Building Automated Security Pipeline for Containerized Microservices

Abstract: *DevOps and microservices architectures are increasingly being adopted by organizations to improve software delivery agility, scalability, and speed. These techniques transform development workflows, resulting in faster innovation and deployment. However, they pose distinct security challenges, especially in containerized systems. Containers, as lightweight and scalable solutions, are essential to microservices; nonetheless, they are prone to hazards like misconfiguration, insecure images, and security vulnerabilities during execution. Addressing the security of these systems is critical for maintaining trust, safeguarding sensitive data, and allowing new cloud-native ecosystems. This comprises automating security checks including vulnerability scanning, security testing, and policy enforcement within the CI CD pipeline. Securing container images, ensuring compliance with corporate regulations, achieving runtime monitoring, and aiding developers with constant input are all key approaches. Automating these procedures allows firms to proactively mitigate risks, assure consistent protection, and build a culture in which security is seamlessly incorporated into the development lifecycle. This method enables enterprises to accomplish both faster delivery and comprehensive security, paving the door for scalable, safe, and high-performance microservices implementations. This article investigates the revolutionary impact of DevOps and microservices, the vital requirement for security in containerized settings, and solutions for developing automated security pipelines using tools like SAST and DAST, without sacrificing the speed and efficiency of CI/CD workflows.*

Keywords — DevSecOps - Development Security & Operations, CI CD - Continuous Integration and Continuous Deployment, SAST/DAST-Static/DynamicApplicationSecurityTesting.

1.Introduction

In a contemporary software landscape, containerized microservices have revolutionized application development by enabling modular, scalable, and efficient systems. Containers, managed through platforms like Docker and Kubernetes, allow developers to package applications with their dependencies, ensuring consistency across development, testing, and production environments. However, as microservices continue to proliferate, they also introduce a more complex and distributed attack surface, posing significant security challenges.

To address these challenges, security must evolve alongside the development and deployment practices. Traditional, manual security measures are insufficient in the fast-paced, continuous delivery pipelines used for containerized applications. This has necessitated the integration of automated security pipelines into the DevSecOps workflow.

The Automated Protection Pipeline for containerized microservices attempts to integrate security across the program development lifecycle. It includes systems for proactive vulnerability assessment, policy enforcement, and runtime threat detection, ensuring that security is not an afterthought but rather an ongoing activity. By employing automated tools and frameworks, this pipeline improves security while keeping the agility and speed required by modern development processes. This study focuses on creating and implementing an automated security pipeline for containerized microservices. This pipeline will allow:

- **Continuous Vulnerability Assessment:** Examining source code and container images for possible threats**.**
- **Policy compliance:** Corporate security and legal requirements.
- **Runtime Security:** Observing microservices for unusual post-deployment activity.
- **Smooth Integration:** Keeping security procedures in line with current CI/CD workflows to reduce interference.

Important issues including privilege escalation threats, dependency management, and container isolation vulnerabilities are addressed by the suggested pipeline. The technology guarantees that security becomes an essential component of the microservices architecture by automating these procedures, improving production environments' resilience and trust.

This paper aims to provide a comprehensive guide to building such a pipeline, analyzing the associated challenges, tools, and best practices. It emphasizes the importance of creating secure yet efficient workflows for managing containerized microservices, ultimately bridging the gap between development speed and robust security.

2.Related Works

Given that cloud-native architectures are being adopted so quickly, there is a lot of interest in protecting containerized microservices in DevOps environments. Organizations confront several difficulties in resolving vulnerabilities, guaranteeing compliance, and upholding solid safety across the software development lifecycle as an outcome of the shift to microservices and container-based deployments. To effectively tackle these concerns, a number of research initiatives have examined automated security techniques integrated into CI/CD pipelines.

Among the numerous options, including security technologies for vulnerability screening, policy enforcement, and runtime monitoring has proven beneficial in safeguarding containerized workloads. During the early phases of development, techniques like container image scanning and runtime threat detection were used to identify and manage hazards. Additionally, policy-as-code frameworks and automated compliance checks have been widely studied for their ability to enforce consistent security standards. These strategies aim to maintain the agility of DevOps. operations and the strict security needs of modern software systems.

The emergence of containerized apps emphasizes the importance of including security measures into CI/CD pipelines. Recent studies focus on implementing automated security scanning techniques for containerized environments. Research highlights those methods such as SAST and DAST, when combined with tools like Snyk and StackHawk, can effectively identify vulnerabilities and automate fixes during the build process [1]. Another study emphasizes the need for robust security frameworks in container orchestration, addressing lifecycle challenges through tools like Grype and Anchore [2].
Moreover, the integration of AI in security-first DevOps pipelines has been proposed to enhance real-time threat detection [3]. This approach leverages machine learning for proactive mitigation, offering resilience against security breaches. Similarly, security automation in CI pipelines, as detailed in enterprise-driven open-source projects, is proven to enhance the detection of vulnerabilities and adherence to compliance standards [4].

Despite these advancements, challenges such as high false positive rates in static analysis and the trade-off between security and cost remain [5]. The adoption of shift-left security practices further emphasizes the need for early vulnerability detection to secure container images effectively [6]. Security policies integrated as a foundation of the DevSecOps pipeline minimize risks and vulnerabilities, fostering collaboration between teams to enhance development efficiency [7].

The continuous enforcement of policies through automated vulnerability detection has shown promising results in mitigating runtime risks in containerized environments [8]. However, a lack of security automation in microservice applications still poses challenges, as their distributed nature increases monitoring complexity and the potential for attacks [9]. By utilizing techniques such as host-based intrusion detection systems, dynamic adaptation, and integration of dynamic security testing tools, the overall security posture of CI/CD pipelines can be significantly improved [10]. Furthermore, combining AI-powered threat detection and predictive analytics into these security measures might improve the detection of emerging threats, allowing for more proactive responses to vulnerabilities [15].
DevSecOps integration underscores the importance of proactive measures in securing containerized applications. These methodologies contribute to reducing time-to-detection, streamlining remediation, and ensuring compliance with security standards in dynamic environments.

3.Proposed System

The proposed solution involves creating an end-to-end automated security pipeline that integrates security scanning, vulnerability management, and policy enforcement into the CI CD workflows. The main components of this approach are:

3.1 Shift Left Secu. Practices:

Include Static Appli. Secu. Testing (SAST) and Software Composition Analysis (SCA) in the early phases of the development process.Use tools like SonarQube and Snyk for auto. scanning of source code and third-party dependencies to detect vulnerabilities early.

3.2 Dynamic Appli. Security Testing (DAST):

Using DAST technologies like StackHawk during the pre-deployment phase allows you to replicate real-world assaults on staging systems while also finding runtime vulnerabilities like misconfigurations, injection problems, and insecure APIs. DAST does black-box testing, which evaluates running applications from an attacker's perspective and identifies vulnerabilities. Integrating DAST into the CI/CD pipeline enables enterprises to resolve vulnerabilities proactively prior to deployment.

3.3 Container Image Scanning:

Automate container image scanning using tools like Trivy or Grype to identify vulnerabilities in base images and application layers.Enforce security regulations that prohibit the usage of susceptible images in the CI/CD pipeline.

3.4 Constant surveillance and Threat Detection:

Incorporate runtime security solutions such as Falco or Aqua Security to ensure continuous tracking of deployed microservices in production. Utilize AI/ML models to detect abnormalities and actively mitigate risks in real time.

3.5Policy Enforcement and Compliance Checks:

Use policy-as-code frameworks like Open Policy Agent (OPA) to enforce security rules and compliance requirements across the pipeline. Automate compliance checks for regulatory standards such as GDPR, PCI DSS, or HIPAA.

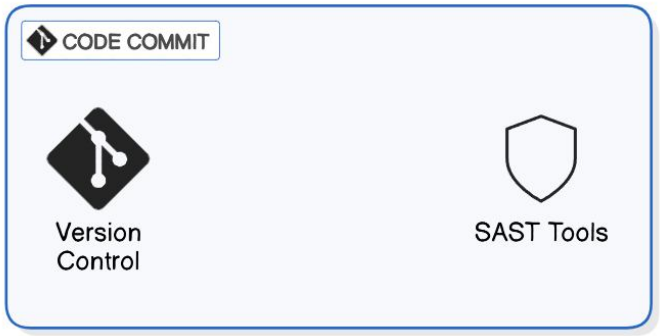
3.6 Secure Infra.-as-Code (IaC):

Scan Infra.-as-Code scripts (e.g., Terraform, Ansible) Using technologies like Checkov or Terrascan to detect insecure configurations before provisioning cloud resources.

4.Automation Process

4.1 *Code Commit*

The Code Commit stage is where developers push their code changes to a version control repository (e.g., Git). This is the initial place where automated security checks can be initiated, guaranteeing vulnerabilities get identified early in the development cycle.. This is often accomplished by incorporating static appli. sec. testing (SAST) technologies into the CI/CD workflow. Use a



CI/CD tool (such as Jenkins, GitLab CI, CircleCI, or Travis CI) to automatically start the pipeline when a commit is made.

Fig 1 : SAST Config.

- Setting up SAST tools such as SonarQube, Snyk, and Checkmarx during the build process enables for automated scanning of the source code to detect vulnerabilities such as SQL injection, XSS, and other security problems. Integrating these technologies identifies possible flaws early in development, allowing for speedy repair before the code reaches later stages of deployment.

- **GitLabCI** : Use a .gitlab-ci.yml configuration to run SAST after each commit

.4.2 Build Stage

The **Build Stage** is where the application is compiled, dependencies are resolved, and the software is packaged in to deployable units, such as container images or artifacts. Once the code is built, it can automate the process of scanning the dependencies and static analysis during the **build stage**.

- Integrate dependency scanning tools (e.g., Snyk, OWASP Dependency-Check) to automatically scan the project's dependencies for known vulnerabilities.
- Integrate SAST tools into the build process to scan both the application code and any dependency files (e.g., package.json, pom.xml).
- Set the pipeline to **fail if any vulnerability** above a set threshold is found (i.e., vulnerabilities in critical dependencies or code).
- **Snyk**: Integrate Snyk for dependency scanning.

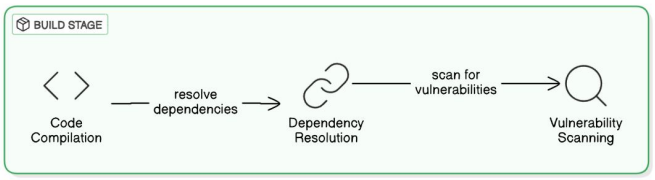


Fig 2 Build Stage Configuration

4.3 Containerization Stage

The **Containerization** stage involves creating container images (e.g., Docker images) that package the application and its environment. This stage includes ensuring that the container is secure before it’s deployed.

In the **containerization stage**, automate the scanning of the container image and enforce security policies before it is used.

- **Integrate container image scanning tools** (e.g., **Trivy**, **Grype**) into the pipeline to scan the container image for known vulnerabilities in the base image, libraries, and dependencies.
- **Policy enforcement** using tools like **Open Policy Agent (OPA)** to ensure that container configurations are secure. Checking that containers do not run with root privileges, unnecessary ports are closed, or sensitive data is not exposed.
- **Fail the pipeline** if the container image scan fails or security policies are violated.

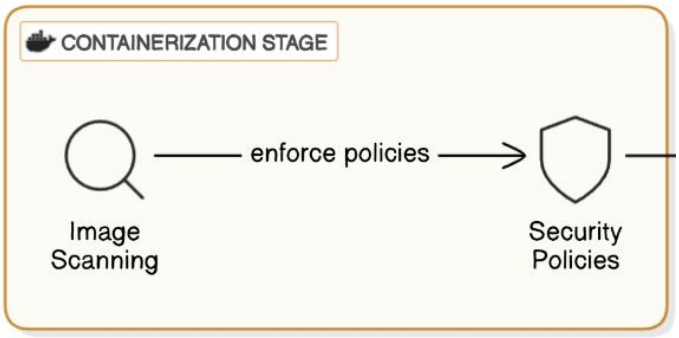


Fig 3 Containerization Stage Configuration

4.4 Testing Stage (Pre-Deployment)

The **Testing Stage** ensures that the application works correctly and securely by running integration tests, system tests, and security tests in a staging environment that mirrors the production environment. The **testing stage** includes dynamic appl. sec. testing (DAST) and infra-as-code (IaC) scanning. DAST identifies issues that only become apparent during runtime, which are often missed by static analysis tools. IaC scans configuration files to detect misconfigurations, insecure settings, or compliance violations before they affect production systems.

- **DAST tools** (e.g., **OWASP ZAP**, **StackHawk**) can be automated to run during integration or system testing. This will simulate real-world attacks on the app, such as SQL injections or cross-site scripting, while the app is running in a staging environment.
- **IaC scanning tools** (e.g., **Checkov**, **TFSec**) can automatically scan infrastructure code (e.g., Terraform, Kubernetes YAML files) for misconfigurations or security vulnerabilities before deployment.
- **OWASP ZAP**: Integrate OWASP ZAP for DAST during testing.

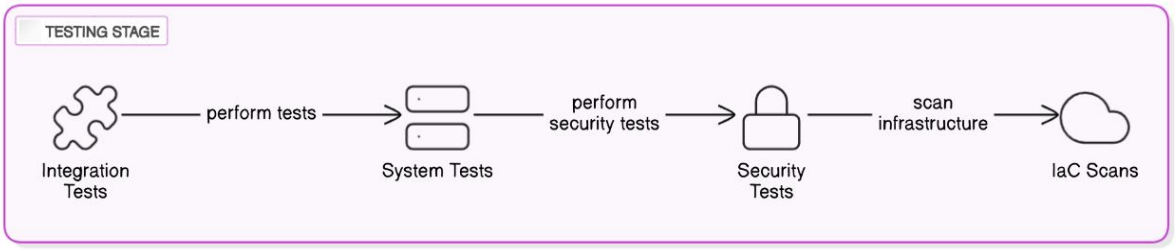


Fig 4 : Testing Configuration

4.5 Compliance and Approval Stage

The **Compliance and Approval** stage involves ensuring that the application complies with industry regulations and security best practices before it can be deployed to production. Ensures the application meets regulatory requirements (e.g., GDPR, HIPAA, PCI-DSS) regarding data privacy, security, and logging. The system checks for data protection rules, user consent tracking, encryption standards, and more, depending on the regulations the company is subject to.

- **Set up automated compliance checks** (e.g., **Prisma Cloud**, **Qualys**) to validate that the application meets required security policies (e.g., GDPR, HIPAA).
- Use **automated gates** in the pipeline for **denying** deployment if a regulatory check or security criteria fails. These entry points can be included with tools such as Jenkins. orGitLab.
- **Jenkins**: Use Jenkins to add a security gate based on a compliance threshold.

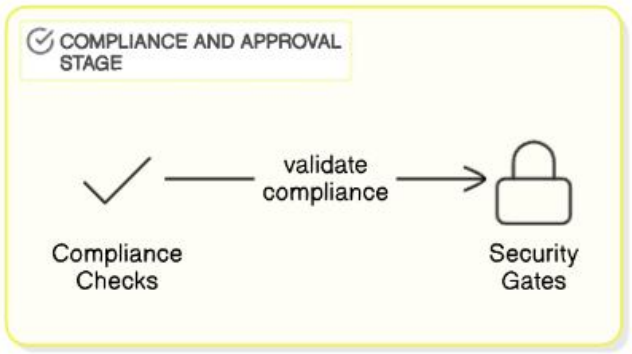


Fig.5 Compliance And Approval

4.6 Deployment to Production

The **Deployment to Production** stage is where the application is released to live environments, such as cloud infrastructure or on-premises servers. This stage often involves rolling updates to ensure minimal disruption.

- **Secrets management** (e.g., **HashiCorp Vault**, **AWS Secrets Manager**) can be linked into the deployment pipeline, securely storing and managing API keys, database credentials, etc. This ensures sensitive data is never hardcoded in the source code or configuration files.
- Automate the **deployment to Kubernetes** or other orchestrators, using **deployment scripts** that check access control policies before pushing the application live.
- **Vault**: Use HashiCorp Vault to inject secrets into the container.

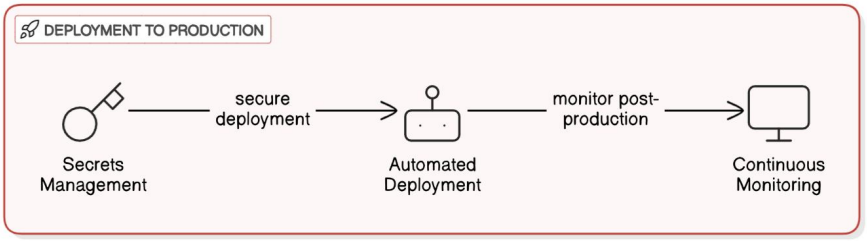


Fig 6 Deployment Configuration

4.7 Continuous Monitoring in Production

After the application is deployed to production, continuous monitoring is implemented to detect any potential security incidents or abnormalities that may signal an attack..

- **Set up continuous monitoring tools** (e.g., **Falco**, **Aqua Security**, **Datadog**) to look for unusual behaviors in the runtime environment, such as illegal access, malware, or strange system calls.
- **Automate alerting** When a suspected security breach is found, notify the relevant teams..
- **Falco** : Automate real-time monitoring to discover and alert on anomalous activity and potential security threats..

Implementing security in the CI/CD process demonstrates crucial in identifying vulnerabilities prior to the development phase and ensuring that security is built into the whole software development lifecycle. Jenkins, GitLab CI, and CircleCI enable automated security checks at each stage of code commit, build, test, and deployment. Automating security chores such as static analysis and vulnerability scanning and policy enforcement, teams can continuously test their code for security risks and vulnerabilities. If any issues, such as vulnerabilities or compliance violations, are detected, the pipeline automatically fails, preventing insecure code from being deployed. This automatic feedback loop ensures that security is a priority, lowering the likelihood of vulnerabilities reaching an entry into production environments. As a result, the development process gets more productive and safer, ensuring that code is consistently up to date.

Incorporating security checks directly into the CI/CD pipeline facilitates teams to shift left on security, fixing potential problems as soon as possible and drastically minimizing the risk of security breaches. This proactive security approach gives a security-conscious culture, where developers are empowered to address vulnerabilities swiftly and efficiently, minimizing their impact. The integration of security into the pipeline helps teams build a more robust, resilient system by automating tasks that would otherwise require manual intervention. Ultimately, automating security in the CI/CD pipeline not only ensures that only secure, compliant code reaches production but also streamlines the development process, leading to faster and safer software delivery. This continuous integration of security strengthens the organization's overall security posture while maintaining agile development workflows.

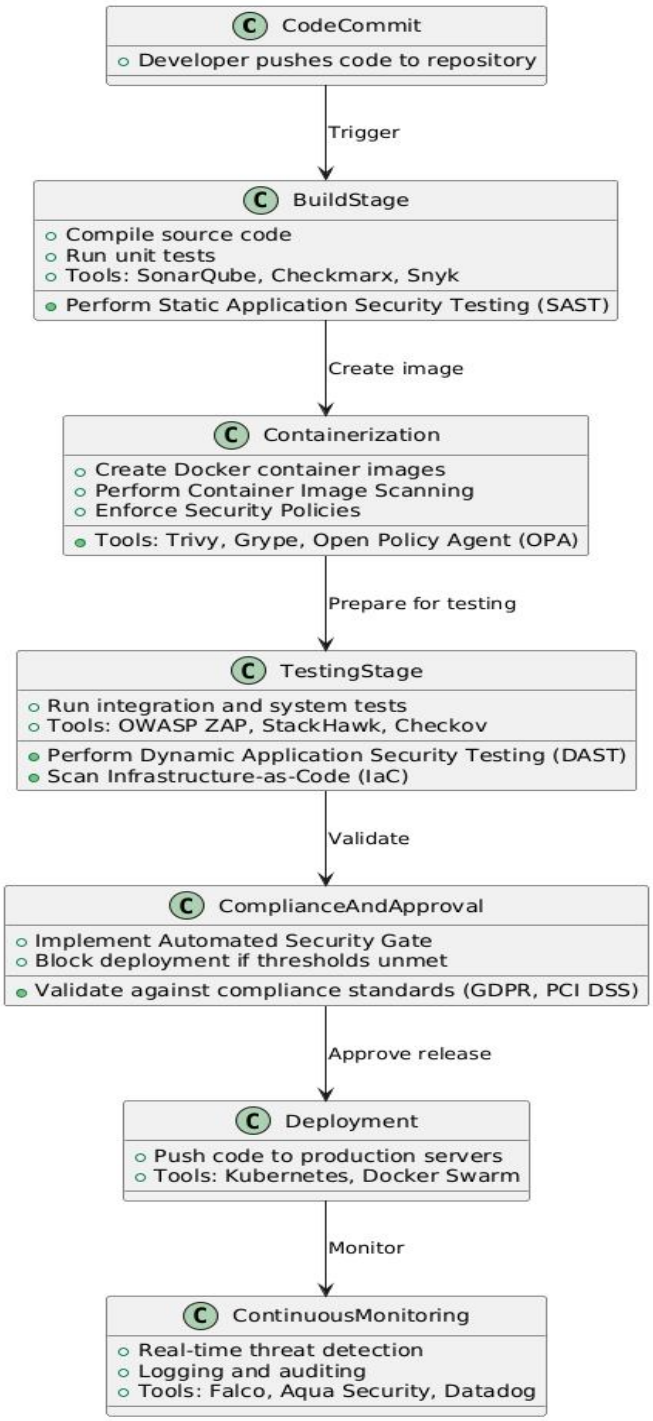


Fig 7 Pipeline with Integrated Security

As illustrated in the figure 7, combining these steps and integrating the tools into our CI/CD pipeline ensures that security is automatically enforced throughout the software delivery lifecycle. By leveraging automation, we can seamlessly incorporate security checks at every stage, from development to deployment. This integration strengthens the overall security of containerized microservices, minimizing risks and vulnerabilities.

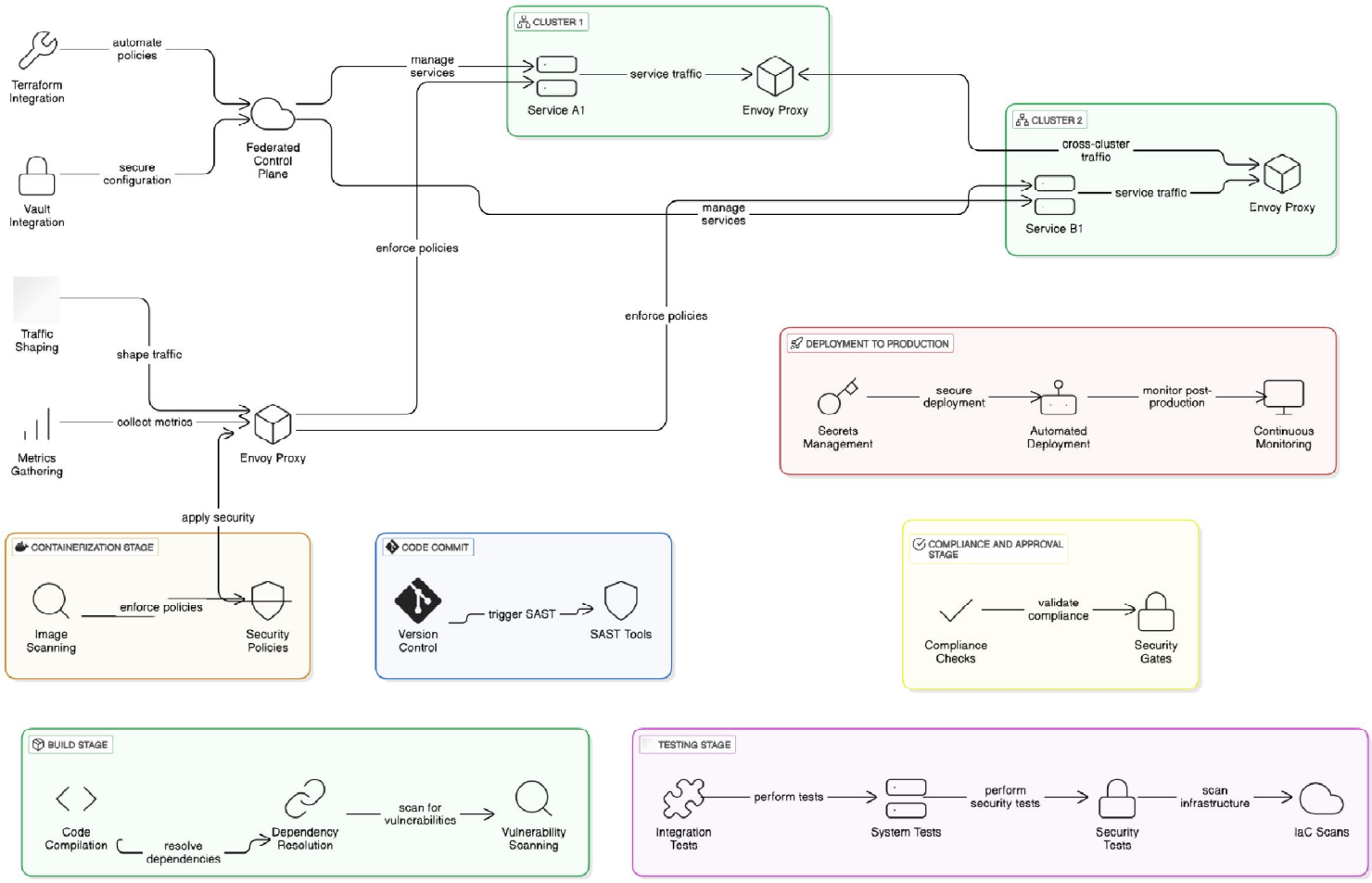


Fig 8 Overall Process Flow of Integrated Security

5. RESULTS

Development of an automated security pipeline for containerized microservices greatly boosts security, efficiency, and scalability. Integrating security testing at each level of the CI/CD pipeline guarantees early vulnerability identification, real-time identification of threats, and consistent compliance, all while cutting expenditures and reducing time to market.The quantifiable advantages of the automated security procedure are displayed in the statistics below..

Automated security checks are used at each stage of the development lifecycle, including code commit, build, containerization, testing, and deployment to detect vulnerabilities in a timely manner. This early detection reduces the risk of discovering security issues later in the development process and prevents unsafe code from reaching production, lowering the cost of addressing vulnerabilities by up to 98% because issues are handled early on rather than after deployment.In production, real-time monitoring ensures that any unexpected behavior or potential security breaches are discovered and corrected as soon as feasible. Aqua Security and Falco are two systems that monitor runtime activity, detect anomalies, and generate immediate alerts that trigger automatic countermeasures to avoid attacks. This proactive solution protects data and the company's image by reducing downtime caused by security incidents by up to 90%.

Automated security pipelines also continuously enforce compliance with regulatory standards such as GDPR, PCI DSS, and HIPAA. This eliminates the necessity to conduct human compliance checks, which ensure continued policy adherence and drastically lower risk of regulatory violations and associated legal consequences. This automation saves up to 85% of the time spent on manual compliance checks. Furthermore, automated tools like SAST, DAST, and container scanning enable accelerated development by catching vulnerabilities without human intervention. SAST analyzes source code for vulnerabilities early in

development, whereas DAST examines apps during runtime to detect security flaws. Security gates block vulnerable builds automatically, ensuring only secure releases progress to production, improving development velocity by up to 70%, reducing deployment delays, and accelerating time-to-market. Fixing vulnerabilities early in the CI/CD pipeline also reduces the financial burden compared to fixing problems after deployment, as after-production changes are 10-100 times more expensive than early repairs. Automated checks significantly reduce labor costs associated with manual security reviews, decreasing overall security management costs by up to 50%.

Automated pipelines ensure that every code change, container image, and configuration file is subjected to consistent security checks, eliminating human error and inconsistencies in manual reviews. This maintains uniform security standards across the application lifecycle and improves the accuracy and reliability of security assessments, reducing false positives by up to 70%.

These pipelines are also highly scalable and adaptable, effortlessly handling increasing workloads and the dynamic nature of containerized environments. They support microservices architectures with growing numbers of containers and services and achieve a 10x improvement in scalability, enabling secure growth without additional manual effort.

By introducing security into the CI CD process, automated pipelines promote coordination between the development, operations, and security teams. Early fixes for security flaws might be made by developers. and without affecting workflows and security becomes a shared responsibility rather than an afterthought. This improves communication and responsiveness to security incidents, increasing team productivity by up to 60%. Automated tools identify and remediate vulnerabilities quickly, reducing detection time from weeks to minutes. Automated patching ensures systems remain up-to-date with security fixes, reducing vulnerability exposure time by up to 95% and ensuring minimal disruption to operations.

Automated pipelines also significantly lower the risk of security breaches by proactively detecting and addressing vulnerabilities. Security risks are identified and mitigated before deployment, and automated alerts ensure prompt response to incidents, reducing the likelihood of breaches by up to 80%. This protects sensitive data, maintains customer trust, and safeguards the organization's reputation.The transition from traditional security approaches to modern automated security pipelines marks a significant shift in how security is managed in software development. In the past, security was often handled in a siloed, manual, and reactive manner, particularly in monolithic systems. Security checks were performed intermittently, usually after an incident or during dedicated security review phases, which meant that vulnerabilities were often discovered late in the development process or, worse, in production.

Aspect	Traditional Security (Past)	Automated Security Pipeline (Future)	Improvement (%)
Vulnerability Detection Time	10–20 days after deployment	5–15 minutes (automated tools)	**~95% faster**
Average Cost of Fixing Bugs	$7,000 in production stages	$50–$150 during development stages	**~98% cost reduction**
Incident Response Time	Hours to days (manual)	Seconds to minutes (automated alerts)	**~90% faster**
Compliance Check Effort	Manual, 3–5 weeks for audits	Automated, continuous	**~85% less effort**

False Positive Rate	40–60% (manual reviews)	10–20% (with advanced scanning tools)	**~70% reduction**
Security Breaches	30% higher risk (post-deployment flaws)	Reduced to ~5% with proactive checks	**~80% fewer breaches**
Scalability	Limited scalability (requires more manpower)	Highly scalable (automated)	**~10x scalability increase**

Table 1 Key Statistical Improvements

From these key stats, we can conclude that automated security pipelines lead to a 95% faster vulnerability detection, a 98% reduction in bug-fixing costs, and a 90% faster incident response time. Additionally, they reduce compliance check efforts by 85%, lower false positives by 70%, and decrease security breaches by 80%, all while providing 10x greater scalability.

With the rise of microservices architectures, which are distributed and containerized, the need for a more agile and efficient security strategy became evident. Modern automated security pipelines, Integrating within theThe CI CD procedure offers a far more effective and ongoing approach to security. This step allows firms to deal with vulnerabilities promptly and consistently throughout the development process.

Amongst the most important advancements is the incorporation of automated security checks directly into the CI CD pipeline, ensuring security to be an ongoing aspect of development, rather than something to be addressed after the fact. Tools such as container scanners (e.g., Trivy, Clair) and orchestration tools are now standard in detecting vulnerabilities at various stages, from code commit to deployment, ensuring that potential threats are identified and mitigated before they reach production.

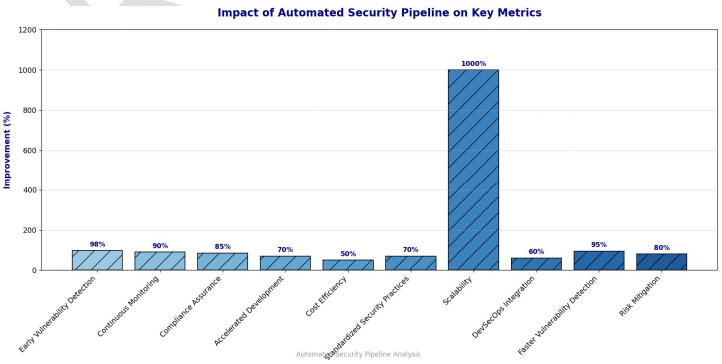


Fig 9 Improved Key Metrics

As shown in the figure the focus of security also evolves with this change. Traditional security relied heavily on endpoint protection and network firewalls to secure monolithic applications. In contrast, modern security pipelines focus on application-level security,

specifically container security and runtime monitoring, which are crucial for distributed, cloud-native applications. This shift reflects the growing complexity of systems and the need for a more nuanced approach to security.

Automation in modern security pipelines not only eliminates manual intervention but also accelerates threat detection and response. Security tools continuously monitor the system, and any anomalies are detected in real time, Triggering automated responses to reduce potential dangers. This reactive execution significantly lowers the time required to recognize and resolve problems as compared to traditional methods., where responses were often slow and reactive.In terms of compliance, modern pipelines offer dynamic policy enforcement and automated compliance checks, streamlining what was previously a manual and static process. This is crucial for organizations that must adhere to complex regulatory standards, as it minimizes human error and ensures compliance is continuously monitored.

6.CONCLUSION

Implementing an automated security pipeline for containerized microservices has a profound impact on the security, efficiency, and cost-effectiveness of the development lifecycle. It not only improves the security posture by detecting vulnerabilities early but also accelerates the development process, cuts down on manual labor, and reduces long-term costs. Integrating security into the CI/CD workflow allows teams to guarantee that microservices architecture remains resilient against attacks while delivering value to customers faster.

This approach leads to improved collaboration across teams, better compliance, and a more robust security posture, making certain that security is a consciously done, continual procedure that is built into the development and deployment process rather than a last-minute consideration. The incorporation of security into the DevSecOps model fosters a security-first culture. Where the development, security & operations teams collaborate to discover and eliminate problems before they reach production.The benefits of implementing an automated security pipeline are numerous. It significantly improves security by enabling early detection and remediation of vulnerabilities, thus reducing the likelihood of costly breaches. The pipeline also accelerates time-to-market by allowing for more frequent, faster releases without compromising security. Additionally, automating security processes reduces manual intervention, resulting in operational cost savings. As the number of microservices grows, the pipeline ensures that security practices remain consistent and scalable across the system.

However, there are challenges to consider. Setting up an automated security pipeline can be complex, requiring significant time, resources, and careful tool selection. Initial configuration might lead to false positives, which can cause delays, and integrating multiple security tools can sometimes create compatibility issues. Ongoing maintenance of security tools to keep up with emerging threats or changes in the pipeline is also necessary and can add to the workload of security teams.

Despite these challenges, the long-term benefits of implementing an automated security pipeline far outweigh the initial effort. As organizations increasingly adopt microservices and containerization, the need for integrated, automated security becomes even more critical. With the right strategies and tools in place, security becomes a seamless part of the development lifecycle, enabling faster, more reliable, and cost-efficient software delivery while safeguarding applications from potential threats.

COMPETING INTERESTS DISCLAIMER:

Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

Option 2:

Author(s) hereby declare that generative AI technologies such as Large Language Models, etc. have been used during the writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

Details of the AI usage are given below:

1.

2.

3.

7. REFERENCES

[1] Manohar Marandi, A. Bertia, Salaja - Implementing & Automating Security Scanning to a DevSecOps CI/CD Pipeline - IEEE Xplore 04 Sep 2023.

[2] Zlatan Moric, VedranDakic, MatejKulic - Implementing a Security Framework for Container Orchestration - IEEE Xplore 02 August 2024.

[3] Zachary Wadhams, Ann Marie Reinhold, ClementeIzurieta - Automating Static Code Analysis Through CI/CD Pipeline Integration - - IEEE Xplore 15 August 2024.

[4] Bharath Chandra Vadde, VamshiBharathMunagandla -Security-First DevOps: Integrating AI for Real-Time Threat Detection in CI/CD Pipelines -IJAETI 21 December 2021.

[5]Arvind Kumar Bhardwaj, P.K. Dutta, Pradeep Chintale -Securing Container Images through Automated Vulnerability Detection in Shift-Left CI/CD Pipelines -Babylonian Journal of Networking - 20 Aug 2024.

[6] Pradeep Chintale, GopiDesaboyina, Rajashekhar Reddy Kethireddy - Automating Vulnerability Detection in Container Images - J. Harbin Eng. Univ. - Sep 2024 .

[7] AnđelaTrajković, Milan Stojkov, MilošSimić, Goran Sladić Knowledge base driven pipelines for security enforcement-ICIST 2023 Proceedings, pp.53-64, 2023.

[8]BurakÜnver, Ricardo -Autom. Detection of Secu. Deficiencies and Refactoring Advice for Microservice - IEEE XPLORE 06 July 2023.

[9] Ugale,Potgantwar A - A Comprehensive Analysis and Future Directions for DevSecOps - Engineering Proceedings. 2023; 59(1):57.

[10] Jessica Castro, NunoLaranjeiro, Marco Vieira - Techniques and Tools for Runtime Security Monitoring and Analysis of Microservice- IEEE Xplore- 10 August 2023.

[11] SumanthTatineni - Compliance And Audit Challenges In Devops: A Security Perspective - Irjmets - October-2023.

[12] Florian Angermeir, Markus, Fabiola, Daniel - Enterprise-Driven Open Source Software: Study on Security Automation - IEEE Xplore -07 May 2021.

[13] Jose Flora - Improving the Secu. ofMicroservice Systems Detecting and Tolerating Intrusions - IEEE Xplore 04 January 2021.

[14] Thorsten Rangnau, Remco, Frank,Turkmen - Continuous Secu. Testing: Integrating Dynamic Secu. Testing Tools in CI CD Pipelines - IEEEXplore - 23 October 2020.

[15] Pattanayak, Suprit, Pranav Murthy, Aditya - Integrating AI into DevOps pipelines: Continuous integration, continuous delivery, and automation in infrastructural management: Projections for future. - IJSRA - 30 September 2024