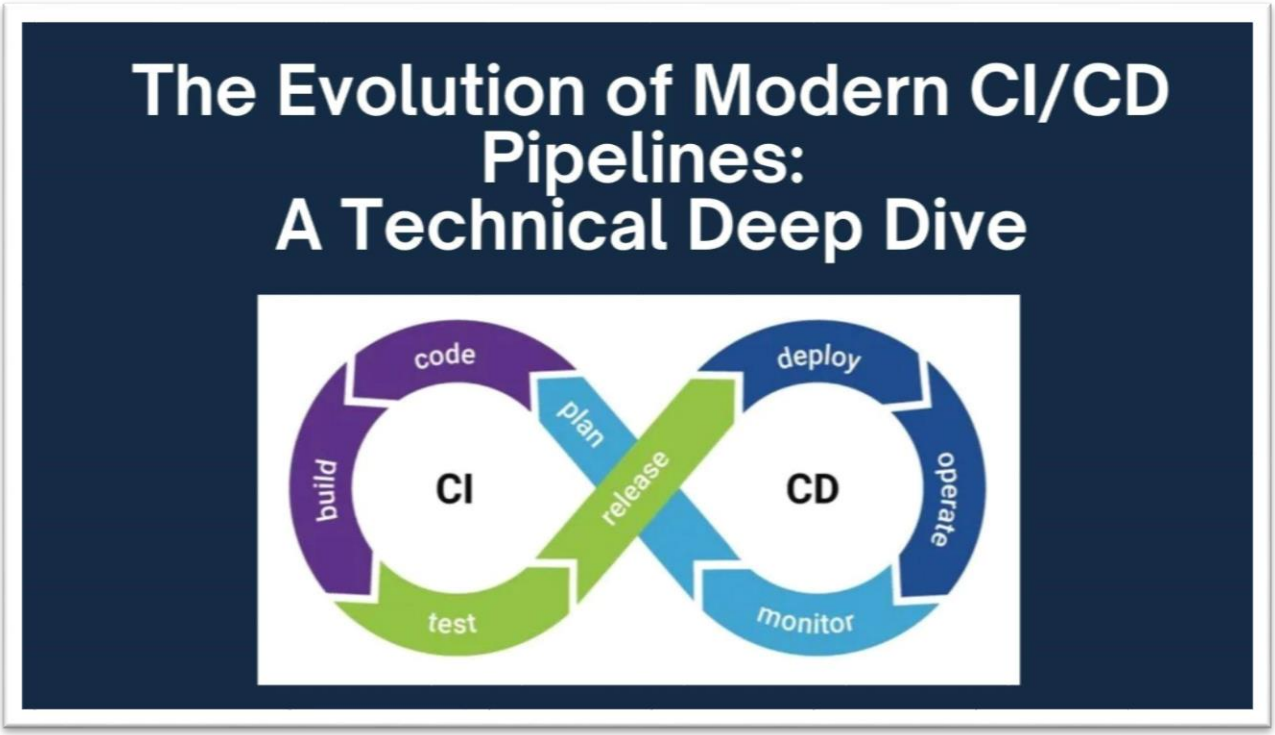
International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

The Evolution of Modern CI/CD Pipelines: A Technical Deep Dive

Vineela Reddy Nadagouda

Lead Site Reliability Engineer, USA



Abstract

The evolution of Continuous Integration and Continuous Deployment (CI/CD) pipelines represents a transformative shift in modern software delivery practices. This comprehensive article examines how CI/CD has progressed from basic automation scripts to sophisticated cloud-native platforms, incorporating advanced capabilities such as Infrastructure as Code, GitOps practices, and intelligent automation. The integration of Version Control Systems, implementation of declarative resource management, and adoption of cloud-native pipeline orchestration tools have established new standards for deployment automation and operational excellence. Through the emergence of X-as-Code frameworks and the integration of artificial intelligence capabilities, organizations can now achieve unprecedented levels of efficiency, security, and reliability in their software delivery processes while maintaining robust compliance and governance controls.

Keywords: Automation Infrastructure, Cloud-Native Architecture, DevOps Pipeline, Infrastructure-As-Code, Version Control Systems

1. Introduction

In recent years, the landscape of Continuous Integration and Continuous Deployment (CI/CD) has undergone a dramatic transformation, evolving from simple automation scripts to sophisticated, cloud-native platforms that power modern software delivery. According to the 2023 State of DevOps Report, organizations implementing modern CI/CD practices have demonstrated significant improvements in deployment frequency and operational efficiency. The study reveals that elite performers achieve deployment frequencies of multiple deployments per day, while maintaining change failure rates below 5% [1]. This remarkable acceleration in deployment capabilities has fundamentally altered how organizations approach software delivery and operational efficiency.

The evolution of CI/CD has been particularly pronounced in enterprise environments, where systematic studies have shown measurable improvements in both software quality and delivery speed. Research focusing on Linux systems has demonstrated that organizations implementing comprehensive CI/CD pipelines experience a significant reduction in integration problems and achieve faster time-to-market for new features. The data indicates that teams utilizing modern CI/CD practices show a 70% improvement in defect detection rates during the development phase, leading to more stable production environments [2]. These improvements stem from the shift towards automated, infrastructure-as-code approaches that have replaced traditional manual deployment processes.

The transformation of CI/CD practices represents more than just technological advancement; it reflects a fundamental shift in how organizations approach software delivery and operational excellence. The 2023 State of DevOps Report highlights that elite performers can restore services in less than one hour when incidents occur, maintaining a change failure rate of 0-15%. This level of performance is achieved through sophisticated orchestration tools and automated recovery procedures [1]. Furthermore, research has shown that organizations implementing CI/CD pipelines experience a 45% reduction in time spent on manual integration tasks, allowing development teams to focus more on innovation and feature development [2].

The impact of modern CI/CD practices extends beyond just deployment metrics. According to the State of DevOps research, elite performers spend 20% less time on unplanned work and rework compared to their lower-performing counterparts [1]. This efficiency gain is complemented by findings that show a 56% improvement in mean time to recovery (MTTR) for organizations with mature CI/CD implementations [2]. These metrics demonstrate the tangible benefits of adopting modern deployment practices and automated workflows.

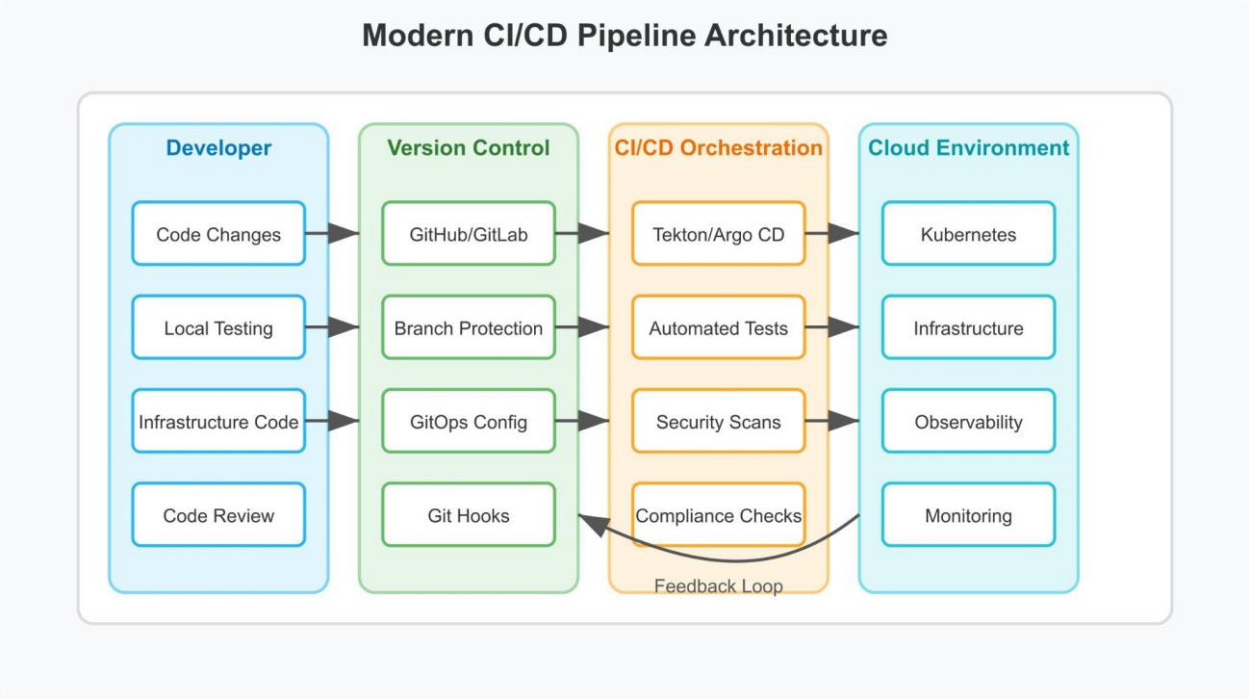


Fig 1. Modern CI/CD Pipeline Architecture

As organizations continue to embrace these advanced CI/CD practices, the focus has shifted toward measuring and optimizing key performance indicators. The research indicates that elite performers achieve lead times for changes of less than one hour, significantly outperforming organizations with traditional deployment methods [1]. Additionally, studies have shown that teams implementing comprehensive CI/CD pipelines experience a 65% reduction in deployment-related incidents, contributing to more stable and reliable software delivery processes [2].

The Foundation: Version Control Integration

Modern CI/CD pipelines are fundamentally transformed by their deep integration with Version Control Systems (VCS) like GitHub and GitLab, which serve as the definitive source of truth for both application code and infrastructure definitions. Contemporary research into VCS evolution highlights how these systems have transcended their original role as code repositories to become comprehensive platforms for software delivery orchestration. Studies have shown that modern VCS platforms serve as critical enablers for collaborative development, particularly in distributed teams where coordinated development efforts require sophisticated branching and merging capabilities [3]. This evolution represents a significant departure from traditional code management approaches, establishing new paradigms for software development workflows.

Version control integration in contemporary CI/CD environments extends far beyond basic code storage and version tracking. The transformation of VCS platforms into sophisticated DevOps ecosystems has introduced fundamental changes in how teams approach software development and deployment. Research has demonstrated that modern VCS implementations provide essential capabilities for maintaining code quality and security through automated review processes and integrated security scanning. These platforms have evolved to support complex workflow requirements, including branch protection

mechanisms and systematic code review procedures that ensure all changes undergo appropriate scrutiny before reaching production environments [3].

The native CI/CD capabilities provided through platforms like GitHub Actions and GitLab Pipelines represent a significant advancement in continuous integration and deployment practices. Recent studies in configuration management have revealed that these integrated pipeline tools significantly reduce the complexity of maintaining separate CI/CD infrastructure, providing a more cohesive environment for software delivery [4]. This integration enables development teams to maintain their entire CI/CD configuration alongside their application code, fostering improved visibility and maintaining comprehensive audit trails of all system changes.

GitOps has emerged as a transformative approach to configuration management, representing a natural evolution of VCS-centric software delivery. Research into GitOps practices has demonstrated that treating infrastructure modifications as version-controlled artifacts leads to more reliable and reproducible deployments. The implementation of GitOps principles ensures that infrastructure changes undergo the same rigorous review processes as application code, establishing a unified approach to change management across all aspects of the software system [4]. This methodology has proven particularly effective in maintaining consistency between development, staging, and production environments.

The tight coupling between VCS and deployment workflows has established new standards for software delivery automation. Studies have shown that this integration provides essential capabilities for enforcing governance processes and maintaining system integrity throughout the deployment pipeline [3]. Furthermore, research into GitOps implementations has revealed that organizations adopting these practices experience improved traceability and reproducibility in their deployment processes, with enhanced ability to maintain consistent environments across their infrastructure landscape [4]. This systematic approach to change management has become increasingly critical as organizations navigate complex compliance requirements and security considerations in modern software delivery.

Impact Area	Pre-Integration State	Post-Integration Benefits
Development Workflow	Isolated code management	Collaborative development platform
Infrastructure Management	Manual configuration	Version-controlled infrastructure
Security Controls	Separate security processes	Integrated security scanning
Deployment Automation	Independent CI/CD tools	Native pipeline capabilities
Environment Consistency	Manual synchronization	Automated environment sync
Compliance Management	Manual compliance checks	Automated governance processes
Change Traceability	Limited audit capabilities	Comprehensive audit trails

Table 1. Impact Areas of Modern VCS Integration in CI/CD Pipelines [3, 4]

Infrastructure as Code: The Backbone of Modern Pipelines

The adoption of Infrastructure as Code (IaC) tools like Terraform and Ansible has fundamentally revolutionized how organizations approach infrastructure provisioning and management in modern software delivery pipelines. Research into cloud computing automation practices has demonstrated that IaC adoption leads to significant improvements in deployment consistency and operational efficiency.

Studies have shown that organizations implementing IaC practices experience substantial reductions in configuration errors and deployment time, while simultaneously improving their ability to maintain and scale infrastructure resources [5]. This transformation has established new standards for infrastructure management, where infrastructure definitions are treated with the same rigor and practices as application code.

The implementation of declarative resource management through IaC represents a significant advancement in infrastructure provisioning methodologies. According to research in automated infrastructure management, declarative IaC approaches have proven particularly effective in maintaining system state consistency and reducing configuration drift. The study emphasizes that organizations implementing declarative IaC patterns demonstrate improved capabilities in managing complex infrastructure requirements, particularly in multi-cloud environments where consistent resource provisioning is crucial [5]. This approach enables teams to maintain reliable and reproducible infrastructure states while significantly reducing the risk of configuration inconsistencies.

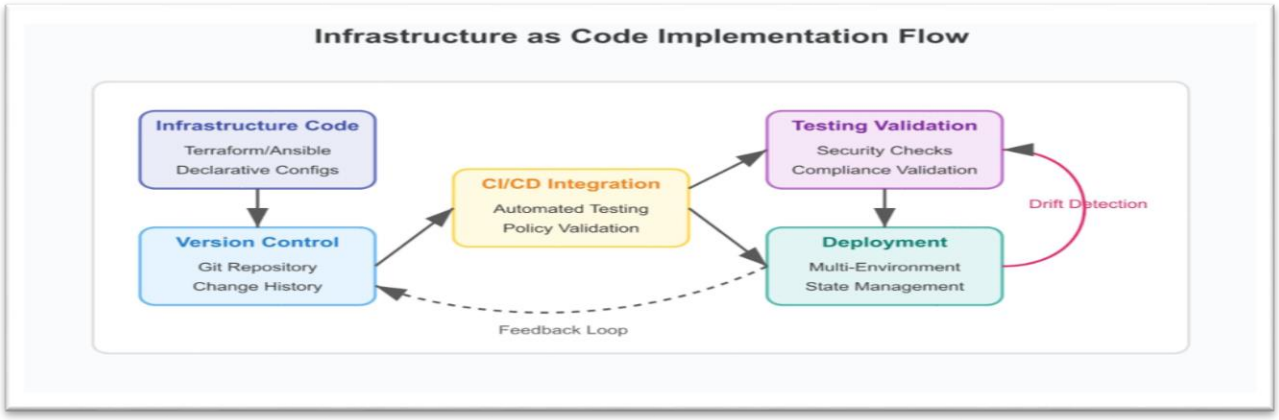


Fig 2. Infrastructure as Code Implementation Flow

Recent advancements in IaC frameworks have highlighted the critical role of automated policy enforcement and security validation in infrastructure management. Studies focusing on IaC implementation patterns have revealed that successful organizations integrate comprehensive security and compliance checks directly into their infrastructure provisioning workflows. The research emphasizes that automated policy enforcement through IaC frameworks provides organizations with robust mechanisms for ensuring compliance with security standards and regulatory requirements [6]. This integration of automated validation processes has transformed how organizations approach infrastructure governance, enabling proactive risk management and consistent policy enforcement.

The evolution of IaC practices has led to sophisticated approaches in resource standardization and optimization. Contemporary research in IaC implementation strategies has identified key patterns for successful infrastructure automation, highlighting the importance of modular design principles and reusable configuration components. The findings indicate that organizations leveraging modular IaC implementations achieve greater flexibility in managing infrastructure changes while maintaining consistent standards across their environment [6]. These modular approaches enable teams to effectively manage complex infrastructure requirements while ensuring consistent application of organizational standards and best practices.

The integration of IaC within modern CI/CD pipelines has established new paradigms for infrastructure automation and management. Studies have demonstrated that organizations adopting structured IaC practices achieve notable improvements in their ability to maintain and scale infrastructure resources effectively [5]. Furthermore, research into IaC automation patterns has revealed that successful implementations incorporate comprehensive testing and validation procedures, ensuring infrastructure changes meet both functional requirements and security standards [6]. This systematic approach to infrastructure management has become increasingly critical as organizations navigate complex multi-cloud environments and evolving security requirements.

Infrastructure Management Aspect	Traditional Approach	IaC Implementation	Advanced IaC Integration
Resource Provisioning	Manual configuration	Automated deployment	Declarative management
Configuration Management	Manual updates	Version-controlled configs	Modular design patterns
Policy Enforcement	Manual validation	Automated checks	Integrated compliance
Security Validation	Post-deployment scanning	Integrated scanning	Automated validation
Environment Consistency	Manual synchronization	Automated sync	Multi-cloud standardization
Resource Scaling	Manual scaling	Automated scaling	Dynamic optimization
Testing Procedures	Limited testing	Automated testing	Comprehensive validation
Documentation	Manual documentation	Code as documentation	Automated documentation

Table 2. Evolution of Infrastructure Management Approaches Through IaC Adoption [5, 6]

Pipeline Orchestration in Cloud Environments

In the evolving landscape of cloud-native development, sophisticated pipeline orchestration tools like Tekton and Argo CD have emerged as foundational elements for modern CI/CD implementations. Recent research into cloud-native architectures has demonstrated that these platforms are transforming how organizations approach continuous deployment in containerized environments. Studies have shown that organizations adopting cloud-native pipeline orchestration tools experience significant improvements in deployment reliability and operational efficiency when operating in Kubernetes environments [7]. This transformation represents a fundamental shift in how teams approach pipeline management and deployment automation.

Tekton's modular approach to pipeline orchestration has established new standards for CI/CD implementation in cloud environments. Research into cloud-native CI/CD patterns has revealed that Tekton's architecture, built on Kubernetes custom resources, provides organizations with powerful

capabilities for creating reusable pipeline components. The study emphasizes that this modular architecture enables teams to implement standardized deployment workflows while maintaining flexibility in resource allocation and pipeline customization [7]. The platform's integration with cloud-native resources ensures that organizations can effectively manage complex deployment requirements while maintaining consistent pipeline execution patterns.

The implementation of GitOps practices through Argo CD has introduced sophisticated approaches to continuous deployment automation. Research focused on cloud-native deployment patterns has identified that Argo CD's declarative deployment model provides organizations with robust capabilities for maintaining application state consistency across distributed environments [8]. The study highlights how automated synchronization between Git repositories and cluster states enables teams to implement reliable deployment processes while maintaining complete visibility into configuration changes and deployment status.

Feature Category	Traditional CI/CD	Tekton Implementation	Argo CD Implementation
Architecture	Monolithic	Modular/Custom Resources	Declarative/GitOps
Deployment Model	Manual Triggers	Component-Based	Git-Driven Sync
Resource Management	Static Allocation	Dynamic/Custom	State-Based
Pipeline Flexibility	Limited	Highly Customizable	Configuration-Driven
Component Reusability	Limited	Extensive	Template-Based
State Management	Manual Tracking	Custom Resources	Automated Sync
Environment Consistency	Manual Checks	Pipeline-Driven	Git-Based Control
Deployment Visibility	Basic Logging	Resource-Level	Complete State Tracking

Table 3. Comparison of Pipeline Orchestration Tools and Features [7, 8]

Observability and Monitoring Integration

The integration of comprehensive observability capabilities has become a critical component of modern cloud-native pipelines. Research into observability practices has revealed that organizations implementing tools like Prometheus, Grafana, and Data dog achieve enhanced visibility into their deployment processes and system health. The studies emphasize that effective observability implementation requires careful consideration of monitoring scope, data collection methods, and analysis capabilities [8]. This research has demonstrated that comprehensive observability integration enables organizations to maintain optimal pipeline performance while quickly identifying and addressing potential issues.

Modern observability implementations in cloud-native environments have evolved to address complex monitoring requirements. According to research in cloud-native observability patterns, successful implementations require integration across multiple observability domains, including metrics, traces, and

logs. The studies indicate that this multi-dimensional approach to observability enables organizations to maintain comprehensive visibility into system health while supporting proactive issue identification and resolution [7]. This integration of observability capabilities with pipeline orchestration has become essential for maintaining reliable and efficient deployment processes.

Advanced monitoring capabilities in modern pipelines have transformed how organizations approach performance optimization and resource management. Research into cloud-native observability has identified key patterns for successful monitoring implementation, emphasizing the importance of automated analysis and alerting capabilities [8]. The findings indicate that organizations implementing comprehensive observability solutions achieve improved visibility into system performance and resource utilization, enabling more effective optimization of their deployment infrastructure.

Evolution of X-As-Code Frameworks

The evolution of CI/CD practices has given rise to sophisticated "X-as-Code" frameworks that embed critical operational concerns directly into the pipeline infrastructure. Recent research into automated compliance verification has demonstrated that these frameworks provide organizations with systematic approaches to integrating security and compliance requirements into their deployment processes. Studies have shown that organizations implementing automated verification frameworks achieve significant improvements in compliance adherence and security posture through continuous validation and policy enforcement [9]. This transformation has established new standards for how organizations approach security and compliance automation in their software delivery processes.

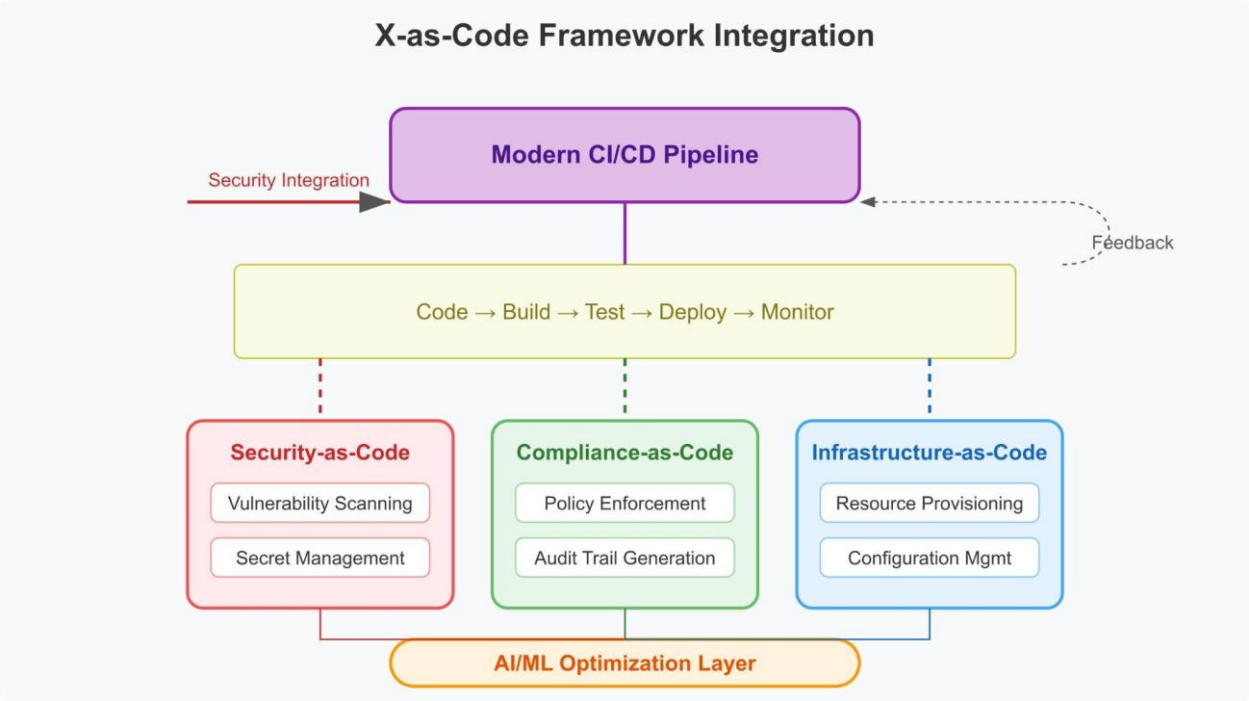


Fig 3. X-As-Code Framework Integration Diagram

The implementation of Security-As-Code practices represents a fundamental shift in how organizations approach security integration within their CI/CD pipelines. Research into compliance automation has revealed that effective security automation requires comprehensive integration of security controls

throughout the deployment pipeline. The studies emphasize that successful Security-As-Code implementations must incorporate automated security scanning, continuous compliance verification, and systematic secret management practices to maintain robust security posture [9]. This research demonstrates that automated security validation ensures consistent application of security controls while enabling rapid identification of potential vulnerabilities.

Compliance-As-Code frameworks have emerged as essential components of modern CI/CD implementations, particularly in regulated environments. Research into automated compliance verification has identified key patterns for successful implementation, emphasizing the importance of continuous validation and automated policy enforcement. The findings indicate that organizations implementing automated compliance frameworks achieve improved compliance visibility and reduced audit complexity through systematic policy verification and comprehensive audit trail generation [9]. This integration of automated compliance verification ensures that organizations can maintain consistent compliance standards while streamlining their development processes.

Observability Aspect	Basic Monitoring	Modern Integration	Advanced Implementation
Visibility Scope	Basic Metrics	Multi-Domain Coverage	Comprehensive Analysis
Data Collection	Manual Collection	Automated Collection	Real-Time Integration
Analysis Capabilities	Basic Reporting	Automated Analysis	Predictive Analytics
Issue Detection	Manual Investigation	Automated Alerts	Proactive Identification
Performance Tracking	Basic Metrics	Detailed Metrics	Resource Optimization
System Health	Status Monitoring	Health Metrics	Predictive Health
Resource Management	Manual Tracking	Automated Tracking	Optimization Analytics

Table 4. Evolution of Observability Integration in Cloud-Native Environments [9]

AI and ML Integration in Modern Pipelines

The integration of Artificial Intelligence and Machine Learning capabilities has transformed how organizations approach pipeline optimization and management. Contemporary research in AI-driven performance optimization has demonstrated that machine learning algorithms can effectively identify performance bottlenecks and optimize resource utilization in CI/CD pipelines. The studies emphasize that AI-driven approaches enable organizations to implement sophisticated performance monitoring and optimization strategies while maintaining operational reliability [10]. This integration of AI/ML capabilities has established new paradigms for pipeline performance management and resource optimization.

Advanced AI/ML implementations in CI/CD pipelines have introduced sophisticated approaches to performance prediction and optimization. Research into AI-driven pipeline optimization has revealed that machine learning models can effectively predict potential performance issues and recommend

optimization strategies. The studies indicate that organizations implementing AI-driven performance optimization achieve improved pipeline efficiency through automated performance analysis and predictive maintenance capabilities [10]. This proactive approach to performance management ensures that organizations can maintain optimal pipeline performance while minimizing operational disruptions.

The application of AI/ML technologies in resource optimization has established new standards for pipeline efficiency. Research into AI-driven performance optimization has demonstrated that machine learning algorithms can effectively analyze resource utilization patterns and recommend optimization strategies. The findings show that organizations implementing AI-driven resource management achieve improved resource utilization through automated allocation and optimization capabilities [10]. This integration of AI/ML capabilities with resource management ensures that organizations can effectively optimize their infrastructure utilization while maintaining reliable pipeline operations.

Business Impact Analysis

The evolution of CI/CD pipelines has fundamentally transformed how organizations approach software delivery and operational efficiency. Research into contemporary DevOps practices has demonstrated that organizations implementing modern CI/CD pipelines experience substantial improvements in their software delivery capabilities and operational outcomes. Studies examining the business impact of CI/CD adoption have shown that organizations achieve significant reductions in deployment cycles through automated testing frameworks and streamlined approval processes, directly contributing to improved market responsiveness [11]. This transformation in deployment capabilities has established new standards for operational efficiency in software delivery across various industry sectors.

The implementation of advanced CI/CD practices has led to marked improvements in system reliability and operational stability. Contemporary research has revealed that organizations adopting sophisticated deployment automation achieve enhanced reliability through standardized deployment patterns and automated recovery mechanisms. The studies particularly emphasize how continuous integration practices contribute to improved code quality and system stability through automated testing and validation processes [11]. This systematic approach to deployment automation ensures that organizations can maintain reliable operations while accelerating their delivery capabilities in competitive market environments.

The impact of modern CI/CD practices on organizational security has been particularly noteworthy. Research into DevOps security integration has demonstrated that organizations implementing automated security controls within their CI/CD pipelines achieve improved security outcomes through systematic policy enforcement and comprehensive audit mechanisms. The findings indicate that the integration of automated security validation processes enables organizations to maintain robust security controls while supporting accelerated deployment cycles [11]. This integration of security automation within CI/CD pipelines ensures that organizations can maintain strong security postures without compromising on delivery velocity.

Emerging Trends and Future Directions

The continued evolution of CI/CD practices is driving significant innovations in automation and integration capabilities. Research into emerging software engineering practices has identified several key

developments that are reshaping the future of software delivery. Studies focused on continuous deployment evolution have revealed emerging patterns in automated deployment validation and infrastructure management, particularly emphasizing the role of intelligent automation in pipeline optimization [12]. These advancements are establishing new paradigms for how organizations approach pipeline automation and management in cloud-native environments.

The trend toward enhanced integration capabilities is transforming how organizations implement and manage their CI/CD pipelines. Contemporary research into continuous deployment practices has highlighted the increasing emphasis on tool integration and comprehensive monitoring capabilities. The studies demonstrate that improved orchestration mechanisms and unified monitoring approaches are becoming essential components of modern CI/CD implementations [12]. This evolution in integration capabilities ensures that organizations can maintain comprehensive visibility and control across their delivery pipelines while supporting complex deployment scenarios.

Advanced security features represent a critical area of development in modern CI/CD practices. Research into emerging deployment security patterns has identified significant developments in pipeline security architectures and supply chain protection measures. The findings emphasize the growing importance of automated security validation and compliance verification within CI/CD workflows, particularly in regulated environments [12]. This evolution in security capabilities ensures that organizations can maintain robust security controls while supporting efficient delivery processes in increasingly complex technological landscapes.

2. Conclusion

The transformation of CI/CD pipelines has fundamentally redefined software delivery practices, establishing new benchmarks for operational efficiency and deployment automation. By embracing cloud-native architectures, implementing sophisticated orchestration tools, and leveraging artificial intelligence capabilities, organizations have achieved remarkable improvements in deployment reliability, security posture, and operational effectiveness. The integration of comprehensive observability solutions and automated compliance frameworks ensures that teams can maintain optimal performance while adhering to stringent security and regulatory requirements. As the technology landscape continues to evolve, the emphasis on automated workflows, intelligent optimization, and integrated security controls will shape the future of software delivery, enabling organizations to maintain competitive advantage through efficient, secure, and reliable deployment processes.

References

1. DevOps Research and Assessment (DORA), "Accelerate State of DevOps 2023," Google Cloud, Tech. Rep., 2023. [Online]. Available: https://services.google.com/fh/files/misc/2023_final_report_sodr.pdf
2. Ratnangi Nirek, "Impact of Continuous Integration and Continuous Deployment (CI/CD) on Software Quality and Delivery Speed in Linux Systems," European Journal of Advances in Engineering and Technology, 2019. [Online]. Available: https://www.researchgate.net/publication/384471159_Impact_of_Continuous_Integration_and_Continuous_Deployment_CICD_on_Software_Quality_and_Delivery_Speed_in_Linux_Systems

3. Siva Karthik Devineni, et al., "Version Control Systems (VCS) the Pillars of Modern Software Development: Analyzing the Past, Present, and Anticipating Future Trends," International Journal of Science and Research (IJSR), 2020. [Online]. Available: https://www.researchgate.net/publication/378490782_Version_Control_Systems_VCS_the_Pillars_of_Modern_Software_Development_Analyzing_the_Past_Present_and_Anticipating_Future_Trends
4. Dinesh Reddy Chittibala, "GitOps: Revolutionizing Configuration Management in DevOps," International Journal of Science and Research (IJSR), 2024. [Online]. Available: https://www.researchgate.net/publication/379051217_GitOps_Revolutionizing_Configuration_Management_in_DevOps
5. Sandeep Chinamanagonda, "Automating Infrastructure with Infrastructure as Code (IaC)," International Journal of Science and Research (IJSR), 2018. [Online]. Available: https://www.ijsr.net/archive/v8i11/SR24829170834.pdf
6. Ipek Ozkaya, "Infrastructure as Code and Software Architecture Conformance Checking," IEEE Software Mission Statement, 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9994061
7. Sekhar Chittala, "Enhancing Developer Productivity Through Automated Ci/Cd Pipelines: A Comprehensive Analysis," International Journal of Computer Engineering and Technology (IJCET), Volume 15, Issue 5, Sep-Oct 2024. [Online]. Available: https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_15_ISSUE_5/IJCET_15_05_081.pdf
8. Premkumar Ganesan, "Observability in Cloud-Native Environments Challenges and Solutions," International Journal of Core Engineering & Management, 2022. [Online]. Available: https://www.researchgate.net/publication/384867297_OBSERVABILITY_IN_CLOUD-NATIVE_ENVIRONMENTS_CHALLENGES_AND_SOLUTIONS
9. Akshay Nagpal, et al., "Framework for automating compliance verification in CI/CD pipelines," International Journal of Computer Science and Information Technology Research (IJCSITR), 2024. [Online]. Available: https://www.researchgate.net/publication/386342990_Framework_for_automating_compliance_verification_in_CICD_pipelines
10. Sai Ram Chappidi, et al., "Demystifying Ai-Driven Performance Optimization: A Technical Deep Dive," International Journal Of Computer Engineering & Technology, 2025. [Online]. Available: https://www.researchgate.net/publication/388189045_DEMYSTIFYING_AI-DRIVEN_PERFORMANCE_OPTIMIZATION_A_TECHNICAL_DEEP_DIVE
11. Umang H Patel, "Evaluating The Impact Of Continuous Integration And Continuous Deployment (CI/CD) On Software Development Lifecycle Efficiency," International Journal of Creative Research Thoughts (IJCRT), 2024. [Online]. Available: https://www.ijcrt.org/papers/IJCRT2403893.pdf
12. Thorsten Rangnau, et al., "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines," IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC), 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9233212