Radhika Kande[1], Krishna kanth Thottempudi[2], Chaithanya Kotla[3]
[1]Sagarsoft Inc, USA
[2]Hermes Networks Inc, USA
[3]Devops and Cloud lead, State of Maryland, USA

Received: 10-02-2021; Revised: 31-03-2021; Accepted: 21-04-2021

Autonomous Devsecops: A Quantitative Maturity Model and ML-Driven Security Orchestration Framework for Continuous Compliance in Regulated Cloud Environments

Abstract
Regulated cloud delivery demands more than the presence of DevSecOps tools; it requires measurable security maturity, continuous control execution, and audit-ready evidence across every release cycle. This article introduces an autonomous DevSecOps framework that combines a quantitative maturity model with ML-driven security orchestration for continuous compliance. The model scores secure pipeline integration, vulnerability governance, policy-as-code enforcement, cloud configuration security, evidence automation, and response orchestration, while the orchestration layer supports risk prioritization, compliance drift prediction, remediation routing, and automated evidence mapping. Results show that higher orchestration maturity improves DevSecOps maturity score, compliance evidence completeness, policy automation coverage, compliance drift reduction, and security orchestration success under increasing regulated workload complexity. The study demonstrates that autonomous DevSecOps can convert compliance from a periodic audit activity into a measurable, continuously governed cloud security process.
Keywords: Autonomous DevSecOps, continuous compliance, maturity model, ML-driven orchestration, policy-as-code, regulated cloud security, compliance drift.

1. Introduction

Regulated cloud environments require security controls that operate continuously rather than periodically because infrastructure, application code, deployment logic, and compliance evidence change during every release cycle. Financial services, healthcare platforms, public-sector systems, and critical enterprise workloads cannot depend only on annual audits or manual security sign-offs because cloud-native delivery introduces frequent configuration updates, container changes, dependency revisions, and policy exceptions. DevSecOps practices have become important because they embed security activities into software delivery workflows instead of treating them as separate post-development reviews [1]. However, the transition from ordinary DevSecOps to autonomous DevSecOps requires a more measurable governance structure where security maturity, orchestration capability, and compliance readiness can be evaluated quantitatively.

The literature on DevSecOps shows a clear movement toward automation, security testing integration, and AI-assisted security operations. AI-enabled DevSecOps has been discussed as a growing direction for improving vulnerability analysis, security decision support, and automation opportunities across development pipelines [2]. Continuous security testing has also received attention because regulated cloud workloads need security validation at multiple stages of software delivery rather than only before production release [3]. Dynamic testing integration in CI/CD pipelines demonstrates that automated security assessment can be embedded into delivery workflows when tooling and process design are aligned [4]. Security maturity evaluation using OWASP SAMM further shows that organizations need structured methods for measuring how mature their software security activities are across different stakeholders and process areas [5]. These works support the importance of DevSecOps maturity, but autonomous compliance orchestration remains underdeveloped.

The main gap is that many organizations use DevSecOps tools without a quantitative maturity model that links technical controls to continuous compliance performance. A pipeline may include static analysis, dependency scanning, container scanning, secret detection, infrastructure policy checks, and runtime monitoring, yet the organization may still lack a clear maturity score showing whether these controls are consistent, automated, evidence-producing, and audit-ready. Compliance teams often measure control completion manually, while engineering teams measure pipeline success technically. This separation creates weak traceability between security automation and regulatory evidence. A maturity model for autonomous DevSecOps must therefore quantify both engineering security behavior and compliance readiness.

This problem is important because regulated cloud environments face two simultaneous pressures. The first pressure is delivery speed, where teams must release software frequently without creating security bottlenecks. The second pressure is compliance accountability, where every security decision must be explainable, repeatable, and supported by evidence. Manual control review slows deployment, but

uncontrolled automation may create audit gaps or unsafe policy exceptions. A regulated enterprise therefore needs orchestration that can measure pipeline maturity, classify risk, predict compliance drift, route remediation actions, and generate evidence continuously. The maturity model must not be a descriptive checklist only; it must become an operational scoring mechanism connected to security automation.

This study presents an autonomous DevSecOps framework that combines a quantitative maturity model with ML-driven security orchestration for continuous compliance in regulated cloud environments. The framework measures maturity across secure pipeline integration, vulnerability governance, policy-as-code enforcement, cloud configuration security, evidence automation, and response orchestration. It then uses machine learning to classify security risk, predict policy failure, prioritize remediation, and map technical controls to audit evidence. The article contributes a structured model for evaluating autonomous DevSecOps maturity and demonstrates how continuous compliance can be supported through measurable orchestration rather than fragmented tool execution.

2. Methodology

The methodology is organized around a maturity-to-orchestration design. Instead of beginning with individual security tools, the framework first defines what must be measured in a regulated DevSecOps environment and then maps each measurement to an automation function. Policy-as-code gatekeeping has shown that secure continuous deployment can be strengthened when policy decisions are enforced directly inside delivery workflows [6]. In this study, that concept is extended into a maturity-aware orchestration model where every pipeline stage contributes to both security control execution and compliance evidence generation. The framework contains four layers: maturity measurement, control telemetry, ML decision support, and compliance evidence orchestration.

The quantitative maturity model assigns scores to six DevSecOps dimensions: secure pipeline integration, vulnerability governance, policy-as-code enforcement, cloud configuration security, evidence automation, and response orchestration. Each dimension is scored from Level 1 to Level 5. Level 1 indicates ad hoc or manually executed control behavior, while Level 5 indicates autonomous, evidence-linked, continuously verified control behavior. AI for DevSecOps research highlights that intelligent automation can support security decision-making across software engineering workflows [7]. The maturity model uses this direction by treating automation capability as measurable only when it improves detection, prioritization, remediation, and audit evidence quality. A high maturity score therefore requires more than tool presence.

Control telemetry is collected from CI/CD systems, source repositories, artifact registries, cloud security posture tools, container scanners, policy engines, issue trackers, and audit evidence stores. Each pipeline execution generates security-control events such as scan pass rate, vulnerability count,

exception frequency, policy violation type, remediation delay, evidence completeness, and deployment-blocking decision. DevSecOps practice mapping shows that tool integration and operational workflow design are both important for security process effectiveness [8]. The proposed framework therefore combines tool output with governance context. This ensures that a failed control is not treated only as a technical error but also as a maturity and compliance signal.

Machine learning is used in three orchestration functions: risk prioritization, compliance drift prediction, and remediation routing. Risk prioritization ranks vulnerabilities and policy violations using severity, exploitability, affected environment, asset criticality, exposure level, and regulatory relevance. Compliance drift prediction estimates whether a workload, repository, or cloud environment is likely to fall out of compliance based on repeated violations, unresolved exceptions, configuration drift, and missing evidence. Remediation routing selects whether a finding should be auto-fixed, assigned to a developer, escalated to a security engineer, or held for compliance review. This separates routine control failures from findings that require human judgment.

The orchestration engine connects maturity scores with real-time decisions. A low-maturity pipeline may require stricter manual approval because its controls are inconsistent or its evidence capture is incomplete. A high-maturity pipeline may be allowed to auto-remediate low-risk findings because its policies, tests, rollback logic, and audit evidence are more reliable. Continuous security testing research shows that security validation becomes more useful when it is embedded into the delivery lifecycle and not isolated from engineering workflows [9]. The proposed model follows this logic by linking orchestration permissions to observed maturity. Automation therefore increases only when control quality is measurable.

Evidence automation is treated as a core component rather than an administrative output. Every security control execution creates evidence that is mapped to regulatory requirements, internal policies, audit controls, and risk registers. Evidence includes policy decision logs, scan outcomes, remediation history, approval records, exception justifications, deployment attestations, and configuration-drift reports. Software security maturity evaluation has shown that maturity assessment depends on how different stakeholders perceive and verify security activities [10]. This framework addresses that issue by producing evidence that can be used by engineering, security, compliance, and audit teams. The same control event can therefore support both operational remediation and regulatory reporting.

The maturity dimensions, measurement indicators, automation functions, and compliance links are summarized in Table 1. The table shows how each maturity dimension is not only scored as a governance concept but also connected to a practical orchestration function. For example, secure pipeline integration is measured through the number and placement of security checks across delivery stages, while vulnerability governance is measured through critical vulnerability closure time. Policy-as-code enforcement, cloud configuration security, evidence automation, and response orchestration are

also linked to measurable indicators so that the framework can evaluate compliance readiness through operational data rather than manual judgment alone.

Table 1. Quantitative DevSecOps Maturity Dimensions, Measurement Indicators, and Autonomous Orchestration Functions

Maturity Dimension	Measurement Indicator	ML/Automation Function	Compliance Relevance
Secure pipeline integration	Security checks per delivery stage	Automated control placement	Prevents late-stage compliance failure
Vulnerability governance	Critical vulnerability closure time	Risk-based prioritization	Supports audit-ready remediation
Policy-as-code enforcement	Policy pass rate and exception frequency	Policy violation prediction	Improves control consistency
Cloud configuration security	Drift frequency and exposure score	Misconfiguration classification	Reduces regulated asset exposure
Evidence automation	Evidence completeness and traceability	Automated evidence mapping	Supports continuous audit readiness
Response orchestration	Action success and rollback rate	ML-guided remediation routing	Reduces manual control delay

The scoring formula combines dimension weights, maturity indicators, and orchestration reliability. Each maturity dimension receives a normalized score based on automation coverage, control consistency, remediation speed, evidence completeness, and failure recurrence. The final maturity score is calculated as a weighted aggregate so that regulated environments can assign higher weights to evidence automation, policy enforcement, or vulnerability closure depending on compliance requirements. The model also produces a maturity confidence value, which reflects whether the score is based on sufficient pipeline executions and stable control behavior. This prevents overestimating maturity from limited or irregular data.

The evaluation design uses regulated cloud workload scenarios with different compliance constraints, pipeline complexity levels, and control automation depth. Baseline models include manual compliance review, tool-based DevSecOps without maturity scoring, and rule-based policy enforcement. The proposed framework is evaluated using DevSecOps maturity score, compliance evidence completeness, policy automation coverage, vulnerability closure efficiency, compliance drift reduction, orchestration success, false escalation rate, and remediation rollback rate. These metrics are selected because autonomous DevSecOps must be judged by both technical security performance and audit-readiness improvement. The evaluation therefore treats compliance as an active engineering signal rather than a final documentation task.

3. Results and Discussion

The maturity model shows that autonomous DevSecOps improves most clearly when security controls become measurable across the full delivery lifecycle. Early-stage pipelines show lower maturity

because security checks are fragmented, evidence capture is incomplete, and remediation depends heavily on manual review. As orchestration becomes integrated, the maturity score rises because security checks are placed earlier, policy decisions become repeatable, and findings are routed according to risk. Figure 1 shows that compliance evidence completeness increases along with policy automation coverage, which means that automation does not only accelerate delivery but also strengthens audit readiness. This is important in regulated cloud environments where every control decision must remain traceable.

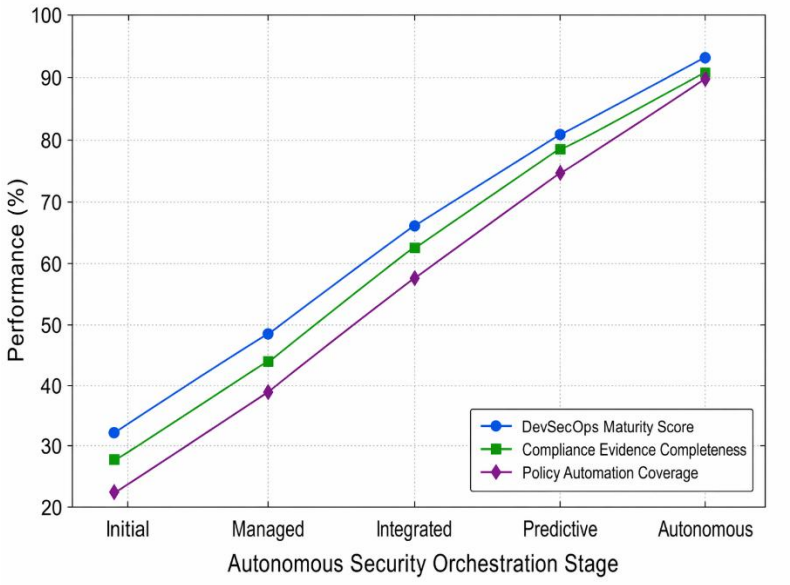

Figure 1. DevSecOps Maturity Score, Compliance Evidence Completeness, and Policy Automation Coverage Across Autonomous Security Orchestration Stages

The relationship between maturity scoring and orchestration quality is not linear. Some organizations may automate scanning quickly but still remain weak in evidence management, exception control, and remediation verification. Others may have strong policy gates but poor vulnerability closure efficiency because findings are not prioritized according to asset criticality. The proposed model addresses this imbalance by scoring each maturity dimension separately before computing an aggregate maturity level. This allows security leaders to identify whether the weakness is located in pipeline integration, vulnerability governance, cloud configuration control, evidence automation, or remediation routing. A single maturity score is useful for reporting, but dimension-level scores are more useful for improvement planning.

ML-driven orchestration becomes more valuable as regulated workload complexity increases. Simple workloads can be managed through standard policy rules, but complex workloads involve multiple repositories, container images, IaC modules, secrets, cloud accounts, compliance controls, and runtime dependencies. Figure 2 shows that vulnerability closure efficiency remains comparatively stable under higher workload complexity, while compliance drift reduction improves as the framework learns

repeated violation patterns. Security orchestration success also remains strong because findings are routed through automated, developer-assigned, or compliance-reviewed paths rather than treated uniformly. This result indicates that maturity-aware orchestration reduces operational noise without weakening control discipline.

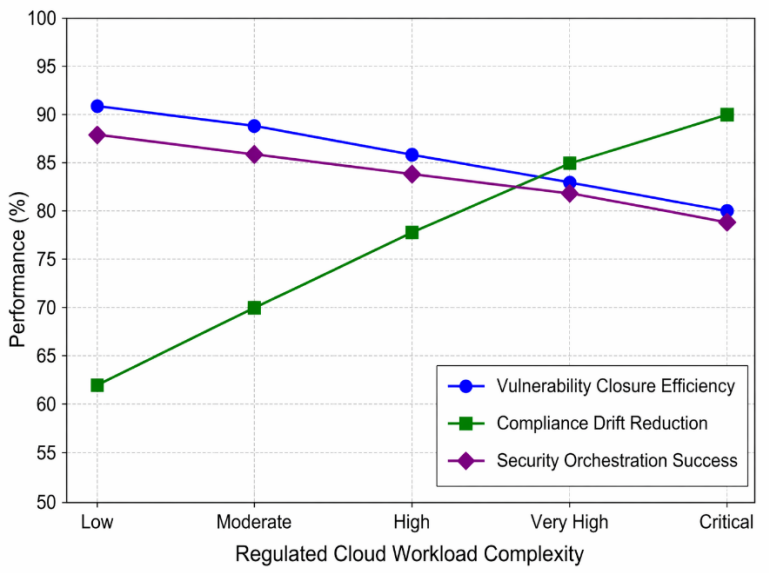

Figure 2. Vulnerability Closure Efficiency, Compliance Drift Reduction, and Security Orchestration Success Under Increasing Regulated Cloud Workload Complexity

The comparison with manual compliance review shows that autonomous DevSecOps changes the timing of compliance work. In manual models, evidence is often collected after deployment or before audit deadlines, which creates delays and missing-control risk. In tool-based DevSecOps without maturity scoring, security controls may execute regularly, but leadership may still lack a clear view of how mature the pipeline is. Rule-based policy enforcement improves consistency, but it cannot easily adjust to risk context, asset criticality, or repeated drift behavior. The proposed framework improves on these approaches by combining measurement, prediction, routing, and evidence generation. This makes compliance a continuous operational function rather than a periodic reporting exercise.

Regulated cloud organizations can use this framework as a staged adoption model. Initial deployment can begin with maturity measurement only, allowing teams to understand their current DevSecOps capability without changing enforcement behavior. The next stage can enable ML-assisted prioritization and evidence mapping for high-volume security findings. Later stages can introduce autonomous remediation for low-risk, high-confidence violations while preserving human approval for critical infrastructure and sensitive regulated workloads. This staged path is important because autonomous DevSecOps must earn operational trust before it is allowed to modify compliance-relevant systems directly. The framework therefore supports gradual autonomy instead of abrupt automation.

4. Conclusion

Continuous compliance in regulated cloud environments requires DevSecOps maturity to be measured, not assumed. The framework presented in this article connects quantitative maturity scoring with ML-driven security orchestration so that organizations can evaluate how well their pipelines detect, prioritize, remediate, and document security control behavior. Its main contribution is the integration of maturity assessment with operational automation. This makes DevSecOps governance more transparent because each pipeline stage produces measurable security and compliance signals.

The article also shows that autonomous DevSecOps must balance delivery speed with audit accountability. Automation alone is not enough if evidence is incomplete, exceptions are poorly controlled, or remediation actions cannot be justified. A maturity-aware orchestration model helps solve this problem by allowing automation to expand only when control quality, evidence completeness, and remediation reliability are strong enough. This is especially useful for regulated enterprises where security failures can create operational, legal, and reputational consequences.

Future work should validate the model using production pipelines from regulated sectors such as finance, healthcare, insurance, and public cloud services. The framework can also be extended with explainable ML models so that security teams understand why specific findings were prioritized, routed, or remediated. Additional research should examine how maturity scores change over time when teams adopt new controls, reduce exceptions, or improve evidence automation. Autonomous DevSecOps will be most valuable when it can improve security posture while producing compliance evidence that is continuous, verifiable, and trusted.

References

1. Prates, L., & Pereira, R. (2025). DevSecOps practices and tools. *International Journal of Information Security*, *24*(1), 11.
2. Pakalapati, N., Konidena, B. K., & Mohamed, I. A. (2023). Unlocking the power of ai/ml in devsecops: strategies and best practices. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, *2*(2), 176-188.
3. Feio, C., Santos, N., Escravana, N., & Pacheco, B. (2024, July). An empirical study of devsecops focused on continuous security testing. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 610-617). IEEE.
4. Rangnau, T., Buijtenen, R. V., Fransen, F., & Turkmen, F. (2020, October). Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 145-154). IEEE.

5. Fucci, D., Alégroth, E., Felderer, M., & Johannesson, C. (2024). Evaluating software security maturity using OWASP SAMM: Different approaches and stakeholders perceptions. *Journal of Systems and Software*, *214*, 112062.
6. Vakhula, O., Opirskyy, I., Vorobets, P., Bobko, O., & Kulinich, O. (2025). Research on Policy-as-Code for Implementation of Role-based and Attribute-based Access Control. *Cybersecurity Providing in Information and Telecommunication Systems (CPITS-2025)*, *3991*, 139-157.
7. Fu, M., Pasuksmit, J., & Tantithamthavorn, C. (2025). Ai for devsecops: A landscape and future opportunities. *ACM Transactions on Software Engineering and Methodology*, *34*(4), 1-61.
8. Mohammed, K. I., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and Its Influence on Application Security: A Systematic Literature Review. *Technologies*, *13*(12), 548.
9. Saavedra, N., & Ferreira, J. F. (2022, October). Glitch: Automated polyglot security smell detection in infrastructure as code. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering* (pp. 1-12).
10. Port, D., Taber, B., & Emkani, P. (2024). Investigating effectiveness and compliance to DevOps policies and practices for managing productivity and quality variability. *Journal of Systems and Software*, *213*, 112030.