



Technology diagnostics for building a cost-effective and high-performance insurance enterprise

Defining and measuring expectations vs. reality in insurance IT operations

■ ■ ■
The better the question.
The better the answer.
The better the world works.



EY

Shape the future
with confidence



Insurance chief technology officers (CTOs) and chief information officers (CIOs) have long been under pressure to optimize technology delivery – not only to improve digital experiences for policyholders, agents and underwriters but also to manage rising costs driven by complex legacy systems and regulatory demands.

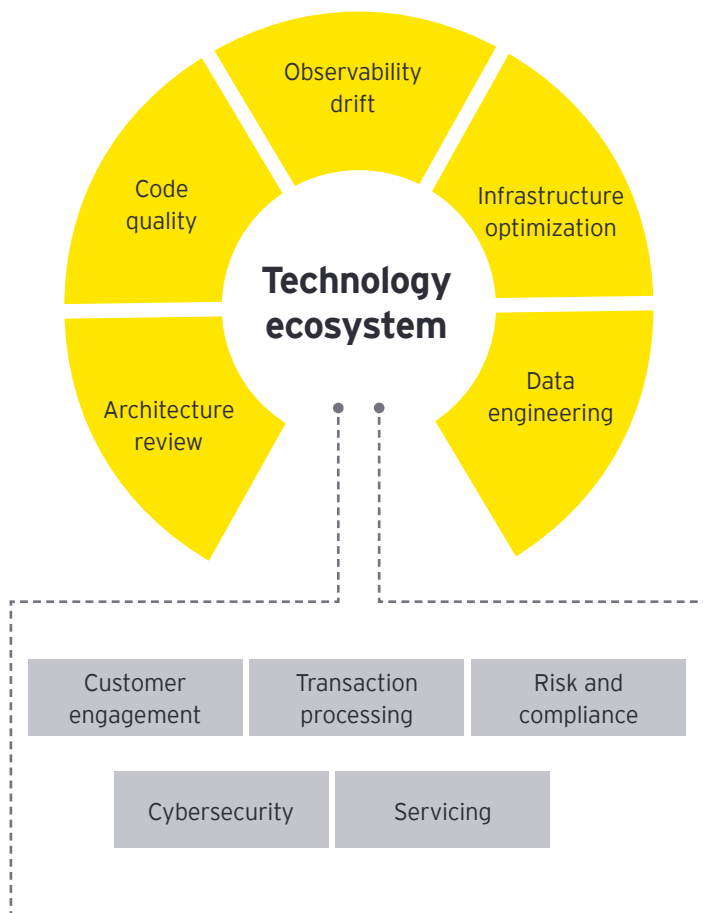
As insurers expand their digital capabilities to remain competitive, the focus often shifts to controlling cost centers such as infrastructure, platforms and enterprise applications. Yet these efforts can amount to short-term relief rather than long-term transformation, as they fail to tackle the systemic inefficiencies embedded in enterprise technology adoption.

To address these unique challenges, EY Financial Services has developed an engineering-led technology diagnostics and cost optimization solution tailored for the insurance sector, anchored in five foundational principles:

- 01 Evaluating architecture design decisions and their impact on total cost of ownership across underwriting, claims and policy administration systems
- 02 Bridging code behavior with enterprise coding standards to facilitate advanced performance of core insurance applications
- 03 Integrating observability into critical insurance workloads for predictable infrastructure and system behavior
- 04 Recommending cost-optimized workload placement across cloud, on-prem and hybrid environments aligned with compliance needs
- 05 Enhancing reliability and recovery of data pipelines supporting actuarial models, fraud detection and customer insights

This paper introduces a modern approach to evaluate how technology design decisions impact operating costs in real-world insurance environments. It also identifies gaps that emerge post-deployment – when actual behavior diverges from intended architecture – affecting claims processing, policy servicing or compliance reporting. The solution helps maintain continued infrastructure and data pipeline health, enabling insurers to support evolving business processes and customer demands with agility and cost efficiency.

Technology ecosystem excellence: drift detection, resolution and proactive cost management



Architecture review:

- Architecture alignment to enterprise standard
- Optimized architecture pattern (based on enterprise and EY team)
- Reduced cost of implementation (forward and backward looking)

Code quality:

- Coding standard match to enterprise standard (or EY supplemented)
- Maintainability and dependency issues
- Test strategy and comprehensiveness
- Impact on maintenance costs (backward looking)

Observability drift:

- Deviation from expected performance standards
- Deviation from business user requirements
- Self-healing infrastructure
- Total infrastructure operations cost

Infrastructure optimization:

- Workload decisioning criteria (application classification, service-level agreement (SLA) criteria, etc.)
- Current state evaluation (licensing, security, etc.)
- Recommendation for placement

Data engineering:

- Data pipeline reliability and recovery
- Self-healing data infrastructure
- Cost reduction opportunities: transformation, modeling





01

CHAPTER

Evaluating architecture design decision and impact on total cost of ownership

Architecture design decisions are often driven by immediate business needs. The decisions do not have access to how the architecture performed in the past or its implications on long-term business value. This is because in application architecture, the gap between design intent and actual production outcomes has been a persistent challenge for decades. Traditionally, architecture reviews focus on standards, controls and best practices – capturing how an application should be designed and deployed. However, these reviews are typically point-in-time exercises, performed in isolation from when the application is actually running in production. This leads to a blind spot where deviations, misconfigurations and unanticipated dependencies emerge post-deployment, often without architects being aware of them until issues surface.

To address this, we propose an engineering-first approach where enterprise standards, industry frameworks, architecture controls, and EY architecture and design considerations are embedded into a machine-readable knowledge base. These standards form the expectation for every application – covering resiliency, security postures, performance baseline and cost. This set of expectations is then compared, continuously, against actual behavior in production by analyzing observability data (metrics, events, logs, traces) and other real-time system signals.

This comparison between what is expected and what is actually happening creates a dynamic feedback loop. It allows enterprise architects to detect deviations early, pinpoint their root causes, and refine both architecture guidance and operational processes in near real time.

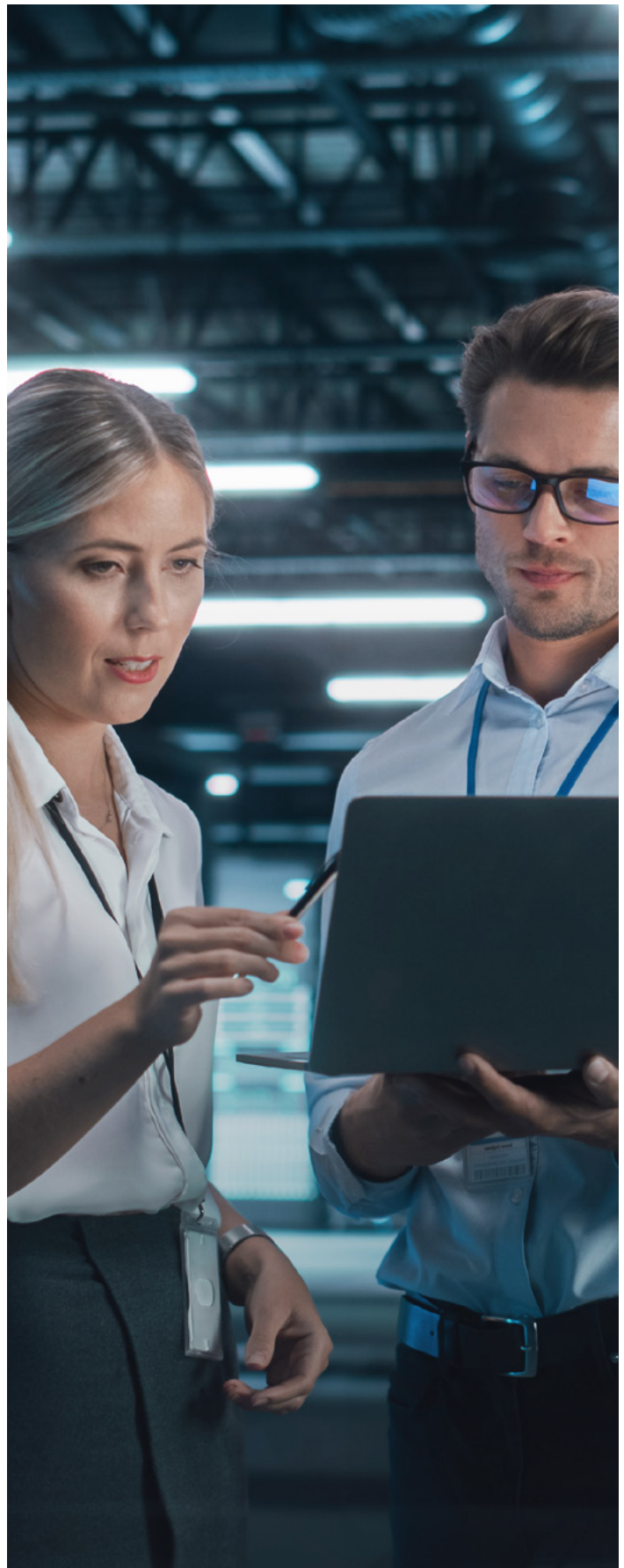
The result is an adaptive architectural assurance process, where design principles are constantly validated against operational realities, leading to faster issue resolution, reduced technical debt accumulation and closer alignment with business outcomes. This paper outlines the methodology, processes and tools used to enable this continuous feedback system, providing a practical roadmap for evolving architecture into a business-aligned, continuously improving practice.

A critical step in maintaining well-governed IT landscapes is the systematic review of application architectures against both internal enterprise standards and recognized industry frameworks. These reviews typically begin by collecting architecture diagrams, design documents and operational guides directly from application teams.

Each document is assessed to confirm alignment with fundamental principles around resilience, performance, security, operational excellence and cost optimization – principles that exist both within the enterprise's internal standards and within widely accepted industry frameworks. The review is not a checklist exercise, but rather a contextual analysis that evaluates whether the architecture's design choices make sense given the business purpose, criticality and future scalability needs of the application.

This review process also identifies gaps, where either required controls (such as data encryption at rest) are missing or where architectural decisions (such as reliance on a single point of failure) introduce unnecessary risk. These gaps are documented, and improvements are recommended to the application owners, with a focus on closing compliance gaps and aligning them to best practices before the application moves into production.

The process is further enhanced through automation tools and intelligent comparison methods. By embedding internal and external standards into a vectorized knowledge base, application architectures can be automatically compared to this knowledge. This allows for rapid identification of potential deviations, freeing up architects to focus on contextual analysis, trade-off decisions and business alignment discussions.



Advanced cost estimation for hybrid and multi-cloud architectures

In addition to reviewing the architecture for standards and controls, we apply a novel approach to estimating the cost of the architecture itself – regardless of its complexity, technology stack or deployment environment. Whether an application spans on-premises systems, public cloud services or third-party SaaS platforms, the EY methodology is designed to produce a comprehensive and reliable cost estimate.

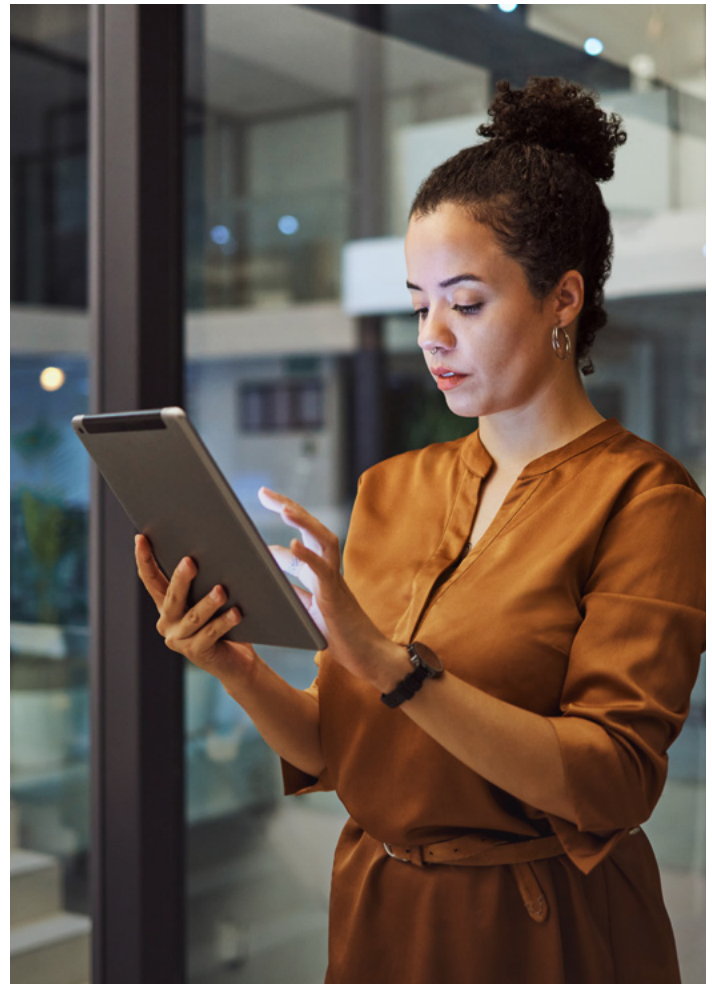
This approach moves far beyond conventional online cloud calculators, which are typically tailored to a single cloud provider and struggle to account for hybrid designs and interconnected third-party components. Instead, workloads are classified into categories, each of which has distinct cost drivers, such as data movement, compute cycles, licensing costs and compliance overhead.

For example, a hybrid financial risk application might include on-prem batch analytics, real-time streaming pipelines in a public cloud provider and data enrichment services powered by external market data providers. The EY cost model breaks down the pricing for each component, factoring in nuances such as inter-region data transfer, application programming interface (API) call charges and specialized licensing models for financial software.

More importantly, by layering this analysis with the results of architecture reviews, we also quantify the cost of technical debt – highlighting how gaps in design or suboptimal technology choices will drive future remediation costs.

This granular cost decomposition allows us not only to predict implementation costs but also to calculate the ongoing operational expense of maintaining the architecture in production.

With this integrated approach, architecture cost estimation becomes a **living process**, evolving alongside changes in design, technology choices and operational realities.





02

CHAPTER

Bridging architecture standards and code behavior

Architecture-to-implementation alignment challenge

Developers often make pragmatic decisions under delivery pressures that drift from architectural standards. Meanwhile, production environments and external dependencies evolve faster than static standards can account for, making continuous alignment between design and implementation critical.

While establishing patterns and standards is essential, verifying adherence throughout the application lifecycle remains challenging.

The value of connecting architectural patterns with actual code implementation lies in effectively translating design intent into real-world execution. While establishing architectural patterns and standards is essential, the true challenge lies in detecting when implementations deviate from these patterns. By automatically identifying deviations, distinguishing between innovation and technical debt, and creating appropriate feedback loops, organizations can evolve their architecture while preventing problematic pattern drift that undermines long-term maintainability.

Triangulation approach to detect pattern deviations

This approach identifies and evaluates deviations through three dimensions:

- 01 **Architectural intent:** Documented patterns, principles, and guardrails defined by architecture teams
- 02 **Implementation reality:** Actual code implementation patterns used by development teams
- 03 **Technical debt impact:** Consequences of pattern deviations on maintainability, performance, and stability

This comparison reveals how closely code adheres to guardrails, where unintentional deviations occur and which represent innovation vs. technical debt.

Beyond static analysis

Traditional compliance assessments use static analysis tools that check syntax and style. This is incomplete, as static analysis cannot detect:

- Implementation patterns circumventing established integration mechanisms
- Security control bypasses or inconsistent implementation
- Unauthorized data access patterns or storage solutions
- Performance optimizations deviating from recommended patterns
- Integration approaches bypassing approved intermediaries like API gateways



Tracking architecture pattern deviations

In production, observability traces show some services calling the external partner directly – bypassing the gateway – perhaps for performance tuning during a production incident. These deviations would be invisible in static analysis alone but show up clearly in observability data.

Implications of the identified drift

This architectural drift has direct business and operational impacts:

- **Security risk:** Direct calls might weaken authentication.
- **Operational risk:** Lack of centralized retries or throttling could overload external services.
- **Compliance risk:** Audit gaps arise if centralized logging is bypassed.
- **Cost risk:** External calls could incur higher egress charges.
- **Technical debt accumulation:** This nonstandard pattern now requires future remediation.

Building the feedback loop to identify and remediate drift

The continuous feedback process takes all these signals – expected design, static analysis findings and tracked deviations – and delivers:

- Real-time compliance scores per application
- Automatic generation of technical debt entries where deviations occur
- Direct feedback to developers through integrated environment development (IDE) plug-ins, pull request comments or architecture dashboards
- Recommendations for improving patterns in future designs – making the standards themselves smarter based on operational learning

This shifts code quality and architecture compliance from being a slow, subjective, manual process into a real-time, evidence-driven system. It achieves:

- Faster feedback for developers (catch issues while they code)
- Objective enforcement of standards (no more reviewer bias)
- Business-context-aware decisions (regulated vs. nonregulated services follow different rules)
- Continuous audit trail for every decision – perfect for regulated industries like insurance
- Early detection of technical debt, so remediation is proactive, not reactive





03

CHAPTER

Integrating observability for predictable infrastructure performance

Updating CMDB accuracy using observability data

A configuration management database (CMDB) is traditionally viewed as the authoritative inventory of infrastructure and applications within an enterprise. However, in practice, CMDB data often lags behind reality. Applications evolve, dependencies shift and new integrations are introduced without thorough documentation. This drift creates gaps between what architects and operations teams believe exists and what actually exists. The solution lies in enriching CMDB data with real-time observability data – metrics, events, logs and traces (MELT) – captured from production systems.

By continuously analyzing MELT data, the architecture review process becomes aware of how the system is behaving, including undocumented services, unregistered APIs or unintended external calls.

For example, if traces show that a core insurance claims system is making repeated calls to a third-party risk-scoring API that was never documented, the observability platform can automatically trigger the creation (or update) of a corresponding configuration item (CI) in the CMDB. This enables the CMDB to reflect the actual operational footprint at all times, avoiding the stale, outdated inventory problems that plague many enterprises.

This enriched CMDB becomes a central intelligence asset for architects, developers and operations teams. Every time an architecture review is conducted, the architecture team can pull live configuration data directly from the CMDB, cross-referenced with observability data, to assess whether the architecture on paper matches the architecture in reality. If discrepancies arise, such as an unexpected dependency or a missing control point, the architecture team can proactively recommend fixes to both the application design and its supporting infrastructure.

The true value emerges when observability data is not only integrated with the CMDB but also linked directly to enterprise architecture standards, design documents and business metrics. Observability 2.0, as defined by the Cloud Native Computing Foundation (CNCF), emphasizes the need to connect technical signals to business outcomes. For example, if logs show degraded response times for policy quotes in an auto insurance platform, the architecture review can link that degradation back to architectural decisions – perhaps an overreliance on synchronous credit checks – and quantify the business impact in terms of reduced policy sales or increased customer abandonment.

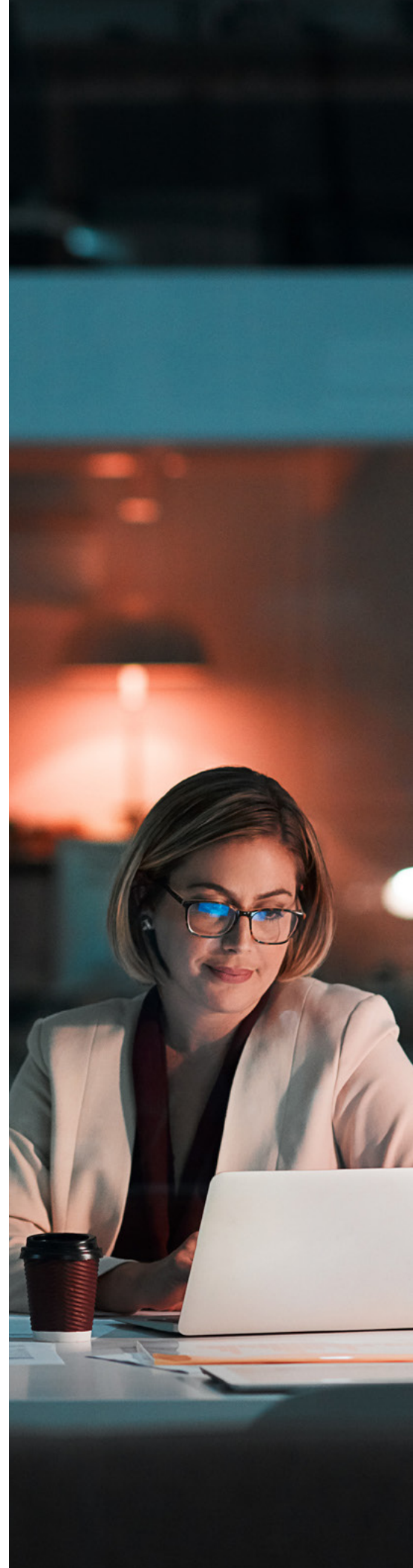
This creates a powerful, closed-loop improvement system, where architecture reviews, observability insights, CMDB accuracy and business performance tracking are all interconnected. Architects no longer work from outdated documentation but instead have real-time architectural intelligence driven by operational data. This means that architecture decisions are directly tied to their real-world business impact, transforming architectural reviews from isolated exercises into continuous contributors to business value delivery.

Integrating observability data with architectural processes

Integrating observability data with architectural processes also enhances configuration management accuracy. Observability data can automatically highlight discrepancies between what the CMDB expects and what is actually deployed.

For example, if production traces show frequent calls to an undocumented third-party API, this triggers an automatic review and update to the CMDB. This creates a self-healing CMDB process that evolves alongside production reality.

By linking observability data directly to both architecture standards and business metrics, architects can detect when technical drift starts affecting customer experience or compliance posture. This makes architecture oversight directly relevant to business risk and opportunity management.



04

CHAPTER

Workload placement is a strategic decision, not a onetime task

As enterprises accelerate their adoption of cloud, SaaS platforms and hybrid infrastructure, the decision of where to place a workload has become increasingly complex – and increasingly impactful.

No longer is workload placement simply about choosing the cheapest infrastructure provider. It now requires balancing:

- 01 Cost efficiency**
Infrastructure costs, licensing and operational expenses
- 02 Performance fit**
Proximity to users, response time SLAs, scalability requirements
- 03 Compliance and security fit**
Data residency, encryption requirements, audit capabilities
- 04 Operational ease**
Fit with current tools, automation and team skills
- 05 Business proximity**
Proximity to key data sources, partners or systems that feed the workload

Traditional static placement – where teams review architecture once, make a decision and assume it will hold for years – is now outdated. Modern technology delivery requires a continuous workload placement optimization process that dynamically revisits these decisions as conditions change. For example, let's assume there are analytical workloads executed from a reporting tool on a cloud-based data platform. One workload requires access to native cloud-based data platform table. A second workload requires similar data but also accesses machine learning models on a public cloud provider.

For each of these workloads, what are the appropriate cloud-based data platform compute clusters required? The choice of these clusters/workload placement dictates the cost of workload execution.

We recognize that more often than not, the balancing factors are not static – new regulatory requirements emerge, cloud pricing changes and workload performance expectations evolve. This makes a onetime placement decision insufficient. Instead, organizations need a dynamic scoring mechanism that continuously reassesses the optimal placement for each workload.



The EY engineering-led approach to placement uses a dynamic, mixed-integer linear programming model combined with a heuristic approach.

Mathematical formulation for workload placement optimization

1. Sets and indexes

$i \in I = \{1, \dots, n\}$: Set of workloads

$j \in J = \{1, \dots, m\}$: Set of target platforms

$k \in K = \{1, \dots, p\}$: Set of resources (CPU, memory, etc.)

$t \in T = \{1, \dots, q\}$: Set of time periods

Greek letter explanations:

- \in (epsilon): "Belongs to" or "is an element of"
- Σ (sigma): Summation
- λ (lambda): Typically used for rates or arrival patterns
- α (alpha): Often used for weighting factors
- β (beta): Secondary weighting factors
- γ (gamma): Tertiary factors or rates

2. Decision variables

Primary decision variable:

$x_{ij} = \{1 \text{ if workload } i \text{ is assigned to platform } j; 0 \text{ otherwise}\}$

Secondary variables:

$y_{jk} = \text{Amount of resource } k \text{ allocated on platform } j$

3. Objective function

Minimize $Z = \sum I \sum J (\alpha C + \beta L + \gamma T) \times x$

Where:

C_{ij} = Infrastructure cost for workload i on platform j

L_{ij} = Licensing cost for workload i on platform j

T_{ij} = Data transfer cost for workload i on platform j

α, β, γ = Weighting factors for different cost components

4. Constraints

4.1 Assignment constraints

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I$$

(Each workload must be assigned to exactly one platform)

4.2 Resource constraints

$$\sum_{i \in I} (r_{ik} \times x_{ij}) \leq R_{jk} \quad \forall j \in J, \forall k \in K$$

Where:

r_{jk} = Resource k required by workload i

R_{jk} = Total available resource k on platform j

4.3 Performance constraints

$$\lambda_{jk} \times x_{ij} \leq \Lambda_i \quad \forall i \in I, \forall j \in J$$

Where:

λ_{ij} = Expected latency for workload i on platform j

Λ_i = Maximum acceptable latency for workload i

4.4 Capacity constraints

$$\sum_{i \in I} (\delta_i \times x_{ij}) \leq \Delta_j \quad \forall j \in J$$

Where:

δ_i = Storage requirement for workload i

Δ_j = Available storage on platform j

5. Additional notations used

- \forall (for all): Indicates the constraint applies to all members of the set
- \leq (less than or equal to): Used in inequality constraints
- Σ (summation): Adds up all elements in the range
- \times : Multiplication operator
- Subscripts ($_{ij}$): Indicate specific elements or combinations

6. Solution space

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J$$

$$y_{jk} \geq 0 \quad \forall j \in J, \forall k \in K$$

7. Time-dependent extension

For time period t:

$$\text{Minimize } Z(t) = \sum_{i \in I} \sum_{j \in J} (\alpha C_{ij}(t) + \beta L_{ij}(t) + \gamma T_{ij}(t)) \times x_{ij}(t)$$

Where:

$C_{ij}(t)$ = Cost at time t

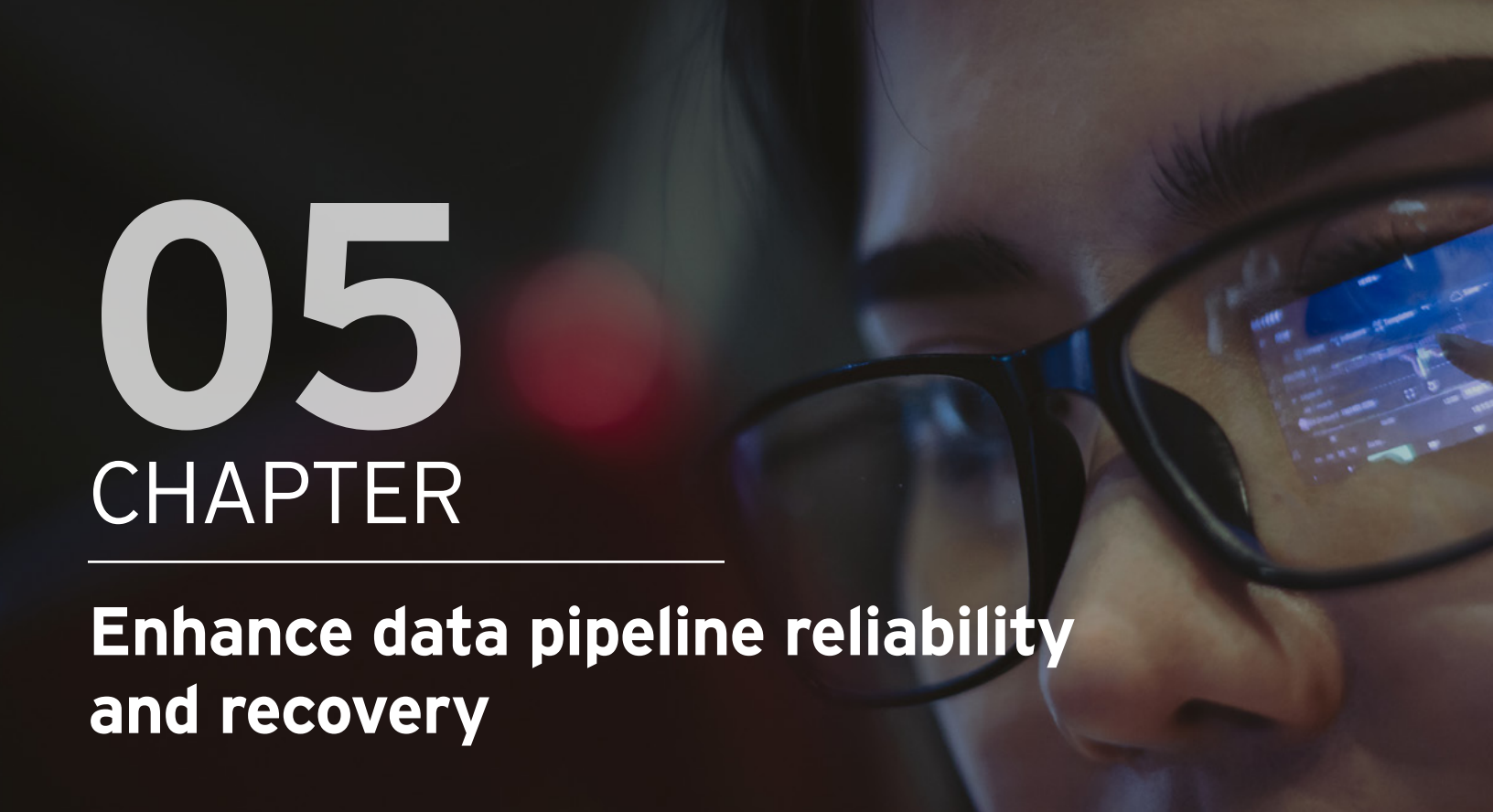
$L_{ij}(t)$ = License cost at time t

$T_{ij}(t)$ = Transfer cost at time t

Platform	Fit score	Placement cost	Placement score
Public cloud provider	85%	\$200k	0.425
Private cloud	75%	\$220k	0.341
SaaS credit platform	92%	\$250k	0.368

The EY approach also encodes other considerations into the decision-making process. Factors such as regulatory requirements, business needs and privacy needs factor into the eventual workload recommendation. However, all of these factors are part of the decision-making engine so workload placements can be addressed at scale anywhere across the enterprise.

In the example above, while the public cloud provider is less costly, the higher fit for regulatory requirements in a financial cloud may shift the placement toward private cloud, especially if regulations change.



05

CHAPTER

Enhance data pipeline reliability and recovery

The concept of self-healing data infrastructure is relatively new and is often mistaken for a mere observability solution focused on data pipelines. However, it represents a far more comprehensive approach.

Drawing a parallel to biochemistry, this solution mirrors the way bacteria defend themselves against viral attacks. Just as bacteria modify their DNA to repair damage and retain a memory of their defense mechanisms, a self-healing data platform leverages historical failure logs and remediation techniques to (semi)autonomously detect, diagnose and resolve pipeline failures. By continuously learning from past incidents, the system evolves, enabling proactive issue resolution and mitigating disruptions.

Self-healing data pipeline infrastructure

Traditional approaches to establishing data pipeline reliability and recovery have included implementing monitoring tools to detect and resolve anomalies, as well as using checkpoint techniques to resume pipelines from points of failure. The EY solution builds upon these methodologies by incorporating historical failure logs to create a more predictive and semiautonomous (trending autonomous) system for self-healing pipelines.

The solution consists of four key components:

- 01 Integrating with existing monitoring tools to establish centralized and appropriate triggers.
- 02 Leveraging machine learning and decision tree techniques to compare historical failure logs with current pipeline issues. This includes establishing an intelligent failure detection technique that classifies the error (security, quality, transient, etc.).
- 03 Providing semiautonomous (and autonomous) recommendations to systematically address failures with limited manual intervention. Leveraging machine learning/decision tree techniques (e.g., random forest or non-linear multistage classifiers) helps in identifying the anomaly and providing appropriate suggestions to remediate the issue.
- 04 Continuously enhancing reference and centralized failure logs to improve the effectiveness of recommendations in component 3.

Let's see how this solution applies in a real-world scenario:

A diversified financial services firm aggregates data from multiple sources (web, mobile, advisors, etc.) to process incoming transactions. These transactions are compiled and loaded into a central repository to maintain an audit trail and ensure chronological execution.

In one instance, the system accurately records a transaction with a value of \$1. However, historically, transactions of this type typically range between \$1,000 and \$10,000 (e.g., a recurring investment pool). Due to the built-in controls, the system processes \$1 as a valid numeric value without triggering any alerts.

With a self-healing pipeline solution, the anomaly would be detected, prompting the system to take the following actions:

- **Cross-system validation:** The system verifies the transaction value by comparing it with data from other enterprise sources to determine if the value was lost, overwritten or dropped during transmission. If the value is consistent across multiple systems, it is recorded as is. Otherwise, using a probabilistic approach, the system corrects the transaction value based on historical patterns and data from other sources while flagging it for manual review.
- **Proactive error prevention:** Based on the root cause analysis, the system generates recommendations for data administrators to refine validation rules and prevent similar discrepancies in the future.

By implementing self-healing capabilities, the pipeline enhances data integrity, minimizes errors and reduces manual intervention, so financial processing is more reliable.



Optimizing existing and proposed data models

Beyond enhancing pipeline reliability and recovery, the solution will also evaluate existing and future data model structures to uncover opportunities for cost optimization and efficiency. By benchmarking against industry-standard models and recommended EY leading practices, the system will systematically generate insights for model improvements.

The evaluation will consider key factors such as data type (structured, unstructured, analytical, operational, transactional), data volume query patterns, update frequency, performance expectations and security/privacy requirements. Based on these parameters, the solution will recommend suitable data serialization approach (e.g., parquet, JSON, key-value pairs as well as the suitable data structure) to improve storage costs and retrieval efficiency.

Additionally, by analyzing frequently accessed or grouped data patterns, the system will suggest partitioning and clustering techniques to improve query performance and reduce aggregation costs. EY domain-led data models will further assist in identifying structural gaps, offering recommendations to refine existing data models, choose between data model approaches for greater flexibility, resilient methods for integration of business requirements, and a more comprehensive representation of the organization's operations.

Closing the loop: continuous feedback between architecture, implementation and operations

As we have explored, the technology ecosystem is fully integrated across all five key areas of the enterprise landscape. No single component operates in isolation – each aspect is interconnected, spanning design, implementation, production performance and business impact. This creates a continuously evolving environment where design decisions are constantly validated, refined and improved based on real-world performance.

A key advantage of our recommended continuous feedback process is its proactive approach to technical debt management. Every deviation between the intended design and actual system behavior is tracked, tagged and quantified in terms of future operational and remediation costs. This enables CIOs and CTOs to make data-driven trade-offs between accelerating delivery today and building future-proof solutions. With each iteration, this process enhances the enterprise's technology knowledge base, leading to better design decisions and reduced uncertainty in cost and risk estimation.

EY teams have helped clients implement these solutions and adopt best practices to improve their technology landscape, lower operational costs and enhance transparency in technology processes. All of the methodologies discussed in this paper are embedded in the cost optimization solution that enables organizations to effectively manage their technology portfolios, enabling long-term efficiency and sustainability.



EY contacts



Manish Dhyani
Principal

Ernst & Young LLP
manish.dhyani@ey.com



Chandra Devarkonda
Managing Director

Ernst & Young LLP
chandra.devarkonda@ey.com

EY | Building a better working world

EY is building a better working world by creating new value for clients, people, society and the planet, while building trust in capital markets.

Enabled by data, AI and advanced technology, EY teams help clients shape the future with confidence and develop answers for the most pressing issues of today and tomorrow.

EY teams work across a full spectrum of services in assurance, consulting, tax, strategy and transactions. Fueled by sector insights, a globally connected, multidisciplinary network and diverse ecosystem partners, EY teams can provide services in more than 150 countries and territories.

All in to shape the future with confidence.

EY refers to the global organization, and may refer to one or more, of the member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients. Information about how EY collects and uses personal data and a description of the rights individuals have under data protection legislation are available via ey.com/privacy. EY member firms do not practice law where prohibited by local laws. For more information about our organization, please visit ey.com.

Ernst & Young LLP is a client-serving member firm of Ernst & Young Global Limited operating in the US.

© 2025 Ernst & Young LLP.
All Rights Reserved.

SCORE no. 27328-251US
2504-10667-CS
ED None

This material has been prepared for general informational purposes only and is not intended to be relied upon as accounting, tax, legal or other professional advice. Please refer to your advisors for specific advice.

ey.com