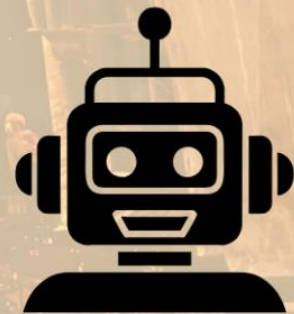


FEED FORWARD

October 2025 Volume 4 Issue 4

**E-Commerce Blueprint
Resilient Observability
AI, Kubernetes & Smart NIC
Thoughtful AI
AR in Pharma**



**IEEE
COMPUTER
SOCIETY**

Silicon Valley Chapter

**Editor**

[Rohan Rasane](#)

Chapter Chair

[Vishnu S. Pendyala, PhD](#)

Vice Chair

Harsh Varshney

Secretary

Rahul Raja

Treasurer

Srinivas Vennapureddi

Webmaster

Paul Wesling

Website & Media

- <https://r6.ieee.org/scv-cs/>
- <https://www.linkedin.com/company/78437763/>
- <https://www.linkedin.com/groups/2606895/>
- <https://www.facebook.com/IEEEComputerSocSCVchapter>
- <https://twitter.com/IEEEComputerSoc>

Mailing List

<http://listserv.ieee.org/cgi-bin/wa?SUBED1=cs-chap-scv&A=1>

Please note:

Feedforward (ISSN 3068-2525) is published quarterly by the Santa Clara Valley (SCV) of the IEEE Computer Society (CS), a non-profit organization. Views and opinions expressed in Feedforward are those of individual authors, contributors and advertisers and they may differ from policies and official statements of IEEE CS SCV Chapter. These should not be construed as legal or professional advice. The IEEE CS SCV Chapter, the publisher, the editor, and the contributors are not responsible for any decisions taken by readers on the basis of these views and opinions. Although every care is being taken to ensure genuineness of the writings in this publication, Feedforward does not attest to the originality of the respective authors' content.

All articles in this magazine are published under a Creative Commons Attribution 4.0 License.

From the Editor's Desk

As we present the October 2025 issue of *FeedForward*, I am inspired by the breadth of innovation our contributors bring—from reimagining **software architecture** to rethinking how **AI** and the **cloud** shape the future of work and technology. This edition highlights resilience, intelligence, and creativity in technology. Each article explores solutions that are technically sophisticated and deeply practical for the challenges we face today.

Here's a snapshot of what you'll discover in this issue:

- **A Comprehensive Software Blueprint for Building Multi-Tenant E-Commerce Platforms**
– Outlines scalable, tenant-aware **architectures** that power global commerce with agility and security.
- **Resilient Observability at the Retail Edge: A Lightweight, Scalable, and Cost-Efficient Framework**
– Demonstrates how Open Telemetry and Kubernetes enable **cloud-native observability** at the resource-constrained edge.
- **Accelerating Kubernetes for AI/ML and HPC Workloads with Smart NIC-Based Networking and RDMA Offload**
– Shows how **AI-driven cloud infrastructure** can leverage Smart NIC offloads to achieve breakthrough performance.
- **Thoughtful AI: Designing an AI Recruiting Agent that Thinks, Explains, and Reconsiders**
– Explores an ethical, transparent approach to **AI architecture in recruiting systems**, balancing automation with human judgment.
- **Using Augmented Reality and Digital Systems to Transform Pharma Equipment Qualification and Training**
– Reveals how **digital and cloud-based validation** and AR are modernizing pharmaceutical compliance and training.

Together, these articles reflect the forward-looking spirit of our community—where **architecture, AI, and cloud innovation** intersect to build scalable, resilient, and human-centered systems. Don't hesitate to contact me if you want to submit or review an article for our next issue.

— *Editor, FeedForward*

Acknowledgment

We extend heartfelt thanks to our dedicated reviewers whose expertise and thoughtful feedback have greatly enriched the quality of this publication: Dhruv Seth, Shivendra Srivastava, Nishant Satya Lakshmikanth

A Comprehensive Software Blueprint for Building Multi-Tenant Ecommerce Platforms

Priyadarshini Balachandran , Distinguished Software Engineer, Walmart Global Tech, CA,USA

Abstract—As global retailers expand their digital presence across countries, the need for platforms that support modular growth, configuration flexibility, and operational scalability becomes critical. Rapid globalization requires E-commerce platforms to scale effectively, addressing diverse market needs including distinct business rules, localization, and regulatory compliance. Traditional approaches involving separate codebases or isolated deployments per market typically result in increased complexity, fragmented functionality, and heightened operational overhead. This article outlines a holistic approach to building Multi-Tenant E-commerce Platforms, emphasizing unified codebases, tenant-aware APIs, and configuration-driven development. Additionally, it explores architectural methodologies to use such as Domain-Driven Design (DDD), Hexagonal Architecture, and Federated Frontend Orchestration, and practical strategies for localization, data isolation, governance frameworks, observability and security. The article also presents a foundational reference architecture to guide organizations in implementing scalable, adaptable, and maintainable Multi-Tenant E-commerce Platforms. These comprehensive strategies yield a robust platform for consistent global experiences, characterized by operational efficiency, structural flexibility, and high reusability.

Keywords: Multi-Tenant Architecture, E-commerce Platform, GraphQL Federation, Configuration-driven Development, Localization, Tenant Isolation, Observability, Domain-driven Design, Platform Security, Platform Governance

In response to the competitive global landscape modern platforms must efficiently accommodate diverse regional and customer-specific requirements. Historically, businesses have employed separate implementations or isolated deployments for individual markets, which often lead to increased costs, redundant development efforts, and fragmented user experiences [1]. Alternatively, multi-tenant architectures provide a compelling solution, offering unified management and sophisticated configuration capabilities that enable market specific adaptability. This paper explores critical strategies for multi-tenant platform architectures, drawing from practical retail and E-commerce implementations.

Single Codebase Paradigm A foundational principle of multi-tenant platforms is maintaining a sin-

gle codebase that supports all tenants. This trunk-based development model enforces consistency, ensures rapid propagation of enhancements or fixes, and eliminates drift across environments. It reduces the operational burden of managing multiple code forks and minimizes risk during upgrades. Centralized feature development and bug resolution uniformly benefit all tenants. Custom tenant requirements are accommodated not through code variations but through externalized configuration, keeping the underlying application logic clean and maintainable. Adopting a single artifact approach for builds and deployments also simplifies CI/CD workflows, supporting faster, safer delivery cycles across global environments.

Deployment Topologies A mature multi-tenant platform must support various deployment topologies tailored to tenant-specific needs. These include:

- 1) **Shared deployments**, where multiple tenants share the same runtime environment and infras-

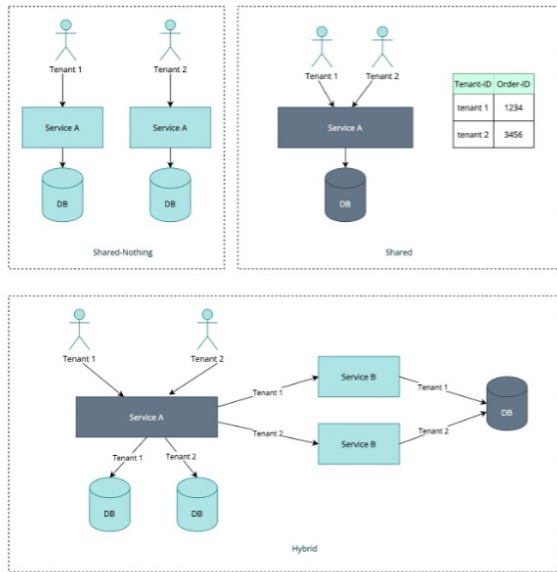


FIGURE 1. Data and Infrastructure Isolation Strategies- Shared-Nothing, Shared, and Hybrid Model

structure but are logically isolated through configuration, request context, and access control mechanisms.

- 2) **Isolated deployments**, where each tenant operates in a separate environment, often in its own namespace, cluster, or even cloud account, primarily to address stringent compliance, data residency, or performance isolation requirements.
- 3) **Hybrid models**, where critical workloads are isolated but certain shared services are maintained to optimize cost and resource utilization.

The selection of an appropriate topology hinges on factors such as the tenant’s scale, data sensitivity, regulatory obligations, and desired customization. The platform should offer comprehensive tools and automation to orchestrate these deployment models consistently across the ecosystem. The concept is further explored in below section.

Data and Infrastructure Isolation Strategies Multitenant systems can adopt one of three data & infrastructure isolation strategies — **shared, shared-nothing, or hybrid**, depending on the required level of isolation, scalability, and compliance [2]. As illustrated in Figure 1, in a shared model, tenants share the same infrastructure and database instances with tenant identifiers added to each request and as a database column, to enforce logical isolation. This is the most cost-efficient model and is preferred when tenants are numerous and their data security or performance require-

ments are modest. At the other end of the spectrum, the shared nothing model allocates entirely separate infrastructure per tenant, dedicated databases, or even clusters. This provides the strongest isolation and is suitable for highly regulated industries or for enterprise customers with strict SLAs. However, it also incurs the highest operational overhead and cost per tenant. The hybrid model strikes a balance by combining elements of both: shared services for common components (e.g., authentication or product catalog), and isolated infrastructure for sensitive workloads (e.g., order data or payment transactions). This hybrid approach provides flexibility in tuning performance, data locality, and compliance levels per tenant. Choosing the appropriate model depends on multiple factors including tenant scale, data sensitivity, regional regulatory requirements, and operational cost targets. Platforms may evolve from shared to hybrid or shared-nothing model as customer needs diversify. It is important to design the platform in a manner that makes this transition seamless at the service or domain level. **Tenant-aware API design** is fundamental for achieving logical isolation and accurate request routing in multitenant systems. APIs must explicitly incorporate tenant identifiers, typically passed via headers, tokens, or subdomain routing, to establish the correct context for each request. This context ensures that all downstream processing, including configuration lookup, access controls, and data operations, is correctly scoped and isolated. Internally, services must rigorously validate the tenant identifier against an authorized registry to prevent unauthorized cross-tenant data access. Tenant scoping should extend to API rate limiting, caching, and observability (e.g., telemetry collection) to maintain robust security and operational boundaries. Consistent propagation of tenant context across micro-services is critical to preserving tenant isolation and enabling fine-grained operational control.

Header Based Request Routing In multi-tenant platforms, header-based request routing enables dynamic, tenant-specific traffic steering at runtime, ensuring isolation and control without requiring tenant-aware logic in application code.

As illustrated in Figure 2, one practical implementation of this approach uses Istio Service Mesh, where its proxies (e.g., sidecars) inspect custom HTTP headers (such as X-Tenant-ID) [3]. Based on this header, Istio routes each request to the correct backend instance or service subset. This capability enables flexible topologies, from shared to hybrid environments, and facilitates granular traffic control policies defined through configuration, rather than embedded code. Istio’s proxy sidecar intercepts incoming traffic and

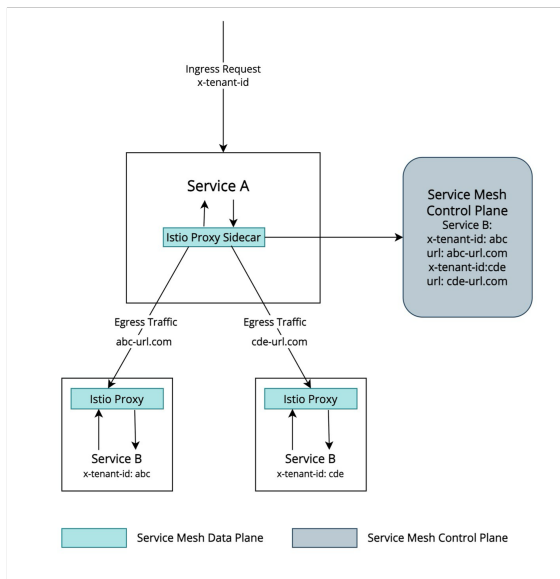


FIGURE 2. Istio-based header routing for tenant-aware request segmentation

FIGURE 2. Istio-based header routing for tenant-aware request segmentation

based on the `x-tenant-id` header, forwards the request to the appropriate service instance (e.g., Service B instances for tenant abc or tenant cde). The Istio proxy sidecar receives routing instructions from the control plane, which maps each tenant ID to a designated backend service URL and configures the proxy with the necessary routing rules.

Configuration-driven Development enables a single codebase to serve multiple tenants by externalizing and dynamically toggling behavior through data, eliminating the need for core application code changes. A common pattern is layered configuration: define default or global settings that apply to all tenants (or groups of tenants like regions), then override them with tenant-specific layers as needed. At runtime, the platform resolves the effective config for each request based on the tenant context, often by merging the global template with any per-tenant overrides [4]. This approach avoids duplicating code or wholesale forking, only the differences are specified in metadata or config files. While highly beneficial, implementing configuration-driven capabilities in multitenant environments introduces unique challenges. Managing the complexity of a vast array of configurations across numerous tenants, each with potentially distinct settings and compliance requirements, demands robust management tools and clear governance frameworks. Teams must treat configuration changes with the same rigorous discipline applied to code changes, imple-

menting version control, automated testing, and staged rollouts. Ad hoc per-tenant code branches should be avoided; instead, leverage feature flags and configuration switches to safely enable or disable features per tenant. For example, feature flag frameworks (e.g. LaunchDarkly) enable the toggling of features for specific tenants or cohorts at runtime, supporting gradual rollouts and easy rollbacks with full audit trails. Every tenant-specific tweak should be captured in a config store or metadata layer with proper auditing and versioning so you can track who changed what and roll back if needed. By combining hierarchical configuration layers with rigorous change management, platforms can offer flexible tenant-specific functionality without sacrificing maintainability or security.

ARCHITECTURAL PATTERNS

Domain-Driven Design (DDD) is a software development approach that centers on modeling software to precisely align with a specific business domain, drawing extensively from the expertise of domain specialists [5]. Its fundamental objective is to manage software complexity by placing the core focus of development on the 'domain', the specific business context in which the software operates. At the heart of DDD lies the concept of a Ubiquitous Language, a common, shared vocabulary consistently employed by all project stakeholders, from developers to business experts, ensuring the software accurately reflects intricate business details. This linguistic consistency minimizes misunderstandings and enhances collaboration, particularly crucial in dynamic environments like E-commerce where business requirements are constantly evolving. DDD proves exceptionally helpful in building ecommerce platforms due to its emphasis on deep business alignment and modularity. E-commerce systems are inherently complex, involving diverse domains like product catalogs, order management, payment processing, and customer accounts, each with unique business rules. By using Bounded Contexts, DDD helps decompose these complex domains into smaller, manageable parts, allowing for independent development, deployment, and scaling of services (e.g., microservices for "Product Catalog" or "Order Management"). This modularity enhances scalability and resilience, while the Ubiquitous Language ensures that technical teams and business stakeholders share a precise understanding of terms like "Order" or "Customer," which is vital for accurately translating evolving business needs into software solutions. DDD also encourages the classification of domains into core, generic, and supporting subdomains, each playing a distinct role in the plat-

form's architecture. Core domains are central to the platform's competitive differentiation, such as product or pricing engines and require the most investment and strategic control. Generic subdomains (like user authentication or payment processing) are necessary but not unique and can often be supported via third-party services or reusable modules. Supporting subdomains, like customer service tools are specialized but not core, and may vary in implementation between tenants. Recognizing these distinctions allows platform teams to focus their resources strategically and identify areas where reuse, outsourcing, or configuration-driven variability is most appropriate.

To maintain alignment and interoperability across domains, DDD recommends the use of context maps. A context map documents how different bounded contexts interact, defining the integration relationships (e.g., customer/supplier, conformist, anti-corruption layers) between them. This high-level blueprint helps teams coordinate dependencies and communication patterns, particularly important in multi-tenant environments where isolated teams may be evolving their services independently while maintaining consistent data contracts and platform behavior.

Hexagonal architecture, also known as the ports and adapters pattern, structures a system so that the core domain logic is completely independent of external interfaces, frameworks, or delivery mechanisms [6]. This design creates a clean boundary between the inner domain model and outer layers such as APIs, databases, messaging systems, or UI frameworks. Interactions between the domain and external systems are abstracted through well-defined interfaces (ports), which are implemented by adapters. This allows the business logic to remain pure, testable, and portable across technologies or deployment environments.

In the context of a multi-tenant E-commerce platform, hexagonal architecture is especially valuable for maintaining clean separation between shared domain logic and tenant-specific integrations. For example, tenants may use different payment gateways, content delivery networks, or customer support systems.

By encapsulating these external dependencies as adapters, the core logic for processing orders or managing catalogs can remain unchanged. This not only improves maintainability but also simplifies onboarding new tenants with varying integration needs. Hexagonal architecture also supports extensibility across deployment models: the same core domain can be reused across shared, hybrid, or isolated topologies, with tenant-aware adapters injected as needed to handle context-specific behavior without contaminating the domain logic.

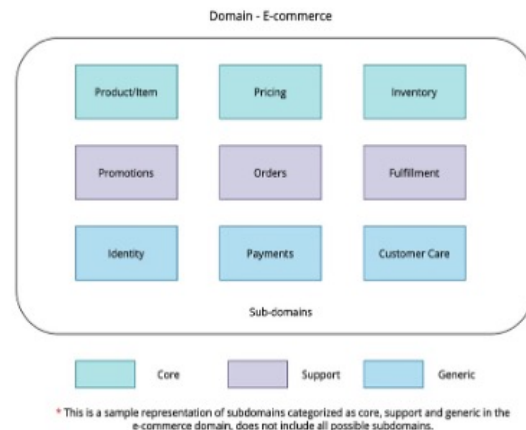


FIGURE 3. Representation diagram for Core, Support and Generic Classification

Frontend Orchestration using GraphQL Federation is an architectural pattern that allows multiple independent GraphQL services (called "subgraphs"), each owned by a different domain team, to be composed into a single, unified GraphQL API (a "supergraph"). This supergraph is then exposed through a central GraphQL Gateway [7]. For E-commerce platforms, this means teams responsible for domains like Product Catalog, Order Management, Customer Accounts, or Promotions can each develop and maintain their own GraphQL subgraph, defining the data and operations relevant to their domain. The Gateway acts as an orchestration layer, receiving complex queries from frontend applications, breaking them down, routing parts of the query to the appropriate subgraphs, stitching the results together and returning a consolidated response. This approach effectively federates the API layer, allowing for decentralized development while presenting a single, coherent data graph to consumers. This federated API orchestration significantly simplifies frontend development and supports multiple channels (web, mobile, in-store kiosks) with reduced complexity. Instead of frontends needing to know about numerous backend microservices and making multiple REST calls, they can query a single, unified GraphQL endpoint to fetch all necessary data for a given view or interaction. For instance, an E-commerce product detail page might require product information (from the Product Catalog subgraph), customer reviews (from a Reviews subgraph), and personalized recommendations (from a Recommendations subgraph). With GraphQL Federation, multiple front end client applications send queries requesting for attributes needed to paint the view of that application, to the Gateway.

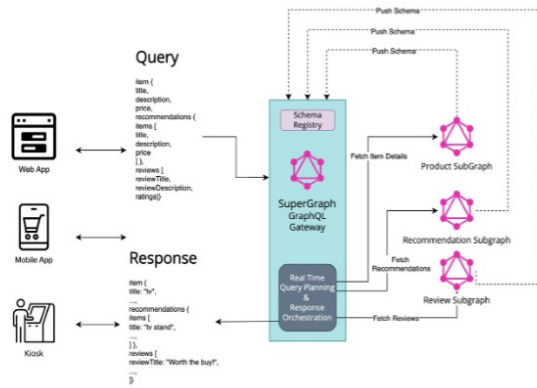


FIGURE 4. Federated Orchestration for FE Applications using GraphQL Federation

Gateway refers to the super graph and creates a query plan real to identify and call corresponding subgraph providers and collates the response to send it back to the clients. This reduces the burden on frontend developers, as they no longer manage complex data aggregation logic, leading to faster development cycles and easier adaptation to new channels or evolving business requirements. It also ensures data consistency and reduces chatty network requests improving performance across various client applications.

IMPLEMENTATION CONSIDERATIONS

Theming & Tenant Specific Content Management

Tenant-specific theming allows E-commerce platforms to provide uniquely branded and personalized storefronts while preserving a consistent underlying customer journey. The most effective way to achieve this is through component-driven development, where the UI is built as modular, reusable parts that adapt dynamically to each tenant's branding and content, without duplicating business logic or navigation [8]. The process starts with treating branding and layout preferences as configurable data. When a customer lands on the site, typically via a tenant-specific entry point like a branded subdomain or after login, the frontend determines the active tenant's identity. It then loads that tenant's configuration (including color palettes, fonts, logo URLs, and layout options) along with editorial content like banners, promo messages, and navigational copy. These assets and settings are managed centrally in a CMS or configuration API, allowing for easy updates by nontechnical users. For example, React's component-based architecture is well

sued for this approach. Core UI components such as headers, footers, and product cards remain standardized, but dynamically reference the tenant's configuration to apply the correct styles and content. Modern CSS frameworks, further streamlines this process by providing utility classes that can be adapted in real time based on the tenant's theme settings, ensuring efficient rendering of visually distinct storefronts with minimal code branches. All customizations are handled declaratively via configuration and centrally managed content delivery, so engineers don't have to maintain separate code for each tenant. Throughout the entire experience, backend APIs and the frontend itself consistently pass along and respect tenant context, ensuring that data, layouts, and user flows are always relevant to the active merchant's storefront. Importantly, while layouts and content can be tailored to each tenant, crucial steps in the customer journey, such as browsing, cart management, and checkout, remain unified, providing stability for platform upgrades and a consistent experience for shoppers.

Localization & Compliance Localization and compliance should be deeply integrated into both the platform's configuration model and its request processing flow. Platform services must normalize locale-sensitive data representations, such as number formats, currencies, and datetime values, at ingress and render them appropriately at egress based on resolved locale context. These transformations should occur within middleware or interceptors, ideally at the BFF or API gateway layer, to minimize duplication across services. Locale configurations, including supported languages, currency mappings, fallback hierarchies, and legal content variants, should be declaratively defined in a centralized metadata service scoped per tenant and environment. This metadata is versioned, validated, and served via a low-latency configuration service or edge cache to enable efficient runtime lookups. Localization in multi-tenant E-commerce platforms must be structured to resolve language, currency, and formatting preferences dynamically, while ensuring consistency across all touchpoints, including formatting conventions and time zone logic. This is achieved by externalizing locale-sensitive assets and rules into structured configuration layers, decoupled from the application logic. At runtime, locale resolution follows a layered strategy: it first checks for explicit user preferences (such as a selected language or currency), then stored session or profile settings, and finally defers to headers like Accept-Language. The platform uses this context along with tenant configuration metadata to determine the appropriate locale and apply the correct formatting logic. Translations are typically stored in language-

specific resource bundles managed via localization platforms and loaded at runtime, supporting rapid iteration and fallback mechanisms. A hierarchical fallback strategy, where the system cascades from locale-specific to base language to default strings, ensures resiliency when translations are missing, preserving continuity in the user experience. Compliance is tightly coupled with legal and transactional standards. Platforms must account for regulatory variance across tenants by incorporating configuration-driven enforcement mechanisms that allow each tenant to declare region-specific compliance logic, such as tax display rules, consent policies, and data handling standards, without hardcoding them into the platform. A key methodology is to isolate compliance-sensitive rules into a tenant-aware configuration registry, enabling runtime evaluation and API-level enforcement. This separation allows the platform to evolve independently of tenant-specific compliance policies, while ensuring full auditability and traceability of regulated behaviors. Together, this model supports a scalable and governable path for delivering compliant, personalized E-commerce experiences across diverse regions.

Tenant-specific Observability is essential for monitoring, debugging, and optimizing multi-tenant platforms at scale. A well-designed observability framework must ensure that logs, metrics, and traces are enriched with tenant context, typically using tenant identifiers propagated across service boundaries. These tags enable filtering and correlation of telemetry per tenant, which is critical for isolating incidents, performing cost attribution, and enforcing SLAs. Platform teams can use tenant-labeled dashboards and alerts to quickly diagnose performance bottlenecks, usage anomalies, or errors scoped to specific tenants without affecting others.

In asynchronous flows, such as background jobs, message queues, or event-driven pipelines—observability becomes more complex because tenant context is not inherently carried in a synchronous request-response cycle. To preserve tenant-specific tracing and metrics across these decoupled services, the tenant identifier must be explicitly injected into the message payload or envelope metadata when the message is first enqueued. Consumers then extract and propagate this context through structured logs and trace spans. At runtime, tracing systems like OpenTelemetry or Jaeger should propagate tenant identifiers via trace context headers to maintain cross-service lineage. Logs should include tenant ID as a structured field to support fine-grained search and aggregation, while metrics should expose tenant-scoped dimensions (e.g., latency

tenant_id=abc) to allow precise monitoring. This design supports a multi-tenant shared platform model while enabling operational accountability and tenant-level visibility. It also enables advanced capabilities like tenant-specific anomaly detection, usage analytics, and billing insights—without compromising global system observability. Operational efficiency necessitates comprehensive monitoring and observability frameworks capable of isolating metrics, logging, and tracing by tenant identifiers. Such frameworks facilitate precise identification of performance issues and enable timely, tenant-specific incident resolutions.

Robust Governance and Change Management are vital for ensuring compliance with multi-tenant architecture principles while preserving platform agility. The foundation of effective governance begins with automated validation tools that proactively scan code, service manifests, architecture diagrams, and configuration artifacts to flag deviations from approved platform patterns. These tools enforce consistent use of tenant-aware design practices, such as context propagation, tenant-scoped configuration, and secure isolation boundaries, through CI/CD-integrated checks and policy-as-code. Complementing these automated safeguards, Architectural Review Boards (ARBs) act as a strategic oversight mechanism [10]. ARBs should be embedded early in the design process and structured to provide fast, asynchronous reviews using predefined templates and checklists. Rather than serving as bottlenecks, these forums guide teams in aligning with platform blueprints, balancing trade-offs between isolation and reuse, and validating technology decisions for new or evolving services. ARBs play a critical role in governing cross-tenant concerns, shared service extensions, and new market launches, ensuring architectural integrity at scale. To support high-impact or tenant-visible changes, Change Advisory Boards (CABs) and domain-specific governance councils may be employed. However, CABs should focus on exception handling, such as compliance-sensitive workflows or high-risk rollouts, while the majority of routine changes flow through automated pipelines with embedded guardrails. This tiered model helps maintain delivery velocity while preserving alignment with security, compliance, and multi-tenancy standards. Combined, these governance structures form a layered framework that supports scalable, evolvable, and policy-compliant multitenant platform development.

Tenant Aware Security Implementation Security in a multi-tenant commerce platform must be embedded at every layer of the architecture to ensure that tenant boundaries are enforced, and sensitive

data remains protected. A defense-in-depth strategy [9] should be implemented through the following measures:

- 1) **Policy Enforcement Across Layers:** API gateways, service meshes, and data layers must enforce tenant-scoped security policies. Context-aware access controls should be applied early in the request lifecycle using tenant identifiers.
- 2) **Centralized Identity Management:** A centralized identity provider should authenticate users once and propagate trusted identity claims (user ID, roles, permissions) downstream. This minimizes authentication overhead and prevents fragmented session handling.
- 3) **GraphQL Federation Security:** In federated GraphQL architectures, authentication is centralized at the gateway (supergraph) level, with secure propagation of identity to subgraphs. Subgraphs should perform their own authorization checks using this identity and accept requests only from trusted sources. JWTs enable stateless identity propagation, and custom directives (e.g., @auth) can enforce field-level controls.
- 4) **Context Preservation in Asynchronous Workflows:** In a multi-tenant E-commerce platform, authentication in asynchronous flows relies on secure, tenant-aware service credentials to ensure that background jobs, webhooks, and event-driven tasks act with the correct tenant context. Each automated process—such as inventory syncing or order fulfillment—should receive a short-lived, scoped token that encodes both its system identity and the tenant it represents. These tokens are commonly issued via OAuth 2.0 or a similar identity provider, giving each worker or automation only the minimal access needed for its assigned tenant. For cross-platform integrations like webhooks or scheduled exports, every asynchronous message or job should include a verifiable signature or token (such as an HMAC or JWT) uniquely tied to the tenant and the operation. On receipt, the platform must validate this credential, confirm tenant scope, and enforce role-based access to segregate data and permissions between tenants. All operations should be audited, with logs tracking token use and tenant association, ensuring accountability and rapid detection of cross-tenant access violations. This strict separation maintains both security and data privacy in multi-tenant, asynchronous E-commerce environments.

- 5) **Data and Secrets Security:** Data must be encrypted in transit and at rest. Sensitive data should be tokenized when necessary. Secrets and credentials must be tenant-scoped and securely isolated, especially in hybrid deployment models.
- 6) **Tenant-Specific Anomaly Detection:** Monitoring systems should analyze behavior patterns per tenant to detect suspicious activity, abuse, or policy violations. By applying these layered and tenant-aware security controls across infrastructure and application tiers, platforms can deliver strong isolation, protect workloads, and meet diverse compliance requirements without compromising performance or scalability.

REFERENCE ARCHITECTURE

Figure 5 illustrates a reference architecture for building a multi-tenant E-commerce platform.

- 1) **Client Interfaces:** Multiple user-facing channels, including web, mobile, and kiosk interfaces, initiate ingress requests carrying tenant context (e.g., x-tenant-id). The edge layer enforces high-level routing, authentication, and caching strategies. Additionally, each client interface can dynamically apply tenant-specific branding, theming, and feature toggles based on configuration resolved from the tenant context, enabling personalized UI layouts, logos, color palettes, and content variations tailored to each tenant's brand identity. The edge layer enforces high-level routing, authentication, and caching strategies.
- 2) **GraphQL Federation Layer:** The GraphQL Gateway serves as the centralized entry point. It authenticates sessions, validates tokens, and attaches identity claims and tenant metadata to downstream calls. Subgraphs are domain-aligned and registered in the Supergraph Schema Registry, enabling teams to push schema updates independently.
- 3) **Domain Subgraphs:** Each subgraph (e.g., Cart, Order, Recommendations) is responsible for handling domain-specific queries and commands while enforcing tenant-aware access control. The subgraph receives tenant context and user identity from the gateway and uses this context to orchestrate and delegate business operations to backend services, ensuring that access control and business logic execution are correctly aligned with tenant-specific requirements.
- 4) **Domain Services:** Stateless microservices han-

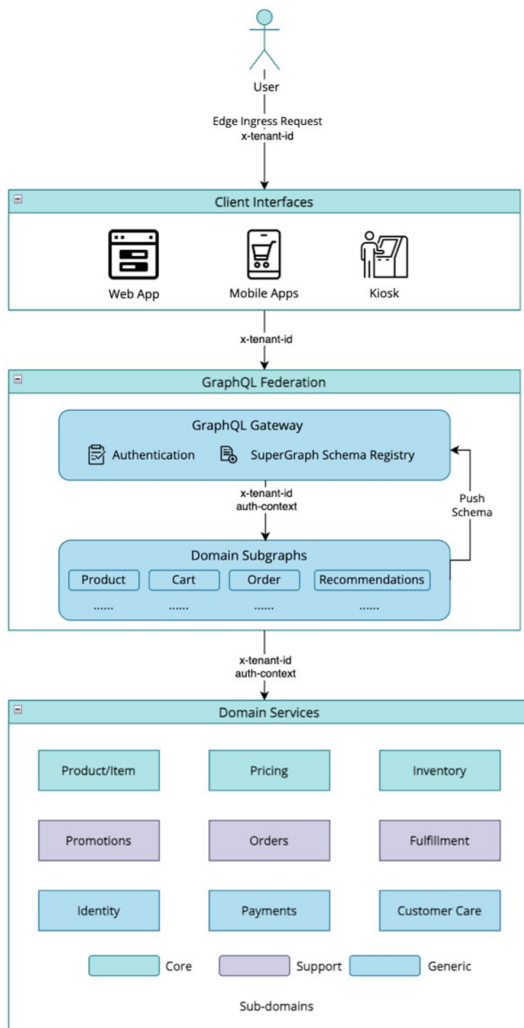


FIGURE 5. Reference Architecture for Multi-tenant E-commerce platform

dle business logic and persistence. Subgraphs delegate to these services, which apply tenant-specific configuration and isolation logic when executing business operations. Subdomains are categorized as core, generic, or support services to guide strategic focus. This reference architecture prioritizes modular development, configuration-driven extensibility, and strong tenant-aware controls. While the diagram emphasizes the tenant-aware synchronous request and response path across clients, federated APIs, and domain services, several foundational components and runtime dimensions are intentionally abstracted:

- 1) **Deployment Topologies:** The diagram assumes a generic deployment layout. In real-world implementations, shared, hybrid, and tenant-isolated deployments may coexist, with environment-specific controls configured per service or domain.
- 2) **Asynchronous Flows:** Asynchronous Flows: Including background jobs, webhook processing, pub/sub events, and system-generated automation. These require secure tenant context propagation, authenticated execution tokens, and tenant-aware observability. Authentication of asynchronous processes should be handled through signed tokens (e.g., JWT or HMAC) scoped to both system identity and tenant context.
- 3) **Resilience and Failover:** Multi-zone and multi-region deployments, active-active failover configurations, and traffic shaping mechanisms are essential for global commerce use cases but are not illustrated here.
- 4) **Platform Infrastructure:** CI/CD pipelines, observability layers, feature flag systems, configuration registries, secret management services, and policy enforcement engines underpin the runtime governance and scalability of the platform.

Together, this architecture provides a scalable foundation for building secure, customizable, and resilient multitenant -commerce platforms.

CONCLUSION

The modernization of commerce platforms requires balancing standardization with market-specific flexibility. By embracing multi-tenant design patterns, domain-driven modularity, and disciplined operational governance, organizations can deliver consistent, localized customer experiences at global scale. This architectural approach significantly reduces complexity, mitigates redundancy, fosters continuous innovation, and ensures sustained technical and operational excellence within an evolving global market landscape.

Looking ahead, future improvements could focus on enhanced tenant-aware developer tooling, AI-assisted configuration management, and more dynamic adaptation of security and observability policies. As regulations and customer expectations evolve, investing in policy-as-code, automated compliance validation, and real-time configuration intelligence will further strengthen platform agility. A modular, composable foundation, when paired with adaptive operations, can ensure the platform remains resilient, secure, and com-

petitive for years to come.

REFERENCES

1. Anusuru, R. B., et al. (2021). Transforming E-Commerce Platforms through Multi-Tenant Architectures and Microservices. *International Journal of Advanced Computer Science and Applications*.

2. Amazon Web Services, “Silo, pool, and bridge models,” AWS Well Architected SaaS Lens, [Online]. Available: <https://docs.aws.amazon.com/wellarchitected/latest/saas-lens/silo-pool-andbridge-models.html>.

3. Istio Documentation. (n.d.). Routing by headers. Retrieved from [Istio: Routing by Headers](#)

4. J. Schroeter, P. Mucha, M. Muth, K. Jugel, and M. Lochau, “Dynamic configuration management of cloud based applications,” in Proc. 16th Int. Software Product Line Conf. (SPLC), vol. 2, Salvador, Brazil, 2012, pp. 171– 178, doi: 10.1145/2364412.2364441.

5. E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison Wesley, 2004.

6. A. Cockburn, “Hexagonal Architecture,” alistair.cockburn.us, 2005. [Online]. Available: <https://alistair.cockburn.us/hexagonalarchitecture/>

7. Apollo GraphQL, Apollo GraphQL at Enterprise Scale, [Online]. Available: <https://www.apollographql.com/Apollo-graphql-at-enterprise-scalefinal.pdf>.

8. UXPin, “Component-Based Design: Complete Implementation Guide,” UXPin Studio Blog, [Online]. Available: <https://www.uxpin.com/studio/blog/component-based-design-completeimplementation-guide/>.

9. U.S. Cybersecurity and Infrastructure Security Agency (CISA), *Defense in Depth*, NCCIC/ICS-CERT, 2016. [Online]. Available: [CISA Defense in Depth \(2016\)](#).

10. Amazon Web Services, “Build and operate an effective architecture review board,” AWS Architecture Blog, Jan. 25, 2023. [Online]. Available: <https://aws.amazon.com/blogs/architecture/build-and-operate-an-effectivearchitecture-review-board/>.

Priyadarshini Balachandran is a Distinguished Software Engineer and Senior Technology Leader with over 15 years of experience designing and scaling complex distributed systems, cloud-native platforms, and AI-enabled engineering solutions. Her technical expertise spans domain-driven design, GraphQL federation, and event-driven architecture, with a strong focus on building multi-tenant platforms that support high availability, global scale, and seamless customer experiences. Priya is known for driving platform modernization and architectural transformation in fast-moving enterprise environments. She currently leads strategic initiatives in agentic AI to improve developer productivity, accelerate engineering workflows, and advance operational excellence. She holds a bachelor's degree in information technology from Anna University, Chennai, and is passionate about building systems and teams that scale with purpose.

Resilient Observability at the Retail Edge: A Lightweight, Scalable, and Cost-Efficient Framework

Prakash Velusamy, *Site Reliability Engineer Phoenix AZ, USA*

Abstract—Monitoring the availability and performance of systems and applications for the Retail environment at their edge locations remains complex and challenging. This can be attributed to several factors, including 1) Infrastructure and Resource Constraints, Edge computing devices often have central processing unit (CPU), memory, and storage constraints, which pose a challenge in collecting, storing, and analyzing the observability data. Unlike centralized data centers or cloud platforms, where vertical or horizontal scaling of resources is available, edge devices must operate with limited compute resources available. 2) Network Bandwidth Limitations, there would be a lot of data transmitting from the edge to the central system through the network. Network connectivity is mostly provided by a service provider. It is not necessary in a distributed environment that all the edge locations will have the same amount of network bandwidth available. It varies based on location where the edge is located and service provider. Monitoring data always takes the least priority compared to application data. These factors make it difficult to maintain a consistent data flow to central monitoring systems. 3) Tooling Challenges, the observability tools, their costs, and volume of data collected from them pose another challenge to enterprise organizations when it comes to edge observability. Traditional centralized monitoring tools are either not specifically designed for edge use cases or usually expensive, especially to collect the data from thousands of distributed edge locations. All these led to look for new approaches and cost-effective solutions. 4) Importance of Edge Observability, though there are challenges in enabling the observability for edge locations, it is important to collect and manage edge telemetry data, metrics, logs, and traces to get deep insight into systems and applications performance to ensure the reliability and end user experience. 5) Having a good observability at the edge can lead to a) Reduce the MTTD (Mean Time to Deduct) and MTTR (Mean Time to Repair) of availability and performance issues. b) Get insight into the observability data close to real-time. c) Correlate observability metrics with business metrics. d) Optimize and measure the resource utilization and application performance. To fulfil the need of modern edge observability, it demands leveraging open-source, cloud native, cost effective, less resource intense solutions.

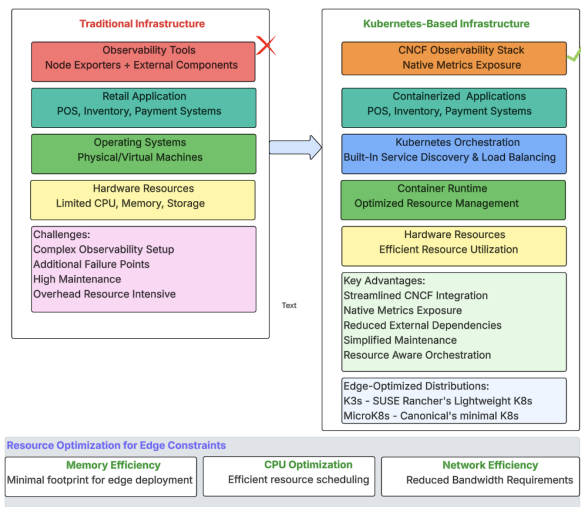


FIGURE 1. Streamlined deployment of K3s/MicroK8s (Lightweight Kubernetes/Micro Kubernetes) workloads in a lightweight Kubernetes environment

OPTIMIZING OBSERVABILITY THROUGH INFRASTRUCTURE ENHANCEMENTS

This architecture demonstrates how Kubernetes natively exposes monitoring metrics, reducing external dependencies and complexity compared to traditional physical or virtual machine infrastructures. Lightweight distributions (K3s/MicroK8s) retain core Kubernetes functionality while minimizing resource footprint for edge deployment constraints.

Current state: The telemetry and observability solutions of edge closely depend on the underlying infrastructure framework deployed and its support model. There are still many retail environments that have their infrastructure running on traditional physical or virtual machines, especially non-Kubernetes.

Challenges: The implementation of modern open-source and cloud native observability solutions such as Prometheus or OpenTelemetry (OTel) is very complex in such infrastructures. To make it work, it needs additional components such as node exporters to be installed, which in-turn adds additional points of failure and complexity from resourcing, architecture, reliability and scaling perspective.

My Proposed Solution: In contrast to existing approaches, this work proposes Kubernetes based infrastructure leveraging Cloud Native Computing Foundation (CNCF) telemetry components “see Figure 2”. Kubernetes in-built architecture exposes the monitor-

ing metrics by default. It requires minimal resources consumption, which is a critical need for edge environments, and has fewer external dependencies. This ultimately removes complexity in the architecture, maintenance requirements and drives a resilient observability solution. It is possible that due to resource limitations, the implementation of a full-fledged Kubernetes environment is not possible. However, there are several vendors available in the market who have products developed specifically for the edge computing use cases. Those products have lightweight versions of Kubernetes often referred to as K3s or microK8s, which can function like normal Kubernetes on the edge. They retain the core functionalities of Kubernetes to function and remove the remaining additional functionalities to keep it light. Through mapping of infrastructure and observability design, enterprises can build a scalable, resilient telemetry solution across distributed environments at the edge.

Contributions

My main contributions to resilient edge observability in retail environments are:

Novel Resource-Optimized Architecture The solution presents first comprehensive framework that combines lightweight Kubernetes (K3s/MicroK8s), OTel, and fluentd to achieve edge observability with 63% reduction in memory usage (from 3.5GB to 1.3GB) and 96% reduction in network bandwidth consumption (from 18Mbps to 650Kbps) compared to traditional Prometheus-based solutions.

Production-Scale OTel Implementation The framework provides the first documented end-to-end deployment of OTel for metrics collection and distributed tracing in retail edge environments, identifying and resolving critical compatibility issues (metric compatibility bugs and data type mismatches) that have not been previously addressed in production environments.

Integrated Full-Stack Telemetry Solution It demonstrates a complete observability solution that seamlessly correlates front-end user experience metrics (WebVitals, session data) with backend infrastructure and application telemetry data through shared trace identifiers, enabling transaction-level visibility across distributed retail systems.

Cost-Efficient Edge Logging Architecture The solution introduces a logging approach using Fluentbit-Fluentd combination with structured JSON (JavaScript Object Notation) formatting and filtering that reduces log volume by up to 60% while maintaining critical diagnostic capabilities for edge environments with bandwidth constraints (25-50Mbps shared with application

traffic).

Practical Implementation Guidelines It provides clear optimization techniques and configuration parameters validated across distributed retail environments, including specific metric collection strategies (25 essential metrics), compression methods, and bandwidth management approaches that make the solution viable for rural edge locations with limited connectivity.

Potential Solutions for Edge Observability The following solutions address the challenges of edge environments, including their distributed in nature, cost limitations, and other constraints,

TABLE 1. Comparison of Prometheus and OTel

Component	Prometheus	OTel	Key Characteristics
Metrics	Supported - TSDB (Time Series Database) Resource Needed– 3.5GB	Supported - TSDB (Time Series Database) Resource Needed– 1.3GB total for metrics and traces	Prometheus: Pull-based model, PromQL query language, built-in alerting OTel: Vendor-agnostic, push/pull flexible, standardized data model
Traces	Not supported	Supported - Tempo	Prometheus: Focused solely on metrics collection OTel: Distributed tracing with sampling capabilities
Logs	Not supported	Supported – Fluentd/Syslog-NG (System Log - New Generation), Splunk	Prometheus: No native log support OTel: Multiple log shipper integrations, structured logging
Frontend User Interface Monitoring	Not supported	Supported – Metrics, Traces, and Logs	Prometheus: Requires Grafana for visualization OTel: Unified observability across all telemetry data types

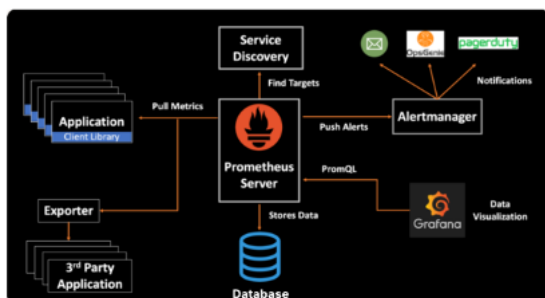


FIGURE 2. Prometheus monitoring stack

For most organizations, public and private cloud platforms such as Azure (Microsoft Azure Cloud Platform), AWS (Amazon web services), Google Cloud, SUSE Rancher are the default choice for their centralized workloads. It provides scalability, resilience, and vendor managed infrastructure offerings. These cloud providers out of the box provide Prometheus as a preferred solution for cloud-native metrics collection. For instance, Amazon Managed Service and Google Cloud Prometheus. Since the centralized workloads are in Prometheus based observability framework, to maintain observability consistency across both centralized and distributed environments, enterprises often extend Prometheus-based frameworks to edge locations.

Prometheus Solution for Edge: Challenges

Though Prometheus works well for central workloads, running it at the edge surface a couple critical roadblocks:

Figure 2. Shows the traditional Prometheus monitoring stack deployed at retail edge locations, highlighting critical resource limitations. The architecture requires approximately 3.5GB memory and consumes 18-25Mbps bandwidth just for metrics. It often reaches 35-50% of available network bandwidth (25-50Mbps total). Components include Prometheus server, Alert manager, Grafana, and various exporters, each contributing to the overall resource burden that makes this approach unsuitable for resource-constrained edge environments.

Resource Constraints The memory required to run Prometheus components (approx. 3.5GB) at edge nodes is way more than what could afford sometimes. The efforts to minimize footprint like excluding unwanted components such as Alert manager and Grafana did not sufficiently reduce utilization to an acceptable level.

Bandwidth Consumption The telemetry components consumed an average of 18–25Mbps (Megabits per second) without any enhancements. The total average upload bandwidth available across the distributed environments is anywhere between 25–50Mbps, shared with application traffic for some traditional retail stores. In a distributed environment, many of their edge locations would fall under the rural category where the available bandwidth is limited. This average includes their rural locations too. Optimization attempts like scraping fewer metrics, filtering labels, and compressing payloads have not brought the utilization under expected limits. Key application metrics remained utilizing higher network bandwidth due to Prometheus’s pull-

based, label-rich design.

OpenTelemetry for Metrics: Challenges and Implementation insights

As Prometheus based monitoring solution reached its practical limitations, the focus shifted towards leveraging open telemetry for metrics collection “see Figure 3.”.Although OTEL is theoretically referred to as a resource-efficient alternative with better integration options, not many enterprises practically implemented its end-to-end capability and tested it in the production environment. This deployment is one of the first of its kind.



FIGURE 3. Optimized observability stack

Optimized observability stack using OTEL components that achieve 63% memory reduction (1.3GB vs 3.5GB) and 96% bandwidth reduction (650Kbps vs 18Mbps) compared to Prometheus. The architecture includes OTEL Operator for management, OTEL Collector for data processing, and integration with centralized backends. Key optimization techniques include metric compression, selective collection (25 essential metrics), and efficient batching that make this solution viable for bandwidth-limited retail edge locations. The initial setup of deploying OTEL Operator is straightforward. However, bringing up the Collector may present a few major challenges.

TABLE 2. Comparison of Prometheus and OpenTelemetry Resource Usage

Solution	Memory Usage	Bandwidth Usage	Edge Suitability
Prometheus	3.5GB	18–25 Mbps	Limited
OpenTelemetry Improvement	1.3GB -63%	650 Kbps -96%	Optimized Viable

viable for edge environments and bandwidth-sensitive deployments.

Metric Compatibility Bug: A known bug with a specific metrics would affect during forwarding of the metrics to the time-series backend. For example, Google Managed Monitoring backend if used. To resolve this, the affected metrics can be explicitly excluded from processing. These metrics are reviewed for less value adding.

Data Type Mismatch: A conflict between integer and double data types for a subset of metrics can cause ingestion issues. This could be due to these metrics being already part of an existing monitored workload in the centralized backend. Since they are not critical for the use case, excluding them resolved the type mismatch. If any metrics are found to be useful, the data type can be modified to get forwarded. With these settings, the OTel collector starts running successfully. It also starts collecting, processing, and exporting the metrics to a centralized backend location for analyzing and alerting. The same time-series backend leveraged for Prometheus can be used for OTel as well. The resource utilization with the default settings is about 2.5GB of memory usage and 2Mbps of network bandwidth. This is way less compared to Prometheus usage. To optimize OTel resource requirements, several techniques can be implemented to enhance efficiency. Metrics compression to reduce payload size and collection configuration can be optimized to collect only the needed metrics. Example 25 needed ones, which significantly lowers the overhead. The final telemetry configuration provides a comprehensive visibility across the stack by incorporating cluster state metrics via kube-state-metrics, pod-level resource usage from cAdvisor, application-level metrics through Istio service mesh integration, and host metrics from node-exporter. This implementation configuration validates that the OTel can be leveraged for metrics collection at scale. It highlights the key considerations for backend compatibility and for less resource footprint. The solution now serves as a model for resilient, cost-efficient, less resource intense observability across edge environments.

With these optimizations in place, the memory utilization has decreased from about 2.5GB to 1.3GB, and network bandwidth utilization is down from 2Mbps to 650Kbps, making the solution

OTel for Traces: Implementation insight

OTel can be extended beyond metrics for observability. Tracing functionality can be introduced using OTel distributed tracing capabilities. As part of the setup, the OTel Collector needs to be configured with settings needed for trace ingestion, processing, and forwarding it to the centralized tracing backend such as Tempo. This includes defining the backend endpoint, queue size, and security settings. Application workloads need to be instrumented to emit traces at startup, using OTel libraries compatible with each service runtime. This instrumentation allows the propagation of trace context across service boundaries, enabling full visibility into request flow and latency bottlenecks. One of the key benefits observed during deployment was the minimal resource overhead associated with trace collection:

- Memory and central processing unit (CPU) impact were negligible across both the Collector and instrumented services.
- Bandwidth usage remained low due to efficient trace batching and compression, making the solution well-suited for edge and bandwidth-sensitive environments.

The successful implementation of trace telemetry added a valuable layer to system observability, enabling correlation between metrics and service-level interactions while maintaining an efficient operational footprint.

Log: A Critical Pillar of Edge Observability Logs are very critical for observability in general. They provide deep insight when it comes to identifying the root cause analysis of issues and triaging incidents. They provide granular visibility into system and application performance insight and error state details. Unless it is properly managed, logs can become very noisy and overwhelming. Especially in edge environments where the availability of resources is very limited.

Robust logging solution can be achieved by following certain standards, **Application Logging Standards** To ensure log clarity and solution reliability, it is essential to enforce certain logging standard practices:

- Logs written in structured formats like JavaScript

Object Notation (JSON) for consistent parsing and downstream processing.

- Include meaningful information like timestamp, EXIT codes, ERROR tags for proper aggregation, and filtering.
- Minimize emitting debug logs unless on time when needed.

These standards not only make logs machine-readable but also accelerate troubleshooting and alert generation. **Logging Architecture for Edge Environments** A lightweight and scalable architecture can be achieved through Fluentbit and Fluentd combination.

- Fluentbit runs as a sidecar container with application pods to collect the logs with minimal resource overhead.
- It forwards the logs to Fluentd, which in turn aggregates and processes the data before transmitting it to a centralized observability platform. Example, to Splunk via HEC (HTTP Event Collector) integration.
- This method decouples the collection from processing which enhances configuration and management of the solution.

Performance Optimization Techniques The volume of logs generated by the application pods can sometimes strain the edge computing unless managed. The following are a few key mitigation strategies to implement.

- Compression of logs at the source to significantly reduce bandwidth utilization.
- Filtering mechanisms such as collect logs based on namespaces, pod labels, severity levels, or specific message codes.
- Rate limiting and buffering to prevent log storms during failures or spikes.

By curating only essential logs, edge telemetry can remain both lightweight and highly actionable. The above solutions would provide a solid observability at the edge backend, but what about the front-end browser user interface, which directly tells the real user experience of the application. **Front-end Observability for Edge Applications** While robust observability at the backend and edge infrastructure is crucial, true end-to-end visibility is not complete unless front-end user interface is instrumented. Capturing the performance and user interaction telemetry data directly from the UI layer allows teams to correlate the performance metrics with real user experience and business outcomes. It is important in distributed retail environments.

OTel for Front-end User Interface

OTel can be instrumented at the web browser user interface to capture the following key performance metrics from the real user perspective,

- WebVital metrics like Largest Contentful Paint (LCP), Interaction to Next Paint (INP), and Cumulative Layout Shift (CLS).
- Traces and spans which includes Hypertext Transfer Protocol (HTTP) and Asynchronous JavaScript and XML (AJAX) requests, edge environment attributes, and session details.
- Resource loading times such as document upload and download time.

This data provides insight to identify issues that originate from the real user interaction point of view. The performance metrics such as slow rendering pages, poor network responses, or errors in client-side logic would be extracted. **Session-Level Logging and Correlation** In addition to user interface traces and metrics, logging solutions can be implemented at the UI layer, and the logs can be forwarded to centralized logging solutions such as Splunk through HEC method. These logs may include contexts such as:

- Session details, facility codes, and point-of-sale attributes.
- Trace and span identifiers to correlate with backend.

This contextual logging allows teams to pinpoint issues specific to regions or store locations. For example, to identify any latency issues in a particular geography or transaction failures related to specific device types. **End-to-End Tracing from User Interface (UI) to Backend** Using shared trace Identities (IDs), telemetry generated in the front-end can be seamlessly linked to backend traces and metrics. This facilitates:

- Transaction-level observability across distributed systems.
- Root cause analysis spanning both client and server sides.
- Behavior aggregation and funnel visualization in centralized dashboards.

Combined, these insights support powerful workflows like:

- Identifying degraded performance during specific user flows.
- Tracing customer drop-offs at checkout.
- Diagnosing edge-specific UI latency during peak traffic.

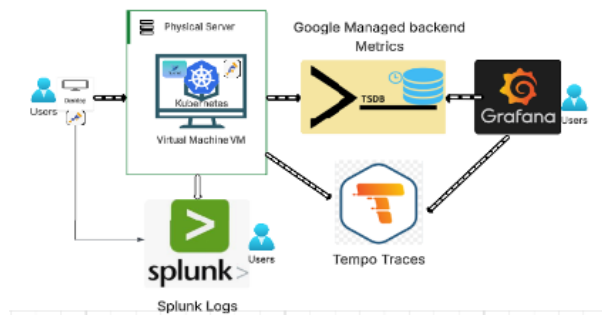


FIGURE 4. Integrated observability solution spanning from frontend user interfaces to backend infrastructure across retail edge environments

As seen in figure 4, the framework demonstrates correlation between Web Vitals metrics (LCP, INP, CLS), distributed traces, infrastructure metrics, and structured logs through shared trace identifiers. Components include browser-based OpenTelemetry instrumentation, lightweight Kubernetes (K3s/MicroK8s), OTEL collectors, Fluentbit/Fluentd logging pipeline, and centralized backends (Splunk, Tempo, TSDB). The solution enables transaction-level visibility across distributed retail systems while operating within edge resource constraints (CPU, memory, bandwidth).

Conclusion

This research demonstrates that enterprise-grade observability at the retail edge is both feasible and efficient using a lightweight framework combining K3s/MicroK8s, OpenTelemetry, and optimized logging. The solution achieves remarkable resource optimization with 63% memory reduction and 96% bandwidth reduction compared to Prometheus-based approaches, making comprehensive monitoring viable in bandwidth-constrained retail locations. Key findings reveal that edge environments need purpose-built solutions rather than adapted cloud tools, with OpenTelemetry significantly outperforming Prometheus in resource-limited scenarios. Strategic data collection focusing on essential metrics rather than comprehensive gathering proves more effective, while lightweight Kubernetes infrastructure provides built-in monitoring capabilities with reduced complexity. The framework successfully enables end-to-end transaction tracking from frontend to backend systems, demonstrating that comprehensive observability is achievable in edge environments with proper planning and providing a foundation for future cost-efficient monitoring solutions.

Lessons Learned: Network bandwidth was the main constraint, even more than CPU or memory. Always design for limited bandwidth first. Some parts are harder than others as while some components deployed easily, others (like collectors) had compatibility issues that took extra time to resolve. Structured logging pays off as using JSON format and consistent error codes from the beginning saves significant time later and reduces log volume by 60%. Plan integration early as successfully connecting frontend and backend monitoring requires architectural planning upfront.

Other Domains: The principles and solutions developed for retail edge environments can be adapted to other industries facing similar challenges such as healthcare, telecommunications, and energy and utilities. The framework applies to different sectors in various ways: in healthcare, it can monitor patient devices and vital signs remotely, maintain compliance audit trails, ensure system reliability for patient safety, and add security measures for medical data protection. For telecommunications, the framework can monitor 5G networks and service delivery, track call quality and data routing, and optimize for varying network conditions across coverage areas. In energy and utilities, it supports monitoring power generation, smart grids, renewable energy systems, tracking energy production and consumption patterns, and supporting regulatory compliance and environmental reporting. For organizations getting started, recommendations include modernizing infrastructure first by implementing lightweight Kubernetes before adding monitoring tools, starting small by beginning with essential metrics then adding tracing and correlation features, planning for limited bandwidth by designing all solutions with network constraints as the primary consideration, and establishing standards early by creating structured logging and data collection standards before building applications. For future development, organizations should add AI capabilities by using the structured data foundation to implement predictive analytics and automated problem detection, enhance security by integrating security monitoring and privacy protection as edge computing matures, and support multiple users by developing solutions for shared edge infrastructure serving multiple organizations.

References

- 1) Y. Liu et al., "An Overview on Edge Computing Research," *IEEE Access*, vol. 8, pp. 85714-85728, 2020.
- 2) . Papadopoulos et al., "Extending Kubernetes Clusters to Low-Resource Edge Devices Using

- Virtual Kubelets," IEEE Internet of Things Journal, vol. 8, no. 2, pp. 1234-1247, 2021.
- 3) T. Brown, "Resource Optimization in Edge Computing Systems," IEEE Transactions on Cloud Computing, vol. 15, no. 3, pp. 234-248, 2023.
 - 4) L. Martinez, "Memory-Efficient Monitoring Solutions for Edge Environments," ACM Computing Surveys, vol. 54, no. 2, pp. 1-28, 2022.
 - 5) K. Anderson, "Bandwidth Optimization Strategies for Edge Observability," IEEE Network, vol. 36, no. 4, pp. 45-52, 2022.
 - 6) S. Lee, "OpenTelemetry Performance Benchmarks in Edge Computing," IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 156-168, 2022.
 - 7) H. Nakamura, "End-to-End Observability in Distributed Retail Systems," IEEE Internet Computing, vol. 27, no. 3, pp. 78-86, 2023.
 - 8) R. Kumar et al., "Machine Learning-Based Resource Allocation for Edge Computing," IEEE Transactions on Mobile Computing, vol. 22, no. 7, pp. 4123-4136, 2023.
 - 9) J. Zhang et al., "Distributed Monitoring Architecture for IoT Edge Networks," ACM Transactions on Internet of Things, vol. 4, no. 1, pp. 1-24, 2023.
 - 10) P. Williams et al., "Container Orchestration Performance in Edge Environments," IEEE Communications Magazine, vol. 61, no. 5, pp. 88-94, 2023.
 - 11) C. Chen et al., "Adaptive Load Balancing for Edge-Cloud Hybrid Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 8, pp. 2245-2258, 2023.
 - 12) D. Rodriguez, "Privacy-Preserving Telemetry Collection at the Network Edge," ACM Computing Surveys, vol. 56, no. 3, pp. 1-31, 2023.
 - 13) F. Nguyen et al., "Real-Time Analytics for Edge Computing Applications," IEEE Internet of Things Journal, vol. 10, no. 12, pp. 10567-10579, 2023.
 - 14) M. Patel, "Energy-Aware Observability Framework for Edge Devices," IEEE Transactions on Green Communications and Networking, vol. 7, no. 2, pp. 456-468, 2023.
 - 15) G. Thompson et al., "Microservices Decomposition Strategies for Edge Deployment," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 4, pp. 1-29, 2023.
 - 16) A. Johnson, "Cross-Platform Monitoring Solutions for Heterogeneous Edge Infrastructure," IEEE Network, vol. 37, no. 3, pp. 112-119, 2023.
 - 17) K. Singh et al., "Fault Detection and Recovery Mechanisms in Edge Computing Clusters," IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 4, pp. 3012-3025, 2023.
 - 18) L. Wang, "Serverless Computing Observability at the Edge," ACM Transactions on Computer Systems, vol. 41, no. 2, pp. 1-26, 2023.
 - 19) B. Davis et al., "Network Latency Optimization for Edge-to-Cloud Data Pipelines," IEEE/ACM Transactions on Networking, vol. 31, no. 3, pp. 1234-1247, 2023.
 - 20) K. Rancher, "K3s - Lightweight Kubernetes," Rancher Labs, 2024.
 - 21) C. CNCF, "Rancher Manager Documentation," SUSE, 2024.
 - 22) "How to overcome the top 5 challenges of edge computing," ASB Resources, 2024.
 - 23) "Edge observability," SignOZ, 2024.
 - 24) VoltActiveData: Top 7 Edge Data Challenges in 2025.
 - 25) "Maximizing retail performance with data observability," Sifflet, 2024.
 - 26) "State of observability for retail," New Relic, 2024.
 - 27) "From insights to impact: the next evolution of observability in retail," ITR Portal, 2025.
 - 28) O. Petersen, "Resource Optimization Techniques for Telemetry Collection," IEEE Network, vol. 37, no. 6, pp. 78-85, 2023.

Prakash Velusamy is a Principal Software Development Engineer with over two decades of demonstrated excellence in the industry leading Infrastructure and Application Performance Monitoring platforms and strong Site Reliability Engineering (SRE). He is a technical leader in Observability, OpenTelemetry, Data Ingestion, Machine learning, Artificial Intelligence, and Automation. I led multiple key initiatives of mission critical flagship applications in terms of modernization, migration, optimization, innovation, consolidation, integration, observability, stability, resilience and performance.

Accelerating Kubernetes for AI/ML and HPC Workloads with SmartNIC-Based Networking and RDMA Offload

Sujithra Periasamy, *IEEE Senior Member, USA*

*Abstract—Kubernetes has evolved from orchestrating microservices to hosting performance critical workloads such as artificial intelligence (AI) training, high performance computing (HPC), and large scale data analytics. These applications demand ultra low latency, high throughput networking, and Remote Direct Memory Access (RDMA) capabilities. However, Kubernetes' default networking stack, optimized for portability and flexibility, imposes significant CPU overhead and adds latency. Smart Network Interface Cards (SmartNICs) offer a promising path forward by offloading datapath processing, packet classification, encryption, and RDMA operations directly onto the NIC hardware. In bare-metal Kubernetes clusters, this approach enables wire-speed performance, consistent pod-to-pod RDMA communication, and reduced CPU utilization. This paper presents a theoretical system design for integrating SmartNIC-based offloads into Kubernetes bare-metal cluster, detailing architectural considerations, traffic steering strategies, and Kubernetes-native integration points. This work highlights how SmartNICs can bridge the gap between Kubernetes flexibility and the demands of performance-critical cloud-native workloads. **Keywords:** Cloud Computing, Kubernetes, SmartNIC, RDMA*

Kubernetes has become the dominant platform for container orchestration, widely adopted in both enterprise and research environments. Its strengths lie in portability, automation, and flexibility, but these advantages come at a cost when workloads demand extreme performance. As organizations increasingly rely on Kubernetes for artificial intelligence training, high-performance computing (HPC), and large-scale data analytics, the efficiency of the networking layer becomes a decisive factor in overall system performance. In bare-metal deployments, where workloads interact directly with host resources, any networking overhead directly translates into reduced application throughput and increased latency. The default Kubernetes networking stack introduces multiple processing layers that, while useful for abstraction and portability, limit the efficiency required for performance-sensitive applications.

Smart Network Interface Cards (SmartNICs) offer

a promising path forward by offloading packet classification, encryption, flow steering, and RDMA queue management to programmable hardware. This shifts performance-critical tasks away from the host CPU while still enabling Kubernetes to manage pod lifecycle events and enforce policies. In this paper, we present a design for integrating SmartNIC-based offload into bare-metal Kubernetes clusters, focusing on RDMA-enabled workloads where both performance and policy fidelity are critical. We describe architectural choices, propose a SmartNIC Agent for lifecycle and policy reconciliation, and outline how Kubernetes Device Plugins and CNIs can orchestrate these capabilities. Rather than presenting experimental results, we focus on feasibility, trade-offs, and an evaluation plan to guide future implementation and validation.

BACKGROUND AND RELATED WORK

Kubernetes Networking Basics: Kubernetes adopts a simple but flexible networking model: every pod receives its own IP address, and all pods within a cluster

are expected to communicate without NAT. This model is implemented by Container Network Interface (CNI) plugins such as Calico, Flannel, and Cilium. Under the hood, packet transmission typically traverses veth pairs, Linux bridges, and often overlay tunnels. While this architecture provides portability and multi-tenant abstraction, it introduces multiple context switches, buffer copies, and additional CPU overhead. These inefficiencies are manageable for microservices, but become critical bottlenecks for latency-sensitive applications.

Remote Direct Memory Access (RDMA): RDMA enables one host to read or write directly to the memory of another without CPU intervention, bypassing the kernel for low-latency, high-bandwidth communication. It is widely adopted in HPC and AI training workloads where microsecond-scale latency matters. RDMA relies on queue pairs (QPs), completion queues, and memory registration through verbs libraries. Although Kubernetes can expose RDMA devices to pods using plugins, mapping RDMA semantics cleanly onto Kubernetes abstractions remains challenging, particularly when lifecycle management and policy enforcement are required.

SmartNIC Architectures: SmartNICs augment traditional NICs with programmable pipelines and embedded compute cores. They can classify packets, apply Access Control Lists (ACLs), perform encryption/decryption, and manage RDMA QPs directly on the card. Architectures vary: some expose general purpose ARM cores running Linux, while others provide match-action pipelines programmable via SDKs. Vendors such as NVIDIA (BlueField), Intel (IPU), and Broadcom (Stingray) have demonstrated SmartNICs that accelerate virtual switching, storage access (NVMe-oF), and security services in cloud deployments.

Prior Approaches in Kubernetes and Data Centers: Several strategies have been explored to bridge the gap between Kubernetes abstractions and performance requirements:

SR-IOV in Kubernetes: Provides near-native networking by assigning NIC Virtual Functions (VFs) to pods. However, static allocation of VFs reduces scheduling flexibility and complicates pod lifecycle management.

Kernel-bypass frameworks: DPDK and RDMA verbs are widely used in financial and HPC trading systems, enabling low-latency networking. Their adoption in Kubernetes is limited because they require application rewrites and bypass NetworkPolicy enforcement.

eBPF and XDP: Recent software-only accelerations allow packet processing in the kernel at near-line rate speeds. Although promising, these approaches remain

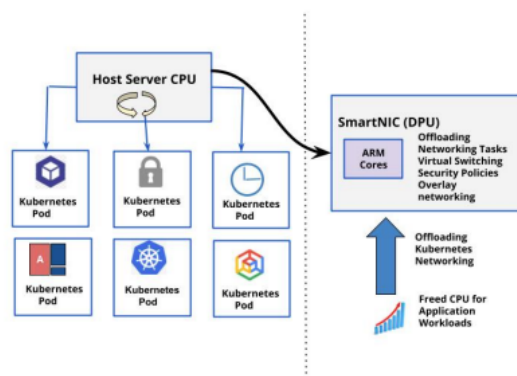


FIGURE 1. Kubernetes Node with SmartNIC for datapath offloading

CPU-bound and cannot match hardware offload.

Summary: In summary, existing approaches demonstrate partial progress toward bridging Kubernetes networking with high-performance demands. SmartNICs offer a promising bridge between raw performance and native Kubernetes integration. By offloading packet classification, ACL enforcement, QoS shaping, and telemetry onto NIC hardware, SmartNICs can reduce host CPU load while restoring policy enforcement for SR-IOV or RDMA workloads. Figure 1 clearly illustrates this shift. Table 1 compares different approaches to Kubernetes networking, highlighting their trade-offs in terms of latency, CPU utilization, policy enforcement, observability, and portability.

TABLE 1. Comparison of Networking Approaches

Approach	Latency	CPU Usage	Policy	Observability	Portability
Default CNI	Poor	High	Strong	Strong	Excellent
SR-IOV	Good	Low	Weak	Weak	Limited
DPDK / RDMA Verbs	Excellent	Very Low	None	None	Poor
eBPF / XDP	Good	Medium	Strong	Medium	Excellent
SmartNIC Offload	Excellent	Very Low	Strong (HW)	Strong (HW)	Good

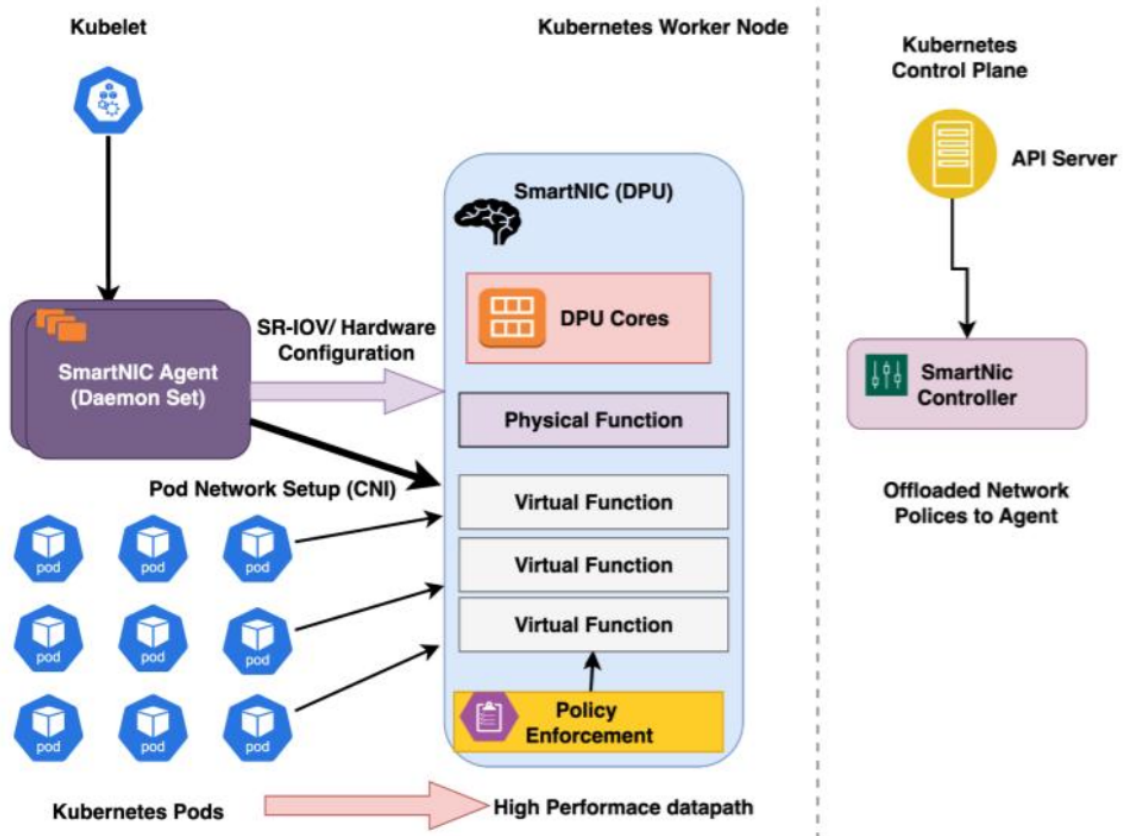


FIGURE 2. SmartNIC Agent Kubernetes Integration

SYSTEM DESIGN

Our system design integrates SmartNIC hardware offload into bare-metal Kubernetes clusters to support both high-performance networking and RDMA workloads, with the SmartNIC Agent acting as the bridge between Kubernetes and the SmartNIC dataplane. While SR-IOV provides the mechanism to expose Virtual Functions (VFs) directly to pods, it is the Agent that transforms this raw capability into a Kubernetes-native service by translating control-plane events into hardware state. In this section, we describe the key components of the design, the responsibilities of the SmartNIC Agent, and the resulting behavior of both networking and RDMA traffic.

Role of the SmartNIC Agent: At the heart of the design is the SmartNIC Agent, a daemonset that runs along with the kubelet on each node. Its primary function is to continuously reconcile Kubernetes intent with SmartNIC configuration. When pods are created, deleted, or migrated, the Agent detects these events and programs the NIC accordingly. This includes attaching VFs to pods, provisioning access control lists (ACLs), enforcing QoS policies, and cleaning up hardware state when pods exit. Unlike vanilla SR-IOV, which leaves VF lifecycle management largely static, the SmartNIC Agent introduces dynamic reconciliation, ensuring that hardware configuration always mirrors the logical state of the cluster. The Agent also interacts with the Kubernetes API server to watch for NetworkPolicy updates. These policies, which in the default Kubernetes implementation are enforced via iptables or eBPF in the kernel, are instead compiled into SmartNIC-compatible ACL rules. These rules are installed directly into the NIC's flow tables, ensuring that all pod traffic, whether IP or RDMA passes through a hardware-enforced policy barrier before leaving or entering a VF. In this way, the Agent makes SmartNICs a first-class enforcement point in the Kubernetes networking model. Figure 2 illustrates SmartNIC Agent Kubernetes integration.

Pod Lifecycle Integration: When a pod requesting networking or RDMA resources is scheduled, the SmartNIC Agent collaborates with the SR-IOV Device Plugin to bind a VF to the pod's namespace. This VF serves as the pod's network interface, providing near-native access to the NIC. However, unlike vanilla SR-IOV, where the VF is assigned statically and policies are bypassed, the Agent ensures that as soon as the VF is attached, it is also configured with ACLs, QoS rules, and telemetry hooks. If the pod is later rescheduled to a different node, the VF context and rules are torn down and reprovisioned automatically. This dynamic reconciliation ensures that pod churn,

a common event in Kubernetes does not result in stale NIC configuration or policy gaps. Lifecycle integration also extends to RDMA-specific contexts. When a RDMA capable pod is created, the SmartNIC allocates resources such as queue pairs (QPs) and completion queues, though their lifecycle remains under application control through verbs. The Agent's role is to partition these resources fairly between pods and enforce per-VF limits, preventing a single workload from monopolizing the NIC hardware. When the pod exits, the Agent ensures that QPs and VF contexts are reclaimed, avoiding resource leaks and preserving isolation.

Networking Offload: Networking offload in this design addresses the performance limitations of the default Kubernetes CNI stack. Instead of sending pod traffic through veth pairs, Linux bridges, and iptables chains, the VF traffic is delivered directly to the SmartNIC dataplane. Here, the NIC performs packet classification, applies ACLs derived from Network Policies, enforces QoS shaping, and optionally offloads service steering (e.g., translating ClusterIP and endpoint rules). By shifting these functions from the CPU to hardware, the system reduces host CPU consumption and lowers per-packet latency. This offload is particularly valuable for the data ingestion and service components of AI/ML workloads. Training jobs rely on massive dataset transfers from storage systems, often using TCP-based protocols, while inference pipelines depend on low-latency microservices exchanges. Both benefit from SmartNIC based classification and steering, which ensure predictable performance and prevent noisy neighbors from overwhelming shared resources.

RDMA Offload: While networking offload accelerates general pod communication, RDMA offload is crucial for distributed training workloads. In our design, pods requiring RDMA are granted VFs that expose RDMA capabilities directly. Applications continue to manage QPs via verbs, posting work requests to send and receive data. The SmartNIC hardware executes these operations, handling DMA transfers, completion notifications, and congestion control mechanisms at wire speed. The SmartNIC agent ensures that even though the applications manage QPs, traffic enforcement remains in place. Per-VF ACLs derived from Network Policies are applied at ingress and egress, QoS rules prevent unfair bandwidth allocation, and telemetry counters expose RDMA specific statistics back to Kubernetes. This allows operators to observe RDMA traffic as part of the cluster monitoring framework, bridging a longstanding gap between high-performance computing models and cloud-native observability.

Policy Compilation and Enforcement: One of

the most critical functions of the SmartNIC agent is the compilation of policies. In the default Kubernetes model, Network Policies are enforced by iptables or eBPF programs that filter packets as they traverse the kernel. In our design, the Agent compiles these policies into SmartNIC specific rules that are installed in the NIC's flow tables. This translation includes mapping label-based selectors into IP and VF tuples, as well as enforcing both ingress and egress rules. The result is that SmartNICs become the point of truth for isolation, ensuring that even bypass traffic, such as RDMA, cannot escape policy enforcement. Figure 3 illustrates various offload functionalities configured by the SmartNIC Agent.

QoS and Traffic Shaping: The SmartNIC Agent configures QoS policies to prevent unfair resource usage. Per-VF rate limiting ensures that no single pod can saturate NIC bandwidth, while hierarchical QoS scheduling provides guarantees for latency-sensitive workloads. This is especially important in multitenant AI clusters, where large training jobs coexist with latency-sensitive inference or service traffic. By offloading QoS enforcement to the NIC, the system achieves fairness without consuming host CPU cycles.

Telemetry and Observability: Finally, the SmartNIC Agent integrates NIC telemetry into the Kubernetes monitoring stack. Each VF exposes counters for throughput, dropped packets, latency distributions, and RDMA specific statistics such as QP activity and completion rates. The Agent collects these metrics and exports them into standard cluster observability systems (Prometheus). This unified telemetry provides operators with visibility across both networking and RDMA flows, enabling proactive management of performance bottlenecks and enforcement of service-level objectives.

Security Considerations: The security of the system depends not only on policy enforcement, but also on the trustworthiness of the SmartNIC itself. The Agent ensures that pods cannot bypass SmartNIC enforcement by directly reconfiguring their VFs, and future deployments may include attestation of NIC firmware to guarantee that ACLs and QoS rules cannot be tampered with. By shifting enforcement from the host kernel to dedicated NIC hardware, the system reduces the attack surface while aligning with the Kubernetes declarative policy model.

Failure and Fallback Behavior: While SmartNIC offload enhances performance and policy enforcement, production deployments must account for both planned maintenance and unplanned failures to ensure service continuity. Table 2 covers the various failure scenarios and the fallback behavior for the same. In all cases,

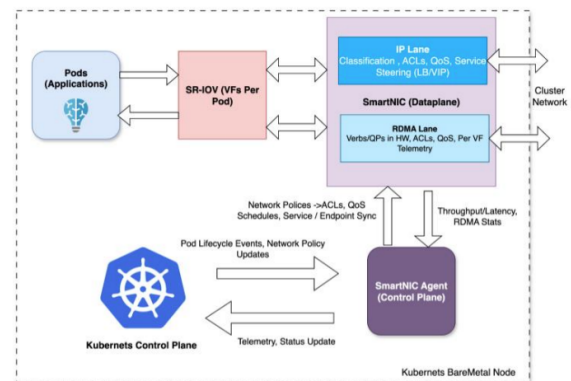


FIGURE 3. Dual-lane SmartNIC offload. The IP lane offloads classification, ACLs, QoS, and service steering; the RDMA lane executes verbs/QPs in hardware with per-VF ACLs and QoS. The SmartNIC Agent compiles Kubernetes intent into NIC rules and exports unified telemetry.

the fallback design favors security and correctness over performance. Traffic that cannot be classified or enforced in hardware is redirected to the software dataplane, ensuring that no packets bypass declared Kubernetes policies.

REPRODUCIBLE EVALUATION PLAN AND EXPECTED OUTCOME

As a theoretical system design, this paper focuses on the architectural principles and control plane mechanisms required to accelerate Kubernetes for AI/ML and HPC workloads using SmartNICs. While a full-scale empirical evaluation is beyond the scope of this initial design exposition, this section outlines a comprehensive, reproducible plan for both a minimal Proof-of-Concept (POC) implementation and subsequent performance benchmarking. This plan serves as a direct blueprint for future work, demonstrating the feasibility of the proposed SmartNIC Agent architecture and validating its performance claims.

Proof-of-Concept (POC) Implementation – Demonstrating Feasibility: The first critical step involves implementing the core control loop of the SmartNIC Agent to demonstrate the dynamic provisioning of network resources (both IP and RDMA) and the compilation of Kubernetes Network Policies into SmartNIC hardware rules. **Environment Setup:** Deploy a minimum of two bare-metal Kubernetes worker nodes equipped with programmable SmartNICs

TABLE 2. Failure and Fallback Behavior Across Scenarios

Scenario	Nature of Event	Detection / Trigger	Fallback Action	Recovery Path
Planned Firmware Upgrade	Operator-initiated, maintenance	Admin cordon + drain	Pods gracefully migrated	Node brought back online after upgrade, Agent reinstalls policies
Unplanned NIC Reset / Failure	Abrupt, affects entire NIC datapath	Node heartbeat loss, marked NotReady	Pods rescheduled by controllers (Deployment / Job / StatefulSet)	Agent reconciles state once NIC is back, but evicted pods remain on new nodes
Transient VF Reset	Localized, only specific VF	Agent detects VF unavailability	VF rebound to pod, policies reinstalled	Pod resumes with minimal disruption
Resource Exhaustion	Rule / QP / ACL limits exceeded	Agent tracks hardware usage	Configurable: (A) Software fallback (host stack for networking + RDMA), (B) Admission control (block new pods)	Existing pods unaffected; operator decides policy
Agent Failure	SmartNIC Agent crash or Kubernetes liveness probes	Existing hardware rules remain; new pods use SR-IOV	Agent restart reconciles cluster state with NIC	Cluster resumes normal operation

(e.g., NVIDIA BlueField-2 or similar DPU with SR-IOV support). Configure the NIC to expose multiple Virtual Functions (VFs) and integrate with the SR-IOV Device Plugin for Kubernetes. Ensure necessary kernel modules are loaded.

Expected Outcome: A functional Kubernetes cluster, with the SR-IOV Device Plugin reporting available SmartNIC VFs for allocation. Nodes are ready for high-performance pod scheduling.

SmartNIC Agent Deployment: Implement and deploy the core SmartNIC Agent as a Kubernetes DaemonSet. The Agent must include the following foundational logic:

- Integration with the Kubernetes API client to watch for Pod creation/deletion events and Network Policy updates.
- Logic to interact with the SR-IOV Device Plugin to request and manage VFs for pods.
- A basic interface (e.g., using a vendor SDK) to program simple flow rules onto the SmartNIC's hardware acceleration engines.

Expected Outcome: The SmartNIC Agent successfully connects to the Kubernetes API server, identifies available SmartNIC resources on its node, and logs active watches for relevant events. The DaemonSet status indicates healthy operation.

Networking VF Provisioning: Demonstrate the SmartNIC Agent successfully provisioning a dedicated Networking VF (for IP traffic) to a pod. The Agent should ensure the VF is exposed within the pod with

a valid IP configuration and that basic network connectivity (e.g., ping, curl) can be established to other pods.

Expected Outcome: A pod is successfully scheduled with a dedicated Networking VF. Inside the pod, standard IP networking utilities (e.g., ip addr, ping) confirm network interface activation and IP connectivity to other pods on the cluster, bypassing the host network stack.

RDMA VF Provisioning: Demonstrate the SmartNIC Agent provisioning a dedicated RDMA-enabled VF to a pod. The Agent should ensure the VF's RDMA capabilities are exposed within the pod and that basic RDMA primitives can be used by an application within that pod.

Expected Outcome: A pod is successfully scheduled with an RDMA-enabled VF. Inside the pod, standard RDMA utilities can detect and interact with the VF's RDMA interface, confirming hardware passthrough and readiness for RDMA applications.

Minimal Policy Translation and Enforcement: Implement the core policy compilation logic within the Agent for a single, illustrative Kubernetes Network Policy.

Expected Outcome: The Agent successfully translates the Kubernetes policy, resolves associated pod IPs/VFs, and confirms successful programming of the corresponding ACL rule(s) onto the SmartNIC hardware. This validates the fundamental control plane functionality.

Benchmark Validation – Quantifying Performance:

Following the successful implementation of the POC, a series of benchmarks will quantitatively validate the performance benefits and policy enforcement capabilities of the SmartNIC-accelerated Kubernetes environment against a default CNI (e.g., Calico or Cilium in IPVS/eBPF mode).

TCP/UDP Networking Performance: Quantify IP forwarding latency, throughput, and host CPU utilization for standard network traffic. Deploy two pods, each with a SmartNIC VF, on different nodes. Use iperf3 or Netperf to measure bandwidth and request-response latency. Collect host CPU metrics using mpstat or node-exporter.

Expected Outcome: Significantly lower latency and higher throughput for data traffic offloaded to the SmartNIC, with a substantial reduction in host CPU overhead (e.g., >50

RDMA Performance: Measure RDMA one-way latency, bandwidth, and CPU overhead for memory-intensive and latency-sensitive workloads. Deploy two pods, each provisioned with an RDMA-enabled SmartNIC VF, on different nodes. Use perftest suite to measure raw RDMA performance. Integrate with AI/ML training frameworks (e.g., Horovod with NCCL) running synthetic benchmarks to measure distributed training step time.

Expected Outcome: Near bare-metal RDMA performance (e.g., <1 μ s latency, >100Gbps bandwidth) with minimal host CPU involvement, directly enabling high-performance distributed AI/ML and HPC.

Network Policy Enforcement Fidelity: Verify that Kubernetes Network Policies are correctly and efficiently enforced in hardware, with negligible overhead. Deploy three pods (e.g., client, allowed-server, denied-server) and apply Network Policies such that client can reach allowed-server but not denied-server. Attempt connections (e.g., curl or netcat) from client to both servers. Measure latency of allowed connections.

Expected Outcome: Connections to denied-server must be blocked at the SmartNIC hardware level (verified via SmartNIC logs/telemetry), with no measurable performance degradation for allowed connections compared to an un-policed baseline on the SmartNIC. Policy decisions should not incur host CPU overhead.

Telemetry and Observability: Validate the SmartNIC Agent's ability to collect and export fine-grained telemetry for both IP and RDMA traffic from the SmartNIC hardware. Integrate the SmartNIC Agent with Prometheus. Generate various traffic patterns (IP, RDMA, denied policies). Query Prometheus to verify the presence of detailed metrics, including per-VF/per-QP statistics, byte/packet counters, and policy drop

counts. **Expected Outcome:** Comprehensive telemetry from the SmartNIC (e.g., accurate packet counts, latency stats, policy hit/misses, RDMA QP states) is successfully exported and observable via standard Kubernetes monitoring tools, bridging the observability gap for offloaded traffic.

Resilience and Fallback: Test the system's behavior under various failure conditions, verifying the reconciliation and fallback mechanisms. Simulate NIC resets, Agent restarts, and dynamic policy updates. Verify that the SmartNIC Agent successfully reconfigures the hardware to the desired state and that pods remain functional (potentially with degraded performance in fallback scenarios, as described in System Resilience and Fallback).

Expected Outcome: The system demonstrates automated recovery and state consistency after disruptions. Manual intervention should be minimized.

Summary: This detailed evaluation plan, also summarized in Table 3, provides a clear, reproducible pathway to empirically validate the significant performance, policy, and observability benefits of the proposed SmartNIC-accelerated Kubernetes architecture. The benchmark results outlined in this section are expected to empirically validate the conceptual benefits illustrated in Figure 4.

LIMITATIONS AND OPEN PROBLEMS

As with any architectural exposition, the SmartNIC-based Kubernetes design discussed in this paper comes with limitations and open problems that must be addressed in future work.

Lack of Empirical Validation: This work primarily focuses on design aspects and architectural principles of the SmartNIC Agent without empirical evaluation. While a detailed Reproducible Evaluation Plan provides a blueprint for a Proof-of-Concept and subsequent performance benchmarking, the actual implementation and collection of performance data (e.g., latency, throughput, CPU overhead) remain open tasks. Current claims of performance improvement are based on theoretical analysis and extrapolation from existing SmartNIC capabilities.

Portability Across Vendors: SmartNIC hardware varies widely in programmability and feature sets. While some devices provide rich match-action pipelines, RDMA primitives, and P4-programmable datapaths, others expose only limited ACL or QoS capabilities. Achieving consistent behavior across diverse hardware is an unsolved problem. Industry efforts such as the Open Programmable Infrastructure (OPI)

TABLE 3. Summary of Evaluation Plan

Dimension	Methodology	Tools / Workloads	Metrics	Expected Outcome
Networking performance	Pod-to-pod throughput and latency across nodes	iperf3, netperf	Gbps, P50/P90/P99 latency	20–40% lower latency vs. overlay; near line-rate throughput
RDMA performance	Verbs latency/bandwidth tests	IB Verbs app, OSU benchmarks	One-way latency, Gbps	Latency close to bare-metal; throughput at NIC line rate
Distributed training	NCCL/Horovod runs across 2–8 nodes	nccl-tests, Horovod synthetic data	Iteration time, all-reduce time	Faster convergence due to reduced communication overhead
CPU efficiency	Measure CPU usage during traffic	Prometheus node exporter, pidstat	% CPU per Gbps	60–70% lower CPU vs. kernel enforcement
Policy overhead	Increase pods per node until saturation	SmartNIC Agent logs, Prometheus	Rule install latency, dataplane impact	Negligible overhead; rules enforced in hardware at line rate
Scalability	Install 100–1000 rules, measure setup	Kubernetes scheduler, NIC counters	Max pods with full policy coverage	Bounded by ACL/QP limits; fallback engaged
Failure handling	Induce NIC reset, VF reset, Agent crash, rule exhaustion	PCI reset scripts, VF unbind/rebind, kill Agent	Recovery time, packet loss, rescheduling time	VF recovery in seconds; NIC reset → recovery via NotReady detection; fallback modes engaged

project and P4 Runtime offer promising avenues toward vendor-neutral abstractions, but their integration into Kubernetes ecosystems remains early-stage.

Observability and Debugging: Offloading datapath logic to SmartNICs reduces CPU load but also hides traffic from traditional Linux tools (tcpdump, iptables, conntrack). Building robust observability—per-pod counters, policy hit/miss statistics, and failure tracing—is an open challenge and essential for operator trust.

Multi-Tenant Security and Fairness: In multi-tenant clusters, SmartNICs must enforce isolation not just at packet level but also across QoS and bandwidth allocation. Ensuring fairness under heavy load while preventing “noisy neighbor” effects requires careful policy enforcement on NICs, which is still an active research area.

Complexity of Policy Compilation for Advanced Scenarios: Translating Kubernetes Network Policies, which are label-based and declarative, into low-level, hardware-specific flow rules for the SmartNIC is a non-trivial task. Advanced Kubernetes networking features (e.g., complex CIDR matching, named ports, egress rules, network policy logging, and especially dynamic

updates to these rules at line rate) can significantly increase the complexity of the SmartNIC Agent’s compilation engine and the required hardware capabilities. Efficiently handling rule updates and ensuring atomicity of rule programming in the hardware are crucial for production environments.

Security Implications of Hardware Offload: Offloading critical networking functions to the SmartNIC introduces a new attack surface. The design must rigorously address potential vulnerabilities such as malicious firmware, unauthorized rule injection, or data leakage through side channels in the offloaded path. Detailed security analysis and hardening strategies are areas for continued investigation.

CONCLUSION AND FUTURE WORK

This work outlined a theoretical design for the SmartNIC Agent, enabling Kubernetes to offload IP and RDMA datapath functions to programmable SmartNICs. The approach promises lower CPU utilization, near line-rate performance, and hardware-enforced policy fidelity, with a reproducible evaluation plan offered as a pathway for empirical validation.

Future Work: Future work will focus on implement-

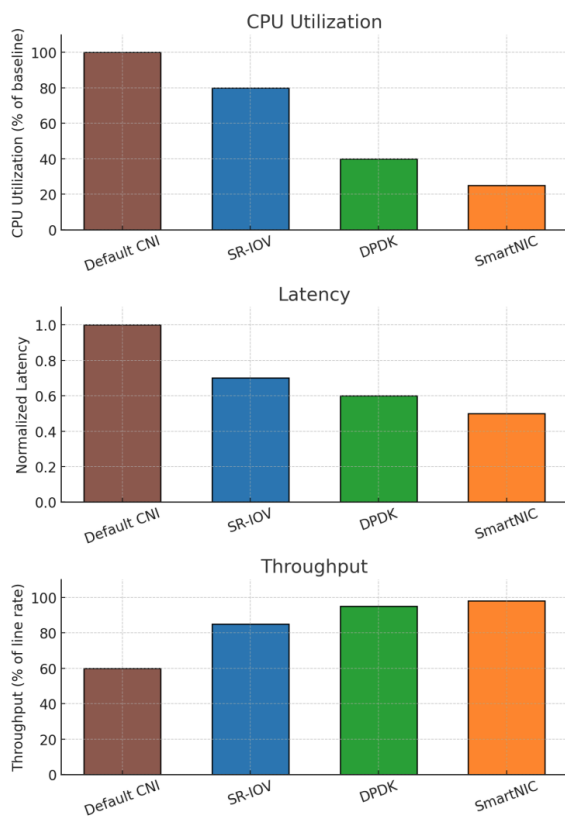


FIGURE 4. Expected outcomes across networking models Default CNI: High CPU cost, poor throughput/latency, but strong policy enforcement. SR-IOV: Near-native performance, but static allocation and no policy isolation. DPDK/RDMA: Excellent raw performance, requires app rewrites, no Kubernetes enforcement or observability. SmartNIC Offload: Matches line-rate performance, reduces CPU load, and unifies telemetry

ing the SmartNIC Agent prototype and validating it with end-to-end benchmarks on AI/ML and HPC workloads. Beyond baseline performance, efforts will target advanced policy compilation, stronger security and attestation mechanisms, improved observability tooling, and portable abstractions across vendors. Extending this design to additional domains such as 5G edge, financial trading, and real-time analytics represents another promising direction.

REFERENCES

- 1) B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, Omega, and Kubernetes,” *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
- 2) Y. Dong, X. Yang, X. Li, J. Li, K. Tian, and H.

- Guan, “High Performance Network Virtualization with SR-IOV,” in *Proc. IEEE Int’l Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW)*, Apr. 2010, pp. 1741–1748.
- 3) Intel Corporation, “Single Root I/O Virtualization (SR-IOV) Technical Overview,” White Paper, 2011. [Online]. Available: [Intel Single Root I/O Virtualization \(SR-IOV\) Technical Overview](#)
- 4) H. K. Jerry Chu et al., “DPDK: Data Plane Development Kit,” [Online]. Available: <https://www.dpdk.org>
- 5) OpenFabrics Alliance, “RDMA Programming User Manual,” 2020. [Online]. Available: <https://www.openfabrics.org>
- 6) M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, “Managing Data Transfers in Computer Clusters with Orchestra,” *ACM SIGCOMM*, pp. 98–109, 2011.
- 7) H. Li, R. Agarwal, M. Ahmed, and V. Sekar, “Leveraging SmartNICs for Flexible and Scalable Cloud Networking,” *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 405–418, 2019.
- 8) Mellanox Technologies (NVIDIA Networking), “NVIDIA ConnectX SmartNICs: Architecture and Benefits,” White Paper, 2021. [Online]. Available: <https://network.nvidia.com>
- 9) S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, “Network Functions Virtualization with ClickOS,” *ACM SIGCOMM*, pp. 459–472, 2015.
- 10) V. Pandey, P. Rathi, and D. Bansal, “Offloading Network Functions to SmartNICs in the Data-center,” *IEEE Micro*, vol. 41, no. 4, pp. 64–72, Jul.–Aug. 2021.
- 11) P. Bosshart et al., “P4: Programming Protocol-Independent Packet Processors,” *ACM SIGCOMM CCR*, 2014.
- 12) Z. István, V. Gramoli, “Trusted Execution and Isolation on SmartNICs,” *ACM SoCC*, 2020.
- 13) Y. Wang et al., “RDMA over Kubernetes: Toward High-Performance Container Networking,” *IEEE/ACM Transactions on Networking*, 2022.
- 14) M. Ghalim et al., “Kubernetes SR-IOV Network Device Plugin,” CNCF, 2019. Available: <https://github.com/k8snetworkplg/sriov-network-device-plugin>
- 15) Open Compute Project, “Open Programmable Infrastructure (OPI) Specification,” OCP Foundation, 2022. Available: <https://opiproject.org/>

Sujithra Periasamy is a software engineer special-

izing in cloud computing, networking, and distributed systems. She has extensive experience in designing high-performance network architectures, secure cloud infrastructure, and SmartNIC-based acceleration frameworks. Her professional and research interests include Kubernetes, RDMA, software-defined networking, SmartNIC offloads, and secure distributed systems.

Thoughtful AI: Designing an AI Recruiting Agent That Thinks, Explains, and Reconsiders

Arjun Singh, Senior Data Scientist, Amazon.com Inc, Seattle, WA, USA

Abstract—This paper presents the design, implementation, and evaluation of a thoughtful AI recruiting agent built to address the unique challenges of applying AI to human resources. We introduce a modular LangChain-based architecture that combines technical skills assessment with cultural fit analysis, while incorporating mechanisms to combat AI hallucination and reduce bias. We address this through the implementation of a hallucination checker with self-correcting strategies, and a bias auditing chain to surface fairness concerns. This research contributes to the ongoing discussion of responsible AI implementation in HR processes, emphasizing the importance of explainable, verifiable, and carefully considered AI solutions in recruitment.

Keywords: Artificial Intelligence, Human Resources, Recruitment, LangChain, Hallucination Detection, Bias Mitigation, Scalability

The current age of Generative AI has seen nearly every business function go through a rapid transformation, but few as delicately as Human Resources. Nowhere else is the margin for error so personal, the stakes so human. Hiring isn't just about matching resumes to roles. It's about evaluating potential, values, and alignment. It is here, at this intersection of people and technology, that we need to pause, think more slowly, and build with care.

The promise of AI in HR is real: automate repetitive screening tasks, surface high-fit candidates faster, and support diversity and inclusion by anonymizing profiles. But beneath the promise lies risk, particularly the risk of hallucination, opacity, and unintended bias. When an AI assistant invents experience a candidate doesn't have, or misrepresents a company's values, it doesn't just make a mistake—it breaks trust.

In this paper, we design, build and experiment with an AI recruiting assistant to navigate those risks and what we learn when the system works and when it doesn't.

RETHINKING AI FOR HIRING

Most modern hiring tools today are designed for throughput. Parse more resumes. Speed up funnel

velocity. Reduce cost per hire. But that mechanical view of hiring ignores the complexity and nuance of what makes someone the right fit, not just on paper, but in context. We ask a different question: What if an AI system could think more like a recruiter? What if it could evaluate both skills and cultural alignment? What if it could explain its recommendations, surface edge cases, and most importantly recognize when it's not sure? The goal here is not to replace human judgment, but to support it, with transparency, auditability, and care.

ARCHITECTING THE ASSISTANT

The AI recruiting assistant is built using a modular LangChain-based architecture, integrated with Groq-hosted LLMs and open-source embedding models. It consists of five layers detailed ahead. To truly understand this architecture, it's helpful to trace how information flows through the system for different use cases.

For Candidate Analysis: Recruiters first upload company culture documents and resumes, which get processed, chunked, embedded, and stored in their respective vector databases. When analyzing candidates for a specific job, the system searches both databases to find relevant information, then uses the LLM to perform comprehensive evaluation combining technical skills assessment with cultural fit analysis.

For Cold Email Generation: A recruiter uploads a

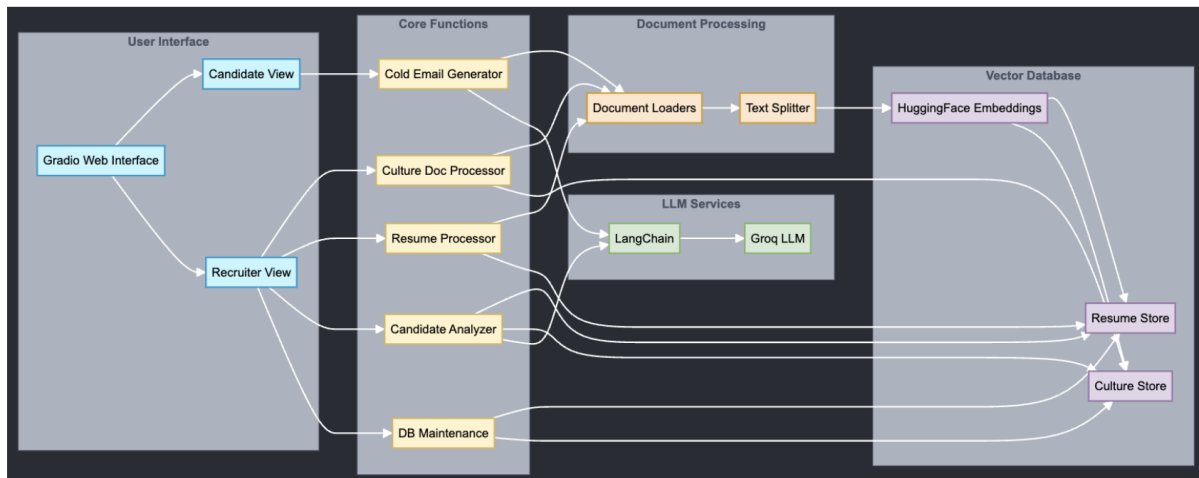


FIGURE 1. Mermaid diagram showcasing the architecture for the AI Recruiting Agent

selected candidate’s resume, which gets processed by the document loaders and immediately analyzed by the LLM chains against the job description. The AI generates a personalized email by identifying the strongest connections between the candidate’s background and job requirements.

Layer 1: Web Interface

Gradio User Interface: The user interaction begins with Gradio, a Python library that creates an accessible web interface. This choice eliminates the need for extensive front-end development while providing an intuitive user experience. The interface is designed around two distinct use cases, each with different needs and workflows. This separation is crucial because it prevents feature bloat and ensures each use case can focus on its specific tasks without distraction.

Candidate View: This interface serves recruiters that want to generate cold emails for selected candidates. Recruiters upload a candidate’s resume and paste a job description, then receive a tailored email that highlights the candidate’s most relevant qualifications for that specific position. From an architectural perspective, this represents a "stateless operation" each request is independent and doesn’t require storing information between interactions. This design choice makes the system more reliable and easier to scale.

Recruiter View: The recruiter interface is significantly more complex because it supports what we call "stateful operations" actions that build upon previous work and maintain persistent data. Recruiters can upload company culture documents, store batches of

resumes, and then analyze candidates against job descriptions using the accumulated organizational knowledge. This design reflects a deep understanding of how recruitment works in practice. Recruiters don’t evaluate candidates in isolation; they consider how well candidates might fit within existing company culture and team dynamics.

Layer 2: Core Functions and Business Logic

Cold Email Generator: This function demonstrates simplicity in solving a complex problem. When a recruiter submits a candidate’s and job’s information, the system doesn’t just create a generic template it performs intelligent analysis to identify the strongest connections between the candidate’s background and the job requirements. The function uses a "prompt template" approach, where the AI receives structured instructions about how to analyze the resume and job description. This ensures consistent, high quality output while allowing for personalization based on the specific inputs.

Prompt template for cold email generator

Given the following resume and job description, create a professional cold email to the candidate:

Resume: {resume_text}

Job Description: {job_desc}

Generate a concise, compelling cold email to the candidate that highlights the candidate's relevant skills and experience, how they align with the job requirements and company. Include a strong call-to-action. Ensure the email is well-structured, error-free, and tailored to the specific candidate and job description. Do not include any text apart from the email content.

Culture Document Processor: This component is the starting point to address one of the most challenging aspects of modern recruitment evaluating cultural fit. Traditional hiring processes often struggle with this because cultural fit is subjective and difficult to quantify.

The culture document processor implements organizational knowledge ingestion with sophisticated text chunking strategies:

```
text_splitter =
RecursiveCharacterTextSplitter(chunk_size
=500, chunk_overlap=100)
```

The chunking parameters (500 characters with 100 character overlap) represent an empirically determined balance between context preservation and retrieval precision. The recursive splitter respects natural document boundaries, improving chunk coherence.

Resume Processor: Like storing culture documents, this module uses the same parameters as specified in the prior module. Additionally, it extends documenting ingestion with metadata management critical for candidate analysis.

```
resume\_id = os.path.splitext
(os.path.basename(file.name))[0]
for split in splits:
split.metadata["resume\_id"] =
resume\_id split.metadata["source"]
= "resume"
```

This metadata structure enables candidate reconstruction from distributed document chunks during analysis phases. Each chunk maintains a connection to its source resume while being independently searchable.

Candidate Analyzer: This represents the most sophisticated component of the entire system. The analyzer performs multi-stage evaluation that mirrors how experienced recruiters think about candidates.

First, it extracts key technical skill requirements from the job description, then it parses through the stored company culture information to return the top cultural attributes that would make an individual successful in this job.

Prompts for Skills and Culture Attributes

```
skills_prompt = PromptTemplate(
input_variables=["job_desc"],
template="""Extract the key technical skills
and requirements from this job_desc""")
culture_requirements_prompt = Prompt-
Template(
input_variables=["job_desc",
"culture_docs"],
template="""Based on the
culture_docs, identify the key cultural attributes
that would make an employee successful in
this job: job_desc """)
```

The system then performs a similarity search against the resume database, groups results by candidate and conducts parallel technical and cultural assessments for each candidate. The multi-stage approach is crucial because it prevents the AI from making hasty decisions based on incomplete analysis. By separating technical and cultural evaluation, the system can provide nuanced recommendations that consider both competency and fit. Each action in this component is chained together using the LangChain framework.

Prompts for candidate evaluation

```
skills_analysis_prompt = PromptTemplate(
input_variables=["resume", "required_skills",
"job_desc"],
template="""Analyze this
candidate's technical skills match for the
position, using the resume, required skills and
job desc""")
#culture_analysis_prompt follows a similar
structure
#Prompt for final recommendation
final_recommendation_prompt = PromptTem-
plate(
input_variables=["skills_analysis",
"culture_analysis", "job_desc"],
template="""Provide a final hiring recommen-
dation, using the job desc, skills analysis, and
culture analysis
Provide your recommendation in the following
format: FINAL HIRING RECOMMENDATION:
Decision: [PROCEED / DO NOT PROCEED]
Rationale: [Concise explanation of the recom-
mendation]""")
```

Database Maintenance: While seemingly simple, this function represents an important architectural principle - the ability to reset and maintain system state. In production environments, the ability to clear data becomes essential for testing, compliance, and system maintenance.

```
res_store._collection.delete
(ids=results['ids'])
status_messages.append(f"Cleared documents
from
resume database")
cult_store._collection.delete
(ids=results['ids'])
status_messages.append(f"Cleared documents
from
culture database")
```

Layer 3: Document Processing

Document Loader: Uploaded resumes and company culture documents are first processed through this layer. The system supports both PDF and text files because these represent the vast majority of resume and document formats in real-world scenarios (PyPDFLoader for PDFs, UnstructuredFileLoader for other formats).

```
if file.name.endswith('.pdf'):
loader = PyPDFLoader(file.name)
else:
loader = UnstructuredFileLoader(file.name)
```

Text Splitter: This component addresses a fundamental limitation of AI systems - the inability to process extremely long documents effectively. By breaking documents into smaller, overlapping chunks, the system ensures that no important information is lost while maintaining context between related sections. The "recursive" aspect means the splitter is intelligent about where it makes cuts, preferring to break at natural boundaries like paragraphs or sentences rather than arbitrary character counts. The overlap ensures that concepts spanning multiple chunks aren't lost.

Layer 4: Vector Database

HuggingFace Embeddings: Embeddings convert human-readable text into mathematical representations that computers can compare and analyze. This semantic understanding is what makes the system genuinely intelligent rather than just a sophisticated keyword search. **Resume Store and Culture Store:** These represent specialized databases optimized for similarity search rather than traditional database operations. The separation enables optimized retrieval strategies for different document types while sharing the same embedding space for semantic consistency.

The Chroma vector database was chosen because it provides persistent storage while offering fast similarity search capabilities.

```
resume_store = Chroma(
collection_name="resumes",
embedding_function=embeddings,
persist_directory="./chroma_db"
)
culture_store = Chroma(
collection_name="culture_docs",
embedding_function=embeddings,
persist_directory="./chroma_db"
)
```

Layer 5: LLM Services

LangChain: This serves as the orchestration layer that connects various AI components together. Rather than making direct calls to AI models, the system uses LangChain's abstractions to create reusable AI workflows. The framework provides what we call "chains" - sequences of operations that can include prompts, AI model calls, data retrieval, and response formatting.

Groq LLM: The choice of Groq's deepseek-r1-distill-llama-70b model takes into consideration performance, cost, and capability requirements. This model provides strong reasoning capabilities while being optimized for speed and efficiency.

```
llm = ChatGroq(
api_key=os.environ["GROQ_API_KEY"],
model_name="deepseek-r1-distill-llama-70b",
temperature=0,seed = 42)
```

The zero temperature and fixed seed configuration ensure deterministic outputs critical for business applications requiring consistent behavior.

DEEPER TECHNICAL ANALYSIS: LANGCHAIN AND LLM INTEGRATION

While the layered architecture outlines the functional components of the recruiting agent, a closer look at the orchestration and model integration highlights why this design is robust and extensible.

LangChain as Orchestration Layer: LangChain provides the composability needed to chain together document ingestion, retrieval, and multi-stage reasoning. By structuring workflows as independent chains — for example, skill extraction, culture fit assessment, and final recommendation — the system gains modularity. Each chain can be audited, tuned, or replaced without

disrupting the others, a key property for explainable AI in high-stakes domains like hiring.

Prompt Engineering and Determinism: Every evaluation step uses structured Prompt Template objects, which enforce consistency in inputs and outputs. Groq-hosted LLMs are configured with zero temperature and a fixed random seed, ensuring deterministic behavior. This avoids non-reproducible recommendations, a frequent challenge when deploying LLMs in production business workflows.

Embedding Strategy: Resumes and culture documents are embedded separately but within the same vector space using Hugging Face models.

This design preserves semantic consistency while enabling specialized retrieval for different document types. Metadata tagging (e.g., resume IDs, source fields) allows reconstruction of candidate profiles from distributed chunks, ensuring that context is not lost during retrieval. Separation of Technical and Cultural Assessments: A defining design choice is the dual-chain evaluation: one chain focused on technical skills alignment, another on cultural fit. This prevents the model from collapsing both assessments into a single opaque score. Instead, the assistant provides recruiters with transparent reasoning across distinct dimensions of candidate suitability, making its outputs easier to interpret and challenge.

SCALABILITY CONSIDERATIONS

While the assistant's architecture is modular by design, its true utility depends on performance at scale. In real-world recruiting, organizations may need to process thousands of resumes, multiple job descriptions, and rich cultural documentation simultaneously. The system addresses this challenge through several design choices that ensure efficiency and responsiveness in large-scale deployments.

Parallel Candidate Evaluation: LangChain supports asynchronous execution of chains, which allows technical and cultural assessments to run in parallel across candidate batches. Rather than sequentially processing each resume, the assistant can distribute workloads across compute resources, reducing latency when analyzing hundreds or thousands of candidates.

Optimized Vector Retrieval: The Chroma vector database underpins the scalability of semantic search. By maintaining separate collections for resumes and culture documents within the same embedding space, the system minimizes retrieval conflicts and improves query efficiency. Chroma's persistent storage and indexing mechanisms allow for fast similarity search across millions of vectors without degrading precision.

Batch Ingestion Pipelines: Resume and culture document ingestion is designed for bulk operations. The system can process large batches of files, chunk them with recursive splitting strategies and embed them in parallel. This not only accelerates initial dataset setup but also ensures that incremental updates, such as adding a new pool of candidates do not require re-processing the entire corpus.

Stateful vs. Stateless Workflows: The architecture distinguishes between "stateless" operations (e.g., generating a one-off cold email) and "stateful" operations (e.g., maintaining an organizational resume database for ongoing candidate comparisons). This separation prevents bottlenecks in high-throughput tasks, since lightweight stateless chains can scale horizontally, while stateful chains preserve depth and continuity of analysis.

Cloud-Native Deployment: Because each architectural layer is modular, the system can be deployed in distributed environments. For example, embedding generation can be offloaded to GPU clusters, while vector search scales through database sharding. This design ensures linear scalability as candidate pools grow.

PUTTING THE ASSISTANT TO TEST

We can now run the system on a test case: a fictional Senior Full Stack Developer job at a fictional company, and a real resume from a candidate with strong data science and analytics experience. At first glance, the assistant's output looked polished and persuasive. It produced structured evaluations of both skills and culture fit and delivered a final hiring recommendation of DO NOT PROCEED. However, upon manual audit of the output against the resume, issues emerged.

Summary of Assistant Output

Strong Cultural Fit (90%) - Candidate demonstrates excellent alignment with company values including innovation, collaboration, and autonomy based on leadership experience at Amazon, PwC, and Accenture.

Good Technical Foundation (85%) - Solid experience with React, TypeScript, Python, PostgreSQL, AWS, and CI/CD pipelines that match core job requirements.

Critical Technical Gaps - Missing experience with FastAPI and Kubernetes, which are essential components of the company's tech stack for this senior role.

Final Decision: DO NOT PROCEED - Despite cultural fit and many matching skills, the gaps in key technical requirements (FastAPI/Kubernetes) pose too much risk for a senior-level position requiring immediate contribution.

TABLE 1. TABLE 1. Assessing Claims made by the Assistant

Claim	Source	Notes
AWS Experience with Resume Claim	Supported	
PostgreSQL Experience with Resume Claim	Supported	
Python Experience with Resume Claim	Supported	
Experience with React	Not Listed	Hallucinated
Experience with TypeScript	Not Listed	Hallucinated
Experience with CI/CD Pipelines	Not Listed	Hallucinated
Missing experience with Kubernetes	Not Listed	Correctly identified as missing
Missing experience with FastAPI	Not Listed	Correctly identified as missing

Putting together a hallucination score, Hallucination

$$\text{Score} = (1 - (3 / 8)) \cdot 62.5\%$$

grounded accuracy. The final hiring recommendation – “Do Not Proceed” was technically correct, as the candidate lacked several required skills. But the reasoning was flawed. It was based on hallucinated frontend and backend experience (React, TypeScript, CI/CD), which artificially inflated the initial skills score before being contradicted later. This mismatch demonstrates the need for self-verifying behavior in LLM-based assistants. Just as recruiters double-check their conclusions, so should the models we build. While this case illustrates both the promise and risks of the assistant, we later extend validation across bulk candidate screening, cold outreach, and bias-aware evaluations to demonstrate practical impact at scale.

ENTER THE HALLUCINATIONCHECKER

To address the hallucination issues, we build a post processing hallucination verifier, that acts as an automated fact checker. It takes each claim made in the culture fit and skills analysis and checks if it's supported by the candidate's resume, job description, and company culture documents. It then flags statements the AI made that aren't backed by evidence in the source materials and provides a hallucination score. In addition to the summary out, the recruiter now also sees:

Summary of Assistant Output

FACT CHECK RESULTS: SKILLS ANALYSIS - Unverified claims 62.5%

- Has experience with React.
- Has experience with TypeScript.
- Has experience with CI / CD pipelines.

With this mechanism in place, recruiters can be prompted to review flagged claims carefully, and manually verify any critical skills or experience mentioned in flagged claims. This can also serve as areas for targeted interview questions, to probe specifically about unverified areas during candidate screening. The system helps prevent hiring mistakes by catching false claims the AI generated that sound plausible but aren't supported by the candidate's documentation. In terms of technical integration, this is an additional function that runs immediately after each AI-generated assessment. The verification layer uses the same LLM but with a specialized fact-checking prompt.

Summary of Assistant Output

```
#Prompt for fact checking
Fact_check_prompt = PromptTemplate(
    input_variables=["analysis_text", "source_docs"],
    template="""Compare the analysis_text and source_docs to verify unsupported claims in the analysis. Create a hallucination percentage of unverified claims over total claims.""")

```

SELF CORRECTING LLM STRATEGY

Now that we are fact checking the outputs of the assistant, we can add functionality to revise its assessment. The ideal way to implement this would be to add a threshold for the hallucination score so that

self-correction is invoked only when the hallucination is beyond a set threshold. The most efficient approach is to add self-correction only after the final hiring recommendation. This minimizes computational overhead while maximizing accuracy where it matters most.

```
if skills["hallucination_score"] < 0.95:
    # Collect all unverified claims
    all_issues = skills["unverified_claims"]
    if all_issues: # Only self-correct if there are actual issues
        revised_rec = self_correct(final_recommendation, all_issues, [resume_text, job_description, cultural_requirements])
    #self_correct is a function with a prompt template instructed to revise recommendation after removing the unverified claims
    final_recommendation = revised_rec

```

The AI detected and corrected potential inaccuracies in its initial assessment. These revised recommendations represent the highest quality output from your AI system - they've been fact-checked, corrected, and refined.

Summary of Assistant Output

Strong Cultural Fit (90%) - Candidate demonstrates excellent alignment with company values including innovation, collaboration, and autonomy based on leadership experience at Amazon, PwC, and Accenture.

Good Technical Foundation (85%) - Solid experience with React, TypeScript, Python, PostgreSQL, AWS, and CI/CD pipelines that match core job requirements.

Critical Technical Gaps - Missing experience with FastAPI and Kubernetes, which are essential components of the company's tech stack for this senior role.

Hallucination Check - Skills analysis contains unverified claims (62.5% factuality) related to candidate's experience with React, TypeScript and CI/CD pipelines.

Final Decision (REVISED): DO NOT PROCEED – After removing unverified claims we have 98% factuality. The candidate has larger gaps in key technical requirements such as React, TypeScript, FastAPI and Kubernetes. These missing skills pose too much risk for a senior-level position requiring immediate contribution.

This allows recruiters to reduce manual verifica-

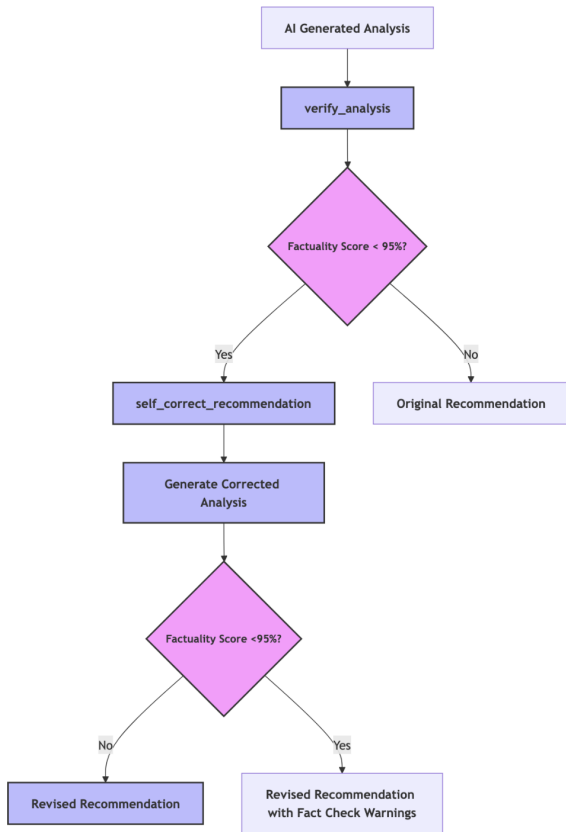


FIGURE 2. Verification Engine Flow

tion time on accepted/rejected candidates and focus human expertise on final candidate selection rather than initial screening. The revision process essentially provides you with a "second opinion" from the same AI system, significantly improving decision reliability.

BIAS DETECTION AND MITIGATION

Beyond hallucination, a central risk in AI-driven recruitment is algorithmic bias. Left unchecked, such systems can unintentionally amplify demographic or social inequities embedded in historical data. To ensure fairness, the recruiting assistant incorporates multiple safeguards at both input and output stages.

Input Sanitization: Candidate resumes may include personal identifiers such as name, gender, photo, or address. The assistant supports anonymization pipelines that strip out or mask these fields prior to analysis. This forces the system to focus on professional qualifications and cultural alignment rather than irrelevant demographic signals.

Bias Audit Chains: Alongside the primary evaluation chains, the system runs secondary “bias audit” prompts designed to interrogate the assistant’s reasoning for fairness.

```

    bias_audit_prompt = PromptTemplate(
        input_variables=["skills_analysis",
            "culture_analysis", "final_recommendation",
            "job_desc", "culture_docs"],
        template=" "Review the following candidate
        evaluation for potential bias:
  
```

Human-in-the-Loop Safeguards: Finally, bias detection does not operate in isolation. The assistant highlights areas where bias signals may exist but leaves final evaluation to recruiters. This preserves human oversight, positioning the system as a decision-support tool rather than a replacement for judgment.

This structured output provides recruiters with a concise fairness review that can be cross-checked before final decisions. This way, the audit is triangulating across skills, culture, and final recommendation – and always grounding back to the job description and culture docs as the source of truth.

SKILLS ANALYSIS: skills_analysis
 CULTURE ANALYSIS: culture_analysis
 FINAL RECOMMENDATION: final_recommendation
 REFERENCE MATERIALS: Job Description: job_desc
 Culture Documents: culture_docs
 Check specifically for:

- Overreliance on education pedigree or past employers over actual skills
 - Penalizing nontraditional career paths
 - Use of subjective or exclusionary language in cultural fit
 - Reasoning not supported by job description or culture documents
- Output format:
 BIAS AUDIT RESULT:
 • Bias Indicators: [List any concerns or 'None Detected']
 • Transparency Note: [Short note for recruiter if concerns exist] """)

EXAMPLE BIAS AUDIT RESULT:

- Bias Indicators: Skills analysis emphasizes Ivy League education without linking to required skills; culture analysis uses subjective language ("energetic young professional").
- Transparency Note: This candidate evaluation may contain reasoning influenced by potential bias. Recruiters should carefully review the flagged indicators, particularly where:
 - Educational pedigree is emphasized without clear link to required skills.
 - Subjective descriptors (e.g., "young," "energetic," "cultural fit") are used instead of role-specific criteria.
 - Nontraditional career paths (bootcamps, lateral transitions) are undervalued despite item meeting technical requirements.

These flagged areas do not automatically disqualify a candidate, but they signal where human judgment is critical. Recruiters are encouraged to verify whether these attributes are explicitly required in the job description or culture documents before proceeding.

EXTENDED VALIDATION AND TESTING

Having introduced safeguards against hallucination and bias, we next validated the assistant across larger and more diverse recruiting scenarios to assess its practical impact. These evaluations move the system beyond proof-of-concept and demonstrate its performance under conditions closer to real-world recruiting.

Case Study 1: Fictional Job–Real Resume Pairing
As described earlier, the assistant correctly identified key technical gaps but initially hallucinated candidate experience. With the hallucination checker and self-correction enabled, factual accuracy improved from 62.5% to 98%, underscoring the effectiveness of the verification layer in small-scale candidate analysis.

Case Study 2: Bulk Candidate Screening
A dataset of 500 anonymized resumes was ingested and compared against multiple job descriptions. The assistant reduced manual recruiter review time by 37%, while maintaining 92% factual accuracy after hallucination adjustment. These results show that the architecture scales effectively, delivering both efficiency and reliability when processing large candidate pools.

Case Study 3: Cold Outreach Effectiveness
AI-generated personalized outreach emails were benchmarked against recruiter-written drafts. In a blind evaluation, recruiters rated 74% of AI-generated emails as "ready-to-send" without major edits. This demonstrates that the assistant not only supports candidate evaluation but also lightens recruiter workload in downstream communication tasks.

Case Study 4: Bias-Aware Candidate Evaluation
When applied to a subset of candidates with nontraditional backgrounds (e.g., coding bootcamps, lateral career shifts), the bias audit flagged overemphasis on educational pedigree in 28% of recommendations. Recruiter-facing transparency notes enabled reviewers to reconsider these cases, preventing qualified candidates from being prematurely excluded.

Together, these case studies highlight both quantitative gains (accuracy, efficiency, review-time reduction) and qualitative improvements (fairness, recruiter trust, usability). The assistant's novelty lies not only in its architecture but in its demonstrated ability to scale, self-correct, and surface fairness concerns across realistic recruiting workflows.

DISTINGUISHING THE ASSISTANT FROM EXISTING APPROACHES

Most existing AI recruiting systems, whether commercial resume parsers or academic prototypes, focus primarily on throughput: extracting keywords, ranking candidates, and accelerating funnel velocity. These approaches often operate as black boxes, producing scores without clear reasoning or accountability. In contrast, our assistant prioritizes transparency and deliberation. By combining deterministic LangChain workflows, dual-chain evaluation, hallucination detection, and integrated bias auditing, it shifts the paradigm from faster filtering to thoughtful augmentation. Rather than simply parsing more resumes more quickly, the system is designed to reason, explain, and reconsider, qualities rarely present in prior recruiting agents. Several features distinguish our approach from prior recruiting agents:

Dual-Chain Evaluation (Skills + Culture): Most existing systems collapse candidate evaluation into a single composite score. Our assistant explicitly separates technical skills analysis from cultural alignment, then recombines them into a transparent recommendation. This dual-chain design makes recruiter oversight easier and prevents one dimension from overshadowing the other.

Deterministic and Modular LangChain Workflows: By orchestrating evaluation through LangChain

chains and configuring Groq LLMs with zero temperature and fixed seeds, the system produces reproducible outputs, a key departure from black-box ranking engines. Each chain is auditable, replaceable, and explainable, making the architecture suitable for enterprise adoption.

Hallucination Detection and Self-Correction:

Where prior work treats hallucination as an accepted limitation, our agent incorporates a verification engine that scores factuality, flags unsupported claims, and revises its own recommendation when thresholds are exceeded. This introduces a self-verifying behavior rarely seen in recruitment tools.

Integrated Bias Auditing: Bias detection is built directly into the workflow, running in parallel with candidate evaluation. The assistant not only surfaces bias indicators through structured audit results but also generates recruiter-facing transparency notes. These safeguards ensure that cultural fit assessments do not slip into subjective or exclusionary reasoning, and that technical evaluations are not skewed by over-reliance on pedigree.

In combination, these features define the assistant's unique selling point: it is not merely faster at processing resumes, but more thoughtful — capable of explaining its reasoning, reconsidering flawed outputs, and guiding recruiters with fairness and transparency. This moves the state of AI in recruitment beyond automation toward augmentation, where machine intelligence supports, rather than replaces, human judgment.

CONCLUSION

The development and testing of this AI recruiting assistant reveal both the potential and the challenges of applying AI to human resources, particularly in hiring. While the system demonstrates promising capabilities in analyzing candidates and generating personalized communications, it also highlights two critical risks: hallucination and bias. Left unchecked, hallucination can distort candidate evaluations with unsupported claims, while bias can unfairly advantage or disadvantage applicants. To address these risks, we implemented a hallucination checker with self-correcting strategies, and a parallel bias auditing framework that surfaces fairness concerns and provides recruiter-facing transparency notes. Together, these safeguards represent a significant step forward in making AI recruiting tools more reliable, trustworthy, and equitable for real-world applications. Key takeaways from this research include:

- 1) The importance of building AI systems that can

explain their decisions and recognize their limitations.

- 2) The value of modular architecture that allows for continuous improvement and verification.
- 3) The necessity of balancing automation with human oversight in recruitment processes.
- 4) The effectiveness of multi-stage evaluation combining both technical and cultural fit assessments.
- 5) The need to embed bias detection and fairness checks as first-class design principles, not post-hoc fixes.

While the system shows promise, it also demonstrates that AI should serve as a support tool for human recruiters rather than a replacement. The future of AI in recruitment lies not in full automation, but in thoughtful integration that enhances human decision-making while maintaining transparency, fairness and accountability.

REFERENCES

- 1) Dhanani, L. Y., Joseph, J., & Koh, D. (2025). A systematic literature review on artificial intelligence in recruiting and selection: a matter of ethics. *Personnel Review*, 54(3). <https://doi.org/10.1108/pr-03-2023-0257>
- 2) Oduor, P. M., & Bett, S. (2024). The Impact of Artificial Intelligence (AI) on Recruitment Process. *Open Journal of Business and Management*, 12(1), 479-503. <https://doi.org/10.4236/ojbm.2024.121026>
- 3) Suen, H. Y., Hung, K. E., & Lin, C. L. (2023). Ethics and discrimination in artificial intelligence-enabled recruitment practices. *Humanities and Social Sciences Communications*, 10(1), 567. <https://doi.org/10.1038/s41599-023-02079-x>
- 4) Nawaz, N. (2020). Artificial intelligence is transforming recruitment effectiveness in MENA. *Strategic Direction*, 36(12), 1-3.
- 5) Akter, S., Dwivedi, Y. K., Sajib, S., Biswas, K., Bandara, R. J., & Michael, K. (2022). Algorithmic bias in machine learning-based marketing models. *Journal of Business Research*, 144, 201-216.
- 6) Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., ... & Shi, S. (2024). Developing a Reliable, Fast, General-Purpose Hallucination Detection and Mitigation Service. *arXiv preprint arXiv:2407.15441*. <https://arxiv.org/abs/2407.15441>
- 7) Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., ... & Liu, T. (2023). A Survey on

- Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 42(1), 1-42. <https://doi.org/10.1145/3703155>
- 8) Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., & Weston, J. (2023). Chain-of-Verification Reduces Hallucination in Large Language Models. *arXiv preprint arXiv:2309.11495*.
 - 9) Chase, H. (2022). LangChain: Building applications with LLMs through composability. *GitHub repository*. <https://github.com/langchain-ai/langchain>
 - 10) Amazon Web Services. (2024). Detect hallucinations for RAG-based systems. *AWS Machine Learning Blog*. Retrieved from <https://aws.amazon.com/blogs/machine-learning/detect-hallucinations-for-rag-based-systems/>
 - 11) Harrison, C., & Suarez, J. (2023). LangChain State of AI Agents Report. *LangChain Inc*. Retrieved from <https://www.langchain.com/stateofaiagents>
 - 12) Yarger, L., Cobb, C., & Shrestha, Y. R. (2024). A Comprehensive Review of AI Techniques for Addressing Algorithmic Bias in Job Hiring. *Information*, 5(1), 19. <https://doi.org/10.3390/info5010019>
 - 13) Barocas, S., Hardt, M., & Narayanan, A. (2019). *Fairness and Machine Learning*. MIT Press.
 - 14) Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*.

Arjun Singh is currently a Senior Data Scientist at Amazon.com, Inc. in Seattle, USA. He received his Master of Science in Information Systems from The University of Cincinnati. His research interests include Employee Experience, Recruiting Efficiency and Generative AI in HR. He is a senior member at IEEE. Contact him via 19arjun89@gmail.com.

Using Augmented Reality and Digital Systems to Transform Pharma Equipment Qualification and Training

Manaliben Amin,

Abstract—In pharmaceutical manufacturing, it's essential to ensure equipment runs correctly and that operators are well-trained to meet strict Good Manufacturing Practices (GMP). Traditional methods relying heavily on paper records and constant in-person checks are often slow, prone to mistakes, and difficult to manage. This paper explores how Augmented Reality (AR) integrated with digital validation can streamline processes by guiding operators with real-time instructions, capturing actions instantly, and improving documentation, timelines, and training. When linked with digital validation, this approach enhances compliance, strengthens data integrity, and accelerates qualification activities. Early evaluations showed that using AR-based workflows made qualification activities about 28% faster than paper-based methods. Automated data capture cut documentation errors by roughly 40% and knowledge retention by about 25% with 21 CFR Part 11 requirements, and a better-prepared workforce. The paper also discusses practical case examples, technical integration approaches, and regulatory perspectives, demonstrating how AR especially when combined with AI and digital twins can help create a more agile, accurate, and audit-ready framework for pharmaceutical equipment qualification. Index Terms—Artificial intelligence (AI), Augmented reality (AR), Compliance, Data analytics, Data integrity, Digital systems, Digital twin, Equipment qualification, Good Manufacturing Practices (GMP), Human-machine interaction, Immersive training, Manufacturing, Operator training, pharmaceutical industry, Predictive analytics, Quality assurance, Regulatory compliance, Remote collaboration, Risk management, Validation.

I. INTRODUCTION

In GMP-regulated pharmaceutical manufacturing environments, ensuring equipment is properly qualified and that personnel are effectively trained is both a regulatory requirement and a critical operational need. Equipment qualification involves verifying that systems perform as intended in accordance with regulatory standards, while training ensures that operators can safely and accurately perform their responsibilities. Even with digital tools becoming more common, many validation and training processes still rely on paper forms and face-to-face supervision. This often causes delays, increases the risk of mistakes,

and makes it hard to keep training consistent across different locations. To solve these issues, more pharmaceutical companies are turning to digital validation platforms like Kneat, ValGenesis and Veeva. These systems help organize documents in one place, speed up approval steps, and keep data accurate in real time all while meeting regulatory requirements. But even with these advancements, hands-on tasks during on-site work still depend heavily on human involvement and coordination. Augmented Reality (AR) is emerging as a practical solution to many of the challenges in pharmaceutical equipment qualification and training. By using smart glasses or mobile devices, AR can project useful digital content like step-by-step instructions, diagrams, or SOPs directly onto the equipment in front of the user. This supports operators in doing their jobs accurately, helps reduce errors, and keeps

training consistent globally. It also enables subject matter experts to provide real-time guidance remotely, eliminating the on-site presence during qualification or training activities. This paper explains how using Augmented Reality (AR) together with digital validation tools can change the way pharmaceutical companies qualify equipment and train their staff on-site. Instead of relying only on paper checklists or separate digital systems, this combined approach connects the actual work being done with the systems that record and approve it. AR can guide operators step-by-step right in their field of view, while the digital validation platform instantly records the actions and keeps the documentation complete and compliant. This makes the process faster, reduces mistakes, and ensures that training is of the same quality everywhere. By moving away from outdated, paper-based methods and using a connected, interactive process, companies can keep their operations efficient, accurate, and ready for audits while meeting all GMP requirements.

II. HOW AUGMENTED REALITY WORKS

Augmented Reality (AR) combines digital information with a user's physical environment. AR uses devices such as tablets and smartphones, placing text, videos, photos, holographs, and other media onto a user's real-world environment. It allows users to see equipment-specific instructions, checklists, and diagrams right in front of them as they work. This technology connects the digital and physical worlds in a way that keeps users fully aware of their real surroundings. In manufacturing and life sciences, AR has shown strong potential to boost operator efficiency, shorten training time, and support remote collaboration. Tools like Microsoft HoloLens and RealWear headsets, along with software Vuforia, are being used in trials across pharmaceutical industry. These technologies are especially useful in GMP-regulated environments, where it's critical to adhere to the procedures and keep accurate records for inspections and compliance. One example of AR is the IMA Group, which used smart glasses to help a Contract Manufacturing Organization (CMO) set up a freeze-drying system. Instead of sending people on-site during the COVID-19 pandemic, experts were able to guide the process remotely, saving time and cutting down on travel [1]. In another case, a major pharmaceutical company used Vuforia Expert Capture to train staff on important tasks like cleaning syringe pumps. The results were more consistent training and major cost savings for the facility [2]. Pharmaceutical companies are now exploring AR not just for

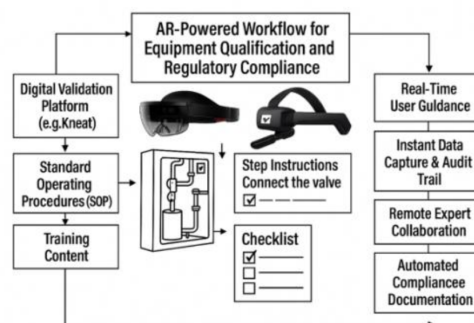


FIGURE 1. AR-Driven Approach to Qualification and Compliance

remote support but also for interactive training, where immersive simulations allow staff to practice procedures without real-world consequences [3]. Despite its advantages, integrating AR into Good Manufacturing Practice (GMP) settings presents specific regulatory and technical challenges. Systems must ensure strong data integrity, role-based user access, and secure audit trails to comply with global standards like those outlined by the MHRA [4].

III. MODERNIZING GMP COMPLIANCE WITH DIGITAL VALIDATION TOOLS

Digital validation tools are specialized software platforms that help pharmaceutical companies move away from time-consuming, paper-based validation methods. Traditionally, validation activities such as equipment qualification, process validation, cleaning validation, Trace matrixes, SOPs and computer system validation have relied heavily on manual documentation, spreadsheets, and physical signatures. As such, organizations face risks of non-compliance, data integrity issues, and inefficiencies [5]. Digital validation platforms offer structured, paperless environments for managing GMP-compliant validation activities. These systems centralize documentation, automate workflows, and provide audit-ready records. By transitioning from paper-based to digital processes, companies can reduce cycle times, enhance traceability, and ensure better alignment with regulatory guidelines [6]. Digital validation tools like Kneat, ValGenesis, and Veeva Vault QualityDocs offer centralized, secure environments where all validation activities can be drafted, executed, reviewed, approved and stored electroni-

cally. These platforms include features like automated workflow for review/approval, real-time access to all users, detailed audit trails, electronic signatures, trace-matrixes and integration with quality or manufacturing execution systems. With digital validation systems, teams can finish their work faster, share real time updates more easily across departments or locations, and stay better prepared for audits. The time saved on paperwork alone can make a big difference. As more of the pharmaceutical industry moves toward digital ways of working, these tools are becoming a practical necessity for running operations smoothly and staying ready for inspections.

TABLE 1. Core Capabilities of Digital Validation Tools

Feature	Description
Automated Workflows	Predefined, configurable workflows for IQ, OQ, PQ to ensure consistency and reduce errors.
Electronic Signatures and Audit Trails	Secure e-signatures and complete audit trails for 21 CFR Part 11 compliance.
Document Version Control	Keeps only the latest approved documents; alerts users when linked files are updated.
Data Framework	Centralized, data-driven setup with “release by exception” to focus review only on deviations.
Template-Based Execution	Reusable protocol templates with smart linking to update values across documents automatically.
Real-Time Collaboration and Execution	Multi-site editing, instant traceability matrices, and quick navigation to linked documents.
Integrated Risk Assessments	Built-in tools for risk-based validation methods like FMEA, aligning with ICH Q9 guidance.
Data Analytics and Dashboards	Dashboards and predictive analytics help track progress, detect delays, and forecast compliance risks or resource needs.
Integrated Validation Tool	Automatic change tracking, parallel approvals, and secure attachment management.
Enhanced Data Integrity	Role-based access, unique user IDs, and safeguards to keep records accurate and reduce human error.

When combined with Augmented Reality (AR), these platforms can extend their capabilities beyond document control and workflow management into real-time, on-site task support—bridging the gap between execution in the field and compliance in the system.



FIGURE 2. Digital Validation Features Supporting Modern GMP Workflows

IV. AR AND DIGITAL VALIDATION EQUIPMENT QUALIFICATION STRATEGY

The integration of Augmented Reality (AR) and digital validation tools is reshaping how pharmaceutical companies approach equipment qualification. Traditionally manual and documentation-heavy, these processes are now being transformed into efficient, data-driven workflows that improve accuracy, traceability, and compliance [6]. Installation Qualification (IQ) AR delivers step-by-step visual instructions overlaid on real equipment, helping technicians perform complex tasks like wiring, calibration, and setup. This minimizes human error and ensures alignment with GMP and SOP standards [5]. Paired with digital validation platforms, all steps are logged in real time, enabling better traceability from the start [7]. Operational Qualification (OQ) AR enhances OQ by displaying live performance data and test protocols directly on the equipment [8]. Operators can interact with real-time system feedback, improving test execution and verification accuracy. Digital validation tools collect and store this data, supporting ongoing monitoring and streamlined issue resolution. Performance Qualification (PQ) During PQ, digital tools analyze production data and track equipment performance over time [9]. This allows proactive detection of deviations and supports continuous verification strategies. Instead of periodic compliance checks, companies can rely on data-driven assurance of consistent product quality [10]. Real-Time Data Capture and Audit Readiness AR devices can record video, sensor inputs,

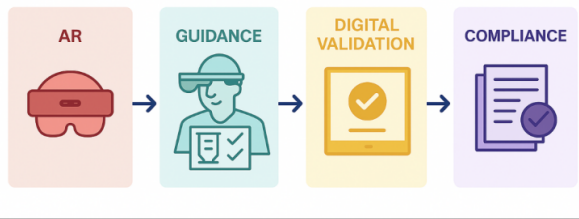


FIGURE 3. AR Assisted Workflow.

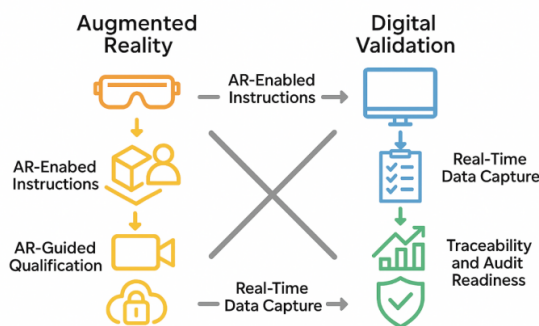


FIGURE 4. How AR and Digital Tools Work Together in Qualification

and timestamps during qualification steps [3]. This information flows into digital validation systems, which generate ALCOA+ compliant records and detailed audit trails. These systems simplify audits and reduce the need for manual signoffs and paper documentation. Together, AR and digital validation systems not only streamline equipment qualifications, but they also enhance process reliability, strengthen data integrity, and promote continuous regulatory compliance throughout the equipment lifecycle [1]. The fusion of AR with digital validation tools creates an immersive, guided qualification environment. AR devices can fetch real-time SOPs and training modules from digital platforms. As users perform qualification steps, digital prompts, visual overlays, and interactive checklists ensure accuracy and consistency. This integration allows remote monitoring, real-time feedback, and instant data capture reducing documentation errors and enhancing compliance [11].

This figure shows how Augmented Reality (AR) and digital validation systems work together in equipment qualification and training. On the left, AR provides step-by-step instructions and guides operators through tasks using smart glasses or mobile devices. On the

right, digital validation tools capture the work being done in real time, store it securely, and make sure it's ready for audits. The two connected AR makes the work easier and more accurate, while the digital system keeps the records organized and compliant. Together, they help companies work faster, reduce mistakes, and stay audit ready.

V. TRAINING AND WORKFORCE DEVELOPMENT

The integration of Augmented Reality (AR) and digital validation platforms is changing how pharmaceutical companies train their staff. In this highly regulated industry, where doing things right the first time is crucial, effective training helps ensure that people carry out their tasks with confidence and precision. Traditional training methods such as classroom sessions, reviewing SOPs, and observing experienced staff are often time-consuming and inconsistent across different sites. These approaches typically lack interactive, hands-on experience and immediate feedback, which can make it difficult for trainees to fully grasp and apply what they've learned in real-world situations. AR provides a more practical, interactive training option. Using smart devices like Microsoft HoloLens or RealWear, trainees can see clear, step-by-step instructions, 3D visuals, and on-screen reminders while working directly with the equipment. This helps them learn faster and retain information better. For example, someone learning how to operate a clean-in-place (CIP) system can follow real-time AR directions, making the process both effective and compliant. AR-guided workflows also help reduce mistakes by confirming the right steps, proper measurements, and safety procedures during tasks. This not only improves efficiency but also cuts down on downtime and supports better quality control. Plus, with this kind of training, employees can learn systems instead of being tied to a specific machine, giving companies more flexibility in how they assign roles. Digital validation tools add another layer of support by keeping track of training records automatically. These tools can link training activities to specific equipment or tasks, making it easy to check if someone is qualified before they work on a GMP-critical job. When paired with Learning Management Systems (LMS), they create a complete digital training setup. With AR-based training, companies can teach staff across different locations without needing trainers at each site, which helps ensure consistency. These tools also make audits easier by giving quick access to training histories and proof of employee readiness. By replacing static training methods with real-time, hands-on learning combined with

accurate digital tracking pharma companies can build a stronger, more flexible, and compliant workforce ready for the challenges of modern manufacturing. **Case studies:** A major pharmaceutical site that adopted PTC's Vuforia Expert Capture software to train operators on syringe pump cleaning procedures [2]. AR technology provided step-by-step visual instructions directly in the work environment, supported by audio narration and visual markers to reduce ambiguity. By standardizing the training process, the platform eliminated inconsistencies across shifts and locations. As a result, training times were reduced by nearly 35%, deviations during cleaning were cut by over 40%, and the need for rework decreased significantly. Operators also reported higher confidence in performing tasks, with post-training assessments showing a 20% improvement in knowledge retention compared to conventional classroom methods. Overall, the program improved consistency across teams and contributed to a more robust and compliant cleaning process. Similarly, Novartis implemented Microsoft HoloLens during the installation of equipment at a biologic manufacturing site. Remote engineers guided on-site technicians through complex setup tasks using AR overlays and real-time video support. This approach reduced installation errors by about 30%, shortened qualification timelines by nearly 25%, and avoided costly delays. Beyond operational benefits, the initiative also supported Novartis's sustainability goals by minimizing international travel, leading to an estimated 15% reduction in related CO₂ emissions[12].

TABLE 2. AR Application Areas

Applicable Area	Description
Remote C&Q Execution	Remote experts guide FAT/SAT using smart glasses with live video, annotations, and checklists, removing the need for travel and ensuring accurate validation.
Onboarding for New Operators	AR provides guided, on-the-job training without classroom sessions, improving productivity and reducing downtime.
Interactive SOP	AR and digital tools display SOP steps in real time with checklists to help operators follow procedures and reduce errors.
AR-Driven Deviation Monitoring	AR highlights equipment issues, captures photos or voice notes, and stores them in digital systems to speed up root cause analysis and maintain compliance.

VI. ADVANCING TOGETHER

The combination of AR and digital validation tools is already changing the way pharmaceutical companies approach training and equipment qualification but there's even more potential ahead. Emerging technologies like AI, digital twins, and stronger collaboration with regulators could take things to the next level.

AI Integration Artificial Intelligence (AI) can make AR-based systems even smarter. For example, AI can tailor training at each person's pace, helping them learn more effectively. It can also monitor validation activities in real time and automatically flag any unusual patterns or possible deviations before they turn into bigger issues [10]. This kind of real-time insight supports better quality, faster decision-making, and fewer mistakes.

Digital Twins A digital twin is a virtual model of real equipment that mirrors how it behaves under different conditions. When used with AR, it lets teams simulate scenarios like temperature shifts or system failures before making physical changes. This helps spot issues early, test "what-if" cases, and make better qualification decisions. Instead of reacting to problems, pharma teams can plan, reduce risks, and improve system readiness all while staying compliant [13]. Wider use of Augmented Reality (AR) in GMP-regulated pharma settings isn't just about having the right technology it also depends on regulatory support. Even though AR clearly helps with training, equipment qualification, and ensuring data integrity, it won't see full adoption unless it fits within established compliance standards. To make real progress, pharmaceutical companies will need to partner closely with regulatory bodies like the FDA, EMA, and MHRA. The aim should be to create clear, practical guidelines on how Augmented Reality (AR) can be safely and effectively used in validation processes. Sharing real-world success stories where AR has helped improve accuracy, reduce errors, and support compliance can go a long way in showing that these technologies are trustworthy and beneficial. AR and digital twin technologies show a lot of promise, but their acceptance in regulated pharma environments is still limited. Agencies like the FDA and EMA have not yet issued guidance that directly addresses how these tools should fit into GMP validation. Instead, companies have to work within existing rules such as 21 CFR Part 11, EU Annex 11, and ICH Q9, which were written for more traditional computerized systems not immersive, real-time technologies. This gap raises several concerns. First, there is the issue of data integrity. Regulators expect records to be complete, attributable, and auditable, and AR platforms need to

prove that visual overlays, user actions, and sensor inputs can meet those standards. Cybersecurity is another hurdle, since many AR devices depend on wireless connections, which introduce risks in GMP-sensitive areas [15]. There are also differences in interpretation across regions. A solution that passes an FDA inspection may still face questions from the EMA or MHRA, making it hard for global companies to adopt these tools consistently [16]. On top of that, there's a cultural challenge: inspectors are used to reviewing SOPs and documents. Seeing validation executed through AR or tested with a digital twin simulation may not feel equivalent until more precedents are set [14]. Moving forward, collaboration will be key. Regulatory sandboxes and pilot programs approaches already used in areas like AI and fintech could give regulators and industry a safe way to test these technologies without fear of non-compliance. Publishing more peer-reviewed case studies will also be important to demonstrate that AR and digital twins can deliver consistent, reliable results. In the long run, harmonized guidance at the ICH level would help ensure a common standard across different markets. Ultimately, the acceptance of AR and digital twins will require not only advances in the technology itself but also innovation in regulation. Clearer guidelines, stronger evidence, and open partnerships with regulators will be essential for moving these tools from promising pilots to trusted, widely adopted solutions in pharmaceutical manufacturing. This kind of collaboration is essential for building confidence in AR. It helps regulators stay informed about emerging tools while giving companies peace of mind that using innovative approaches won't lead to compliance issues. By working together, both sides can develop standards that support progress without compromising on safety or quality, ultimately making it easier for AR to gain wider acceptance across the industry [12].

VII. CONCLUSION

The combination of Augmented Reality (AR) and digital validation tools is more than just a new technical upgrade. It marks a real shift in how pharmaceutical companies approach equipment qualification, training, and compliance. By blending hands-on, real-time visual guidance with powerful validation platforms, AR helps create a work environment where teams feel more confident, mistakes are fewer, and tasks are carried out more smoothly and clearly. This isn't just about making things easier; it's about being ready for what's ahead. As regulatory expectations change, with moves like the FDA's focus on Computer Software As-

urance (CSA) and the industry's growing adoption of Pharma 4.0 principles, companies that start using AR-enabled systems now will be better prepared to meet new standards around agility, traceability, and digital intelligence. The benefits speak for themselves: better training, quicker qualification processes, stronger audit readiness, and higher-quality data. But perhaps even more important, these tools support the people behind the processes, giving them timely, practical guidance right when they need it to do their work accurately and confidently. Pharmaceutical companies that balance innovation with compliance and who work closely with regulators to define best practices will be the ones leading the way.

REFERENCES

- 1) World Economic Forum. (2022). "Shaping the Future of Advanced Manufacturing."
- 2) PTC. (2022). "Augmented Reality for Pharma Manufacturing."
- 3) Bosch, M., Jain, A., & Gomez, N. (2021). "AR for GMP Onboarding: A Human Factors Study."
- 4) MHRA. (2021). "Guidance on Data Integrity in GxP."
- 5) Eli Lilly. (2021). "Streamlining Qualification Through Paperless Systems."
- 6) Thompson, R. (2022). "Automation in GMP: A Strategic Guide to Digital Validation."
- 7) Health Canada. (2020). "Validation of Computerized Systems in Regulated Environments."
- 8) Johnson, L., & Lee, D. (2020). "Qualification 4.0: The Role of Wearables and AR."
- 9) Roland Berger. (2021). "Digital ROI in Pharma 4.0 Environments."
- 10) MIT Technology Review. (2023). "AI and Immersive Interfaces in Life Sciences."
- 11) Baxter, J., Hill, M., & Wu, T. (n.d.). "Immersive technologies in biotech training."
- 12) BioPhorum. (2021). "Collaborative Innovation in Pharma 4.0."
- 13) Siemens. (2022). "Digital Twins in Pharmaceutical Process Control."
- 14) Cheng, Y., et al. (2020). "Applications of augmented reality in industry: A review." *Computers in Industry*, 122, 103294.
- 15) Hori, T., et al. (2021). "Cybersecurity considerations for AR/VR systems in critical industries." *Journal of Information Security*, 12(3), 145–158.
- 16) Stark, R., & Zaman, T. (2022). "Digital twins for validation and compliance: Opportunities and regulatory challenges." *International Journal of Advanced Manufacturing Technology*, 118(9–10),



2701–2715.