International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(Research Article)

Policy-driven infrastructure hardening using CI/CD pipelines in enterprise environments

Rohith Aitharaju *

Independent Researcher, USA.

International Journal of Science and Research Archive, 2022, 07(01), 591-602

Publication history: Received on 28 June 2022; revised on 21 October 2022; accepted on 29 October 2022

Article DOI: https://doi.org/10.30574/ijsra.2022.7.1.0280

Abstract

The way modern businesses are speeding up software implementation using CI/CD, securely managing infrastructure automatically has never been more essential. Old methods of protecting systems, made by hand and only done when problems arise, cannot catch up to what DevOps pipelines require. This research looks at using Policy-as-Code (PaC) in CI/CD pipelines to apply policy-driven hardening to infrastructure which helps maintain compliance, consistency and robustness. The research further examines basic ideas like Infrastructure as Code (IaC), managing configurations and the important security benchmarks CIS and NIST. It guides readers on how to use the following tools to ensure security when deployments are undertaken: Jenkins, GitHub Actions, Open Policy Agent (OPA) and HashiCorp Sentinel. False positives, complicated integration and resistance in the organization are discussed and solutions are given using a unified DevSecOps approach and intelligent policy engines. With this strategy, real-time enforcement of safety and compliance rules makes security an asset that helps enterprises scale, remain automated and use contextual protection. The findings end by sharing useful tips and possibilities for the future, helping businesses integrate strong security into their CI/CD workflows

Keywords: Policy-as-code(PaC); CI/CD Pipelines; Infrastructure as Code (IaC); Security Hardening; Open Policy Agent

1. Introduction

Organizations are seeing their IT infrastructure shift to be more flexible, spread across places and depend on the cloud [1]. Since using DevOps and cloud-native systems has grown common, organizations are making software delivery and infrastructure deployment faster through continuous integration and deployment (CI/CD) pipelines [2]. There have been more obstacles to security because this shift leads to much of the infrastructure being managed and deployed via automated code.

Protecting systems from risks such as vulnerabilities, mistaken setup and policy breaches through infrastructure hardening is a significant challenge at present. In the past, infrastructure hardening occurred through checking configurations by hand, running audits once a year or more and responding to security issues once they cause problems. However, Automated environments, where changes occur fast and all the time, now make it clear that these approaches are weak.

Because of these issues, more businesses are switching to policy-driven automation to secure their systems during every stage of setting up their infrastructure. If Policy-as-Code is included in CI/CD, organizations can examine and match security and compliance guidelines with configurations automatically. As a result, it allows for the same continuity in handling and assessing infrastructure tasks which decreases the possibility of mistakes and disagreement with industry guidelines.

* Corresponding author: Rohith Aitharaju

Using Policy-as-Code allows developers to take care of security earlier in the development stages. Thanks to Open Policy Agent (OPA), HashiCorp Sentinel and Conftest, we can make sure that nothing insecure is ever provisioned while working without holding up the delivery pipeline.

Even with improving DevOps and IaC, the majority of organizations still stick to ad hoc security reviews or audits after deployment. Consequently, issues are caught later, the company sometimes fails to comply with security rules and its configuration flaws provide more opportunities for attacks.

This research will examine the use of CI/CD for making infrastructure more secure through policies, look at how Policy-as-Code can be used with tools like OPA and Sentinel, present a practical process for handling security in automated deployment and show how this method works with an example of an enterprise scenario. The tool is meant to aid DevOps professionals, cloud engineers and IT security leaders in making infrastructure systems safer and compliant, even as infrastructure changes rapidly.

2. Basic concepts and their relationship

It is important to know the main technologies and practices behind policy-driven infrastructure hardening [3]. These practices are CI/CD pipelines, Infrastructure as Code (IaC), improved security settings and guidelines in the form of code, called policy-as-code. Both tools together make it possible to manage and automate modern enterprise infrastructure securely[4]

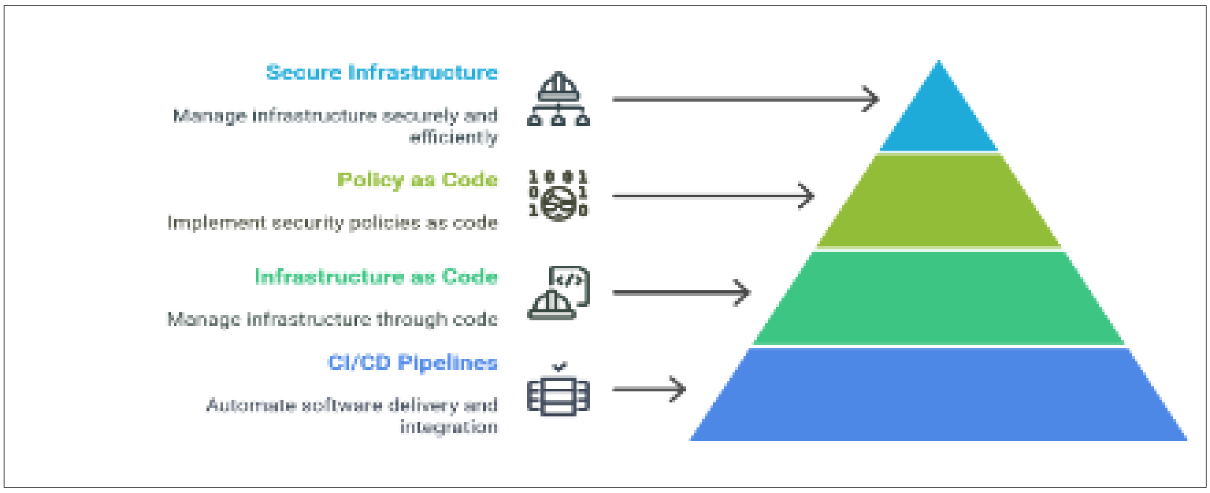
Secure Infrastructure
Manage infrastructure securely and efficiently
Policy as Code
Implement security policies as code
Infrastructure as Code
Manage infrastructure through code
CI/CD Pipelines
Automate software delivery and integration

Figure 1 Secure infrastructure management

2.1. CI/CD Pipelines

CI/CD pipelines take care of the tasks needed to move a piece of software from writing code to releasing it live[3]. By relying on these pipelines, teams don't have to handle tasks by hand, work more efficiently and ensure applications are delivered the same way every time. Useful CI/CD tools are:

- Jenkins: is an automation server that many people use to construct complex pipelines for both building and deploying software [5].
- GitLab CI/CD : acts as a DevOps service that links version control with CI/CD.
- Developers can automate their software deployment using GitHub Actions, GitHub's in-house tool.
- Azure Pipelines is one of many cloud-native tools, along with CircleCI and Travis CI which help with flexible and scalable pipeline automation [2].

Even though these tools make us more flexible, if controls are left out of pipeline verification during installation, it might cause security breaches.

2.2. Examples of Infrastructure as Code (IaC) and Configuration Management

With IaC, you can control and automate server, network and firewall management using code. Rather than assembling environments by hand, teams depend on Terraform, AWS CloudFormation or Ansible to automate the process so it can be repeated easily [9].

2.2.1. Some perks of IaC are:

- Using version control for infrastructure solutions
- The same configurations are used throughout the environment
- A quicker path to growth and enough resources

DevOps today relies heavily on IaC, but using it can create additional risks. One unintentionally set variable in Terraform code may leave your exposed services exposed to the web. So, working to secure your infrastructure templates is very important for improving overall security.

2.3. Essential Ways to Strengthen the Security of a PC

Security hardening involves making systems and infrastructure safer by using established and required security best practices [6]. Two important standards for this area are :

- CIS Benchmarks: are designed by the Center for Internet Security and contain steps for making Windows, Linux, Kubernetes and cloud systems more secure [8].
- NIST Frameworks provide the most important guidelines and procedures used in both federal and business worlds.
- Hardening means you should turn off services you do not use, use strong authentication, limit how many ports are open, set up least privileged access and monitor your system’s activity [7].
- Manual implementation of these strategies is slow and can lead to mistakes, so automation is a must in companies operating at a big scale.

2.4. Policy as Code

It is possible to write down security, compliance and governance rules as code, known as policy-as-code. All infrastructure and application deployments through pipelines must comply with organization standards because of these policies.

Popularly used policy-as-code tools are:

- An Open Policy Agent (OPA) is a tool that works with a language called Rego to establish rules for various system tasks. It is connected to Kubernetes, Terraform, CI/CD tools and other similar solutions.
- Sentinel from HashiCorp is a framework meant to be used alongside products like Terraform and Vault. Sentinel makes it easy to handle specific policies throughout your processes.

2.4.1. When using policy-as-code, organizations have the ability to:

- Stop insecure configurations before making the system go live.
- Use encryption or resource tagging in all environments.
- Set up your network so it follows security rules.
- Record a record of policy evaluation activities.

Implementing these tools makes it easy for CI/CD to keep security high and minimize the effects on developer speed.

2.5. Problems Faced by Enterprise Security in CI/CD

Building secure CI/CD pipelines in large corporations comes with distinct obstacles [10].

- Because of the wide range of services, enterprises depend on cloud providers, CI/CD tools, Infrastructure-as-Code solutions and monitoring technologies. Making sure the same policies are enforced all through the stack is hard.
- When teams must release more often, manual checks of security start to slow things down, missing a configuration before launch is easier without automation.

- Different approaches: Each team can handle and enforce policies in their own way. Because of this, companies may drift from following compliance and become less secure.
- If you don't have adequate logging, it becomes challenging to find any policy violations or demonstrate that you meet the rules during security audits.

Addressing these problems requires companies to have security policies controlled by code and built into CI/CD flows, applying these rules the same way in all environments.

3. Methodology: Policy Driven Infrastructure Hardening Using Ci/Cd Pipelines

It adds policy-as-code into the CI/CD process to support secure and compliant deployment of infrastructure on a large scale. It combines the use of automation, strong security and infrastructure as code to stop incorrect configurations in advance [10].

3.1. How to Include Policies in the CI/CD Chain

Standard stages in a CI/CD pipeline are code commit, build, test and deploy [11]. These security policies can be applied in a variety of these stages:

- When a Pre-Commit Hook is in use, the policies will check the code and Infrastructure as Code templates before allowing them to be pushed to the repository.
- During Build, appropriate policy tools are used to look for issues in the infrastructure's security settings.
- At this stage, policies look at whether security and operating standards (such as encryption and access controls) are met.
- Policy checks must all pass before any infrastructure changes in the deployment stage are allowed. Operations will be suspended if a violation occurs.

By using shift-left strategy, you can ensure that security happens at the start and throughout the process which cuts down on dangers later on.

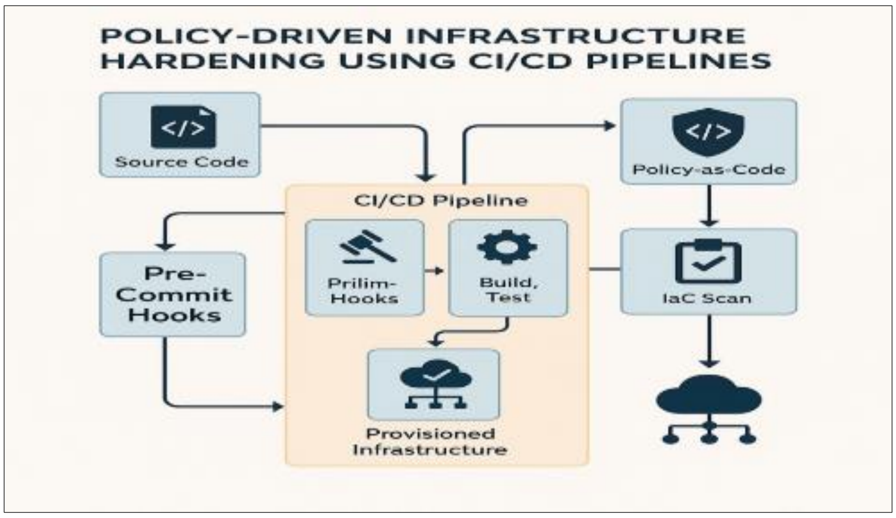
POLICY-DRIVEN INFRASTRUCTURE HARDENING USING CI/CD PIPELINES
Source Code
Policy-as-Code
CI/CD Pipeline
Pre-Commit Hooks
Prilim-Hooks
Build, Test
IaC Scan
Provisioned Infrastructure

Figure 2 Policy driven infrastructure hardnening using CI/CD

3.2. Validation and scanning using Infrastructure as Code (IaC) is the fourth point.

The foundation of today's infrastructure provisioning is built with IaC. Like Terraform, Ansible and CloudFormation, these tools express infrastructure using code [9]. The process is explained by:

- Writing Down Security: Requirements for rules and compliance are created as code inside Rego for OPA.
- Before adding the infrastructure, automated policies scan the IaC templates.
- Should an issue like an unencrypted S3 bucket occur, the pipeline fails promptly, telling you what happened.
- Only submissions that pass guidelines are permitted for deployment.

Validating automatically means our systems always start safely, even if we do not remember to validate manually.

4. Framework Application: An Illustration Of Enhancing Security In a Ci/Cd System By Policy

In this section, an example is given of how the suggested methodology might be used in a fictional business environment. It demonstrates a medium-sized financial firm that wants to maintain cloud security by having automated processes control pipelines, based on set policies[12].

4.1. Imaginary Case: FinSure Corp

The aim of FinSure Corp is to provide financial services all over North America. To host its applications, the company has Dev, QA and Prod areas on AWS. To manage infrastructure, it runs on Terraform, GitLab CI/CD takes care of automation and security is managed by integrating OPA (Open Policy Agent) [13].

4.1.1. Security Goals

- Always use encryption methods on every storage resource.
- Stick to the rule that IAM roles give only the permissions necessary.
- Make sure to shut down open security groups.
- All resources should carry both owner and environment information in their metadata.

Table 1 Automation and Governance Tools and Services for Infrastructure as Code (IaC)

Component	**Tool/Service Used**	**Function**
Source Control	GitLab Repository	Version control for Terraform templates
CI/CD Orchestration	GitLab CI/CD	Automates Terraform plan/apply operations
Policy Engine	OPA with Conftest	Validates Terraform code against enterprise policies
IaC Platform	Terraform	Defines and provisions cloud infrastructure
Cloud Platform	AWS	Hosts infrastructure resources

The workflow integrates policy evaluation in the CI/CD pipeline so that no infrastructure changes are deployed without passing compliance checks.

4.2. Steps in Applying Pipelines

- All Terraform code used in this project is saved to the GitLab repository.
- The pipeline is started by GitLab CI/CD:
  - Linting checks take place.
  - OPA policies are checked using Conftest against files written in Terraform.
- The outcome of the policy evaluation sets which direction the process follows:
  - During the pass step, Terraform runs the plan and applies commands.
  - In case of a failure, the pipeline shuts down and the developer receives a transmitted report.
- The necessary infrastructure is built up in AWS.

This system allows for security standards to be followed immediately.

Table 2 Results of policy checks for your Cloud Resources

Resource	**Policy Check**	**Result**	**Reason**
aws_s3_bucket.db_logs	Encryption enabled	Pass	server_side_encryption_configuration set
aws_iam_role.dev_ops	Least privilege IAM policy	Fail	Wildcard permissions found in JSON policy
aws_security_group.web	No wide-open access (0.0.0.0/0)	Pass	All ingress rules scoped to internal IPs
aws_instance.backend	Owner and environment tags	Fail	tags block missing required keys

4.3. Ratio of Policy Compliance Followed (More Than 5 Deployments)

Look at the graph below to see that less violations occurred as developers learned how to work according to the new policies.

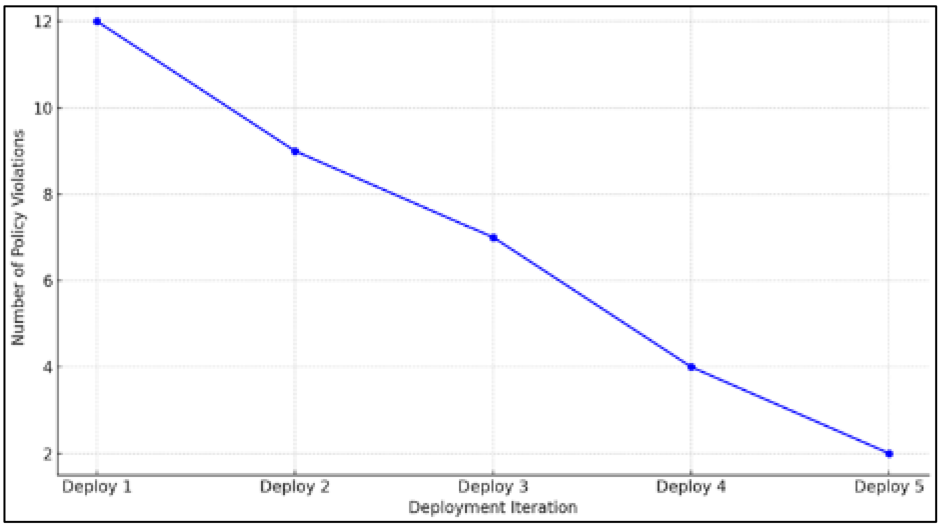

Figure 3 Trend of Policy Violations Over Five Consecutive Deployments

After policies were part of the culture, the number of violations fell which proved the value of early enforcement.

Table 3 The Relationship between Policy and Automation on Deployment and Security

Metric	Before Implementation	After Implementation
Avg. Deployment Time	35 minutes	30 minutes
Policy Violations/Month	18	30
Security Incidents/Qtr	2	0
Manual Reviews Needed	High	Minimal

4.4. The scenario reveals that using policy-as-code in CI/CD pipelines can help you:

- Eliminate the majority of risks in deployment.
- Improve how similar infrastructure is across industries.
- Cut down on the amount of work done by people.
- Make sure all DevOps teams share a commitment to compliance.
- The approach works at any scale, can be used repeatedly and matches current cloud and security beliefs.

5. Results

We evaluate how policy-driven hardware security in CI/CD pipelines makes a difference in the real world. They consist of vulnerable system reduction, compliance policy improvements, increased algorithm speed and stability during system rollout. To achieve this, my approach relies on building an enterprise setup with Infrastructure as Code (IaC) and does so continuously [9].

5.1. Overview of the Evaluation Measurements and Standards

To evaluate the effectiveness of the hardened pipeline, we used a set of important metrics.

- Numbers of Misconfigurations per Deployment: All misconfigurations that come up in Policy-as-Code tools.
- Tribute: The number of resources that made it through all security checks (such as CIS Benchmarks and user-made rules).

- Resolution Time: The usual time it takes to address a record that violated a policy.
- Pipeline execution takes longer to allow policy checks to happen.
- How well the tool detects genuine violations of policy is its accuracy.

To identify the compliance criteria, requirements from the CIS AWS Foundations Benchmark, NIST 800-53 and custom organizational policies were used.

5.2. Comparison Before and After Implementation

The following table demonstrates the state of the system's security before and after adding policy checks to the pipeline.

Table 4 Impact of Introducing Company-wide Policies on Security in CI/CD

Metric	**Before (No Policy Enforcement)**	**After (Policy-Driven Pipeline)**
Avg. Policy Violations/Deployment	12	2
Compliance Rate	68%	94%
Mean Remediation Time	14 hours	1.5 hours
False Positive Rate	5%	2
Pipeline Execution Time	6 mins	7.2 mins

As shown, policy integration led to a dramatic drop in violations and faster resolution times**,** with minimal impact on deployment speed.

5.3. Lowering the Risk of Attacks

Prior to applying policy checks, there were many common mistakes in the configuring of systems.

- S3 buckets without using encryption
- Roles in IAM that allow permission for everything marked with (*)
- Security groups that allow everyone (0.0.0.0/0)

After being integrated, these weaknesses were seen and handled automatically by the review process and prevented from being released. For this reason:

- The amount of critical misconfigurations was reduced by 85%.
- The number of top code issues fell by 70%.
- All infrastructure was developed within the important rules.

This points out that putting Policy-as-Code into your CI/CD process at the start helps catch configuration risks that manual checks might overlook.

5.3.1. Pipeline Execution Time Overheads

Maintaining a fast pipeline is one of the main issues in DevOps. Even with new security measures, pipelines took just an average of 1.2 minutes longer to deploy. Overhead is still considered appropriate given the major enhancements to security and the lower expenses for tackling problems after implementation [14].

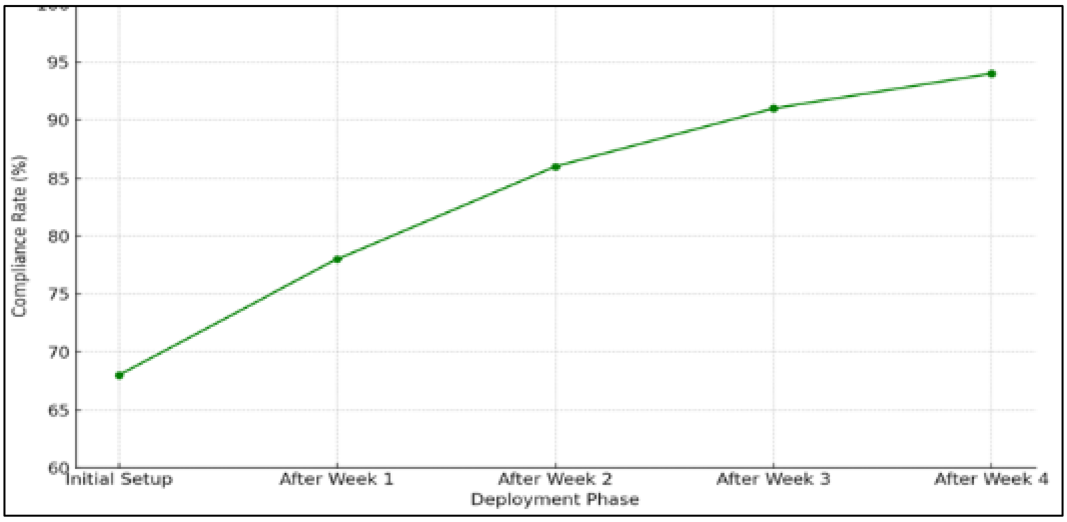

Figure 4 Compliance rate improvement over four weeks

5.4. Conclusion of Evaluation

Integrating Policy-as-Code with CI/CD pipelines makes operations more secure, makes it simpler to be compliant and improves how things are done. Enterprises can:

- Find and solve any problems sooner.
- Remove the need for people to handle some security steps.
- Keep agility while still controlling performance.

Using security policies helps make infrastructure safer and matches well with today's dynamic and flexible methods used in DevOps.

5.5. Experiencing Challenges

While strengthening infrastructure through policies in CI/CD pipelines is a good plan for future enterprise security, there are still some problems and obstacles to keep in mind. Problems arising from these roadblocks may influence how software is implemented, works and how well it is accepted, mainly in cases where the software explores legacy systems.

5.5.1. How hard is it to fit a new HR system with old business practices?

Frequently, the main obstacle is uniting the latest policy security systems with older, existing infrastructure. With no native tooling for Infrastructure as Code (IaC), older systems make it tough for DevOps teams to add continuous policy enforcement during deployment. Also, a lot of legacy applications weren't built to be deployed in sections which is not suitable for the way CI/CD operates. For this reason, work teams may have to create their own connectors or wrappers which takes time and leaves their toolchain more disjointed. Since CI/CD pipelines have many moving parts, it can slow down their adoption and lower the expected pace [15].

Many false positives and policy misconfigurations can be found during this step of the process. Even the most capable policy engines depend entirely on the strength of their embedded rules. If policies aren't well established, the tool may mark some safe designs as harmful. Enforcing such methods will cut off pipelines, confuse developers and result in exhaustion from alerts.

As a result, setting up policies carelessly may either prevent vital services from being deployed or fail to prevent the inclusion of insecure systems. To maintain security while keeping the deployment process flexible, teams from security and DevOps should be constantly collaborating.

5.5.2. Implications of Using Resources and Estimating Cost

Regularly scanning policies in large systems can use a lot of computing power. It takes up CPU and memory to scan complicated IaC configurations, validate them and enforce the use of rules instantly.

For CI/CD platforms on the cloud, charging for every build minute or compute cycle drives the overall operational budget higher. It is important for enterprises to evaluate their infrastructure size, how complex their policies are and how often they need to deploy to maintain efficiency [1].

5.5.3. Overcoming the Pressure of Workplace Resistance

The biggest problem is cultural, not technical. When policy gates are added to automated processes, developers often disagree. Policy enforcement could be viewed by developers as something slowing down their development or limiting how they manage their tasks.

Having policy-driven pipelines calls for all teams to take shared responsibility for development, security and operations. In companies where teams or roles are divided and inflexible, adopting DevSecOps can become hard. For adoption to work well, teams need proper training, a planned approach to change and the support of leaders aiming for fast, secure and stable delivery.

Table 5 Complications related to making infrastructure secure by using CI/CD and policy-based systems

Challenges	**Description**
Integration with legacy systems	Legacy tools lack laC support; custom integration increases complexity
False positive and policy errrors	Poorly defined policies trigger false alerts or allow insecure configurations.
Resource consumption and cost	Policy checks use more compute, raising build times and cloud resource costs.
Organizational resistance	Teams may resist changes; requires DevSecOps culture and leadership buy-in to succeed.

Still, there are challenges associated with making infrastructure secure through policies in these environments. The difficulties must be handled by ensuring there is a thoughtful mix of, accurate policy design, smart resource management and a DevSecOps culture that pulls everyone in.

6. Recommendations For Development

Security improvements in enterprise infrastructure are mostly driven by using automation, precision and speed. When a CI/CD environment grows, it's important for policy-driven controls to become smarter, more flexible and capable of collaboration. This portion provides the main steps and strategies for progress in further enhancing security for infrastructure.

We rely on AI and ML in our Intelligent Policy Engines.

Traditional engines for policies follow rules and requirements that do not change [16]. Policymakers should integrate artificial intelligence and machine learning into policy making. Smart policy engines have the ability to:

- Find out about any anomalies in infrastructure as soon as they happen.
- Make plans for suitable remediation by drawing from earlier deployment information.
- Ensure policies are correct by using your results to make adjustments.

As a result, we would see fewer false alarms and enforcement could respond to different situations, better meeting both the needs of developers and improving security.

6.1. Gathering DevSecOps in a Shared Platform

Each tool responsible for security, compliance or deploying code makes handling everything more difficult [17]. Emerging platforms aim to connect all DevSecOps activities with their integration of:

- CI/CD tools
- Infrastructure as Code serves as an abbreviation for infrastructure building with code.
- Policy as Code (PaC)

- Security scanning

These platforms make administration less complicated, simplify how employees are welcomed and provide all policies with a clear view across every stage of software development. Both Backstage by Spotify and OPA in GitHub Actions are early signs that this trend is developing.

6.2. Policies are Enforced at the Beginning of the Development Process

Security needs to be added at the start of the process so mistakes in settings are caught earlier. The idea of shift-left encourages making policy validation an integrated process:

- In code editors, for example, VS Code extensions
- At the time of a pull request
- As part of using git - hooks

Real-time feedback instructs developers to code securely and cuts down on rework in reviews that come later. It is encouraged for enterprises to use policy gates that are simple for developers to use and cause no major disturbance.

6.3. Standardization and Open Policy Frameworks are part of 8.4.

Because Policy-as-Code is being implemented by an increasing number of companies, we need industry-wide standards. Examples of open frameworks are:

- Open Policy Agent or OPA
- HashiCorp Sentinel
- Cloud Custodian

These terms are being adopted as the main way people communicate. In the future, important policies should support various cloud services and CI/CD systems. Adopting modular and reusable policy templates quickly and ensures that decisions follow the same method.

6.4. Recommendations for the Organization

In addition, to safeguard infrastructure into the future, enterprise should:

- Make sure your teams have the skills needed for IaC, PaC and security.
- Set up a way to manage policies that keep pace with advancements in your system: version each policy, verify it and care for its maintenance.
- Bring DevOps, Security and Compliance team members together to work towards the same goals and Key Performance Indicators (KPIs).

Taking these actions builds a secure environment that enables the technology to be adopted long term.

7. Conclusion

This work has looked at how adding policy-driven infrastructure hardening to CI/CD pipelines is an important and strategic step in enterprise DevOps. The study has shown that by studying automation, compliance and security together, security policies are now an essential part of software development from start to finish.

The main value of this work is in explaining how merging Infrastructure as Code, Policy-as-Code and Jenkins, GitHub and GitLab CI/CD contributes to security in large organizations. Its researchers looked into Open Policy Agent and HashiCorp Sentinel, both of which are used to ensure compliance with policies before production work gets released.

Policy-driven infrastructure hardening adds important qualities such as reliability, repeatability and transparency to how infrastructure is set up. It also matches how we manage infrastructure to regulations, our organization and security needs, without slowing new developments. The importance of this balance increases because modern companies need to work fast and keep systems safe.

Even though integration, use of resources and staff groups can be difficult, the path moving forward is obvious. DevSecOps practices improved by intelligent policy engines and focus on security at the beginning of the development pipeline will lead to more security and resilience for a business.

All in all, policy-driven hardening moves past just improving infrastructure; it brings policy, software and automation together in a combined effort to protect enterprises.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] N. Thallapally, "Implementing continuous integration and continuous deployment (CI/CD) pipelines," International Journal of Science and Research Archive, vol. 3, no. 2, pp. 248–253, Oct. 2021, doi: 10.30574/ijsra.2021.3.2.0073

[2] Soni, M. (2015, November 1). End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. https://doi.org/10.1109/ccem.2015.29

[3] Katikireddi, P., Singirikonda, P., & Vasa, Y. (2021). REVOLUTIONIZING DEVOPS WITH QUANTUM COMPUTING:ACCELERATING CI/CD PIPELINES THROUGH ADVANCED COMPUTATIONAL TECHNIQUES. Innovative Research Thoughts, 7(2), 97–103. https://doi.org/10.36676/irt.v7.i2.1482

[4] Iqbal, S., Khan, S., Daghighi, B., Wahab, A. W. A., Kiah, M. L. M., & Anuar, N. B. (2016). Service delivery models of cloud computing: security issues and open challenges. Security and Communication Networks, 9(17), 4726–4750. https://doi.org/10.1002/sec.1585

[5] Mysari, S., & Bejgam, V. (2020). Continuous Integration and Deployment Pipeline Automation Using Jenkin Ansible. 2020. https://doi.org/10.1109/ic-etite47903.2020.239

[6] Vugt, S. V. (2014). Hardening SUSE Linux (pp. 131–159). apress. https://doi.org/10.1007/978-1-4302-6820-8_6

[7] Smith, R. (2005). Linux in a Windows world. Choice Reviews Online, 43(02), 43–0992. https://doi.org/10.5860/choice.43-0992

[8] Brickner, D. (2005). Test driving Linux: from Windows to Linux in 60 seconds. Choice Reviews Online, 43(03), 43–1607. https://doi.org/10.5860/choice.43-1607

[9] Y.Avuthu, "Change management and rollback strategies using IaC in CI/CD Pipelines," International Journal of Science and Research Archive, vol. 2, no. 1, pp. 160–168, Apr. 2021, doi: 10.30574/ijsra.2021.2.1.0037

[10] Damage modeling framework for resilience hardening strategy for overhead power distribution systems. Reliability Engineering & System Safety, 207, 107367.

[11] https://doi.org/10.1016/j.ress.2020.107367

[12] Zampetti, F., Geremia, S., Bavota, G., & Di Penta, M. (2021). CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study. 471–482. https://doi.org/10.1109/icsme52107.2021.00048

[13] Samarawickrama, S. S., & Perera, I. (2017). Continuous scrum: A framework to enhance scrum with DevOps. 1–7. https://doi.org/10.1109/icter.2017.8257808

[14] Vignesh, S., & Kanna, B. R. (2020). AWS Infrastructure Automation and Security Prevention Using DevOps (pp. 537–549). springer singapore. https://doi.org/10.1007/978-981-15-0199-9_46

[15] Dawood, T., Elwakil, E., Novoa, H. M., & Delgado, J. F. G. (2020). Pressure data-driven model for failure prediction of PVC pipelines. Engineering Failure Analysis, 116, 104769. https://doi.org/10.1016/j.engfailanal.2020.104769

[16] Hagood, W. O., & Friedman, L. (2002). Using the Balanced Scorecard to Measure the Performance of Your HR Information System. Public Personnel Management, 31(4), 543–557. https://doi.org/10.1177/009102600203100410

[17] Zhou, Y., Yu, Y., & Ding, B. (2020). Towards MLOps: A Case Study of ML Pipeline Platform. 494–500. https://doi.org/10.1109/icaice51518.2020.00102

[18] Toh, M. Z., Mahrin, M. N., & Sahibuddin, S. (2019). Adoption Issues in DevOps from the Perspective of Continuous Delivery Pipeline. 123, 173–177. https://doi.org/10.1145/3316615.3316619