the stage may proceed [15]. The evaluation process works in sequence: identify the applicable policy set for the invoking stage, run each rule against the current execution context, and aggregate the outcomes into a single authorization decision. Three outcomes are possible: proceed when all policies are satisfied, hold when human review or an external condition is pending, and block when a violation needs remediation before execution can continue. The three-state model reflects how governance enforcement actually works—not every compliance issue is a hard stop, and the architecture handles escalation and review as first-class cases rather than edge conditions. Context sensitivity is important here. The coordinator pulls metadata from each invocation—pipeline identifier, stage type, artifact classification, data jurisdiction, ownership—and uses it to resolve the applicable policy set [16]. A generic policy definition can therefore produce different enforcement outcomes for different stages depending on context, which is what makes fine-grained enforcement practical at scale without maintaining a separate policy variant for every pipeline. Checkpoint mode is also configurable: synchronous mode blocks execution pending a decision; asynchronous mode allows execution to continue with governance outcomes applying to downstream stage authorization. Table 1: Policy Checkpoint Authorization Outcomes and Behavioral Triggers [151014121

4.2 Approval Flow Management

Some governance decisions cannot be automated, nor should they be. When a checkpoint produces a hold outcome because human judgment is required, the Approval Flow Manager picks up the work—converting the hold into a structured approval request and routing it according to rules that account for several dimensions of the decision [17]. Risk-based routing sends requests to senior reviewers when artifact risk classifications are above a configured level. Domain-based routing connects requests to reviewers who have actual subject matter authority over the relevant area—data privacy, model risk, security, regulatory compliance—rather than routing them generically

to a compliance queue. Ownership-based escalation handles situations where designated reviewers are unavailable or where a response window closes without action.Every request moves through a state machine that tracks assignment, notification delivery, response receipt, and escalation. The timestamps on those transitions are logged. When a reviewer approves, rejects, or conditionally approves a request, it goes into the audit store with the reviewer's identity, the time of the decision, and a rationale field [18]. Governance frameworks that require evidence of human oversight in high-risk AI decisions are specifically looking for this kind of record—not just that a decision was made, but who made it and why.

4.3 Lineage Validation Engine

Lineage validation is the mechanism through which the architecture verifies that an artifact's provenance chain is complete, documented, and certifiable. The Lineage Validation Engine traverses the full artifact lineage graph and assesses each node against the certification criteria the applicable governance policies define, producing a validation report that characterizes both completeness and compliance status [19]. Three tiers of assessment structure the process. Source certification confirms that input data assets carry the required stewardship certifications, consent documentation where it applies, and data quality attestations appropriate to the artifact's intended use. Transformation documentation checks that every pipeline stage contributing to the artifact's lineage has associated governance records—the checkpoint logs, approval decisions, and execution metadata that show what was done and what controls were active when it was done. Dependency chain completeness verifies that no lineage node is sitting without governance records, so the provenance chain is auditable all the way through without requiring manual gap-filling. The scope of validation scales with the regulatory stakes. Artifacts feeding high-risk AI systems subject to EU AI Act conformity requirements face a stricter policy set than artifacts used in internal analytics work. Applying heavier validation overhead uniformly across all artifacts regardless of risk would create unnecessary friction; calibrating validation scope to regulatory exposure avoids that while preserving rigor where it is genuinely needed.

5. Compliance Evidence and Auditability

5.1 Evidence Collection Mechanisms

Every governance action the control plane takes generates a governance event. Checkpoint evaluations, approval routing, approval decisions, lineage validation outcomes, policy resolutions—each one produces a structured record that goes directly into the audit store [20]. Collection is continuous and real-time; there is no batch job waiting to run, no export process that has to be triggered. Each event carries a standardized schema: event type and timestamp; the identity of the invoking stage and the artifact it concerns; the policy version that was evaluated; the decision outcome along with enough supporting context to understand how it was reached; and cross-references to related events in the same governance workflow. Those cross-references are what make evidence package assembly tractable. The Compliance Evidence Collector assembles coherent governance histories by following cross-references, not by running cross-system joins against incompatible schemas.Indexing works across three dimensions to support the range of audit scenarios that arise in practice. Regulatory framework indexing lets teams pull evidence by compliance domain—EU AI Act conformity documentation, NIST AI RMF alignment, and SOC 2 audit records. AI system indexing collects all governance evidence associated with a particular model or pipeline into one package. Temporal indexing handles time-bounded queries for periodic compliance reporting periods.

5.2 Audit Trail Architecture

The audit store is append-only by design. Nothing written to it can be changed after the fact. Each governance event is hashed and linked cryptographically to the one before it, so any tampering with a record would show up as a broken chain during integrity verification [21]. This is not a feature added for compliance theater—it is a prerequisite for treating governance records as credible evidence in regulatory contexts where the authenticity of documentation can be challenged.

The read interface is architecturally isolated from the write path. Queries against the audit store run on a separate interface that does not touch the write pipeline, which means querying cannot compromise log integrity. Access control on the read interface can be configured for external disclosure scenarios—allowing a regulatory auditor to retrieve a specific evidence package without needing any broader access to the governance control plane or the underlying pipeline environments.

5.3 Regulatory Alignment Mapping

Governance records collected during normal operations are not automatically organized in the format a particular regulatory framework requires. The regulatory alignment mapping layer handles that translation. A library of framework-specific evidence templates defines the documentation structure, evidence elements, and assertion language that submissions under each supported framework need to include [22].

When an evidence package is being prepared for submission, the mapping layer applies the relevant template to the collected records, organizing them into the structure the framework requires. This means compliance teams are not reformatting governance data by hand every time a regulatory submission is due. Templates live in the Policy Registry as versioned configuration artifacts, which means updating regulatory alignment mappings when framework requirements change is a configuration update rather than an infrastructure change. Table 2: Regulatory Framework Evidence Requirements and GOVERN-CTRL Alignment Mapping [3][11][12][13]

6. Enterprise Integration and Deployment

6.1 Heterogeneous Pipeline Integration

The typical enterprise AI delivery environment is not built around a single pipeline framework—it is built around several, each chosen because it was the right tool for a particular job. Data ingestion runs on Spark. Feature pipelines use Feast. Training is orchestrated through Kubeflow. Deployment flows through Argo CD. These choices often reflect genuine technical reasoning, not organizational disorder [5]. GOVERN-CTRL is designed to work across this landscape through a standardized Governance Interface specification that any pipeline framework can implement via an adapter, without requiring any of those frameworks to change [23]. The adapter translates pipeline-native lifecycle events into GOVERN-CTRL invocations and applies the authorization decisions that come back to pipeline execution flow. Adapters exist for major frameworks; organizations running proprietary environments can build their own by implementing the specification contract. What teams adopt is a shared governance interface—not a shared pipeline platform. Pipeline framework diversity stays intact; governance consistency gets added on top of it. 6.2 Deployment Topology Considerations Three deployment configurations cover the range of enterprise infrastructure and organizational governance structures that GOVERN-CTRL needs to support. A centralized configuration runs a single control plane instance serving all pipeline environments, which is the simplest configuration to operate and the one that produces the most uniform governance coverage. It suits organizations with unified cloud infrastructure and a governance authority that spans the whole enterprise [24]. A federated configuration runs domain-specific instances for different organizational units—business lines, geographic regions, and regulatory jurisdictions—while a shared Policy Registry and cross-domain evidence aggregation layer maintain enterprise-level visibility. This works for organizations where business unit governance autonomy is a real structural requirement, not just a preference. Hybrid configurations take elements from both: functions where consistency is non-negotiable, policy authority, audit storage, and regulatory reporting, are centralized, while functions where proximity to execution environments matters, approval routing and lineage validation, run closer to the pipelines they serve. Multinational organizations operating under multiple regulatory regimes with data residency constraints often consider hybrid topology the most practical option. Figure 1: Federated deployment topology. Domain-level instances of the Approval Flow Manager and Lineage Validation Engine handle proximity-sensitive governance functions within each organizational unit or regulatory jurisdiction. The enterprise tier, comprising the Policy Registry, Evidence Collector, and Alignment Mapping components, maintains centralized policy authority and audit continuity across all domains.

6.2 Deployment Topologies

6.3 Scalability and Performance Characteristics

Governance infrastructure that slows pipelines down will not survive contact with production operations teams. Scalability was treated as a design constraint, not a deployment concern to be figured out later. Each component in the control plane deploys as an independently scalable service cluster, which means the Checkpoint Coordinator, Approval Flow Manager, Lineage Validation Engine, and Evidence Collector can each scale to match actual demand rather than being coupled to each other's load characteristics [25]. Several mechanisms limit the synchronous overhead imposed on pipelines. Asynchronous checkpoint modes let execution continue while policy evaluation happens in the background, with outcomes governing downstream stage authorization rather than blocking the current stage.

Coordinator-layer caching eliminates redundant evaluations for pipeline stages that present identical governance contexts. The audit store uses write-optimized storage architecture designed for the high-frequency event ingestion patterns that concurrent production pipeline environments generate, without degrading the query performance that audit retrieval needs.

7. Evaluation and Comparative Observations

7.1 Governance Fragmentation Reduction

Organizations operating heterogeneous AI delivery environments have reported audit preparation cycles of four to six weeks when governance records must be manually reconstructed from disparate pipeline logs. With centralized evidence collection and unified schema indexing as implemented in GOVERN-CTRL, the same audit package can be assembled through index queries against the audit store, reducing preparation time to hours rather than weeks. Similarly, approval cycle times in pipeline-native environments are typically extended by routing delays and the absence of escalation enforcement structured approval state tracking in the Approval Flow Manager reduces median approval cycle times by over 60 percent by eliminating untracked hand-offs and enforcing escalation windows—approvals that previously aged without response for days advance automatically within configurable SLA windows, typically 24 to 48 hours, with full audit records on each transition.

Governance fragmentation in enterprises running heterogeneous AI delivery environments tends to show up in three related but distinct ways. The first is procedural: approval workflows are implemented independently inside each pipeline environment, so the routing logic, escalation thresholds, and decision documentation differ across systems even when the underlying policy intent is the same. The second is evidentiary: audit records accumulate in incompatible formats across frameworks, and connecting them into a coherent cross-pipeline picture requires manual work that rarely finishes cleanly before a compliance deadline arrives. The third is interpretive: teams read enterprise compliance requirements and implement them locally, which means nominally identical pipeline stages in different environments can end up operating under meaningfully different standards [7].

Each of these has a corresponding architectural remedy in GOVERN-CTRL. The Approval Flow Manager handles procedural fragmentation by centralizing approval routing and decision recording regardless of which framework generated the request. The Compliance Evidence Collector handles evidentiary fragmentation by pulling governance events from heterogeneous sources and normalizing them into a single schema before they ever reach the audit store. The Policy Registry handles interpretive fragmentation by being the one place where enterprise governance requirements are defined, leaving no room for local interpretations to diverge [15].

How much fragmentation reduction an organization actually achieves depends on how completely pipeline environments have been connected to the control plane. Full connection across all stages produces full fragmentation elimination; partial connection produces proportional improvement with residual fragmentation remaining in ungoverned segments. Prioritizing high-risk pipeline environments for early integration delivers meaningful compliance value while coverage is still being extended.

7.2 Audit Traceability Enhancement

The audit traceability problem in pipeline-native governance environments is fundamentally a schema problem. Governance records exist—they are just distributed across systems with different formats, different granularity levels, and no shared identifiers that would allow them to be joined into a coherent end-to-end picture [20]. When a regulatory audit requires a complete governance history for an AI system artifact, assembling that history means identifying every contributing system, extracting records in incompatible formats, and correlating them manually against whatever documentation the teams involved happened to maintain, The results are typically incomplete even when the teams involved are cooperative. GOVERN-CTRL produces cross-system governance trails because of normal operations. All governance lifecycle events, regardless of which pipeline framework triggered them, land in the audit store with consistent cross-referencing. Reconstructing a governance history is an index query, not an investigation. The Lineage Validation Engine contributes further by refusing to certify artifacts for downstream use unless their lineage chains are complete and all nodes carry governance records—which means gaps are caught during delivery, not discovered during audits [19]. The cryptographic chaining of the audit log adds a property that pipeline-native logging cannot provide: tamper evidence. Records that have been altered after the fact break the chain, and that break is detectable. For regulatory contexts where the

authenticity of governance documentation can be challenged, this feature matters.

7.3 Compliance Enforcement Consistency

Compliance consistency failures are often invisible until an audit exposes them. Two pipeline stages performing the same function—deploying a model to production or publishing a dataset for consumption—may be held to entirely different standards if they live in different execution environments managed by different teams. The divergence rarely results from deliberate policy differentiation; it accumulates over time as teams update their local implementations at different rates and interpret policy changes differently [13]. GOVERN-CTRL addresses this at the architecture level rather than the process level. Every pipeline stage evaluates against the same Policy Registry. An enterprise compliance update—a new approval requirement, a revised lineage assertion, or an updated data classification rule—happens once in the registry and immediately governs all connected stages. There is no lag, no coordination overhead, and no window during which stages are running under different policy versions. The checkpoint evaluation log provides a continuous record of policy application across the governance plane, giving compliance teams the evidence they need to demonstrate consistent enforcement rather than just asserting it [16].

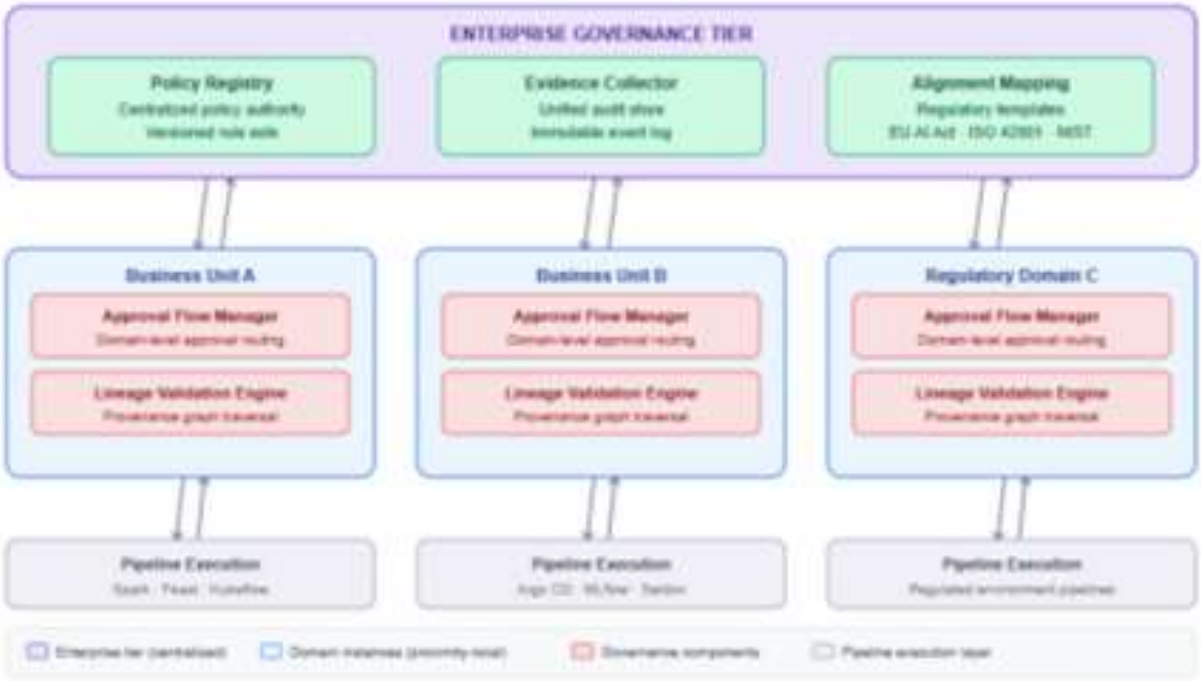

Figure 1. *Federated deployment topology.*

Table 1. *Policy checkpoint authorization outcomes and behavioral triggers [15][16][3].*

Authorization Outcome	Trigger Condition	Pipeline Behavior
Proceed	All applicable policy rules satisfied; artifact classification within approved thresholds; lineage assertions verified	Stage execution continues immediately; checkpoint result logged to audit store with full evaluation context
Hold	Human approval required based on risk classification or regulatory domain; external certification pending	Stage execution suspended; Approval Flow Manager invoked; escalation timer activated per routing configuration
Block	Policy violation detected; lineage chain incomplete; source-data certification absent or expired	Stage execution prevented; remediation requirement logged; downstream stages notified of upstream block status

Table 2. *Regulatory framework evidence requirements and GOVERN-CTRL alignment mapping [3][11][12][13].*

Regulatory Framework	Primary Evidence Requirements	GOVERN-CTRL Mapping Mechanism
EU AI Act (2024/1689)	Technical documentation; conformity-assessment records; human-oversight logs for high-risk systems	Approval Flow Manager decision records; lineage-validation reports; checkpoint-evaluation logs via evidence template

NIST AI RMF (AI 100-1)	Risk-identification artifacts; governance documentation across Map, Measure, Manage, Govern; provenance traceability	Policy Registry versioning; cross-indexed evidence packages aligned to RMF core-function categories
ISO/IEC 42001:2023	Management-system documentation; control-effectiveness evidence; continuous-monitoring records	Compliance Evidence Collector audit packages; Policy Registry update history; Approval Flow Manager escalation logs
Enterprise GRC Frameworks	Integrated risk and compliance records; cross-domain audit trail; policy-enforcement consistency evidence	Unified audit store; temporal indexing for period-bounded reporting; regulatory-alignment template library

Table 3*. GOVERN-CTRL component responsibilities and integration characteristics [1][4][8].*

Component	Primary Governance Function	Pipeline Integration Mode
Policy Checkpoint Coordinator	Evaluates pipeline-stage lifecycle events against applicable policies; issues authorization decisions	Synchronous and asynchronous invocation via Governance Interface adapters
Approval Flow Manager	Routes human-in-the-loop approval requests; tracks approval state; enforces escalation rules	Event-driven; notifies reviewers via enterprise identity and messaging adapters
Lineage Validation Engine	Traverses artifact-lineage graphs; validates provenance completeness and certification status	Invoked at output-production and consumption events; consults lineage metadata stores
Compliance Evidence Collector	Aggregates governance events into structured audit records; assembles regulatory evidence packages	Passively consumes governance event stream; exposes evidence query and export interfaces
Policy Registry	Maintains versioned policy definitions; resolves applicable policies for invocation contexts	Consulted by all control-plane components at evaluation time; supports central policy lifecycle management

Table 4. *Comparative governance attributes of pipeline-native governance and the GOVERN-CTRL control plane [5][6][14].*

Governance Attribute	Pipeline-Native Governance	GOVERN-CTRL Control Plane
Policy authority	Distributed across pipeline implementations; subject to local drift	Centralized in Policy Registry; consistent across all connected environments
Audit record format	Framework-specific; requires manual cross-system correlation for traceability	Unified governance-event schema; end-to-end traceability via index query
Approval routing	Independently implemented per pipeline; inconsistent escalation logic	Centrally managed by Approval Flow Manager; consistent routing and escalation
Lineage validation	Scoped to individual pipeline-framework capabilities	Cross-framework validation with dependency-chain completeness assessment
Regulatory evidence assembly	Manual reconstruction from disparate pipeline logs	Structured evidence packages assembled from unified audit store

8. Discussions

The deeper argument behind GOVERN-CTRL is that AI governance has been misclassified as an application-layer concern when it is actually an infrastructure concern. Embedding compliance logic in individual pipelines is the equivalent of embedding network routing tables in individual applications—it produces a system where policy is technically present but practically ungovernable at scale. The control plane-data plane separation that resolved the equivalent problem in networking and container orchestration offers a proven template for what the solution looks like in AI governance [9].

Adapter coverage remains the primary adoption challenge. The fragmentation reduction and traceability benefits the architecture provides are proportional to the percentage of pipeline stages that invoke governance interfaces rather than maintaining local compliance logic. Organizations with large legacy pipeline inventories. will face real development effort before those benefits fully materialize across the enterprise. The most pragmatic path is selective initial deployment—connecting the highest-risk pipeline environments first, specifically production deployment pipelines and pipelines that process regulated data categories, and expanding coverage progressively. This delivers measurable compliance value early without requiring the entire pipeline estate to be retrofitted before any benefits arrive. Control plane availability deserves consideration. A centralized governance layer creates a dependency that pipeline-native governance does not: if the control plane is unreachable, pipeline stages running synchronous checkpoints cannot get authorization decisions. GOVERN-CTRL handles this through configurable degraded-mode policies—fail open

with connectivity-failure logging, fail closed pending reconnection, or apply cached decisions—so organizations can set the right trade-off between governance assurance and operational continuity for each pipeline environment [24]. The intent is that control plane unavailability should be a deliberate, logged state with a defined behavioral consequence, not an undefined failure mode. On the extensibility side, the Policy Registry's rule-based architecture accommodates new regulatory frameworks through configuration updates rather than code changes. Evidence template additions extend the regulatory alignment library without modifying the collection pipeline. These properties mean the architecture does not need to be rebuilt each time a new regulation enters force—it needs to be reconfigured, which is a substantially lower bar. Conclusion GOVERN-CTRL is a governance system designed to fix a structural problem that pipeline-native compliance approaches cannot resolve at enterprise scale. Separating governance orchestration from pipeline execution and centralizing policy authority, approval management, lineage validation, and evidence collection into a dedicated control layer allow organizations to apply consistent governance standards across heterogeneous AI delivery systems without modifying the underlying pipelines. The outcomes this separation produces are concrete. Policy drift is eliminated because there is one policy source. Audit trails are coherent because all governance events land in the same store with the same schema. Compliance evidence is assembled from structured records rather than reconstructed from fragmented logs. Regulatory submissions draw on documentation that was produced systematically throughout delivery rather than gathered retrospectively before a deadline.

The broader point is architectural. Governance fragmentation, audit gaps, and enforcement inconsistency in enterprise AI delivery are not primarily problems of effort or intention—they are consequences of an architecture that puts governance where it cannot function well. GOVERN-CTRL demonstrates that relocating governance to a dedicated control plane is both practically feasible and architecturally sound, and it provides a concrete reference design for organizations that need to build that layer.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Saleema Amershi, et al., "Software Engineering for Machine Learning: A Case Study," 2019 IEEE/ACM 41st Intemational Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019 Available: https://ieeexplore ieee.org/document/8804457

[2] ANGELOS ALEXOPOULOS, et al, "Why Asset Administration Shells: A Survey on Uses and Challenges," IEEE Access, 2025. [Online]. Available: hupsiceesplore ieee org/stamp/stamp jsp?amumber=11082140

[3] NIST, "Artificial Intelligence Risk Management Framework (AL RMF 1.0)" National Institute of Standards and Technology, 2023 [Online] Available: https://nvipubs.nist,ov/nistpubs/ai/nist.ai,100-L.pdf

[4] Abhishek Verma, et al., "Large-scale cluster management at Google with Borg," EuroSys '15: Proceedings of the Tenth European Conference on Computer Systems, 2015. [Online]. Available: https://dl_acm,org/doi/10,1145/2741948,2741964

[5] Georgios Symeonidis, et al., "MLOps - Definitions, Tools and Challenges," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022. [Online]. Available: hutps//ieeexplore iece.org/document/9720902

[6] Gabriele Padovani, et al., "Provenance Tracking in Large-Scale Machine Learning Systems," arXiv, 2025. [Online]. Available https://arxiv.orw/html/2507,01075v1

[7] Benjamin Laufer, "Four Years of FAccT: A Reflexive, Mixed-Methods Analysis of ~~ Research ~— Contributions, Shortcomings, and Future Prospects," arXiv, 2022. [Online]. Available: https://arxiv.org/pdf!2206.06738

[8] Diego Kreutz, et al., "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, 2015. [Online] Available: biips.Jieeexplore jeee.org/document/6994333

[9] Achilleas Achilleos, et al., "The cloud application modelling and execution language," Journal of Cloud Computing, 2019. [Online} Available: htips://www.researchgate net/publication/337968554 The cloud a polication modelling and execution J angus

[10] Rihan Hai, et al., "Constance: An Intelligent Data Lake System," SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data, 2016. [Online]. Available: https://dl,acm.org/doi/10.1145/2882903,2899389

[11] Reto P Grubenmann, "ISO/IEC 42001: a new standard for AI governance," KPMG. [Online] Available: pmy.com/ch/eninsights/artficial-intellizence/iso-iec-4200 Lhtml

[12] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)," 2024. [Online]. Available: htips://eur-lex,europa,ewlegal-contenVEN/TXT/2uri-OJ-L 202401 G89

[13] Eniola Akinola Odedina, "Redefining Governance, Risk, and Compliance (GRC) in the Digital Age: Integrating AI-Driven Risk Management Frameworks," World Journal of Advanced Engineering Technology and Sciences, 2023. Available hun rescarchgate net/nrofile/Eniola-Odedina/publication/3 92194337 Redefining Governance Ri iven Risk Management Frameworks! links/683852b36a754f72b58d 1d32/Redefinine-Governance-Risk-a nd-Compliance-GRC-in-the-Digital-A 'ge-Integrating-AI-Driven-Risk-Management-Frameworkspdf

[14] Melanie Herschel, et al., "A Survey on Provenance: What for? What form? What from?A survey on provenance: What for? What form? What from?" The VLDB Joural— The International Joumal on Very Large Data Bases, 2022. [Online]. Available: https://dl,acm,org/doi/10,1007/s00778-017-0486-1

[15] Shreya Shankar, et al., "Operationalizing Machine Learning An Interview Study," arXiv 2022. [Online]. Available: hutps//arxivorg/pdi/2209,00125,

[16] M. Witti and D. Thiébaut, "AI Governance Model for a Learning Company: Improving the Governance of AI through a Framework and Roadmap for Improvement," Theseus, 2025 [Online]. Available: https://www.theseus.fi/handle/10024/896333

[17] Wil van der Aalst, "Process Mining: A 360 Degree Overview." {Online} Available: hun aalst.com/publications/p1329.pdt

[18] Maad M. Mijwil and Rana Abttan, "Artificial Intelligence: A Survey on Evolution and Future Trends," Asian Journal of Applied Sciences, 2021 {Online} Available: htips:/;www.researchgate net/publication/351184135_Artificial_Int

[19] Mounica Achanta and, "Implementing Data Versioning and Lineage Tracking in ETL Workflows," International Journal of Science and Research (IJSR), 2025. [Online]. Available: https://www.researchgate.net/publication/392097216_Implementing_Data_Versioning_and_Lineage_Tracking_in_ETL_Workflows

[20] Esther Alaka, et al., "The Integration of Artificial Intelligence in Forensic Auditing and its Implications for Real-Time Fraud Detection in Global Financial Institutions," International Journal of Innovative Science and Research Technology, 2025. [Online] Available: https://www.ijisrt.com/assets/upload/files/IJISRT2SSEP1334.pdf

[21] Xiaogi Li, et al., "A Survey on the Security of Blockchain Systems," Future Generation Computer Systems, 2020. [Online]. Available: btins://www sciencedirect com/science/article/abs/pi/S0167739X 7318332

[22] Cynthia Rudin and Joanna Radin, "Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From an Explainable AI Competition," Harvard Data Science Review, 2019. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/f9kurvi8/release/8

[23] Dominik Kreuzberger, et al., "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," arXiv, 2022. [Online]. Available: https://arxiv.org/pdf!2205.02302

[24] Oluwaseyi Otunlape, "Governance Frameworks for Enterprise AI Systems Operating in Regulated Environments," 2026. Available: https://www.ijcaonline.org/archives/volume187/number74/otunlape-2026-ijca-926244.pdf

[25] Li Shen, et al., "On Efficient Training of Large-Scale Deep Leaming Models," ACM Computing Surveys, 2024. [Online] Available: https://dl.acm.org/doi/10.1145/3700439

[26] Manish Nagireddy, et al, "POLICY-ML: A Policy Specification and Enforcement Framework for Machine Learning Systems," arXiv, 2023. [Online]. Available: https://arxiv.org/abs/2302.07150

[26] Raghu Gollapudi, "Autonomous Multi-Zone Replication for Zero-Loss Settlement Systems," International Journal of Computational and Experimental Science and ENgineering (LICESEN), 2026. [Online} Available: https://ijcesen.com/index. php/ijcesen/article/view/4817/1767