International Journal of Science and Engineering Applications
Volume 12-Issue 09, 169 – 176, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1209.1049

Zero-Trust DevOps for Electronic Health Records: Continuous Verification in Multi-Cloud Environments

Nagarjuna Nellutla
Independent Researcher
Eagan, MN, USA 55123

Abstract: Electronic Health Records (EHR) are increasingly hosted across hybrid and multi-cloud infrastructures to support scalable analytics, cross-institution collaboration, and distributed care delivery. While these deployments enable high availability and integration of diverse digital services, they also introduce a broader attack surface in which authentication, service communication, and deployment operations become persistent security risks. Zero-Trust DevOps extends the classical DevOps life cycle with a security model that assumes no implicit trust for users, services, or network boundaries, mandating continuous verification throughout CI/CD execution, identity management, and infrastructure orchestration. This work proposes a continuous verification framework for secure EHR deployments, incorporating runtime service attestation, policy-driven identity controls, multi-cloud trust segmentation, and immutable audit trails that enforce defensive guarantees during both deployment and operation. The proposed approach integrates Zero-Trust principles into DevOps workflows to harden the EHR ecosystem against evolving threats without compromising interoperability or system agility.
Keywords: Zero-Trust Architecture, DevOps Security, Electronic Health Records, CI/CD Security, Multi-Cloud Deployment, Continuous Verification

1. INTRODUCTION

Electronic Health Record (EHR) systems have evolved from isolated, on-premises servers into distributed digital platforms that integrate laboratory information systems, clinical decision engines, telemedicine interfaces, imaging archives, billing systems, and population health analytics. These interconnected environments increasingly span hybrid and multicloud deployments, enabling scale, high availability, and cross institutional data exchange. However, distributing clinical workloads across multiple infrastructures also reshapes the threat surface. Patient data, identity tokens, infrastructure metadata, and deployment artifacts move through shared networks, dynamic routing paths, and heterogeneous compute environments where no implicit security boundary can be safely assumed.

Historically, EHR systems operated behind secure network perimeters where internal traffic was implicitly trusted and external entities were subject to strict access controls. This perimeter-based defense model is insufficient for multi-cloud environments where workloads migrate between providers, virtual boundaries shift with orchestration engines, and adversaries frequently exploit trusted channels or internal misconfigurations rather than attacking public endpoints. The rise of credential compromise, insider misuse, lateral movement attacks, pilfered container registries, and poisoned deployment artifacts demonstrates that threats originate from both outside and within organizational infrastructure. Trust based on location, network origin, or historical access patterns is no longer a reliable assumption.

Zero-Trust Architecture (ZTA) responds to these challenges by establishing a policy that no user, device, service, or deployment component is trusted by default, regardless of network location or prior authentication. Instead, identities must be continuously verified based on contextual signals such as workload integrity, device health, configuration rules, and policy-defined access constraints. When applied to DevOps, Zero-Trust alters how software is built, validated, and deployed. CI/CD pipelines must verify artifact authenticity before execution, evaluate configuration changes before rollout, and ensure that runtime services authenticate each other using principles that do not rely on shared network trust. Security shifts from perimeter enforcement to continuous verification embedded throughout the lifecycle.

Multi-cloud adoption reinforces the need for continuous verification, as EHR workloads increasingly scale across cloud vendors, regional data centers, and federated research networks. Each cloud environment introduces unique access controls, cryptographic capabilities, operational policies, and logging semantics. Moving a service across environments may unintentionally relax constraints, duplicate credentials, or modify routing trust relationships, creating openings for privilege escalation or unauthorized replication. Identity, policy enforcement, and workload attestation must therefore travel with the service rather than remain bound to a static network boundary. The more dynamic the deployment, the more dangerous implicit trust becomes.

Integrating Zero-Trust with DevOps transforms the deployment workflow into a defensive enforcement plane. Security becomes proactive: a pipeline may build rapidly, but it will not deploy without verified provenance, validated secrets, signed containers, and runtime identity attestation. Immutable logs ensure that artifact lineage cannot be rewritten, mutual authentication protects east-west communication, and policy driven gatekeeping blocks insecure updates even if functional tests succeed. Continuous verification hardens distributed EHR systems not by slowing development, but by embedding defensible guarantees directly into build automation, orchestration, and operational runtime. This convergence of Zero-Trust and DevOps offers a scalable

pathway to secure EHR operations across increasingly complex multi-cloud environments.

2. ZERO-TRUST PRINCIPLES IN CI/CD SECURITY

Integrating Zero-Trust Architecture (ZTA) with CI/CD pipelines transforms the software delivery process into a continuously validated trust boundary [1]. Unlike perimeter-driven deployments where authenticated users, network zones, or previously trusted artifacts are implicitly approved, Zero-Trust enforces a stance in which no entity—human or machine—is allowed to act without renewed verification [2]. This mandates stricter controls over source code origin, container lineage, identity permissions, signing requirements, infrastructure definitions, and inter-service communication.

In a Zero-Trust CI/CD model, all actions are evaluated against contextual evidence and policy-based constraints. Pipeline agents can only execute with short-lived credentials, build artifacts must be cryptographically signed before promotion, deployment workflows must verify workload integrity, and runtime services authenticate each other even when the network has already verified their access. This approach prevents malicious binaries, tampered manifests, stolen keys, and compromised build runners from infiltrating production workloads. Continuous verification therefore shifts security left and right simultaneously: validating upstream build inputs and downstream runtime behavior using identical trust primitives.

Zero-Trust security also embeds digital identity into each deployment stage. Developers, automation agents, Kubernetes controllers, API gateways, test utilities, and analytics workloads all function as independent identities that require mutual authentication. Role assignments are no longer static but evaluated based on policy rules, workload state, device health, and temporal authorization windows. Instead of overprivileged service accounts or shared keys, Zero-Trust CI/CD enforces least-privilege access with identity-based segmentation. This prevents lateral movement across microservices even if a single credential or node is compromised.

To operationalize these principles, continuous verification is applied before any CI/CD transition, including build issuance, staging deployment, canary rollout, and runtime scaling. Verification gates assess provenance, configuration drift, signature validity, and integrity attestation. Fig. 1 illustrates how continuous verification intercepts each CI/CD stage, embedding mandatory validation loops that deny unverified artifacts and workloads.

Continuous verification in CI/CD enables defensive deployment practices without delaying delivery cycles. By enforcing identity segmentation, cryptographic lineage, and mutual attestation at each transitional stage, the pipeline becomes an active validator rather than a passive delivery mechanism.

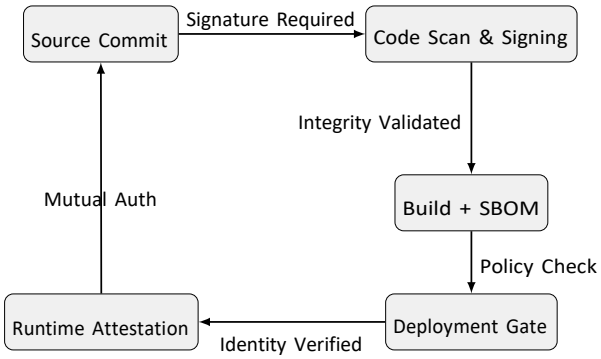

Fig. 1: Continuous verification enforcing Zero-Trust controls across CI/CD stages.

These Zero-Trust principles ensure that only verified workloads are built, only authenticated artifacts are deployed, and only attested services may operate, forming the foundation for secure multi-cloud EHR operations.

3. MULTI-CLOUD THREAT SURFACE FOR EHR DEPLOYMENTS

EHR systems in multi-cloud environments distribute sensitive workloads across diverse compute and network infrastructures, often spanning private data centers, multi-vendor public clouds, regional failover clusters, and federated research networks [3]. This diversity improves availability and scalability but expands the set of locations where identities, keys, certificates, and deployment assets may be exposed. A single misconfigured trust assumption can compromise multiple regions or propagate across cloud providers, especially when DevOps pipelines deploy EHR services into multiple environments using identical configurations and shared credentials [4]. Multi-cloud security therefore requires Zero-Trust controls that treat each cloud as an untrusted domain with independently verifiable identities and policies.

Unlike traditional deployments where a centralized security model governs all traffic, multi-cloud infrastructure produces fragmented trust boundaries [5]. Each cloud vendor enforces its own access controls, certificate authorities, RBAC semantics, audit formats, and infrastructure APIs. When EHR workloads migrate or scale dynamically, access rights can unintentionally widen, secrets may replicate into new regions, and stale cryptographic material may persist beyond its intended lifecycle. Additionally, mutable cloud services such as serverless functions and ephemeral nodes introduce runtime variability that can bypass static firewall rules or perimeter assumptions.

A Zero-Trust model mitigates these vulnerabilities by replacing inherited trust with inter-cloud verification. Identity and access management (IAM) becomes portable and continuously validated, rather than bound to the security model of a single cloud. Deployment artifacts must present verifiable provenance regardless of where they are executed,

International Journal of Science and Engineering Applications
Volume 12-Issue 09, 169 – 176, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1209.1049

and east-west traffic must authenticate across clouds using workload identity rather than network location.

Table I summarizes common attack opportunities in multicloud EHR deployments and highlights Zero-Trust countermeasures.

TABLE I: Multi-Cloud Threat Surface for EHR and Zero-Trust Mitigation

Attack Opportunity	Vector	Zero-Trust Mitigation
Stolen credentials reused across clouds	Identity	Short-lived tokens; identity segmentation
Unverified containers deployed in another region	Artifact	Image signing + SBOM provenance checks
Lateral movement through shared VPC or tunnels	Network	Mutual TLS enforced between workloads
Duplicate secrets in replicated clusters	Secrets	Dynamic secret rotation per environment
Blind spots from heterogeneous logging	Visibility	Immutable, unified audit and verification logs
Overprivileged automation agents	CI/CD	Policy-driven least privilege for pipeline identities

Zero-Trust transforms multi-cloud deployment from a federation of partially trusted networks into a continuously verified ecosystem. Each cloud environment becomes an untrusted zone until proven otherwise, and every workload validates identity and provenance before interaction. This perspective eliminates inherited trust across providers, reduces lateral exploitation paths, and ensures that EHR services can scale globally without weakening their defensive posture.

4. IDENTITY-BASED SEGMENTATION IN ZERO-TRUST DEVOPS

Identity is the fundamental trust anchor of Zero-Trust DevOps. Rather than securing networks or locations, Zero-Trust enforces least-privilege policies based on workload identity, user role intent, and temporally validated authorization rules. Every pipeline component—developers, automation agents, build runners, admission controllers, container orchestrators, API gateways, and microservices—must operate using scoped identities with independent authentication. This separation prevents overprivileged access from cascading across the CI/CD workflow or multi-cloud deployment surface [6].

Identity-based segmentation isolates actions not by where they originate, but by who and what is performing them. Instead of shared service accounts or static keys, Zero-Trust pipelines use short-lived credentials tied to contextual conditions such as workload signature, policy status, device health, and build provenance. Automation utilities cannot impersonate developers, pipeline agents cannot assume operator privileges, and containers cannot inherit credentials simply because they were deployed by a trusted node. Identity must be explicitly presented and validated before any interaction, including intra cluster traffic, inter-cloud migration, and pipeline execution.

Segmentation also extends to runtime workloads. Microservices communicating inside a cluster or across multiple clouds must authenticate via workload identity, rather than trusting internal IP addresses, VLANs, or VPN boundaries. Mutual authentication using certificates, attestation tokens, or signed service identities eliminates east-west trust assumptions that attackers often exploit after gaining a foothold. Fig. 2 illustrates how identity delineates trust across both DevOps workflows and distributed runtime traffic.

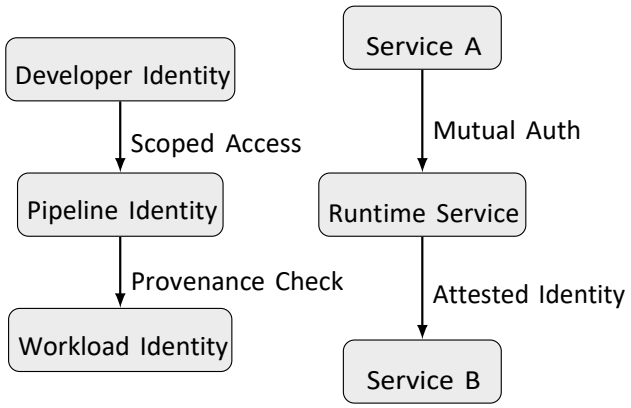

Fig. 2: Identity segmentation across CI/CD actors and runtime services.

Identity-based segmentation reduces reliance on perimeter controls and blocks attackers from escalating privileges once they compromise a single credential or machine. By treating each user, agent, and workload as an independently validated identity, Zero-Trust DevOps prevents lateral movement, enforces contextual access, and aligns operational security with multi-cloud variability [7]. Trust is not inherited from the network or deployment workflow; it must be continuously proven by every actor during build, deployment, and runtime execution.

5. CONTINUOUS VERIFICATION WORKFLOW FOR EHR PIPELINES

Zero-Trust DevOps requires every CI/CD stage and every runtime transition to prove identity, integrity, and policy compliance before execution continues. For EHR systems, this continuous verification process must validate more than build correctness; it must authenticate service provenance, enforce identity segmentation, confirm artifact lineage, and ensure that secrets or credentials are not inherited by default [8]. Verification becomes a gating mechanism embedded between

International Journal of Science and Engineering Applications
Volume 12-Issue 09, 169 – 176, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1209.1049

pipeline stages and an enforcement checkpoint between runtime operations.

Continuous verification begins at source submission. Commits must be signed with trusted keys before they are allowed to trigger builds. The verification pipeline inspects dependencies, validates code provenance, and rejects unsigned binaries or unverified base images. Build outputs are subjected to manifest verification, Software Bill of Materials (SBOM) generation, and cryptographic signing prior to being stored in registries. These registries likewise enforce signature validation, blocking tampered images or artifacts compiled outside approved workflows.

The deployment stage introduces additional verification layers. Deployment manifests, infrastructure-as-code files, and configuration overlays must present unaltered signatures [9]. Kubernetes admission controllers, cloud deployment policies, or service meshes evaluate whether the workload includes identity tokens, attestation evidence, or verified configuration baselines [10]. If any portion of the runtime payload fails validation—due to configuration drift, expired tokens, or incorrect signature chains—the deployment is denied [11]. At runtime, service-to-service interactions likewise undergo identity-based validation, ensuring that east-west traffic cannot bypass Zero-Trust rules even if already deployed.

Fig. 3 illustrates a decision workflow that governs EHR pipeline transitions. Every step is bounded by verification gates that allow or deny progression based on cryptographic, identity, and policy checks.

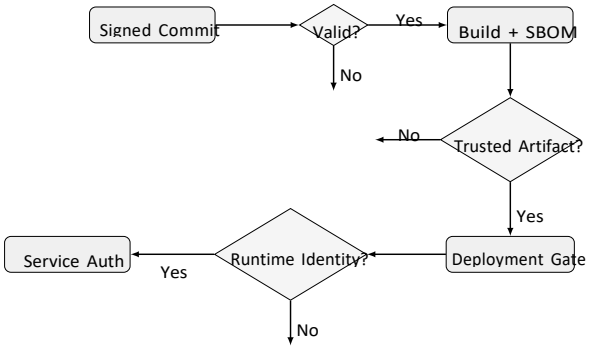

Fig. 3: Continuous verification gates across build, deployment, and service runtime.

Continuous verification ensures that no EHR workload is trusted merely because it has been built or deployed by a recognized system. Each stage enforces independently validated identity and signed provenance, preventing compromised infrastructure components or insider misuse from bypassing security controls. In this model, CI/CD pipelines evolve from delivery automation into dynamic enforcement systems that reject untrusted artifacts, deny unauthorized service interactions, and preserve the confidentiality and integrity of patient records across multi-cloud deployments.

6. POLICY-DRIVEN DEPLOYMENT CONTROLS FOR MULTI-CLOUD EHR WORKLOADS

Zero-Trust DevOps requires deployment decisions to be governed by policy rather than operator discretion, static configuration, or assumed network trust. In a multi-cloud EHR ecosystem, policy serves as a cryptographically enforceable contract that must be satisfied before a workload can be deployed, scaled, migrated, or allowed to communicate with another service [12]. Policy enforcement blocks privileged misuse, eliminates reliance on manual reviews, and provides deterministic outcomes for security-critical decisions.

Deployment policies operate across three dimensions: *identity enforcement*, *artifact provenance*, and *runtime compliance*. Identity enforcement verifies that requestors, deployment agents, and target services are authenticated using short-lived credentials that cannot be reused outside their intended scope. Artifact provenance ensures that containers, IaC manifests, and binaries present a traceable signature chain tied to a recognized build process rather than a potentially compromised registry. Runtime compliance evaluates whether a workload adheres to network segmentation policies, encryption requirements, and secret management rules when the service begins execution or migrates between clouds.

Unlike traditional DevOps workflows, policy in a Zero-Trust environment cannot be partially enforced. Policies must be machine-verifiable, automatically audited, and immutable once applied to production. Cloud orchestration engines, Kubernetes admission controllers, and service meshes assume the role of enforcers rather than passive routers or schedulers. If a workload lacks a valid identity token, contains unsigned components, or presents configuration drift, the deployment is rejected—even when the application would otherwise function correctly. Table II illustrates representative policy rules that govern EHR workload deployment in multi-cloud environments.

TABLE II: Examples of Enforceable Zero-Trust Policies for EHR Deployment

Policy Requirement	Scope	Enforced Action
Short-lived workload identity	Runtime	Deny access if token expires or lacks attestation
Mandatory image signing	CI/CD	Block deployment of unsigned or altered containers
Encryption on all east-west traffic	Network	Force mutual TLS between microservices
Scoped secret access	Secrets	Limit secrets to specific workloads and durations
SBOM presence for arti-facts	Build	Reject artifacts lacking provenance manifest

Immutable audit events	Logging	Prevent tampering with commit or deployment history

Policy-driven enforcement transforms deployment pipelines into autonomous security guardians instead of operational conduits. Rather than trusting the environment or operators, policy requires every workload to prove identity, provenance, and compliance before it is allowed to run. In multi-cloud EHR systems, this approach prevents configuration gaps, deters privilege escalation across providers, and ensures that sensitive workloads cannot execute unless they actively demonstrate continuous alignment with Zero-Trust security guarantees.

7. TRUST ANCHORS AND KEY MANAGEMENT FOR ZERO-TRUST DEVOPS

Zero-Trust DevOps relies on strong cryptographic trust anchors that authenticate identities and validate workload integrity throughout CI/CD and runtime operations. Trust anchors include root certificate authorities, signing keys for supply-chain artifacts, attestation tokens issued to runtime workloads, and hardware-bound secrets stored in secure enclaves or cloud key vaults [13]. These anchors enable deterministic verification of who generated an artifact, how it has changed, and whether the workload requesting execution or network access is genuine.

A critical property of trust anchors in Zero-Trust pipelines is that they cannot remain static. Long-lived secrets, persistent SSH keys, or shared service accounts violate Zero-Trust principles by creating targets that attackers can reuse. Instead, ephemeral identities are issued using short-duration cryptographic credentials tied to contextual conditions, such as workload health, configuration baselines, and signature lineage. Once a deployment ends, the identity expires automatically, preventing misuse even if credentials are intercepted [14].

Encryption must follow the identity rather than the network. Communication between microservices, PACS nodes, database proxies, and analytics engines cannot rely on internal network trust zones. Certificates and rotating tokens issued during build and deployment govern communication paths, ensuring that secure channels are enforced without reliance on static VLANs or VPN boundaries. Similarly, container registries cannot function as trusted sources unless they enforce key expiration, signature scanning, and SBOM compliance at the time of image pull—not merely during storage.

To maintain continuity across multi-cloud EHR deployments, key management must remain portable. Each workload must be able to renew its identity even when migrating across cloud providers or clusters. Trust anchors are therefore distributed through a federated model, where signing policies, rotation intervals, certificate transparency logs, and attestation roots follow the service wherever it executes. This decouples trust from cloud ownership and prevents exploitation of cloud specific weaknesses. When keys, certificates, and attestation evidence evolve with the workload itself, Zero-Trust becomes a property of the identity, not the infrastructure.

8. SECURE OBSERVABILITY AND IMMUTABLE AUDIT TRAILS FOR EHR PIPELINES

Secure observability converts audit data from diagnostic artifacts into tamper-proof evidence. In Zero-Trust DevOps, observability must authenticate not only what occurred, but also who caused it and under which identity. Logging systems cannot treat pipeline events as trustworthy simply because they originate inside the network. Instead, every log entry, audit event, SBOM metadata file, and code-scanning result must carry verifiable signatures tied to specific workload identities.

Immutable audit trails are foundational for legal, clinical, and forensic analysis. When an imaging gateway pushes data to archival storage, the event must be accompanied by an auditable record of the workload identity responsible for the transfer, the cryptographic lineage of the software used to perform the operation, and the configuration baseline active at the time of execution. The archive itself becomes meaningless without defensible provenance: a compliant audit system must demonstrate how a workload was deployed, what policies were enforced, and how the receiving service authenticated itself.

Observability components must also avoid leaking sensitive metadata. Debug logs that expose request contents, patient identifiers, cloud credentials, or routing tokens become attack vectors that violate Zero-Trust discipline [15]. Secure observability filters information based on identity privileges and encrypts metadata before distributing it to collection agents. Even pipeline metrics must be treated as confidential data when they reveal operational patterns or storage movements that could be exploited.

Immutable logging strengthens enforcement by linking forensic evidence back to workload identity. If an attacker compromises part of the pipeline and attempts to modify audit events, they lack access to the signing keys required to forge legitimate events. Similarly, if malicious operators attempt to deploy altered workloads without proper provenance, they cannot generate verifiable trace entries and the deployment will fail policy checks. Secure observability therefore transforms logs into enforcement assets, ensuring that audit data cannot be falsified to hide breaches or insider misuse.

9. FEDERATED ACCESS CONTROL AND CROSS-CLOUD TRUST BOUNDARIES IN EHR SYSTEMS

Multi-cloud EHR infrastructures often span different ownership domains, including regional hospital networks, cloud based analytics platforms, clinical archives, and vendor managed subsystems such as PACS, pharmacy logistics, telemedicine endpoints, or diagnostic decision engines. Each

environment enforces its own identity mechanisms, certificate authorities, routing policies, and secret storage models. When these services interact, trust cannot be inherited from one cloud provider to another; instead, Zero-Trust DevOps requires *federated access control*, where each domain authenticates workloads using portable identities and verifiable credentials without implicitly trusting external frameworks or network origins [16].

Federated access control functions through policy-driven delegation. Rather than allowing cloud A to trust cloud B's identities directly, workloads must carry proof of provenance and attestation that is independently verifiable. This differs from federated login models, which often transfer trust based on identity provider agreements. In Zero-Trust deployment pipelines, federation is rooted in workload cryptographic authenticity rather than inter-organizational agreements. Workloads must present identity tokens issued by trusted signing authorities and must be continuously revalidated as they migrate between clusters or clouds.

Zero-Trust federation also prevents excessive credential sprawl. Without portable identities, multi-cloud deployments often duplicate secrets or API keys across regions, dramatically increasing the attack surface. When access is tied to cryptographic workload tokens that expire and rotate automatically, federated systems eliminate the need for static credentials that persist across domains. This reduces the risk of unauthorized replication, insider misuse, or credential harvesting.

Cross-cloud trust boundaries must remain intentionally hard. Instead of enabling shared networks or privileged VPN tunnels, Zero-Trust enforces mutual authentication between services even when they are part of the same EHR ecosystem. When imaging archives request anatomy-rendering services, when telehealth systems transmit clinical documents to EHR decision engines, or when research workloads consume deidentified patient data, each request must validate identity, policy compliance, encryption, and runtime attestation before establishing a communication channel [17]. Federation therefore becomes a property of cryptographic confidence, not shared infrastructure.

In this model, the cloud becomes merely a hosting substrate, not the security authority. Trust follows workloads through cryptographic proofs rather than through ownership geography. As a result, federated access control prevents the most common class of multi-cloud breaches—those caused by overprivileged networks, shared private routes, and inherited trust models. Zero-Trust federation allows distributed EHR services to collaborate securely without diluting accountability or weakening cloud separation boundaries.

10. CASE STUDY: ZERO-TRUST CI/CD FOR CLINICAL IMAGING SERVICES

Medical imaging systems rely on high-resolution DICOM objects, large storage transfers, modality-specific metadata, and time-synchronized workflows between radiology workstations, PACS archives, and EHR platforms. Multi-cloud deployments are increasingly used to support elastic compute for image rendering, AI-assisted diagnostics, and archival compression, yet they confront a unique security challenge: large imaging files frequently transit between autonomous clusters where implicit network trust or shared credentials may be mistakenly assumed [18]. A breach or misconfiguration in one cluster could expose radiological identifiers, patient annotations, anatomical studies, or reconstructed 3-D models if Zero-Trust controls are not embedded into the CI/CD pipeline and runtime environment.

To secure these workflows, Zero-Trust DevOps enforces identity validation for imaging services before any compute or storage resources are allocated. GPU nodes must authenticate as workload identities rather than machines; PACS gateways must verify mutual TLS before exporting image frames; and transcoding services cannot inherit persistent keys during deployments. Continuous verification at build time ensures that DICOM parsers, anonymizers, and reconstruction algorithms are cryptographically signed and originate from approved registries. Image-processing containers cannot be deployed until they produce trusted SBOM artifacts, validated source identity, and policy-defined configuration baselines. Runtime identity segmentation prevents malicious components from eavesdropping on image transfers, altering metadata, or spoofing modality origins.

Fig. 4 illustrates how Zero-Trust controls regulate image processing across build, deployment, and inter-service transfer stages. Each phase enforces identity and provenance checks

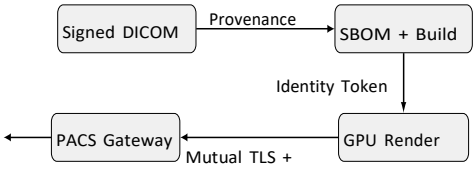

Fig. 4: Zero-Trust enforcement across imaging pipeline stages.

before permitting any access to sensitive imaging data or PACS-linked endpoints.

Zero-Trust CI/CD protects clinical imaging workloads by transforming image-processing services into authenticated, policy-verified entities. Even if a compromised registry component or rogue GPU node attempts to perform unauthorized rendering or archival interactions, continuous verification blocks access because the workload identity, provenance chain, and runtime attestation do not satisfy required policies. Rather than defending the network perimeter, Zero-Trust secures the imaging workflow itself, ensuring that only validated services can transform, transport, or store radiological data across distributed multi-cloud infrastructures.

11. DISCUSSION AND LIMITATIONS

Zero-Trust DevOps substantially improves the defensive posture of multi-cloud EHR deployments by removing implicit trust, enforcing identity segmentation, and requiring workload verification before execution. These controls provide a consistent security foundation that is portable across cloud vendors, deployment regions, and heterogeneous compute resources such as GPUs, PACS gateways, or analytics clusters. Rather than relying on network location or administrative privileges, Zero-Trust ties authorization to workload provenance, runtime identity, and verifiable policy compliance. This enables EHR systems to scale without weakening their defensive posture, even when workloads dynamically migrate or interact across clouds.

However, integrating Zero-Trust into DevOps introduces operational and engineering trade-offs. Continuous verification increases build and deployment latency when heterogeneous signing, SBOM generation, image scanning, and policy evaluation are executed across multiple clusters. Identity segmentation requires domain-aligned policies that must evolve as workloads change, especially when clinical services introduce new modalities, external integrations, or vendor applications. The infrastructure needed for attestation, mutual authentication, and immutable logging must be deployed consistently across all cloud providers, increasing operational overhead and requiring consistent identity governance. Fig. 5 summarizes the core benefits and constraints of Zero-Trust DevOps.

Zero-Trust provides a strong security framework, but its effectiveness depends on the discipline and consistency with which it is implemented. Organizations must not only integrate continuous verification into CI/CD but also maintain policy lifecycles, attestation trust anchors, key rotation mechanisms, and audit pipelines. The model is not a one-time configuration; it requires operational maturity, continuous policy review, and

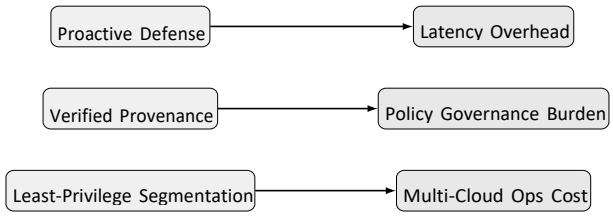

Fig. 5: Trade-offs in Zero-Trust DevOps for EHR systems.

cross-team collaboration between security engineers, DevOps operators, and clinical IT stakeholders. When these responsibilities are met, Zero-Trust yields resilient, scalable, and defensible EHR systems in multi-cloud environments, but if neglected, its protections can erode, creating a misleading sense of safety without true enforcement.

12. CONCLUSION

Zero-Trust DevOps transforms EHR system security by eliminating inherited trust, enforcing workload identity, and requiring continuous verification of both software and infrastructure. Rather than defending static boundaries, this model authenticates every build artifact, deployment request, runtime interaction, and cross-cloud transfer based on cryptographic provenance and policy compliance. Each workload must continuously prove who it is, how it was created, and whether it is still compliant with organizational expectations. Identity segmentation, policy-driven enforcement, cryptographic trust anchors, and immutable observability together create a defensive execution environment that protects clinical workloads even when cloud locations, hardware resources, or orchestration strategies change.

As EHR infrastructures expand across multiple clouds, federated control becomes essential. Trust cannot depend on shared networks, vendor contracts, or regional routing

Domains; security must follow the workload itself. Portable identity, short-lived credentials, automated attestation, and continuous deployment governance ensure that cloud resources act purely as compute substrates rather than implicit trust providers. In this model, movement between clouds does not dilute accountability or increase risk, because every environment must independently verify identity, provenance, and policy adherence.

Zero-Trust DevOps also redefines pipeline responsibilities. CI/CD systems evolve from delivery automation into active enforcement engines. Builds cannot produce unverified artifacts, deployment workflows cannot propagate misconfigured workloads, and runtime services cannot communicate without mutual authentication. Observability transitions from diagnostics to evidence generation, producing tamper-resistant audit artifacts that defend clinical decisions, operational investigations, and regulatory obligations. Instead of assuming correctness after deployment, Zero-Trust protects EHR data throughout the lifecycle, making violations detectible, preventable, and attributable.

However, Zero-Trust is not an installable feature. It is a rigorous operational discipline that requires policy governance, identity hygiene, continuous key rotation, secure observability, domain-specific segmentation, and long-term institutional commitment. Hospitals, vendors, and research partners must align not only on technical controls but also on governance authority, operational practices, and shared definitions of trust. When implemented consistently, Zero-Trust enables scalable and defensible multi-cloud EHR ecosystems where compliance, clinical safety, and operational agility are not competing goals, but mutually reinforcing outcomes of continuous verification.

13. REFERENCES

[1] M. A. Akbar, K. Smolander, S. Mahmood,and A. Alsanad, "Toward successful devsecops in software development organizations: A decision-making framework," *Information and*

Software Technology, vol. 147, p. 106894, 2022. [Online]. Available: https://doi.org/10.1016/ j.infsof.2022.106894

[2] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-207, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207

[3] I. J. Kanani, "Implementing devsecops in cloud-native workflows," *World Journal of Advanced Research and Reviews*, vol. 16, no. 3, pp. 532–541, 2022. [Online]. Available: https://wjarr.com/content/ implementing-devsecops-cloud-native-workflows

[4] R. Kumar and R. Goyal, "Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc)," *Computers & Security*, vol. 97, p. 101967, 2020. [Online]. Available: https://doi.org/10.1016/j.cose.2020.101967

[5] "Software supply chain and devops security practices: Devsecops for secure software development," National Cybersecurity Center of Excellence (NCCoE), NIST, Tech. Rep., 2022, project Description, National Institute of Standards and Technology. [Online]. Available: https://www.nccoe.nist.gov/sites/default/files/2022-11/ dev-sec-ops-project-description-final.pdf

[6] C. S. I. G. P. S. S. Council, "Information supplement: Pci ssc cloud computing guidelines, v3.0," PCI Security Standards Council, Tech. Rep., 2017. [Online]. Available: https://www.pcisecuritystandards.org/ pdfs/PCI SSC Cloud Guidelines v3.pdf

[7] A. Khurshid, R. Alsaaidi, M. Aslam, and S. Raza, "Eu cybersecurity act and iot certification: Landscape, perspective and a proposed template scheme," *IEEE Access*, vol. 10, pp. 129932–129948, 2022.

[8] "Zero trust in healthcare," U.S. Department of Health and Human Services, HHS Cybersecurity Program, Tech. Rep., October 2020, hC3 Cybersecurity Report #202010011030. [Online]. Available: https://www.hhs.gov/sites/default/files/zero-trust.pdf

[9] F. Ullah, A. J. Raft, M. Shahin, M. Zahedi, and M. A. Babar, "Security support in continuous deployment pipeline," in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, 2017, pp. 57–68. [Online]. Available: https://doi.org/10.5220/0006318200570068

[10] R. Chandramouli and Z. Butcher, "Building secure microservices-based applications using service-mesh architecture," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-204A, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP. 800-204A

[11] R. Chandramouli, Z. Butcher, and A. Chetal, "Attribute-based access control for microservices-based applications using a service mesh," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-204B, 2021. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-204B

[12] A. Prasad, V. S. Bhatia, N. Tyagi, A. Sengupta, and H. Singh, "Zero trust in multi-cloud environments: A framework for identity-aware microsegmentation," Jan. 2022, sSRN Working Paper, Date Written: January 27, 2022. [Online]. Available: https://ssrn.com/abstract=5285824

[13] R. Chandramouli and Z. Butcher, "A zero trust architecture model for access control in cloud-native applications in multi-cloud environments," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-207A, 2023. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207A

[14] "Cloud security for healthcare services," European Union Agency for Cybersecurity (ENISA), Tech. Rep., January 2021.

[15] V. Sadhwani, "Cloud container security' next move," Master's thesis, Harrisburg University of Science and Technology, Harrisburg, PA, 2022, master's Thesis. [Online]. Available: https://digitalcommons. harrisburgu.edu/csms dandt/3

[16] R. Nadipalli, "Cloud-native devsecops: A framework for secure continuous delivery," *International Journal of Computing and Engineering*, vol. 3, no. 2, pp. 1–9, 2023. [Online]. Available: https://doi.org/10.47941/ijce.3104

[17] R. Ganiga, R. M. Pai, M. M. Manohara Pai, and R. K. Sinha, "Security framework for cloud based electronic health record (ehr) system," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, pp. 455–466, 2020. [Online]. Available: https://doi.org/10.11591/ijece.v10i1.pp455-466

[18] M. Mehrtak, S. Seyedalinaghi, M. Mohsseni Pour, T. Noori, A. Karimi, A. Shamsabadi *et al.*, "Security challenges and solutions using healthcare cloud computing," *Journal of Medicine and Life*, vol. 14, no. 4, pp. 448– 461, 2021. [Online]. Available: https://doi.org/10.25122/jml-2021-0100