Gomboc

Turning Code into Control:

The Path to Secure IaC

TABLE OF CONTENTS

04 The IaC Maturity Model: Overview and Dimensions

05 Level 1: Emergent

06 Level 2: Reactive

07 Level 3: Proactive

08 Level 4: Adaptive

09 Level 5: Resilient

10 DevSecOps Maturity and IaC: A Symbiotic Journey

10 Business Benefits of IaC Maturity

12 Common Blockers and How to Overcome Them

14 Gomboc in Action: Accelerating Maturity with Automated Remediation

15 About Gomboc

Infrastructure as Code (IaC) has rapidly become the backbone of modern cloud operations.

With its ability to turn infrastructure into repeatable, version-controlled, and testable code, IaC empowers teams to move faster, deploy more reliably, and scale with confidence. As businesses embrace digital transformation and cloud-native development, IaC adoption is no longer a luxury. It's a necessity.

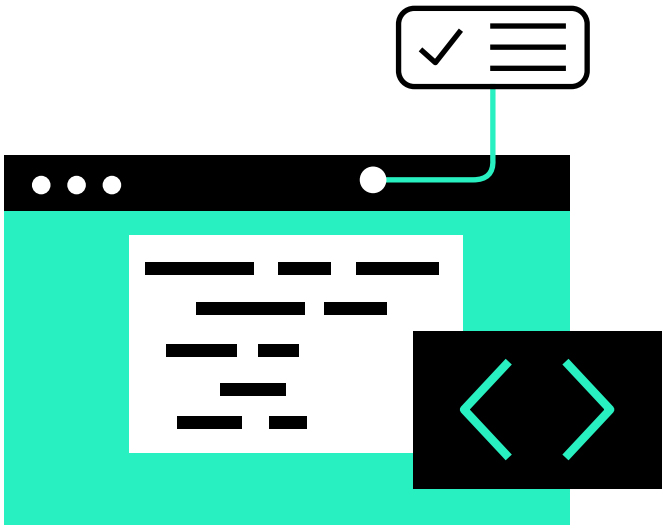
Yet, with this newfound speed and flexibility comes a hidden cost.

The very characteristics that make IaC so powerful are its speed, dynamism, and accessibility, which also introduce new complexities and risks. Cloud environments are ephemeral by design, capable of changing with a single line of code. In such fluid ecosystems, a single misconfiguration can lead to exposure, compliance violations, or worse, a costly breach. Traditional approaches to securing infrastructure are too slow and manual to keep up. Relying on downstream audits or ticket-based remediation simply can't scale in environments where infrastructure changes hundreds of times daily.

Security teams are stuck playing catch-up, while developers are left to interpret vague policy requirements, often without the context, time, or expertise to get it right. This disconnect creates bottlenecks, slows innovation, and accumulates risk over time.

To navigate these challenges, organizations need a framework that captures where they are today and guides where they need to go. The IaC Maturity Model offers exactly that: a lens through which teams can evaluate their current state of infrastructure management, security integration, and operational efficiency.

The model helps organizations benchmark their practices by defining five progressive stages and charting a realistic path toward secure, scalable, and automated infrastructure.



THE IAC MATURITY MODEL: OVERVIEW AND DIMENSIONS

Infrastructure as Code is no longer a niche practice.

It has become foundational to how modern engineering teams build, manage, and scale cloud environments.

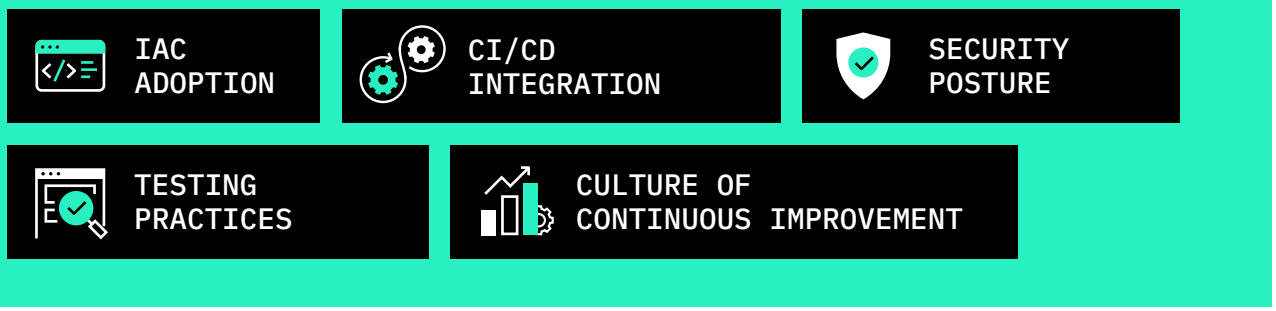
However, not all IaC adoptions are created equal. Some teams are just beginning to define infrastructure in code, while others have embedded it deeply into secure, automated workflows.

To help organizations understand where they stand—and what progress looks like—we introduce the IaC Maturity Model.

This model outlines five levels of maturity, each representing a step forward in how infrastructure is defined, deployed, secured, and improved over time. Rather than focusing solely on tooling, it captures a broader view of organizational capability, including process, security alignment, and automation.

IAC MATURITY MODEL	IAC ADOPTION	CI/CD	SECURITY POSTURE	TESTING	CONTINUOUS IMPROVEMENT
Level 1: Emergent	Limited	None	None	None	None
Level 2: Reactive	Ad Hoc	Version Control (no CI)	Investigating	By Usage	Ad Hoc
Level 3: Proactive	Developing	Linting and Review (no CD)	Basic Hygiene	Investigating	Measuring (Dora)
Level 4: Adaptive	Full Adoption	Full CI/CD	Basic Security	Developing	Reporting
Level 5: Resilient	Drift Detection	Developer Enablement via Automation	Enhanced Security	Automated	Improvement

At the heart of the model are five key dimensions:



Each dimension reflects a core aspect of operating cloud infrastructure at scale, providing a lens for evaluating maturity and identifying the practices needed to advance.

LEVEL 1: EMERGENT

Every journey begins with a first step, and for many organizations, that step into Infrastructure as Code is informal and experimental. At this Emergent stage, teams are just beginning to explore IaC and its potential. Cloud infrastructure is provisioned manually, often through scripts or command-line interfaces. Changes are applied directly to cloud environments without any version control or deployment automation in place.

There may be isolated attempts to define resources in code, but practices are inconsistent and ungoverned. Infrastructure lives in the cloud rather than repositories, making it difficult to track changes, enforce standards, or even know what exists at any given time. As a result, environments are prone to drift, duplication, and shadow infrastructure that slips out of view.

Security, if considered at all, is reactive. Teams might respond to alerts or incidents as they arise, but there are no guardrails to prevent misconfigurations before they happen. Without visibility or a formal process, risk accumulates quietly in the background.

This stage is not inherently negative, though it may initially sound like it. It reflects a phase of discovery. Organizations are testing tools, learning how cloud infrastructure works, and beginning to understand the benefits of codifying their environment. The key challenge is that without structure, the pace of change can quickly outstrip the ability to manage it safely.

LEVEL 2: REACTIVE

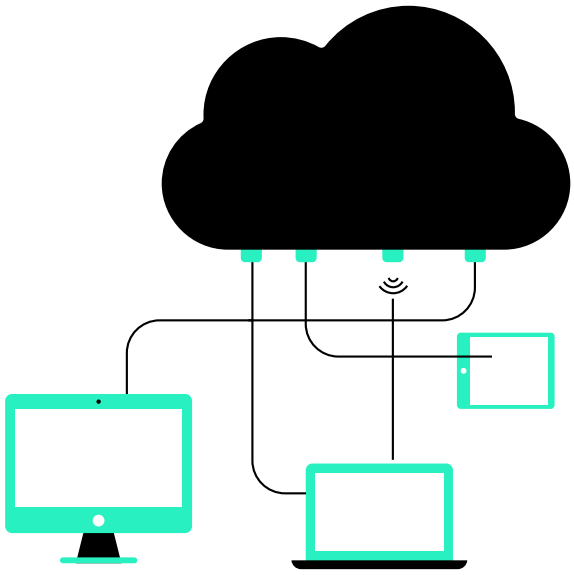
The move from experimentation to early structure marks a turning point in an organization's Infrastructure as Code journey. At the Reactive stage, teams have embraced defining infrastructure in code and taken foundational steps to improve visibility and control. Most notably, they have introduced version control for IaC, typically using Git, to track changes and collaborate more effectively.

This shift introduces consistency and accountability, which were missing in the Emergent phase. Infrastructure can now be reviewed, reverted, and shared across teams. However, deployments are still largely manual. While the infrastructure is written as code, it is not yet connected to continuous delivery pipelines. Teams may apply changes through one-off commands or scripts, but automation is limited or absent.

Security considerations begin to enter the conversation but are usually addressed after the fact. Reviews may catch obvious misconfigurations, or alerts may prompt teams to investigate issues in production. Still, there is no consistent or proactive mechanism to ensure security is built into infrastructure before deployment. Compliance, too, is often seen as a checkbox to revisit during audits rather than a continuous responsibility.

This stage reflects a growing awareness of what is possible but also reveals the limitations of partial adoption. Teams are working toward better practices, but progress remains uneven without automation, enforcement, or shared standards. The groundwork is there, but the tools and processes have not yet matured into systems supporting scale, speed, and security.

Organizations at the Reactive level are poised for transformation. They have the building blocks in place and are beginning to recognize the need for automation, security integration, and more efficient ways to manage change. The next step is turning good intentions into repeatable, enforceable practices.



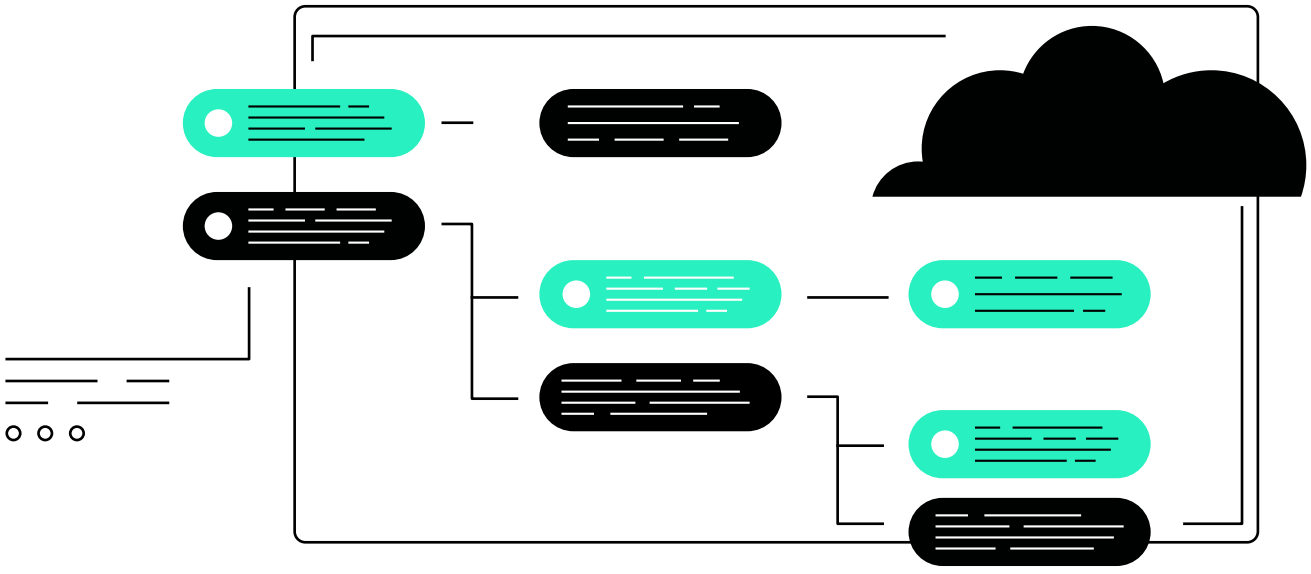
LEVEL 3: PROACTIVE

At the Proactive stage, Infrastructure as Code is no longer an experiment. It has become the standard for managing cloud infrastructure across teams. Most environments have been migrated from manual setups to code-based definitions, bringing greater consistency and control.

With broader adoption comes a stronger emphasis on quality and security. Static analysis tools, such as linters and basic scanners, are introduced to catch errors and misconfigurations early. Infrastructure code is now reviewed more systematically, following patterns similar to application development.

Security starts to shift from reactive to preventive. Teams begin referencing frameworks like CIS and NIST to shape how infrastructure is written and validated. While adoption may still be uneven, there is a growing awareness that security should be built into the development process, not bolted on afterward.

Basic CI integration supports this evolution. Infrastructure code is automatically tested and validated before deployment, reducing the risk of changes breaking environments. Though full delivery automation may still be developing, CI lays the groundwork for more mature practices ahead.



This stage marks a clear turning point. Organizations begin to move beyond manual safeguards and toward structured, scalable processes. The focus shifts from writing code to managing it well, with security in mind.

LEVEL 4: ADAPTIVE

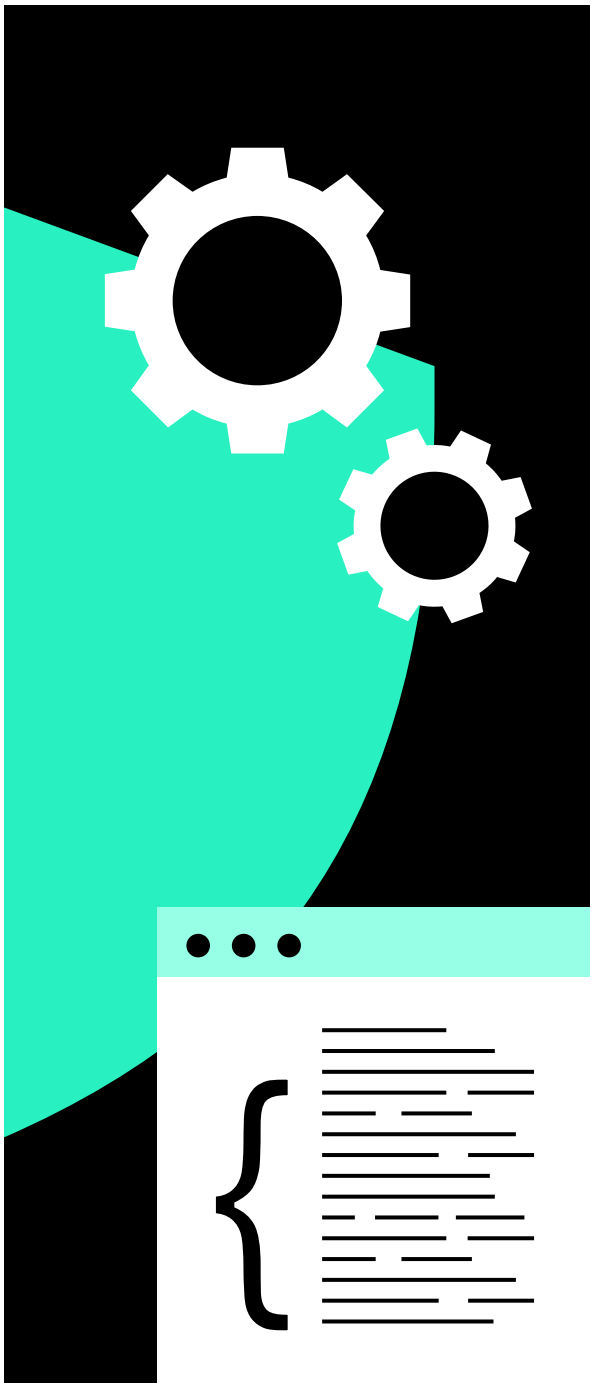
At the Adaptive stage, infrastructure management becomes deliberate and scalable. CI/CD pipelines are fully established, enabling infrastructure changes to flow through automated, testable, and repeatable processes. Teams no longer rely on manual gates as deployment is governed by well-defined workflows.

Security shifts from guidance to enforcement. Policy-as-code introduces automated guardrails that evaluate every change against defined rules. If a proposed configuration violates policy, it's flagged before reaching production. This ensures consistent adherence to security standards without slowing development.

Security shifts from guidance to enforcement. Policy-as-code introduces automated guardrails that evaluate every change against defined rules. If a proposed configuration violates policy, it's flagged before reaching production. This ensures consistent adherence to security standards without slowing development.

Uniform security baselines are applied across teams, reducing inconsistency and risk. Lab environments provide a safe space to test changes before rollout, helping catch issues early. Organizations at this level begin tracking DORA metrics such as Mean Time to Remediate and deployment frequency. These insights highlight performance and guide improvements across both engineering and security practices.

The Adaptive stage reflects a balance between speed and control. Infrastructure is managed as a system, with security built-in, not added later.



LEVEL 5: RESILIENT

Infrastructure as Code is no longer just a practice at the Resilient stage. It is an operating model. Automation, security, and scalability are fully embedded across the organization. Infrastructure is managed with the same rigor as software, governed by policies, continuously validated, and designed to adapt to change without disruption.

Drift detection and remediation are now standard. The system can identify when deployed infrastructure diverges from its intended state and automatically propose or apply fixes. This ensures that environments remain secure and compliant over time, even as cloud services evolve or external teams make changes.

Testing is no longer limited to syntax or security checks. Full end-to-end validation ensures that infrastructure changes behave as expected before they reach production. Fix verification becomes part of the feedback loop, giving teams confidence in every deployment.

Developer enablement also takes center stage. Teams can provision and manage their own infrastructure through self-service tools, with guardrails in place to prevent misconfigurations. This balance of autonomy and control allows teams to move quickly without compromising security or consistency.

Finally, infrastructure security becomes tightly integrated with broader operational workflows. Connections with the Security Operations Center (SOC) and vulnerability management processes ensure that infrastructure risks are monitored, prioritized, and addressed as part of the larger security strategy.

> **Resilience at this level is not just about preventing failure but building systems that adapt, recover, and improve over time. It reflects an organization where infrastructure, development, and security move together, guided by automation, trust, and continuous feedback.**

DEVSECOPS MATURITY AND IAC: A SYMBIOTIC JOURNEY

As organizations progress in their Infrastructure as Code practices, they often advance along the DevSecOps path.

These two journeys are tightly linked. Improving one tends to accelerate the other.

IaC maturity enables DevSecOps by making infrastructure more consistent, visible, and collaborative. Once infrastructure lives in code, it can be reviewed, tested, and integrated into pipelines like application code. This shared workflow brings development, security, and operations onto the same page.

Policy-as-code adds the next layer. Instead of relying on manual checks, security requirements are codified and enforced automatically. Developers get early feedback when something falls outside policy, enabling secure decisions without extra back-and-forth. Security shifts from a gatekeeper to a guide.

Automation is the necessary glue that ties it all together, helping scale to handle the size and complexity of the problem as the organization matures. It ensures that security standards are applied consistently and that environments stay aligned with expectations. Remediation, compliance checks, and testing become part of the system rather than an afterthought.

BUSINESS BENEFITS OF IAC MATURITY

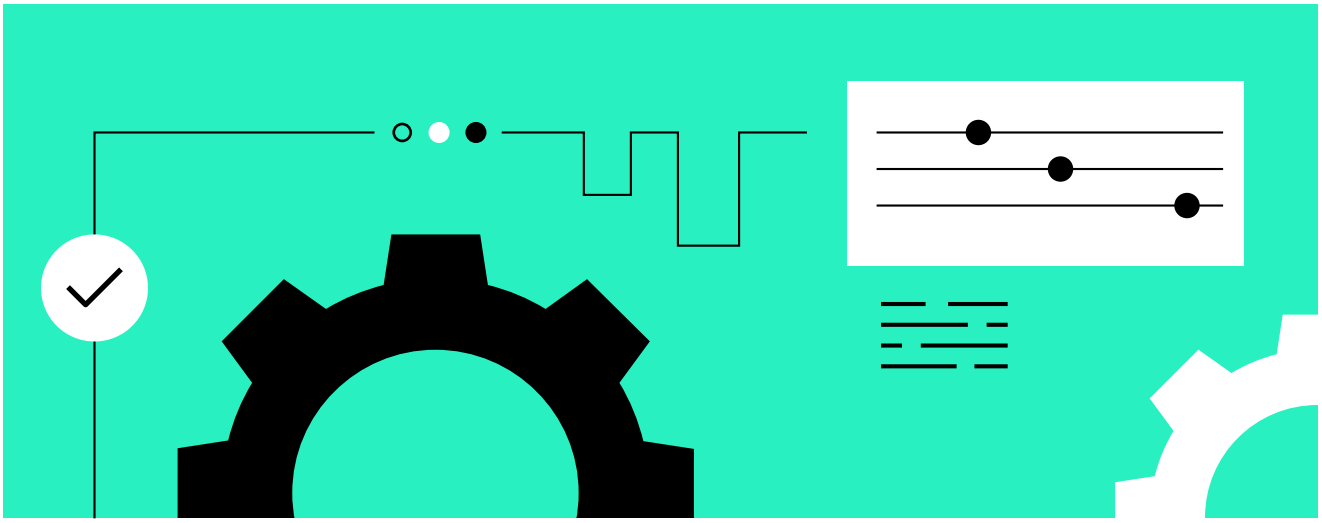
Maturing IaC practice is not just a technical achievement. It delivers meaningful business impact. As organizations climb the IaC maturity ladder, they unlock efficiencies that translate directly into cost savings, reduced risk, and faster execution.

One of the most immediate benefits is cost and risk reduction. Mature IaC reduces the need for time-consuming manual work, freeing engineers to focus on innovation rather than troubleshooting or rework. At the same time, it lowers the likelihood of misconfigurations that can lead to outages, data exposure, or compliance violations. Preventing these incidents before they happen avoids the high financial and reputational costs of security failures.

IaC maturity also supports faster time-to-market. When infrastructure is versioned, tested, and deployed through automated pipelines, new environments can be provisioned in minutes, not days. Teams can launch features, respond to customer needs, and scale operations without waiting on infrastructure bottlenecks.

From a compliance standpoint, mature IaC makes proving that systems meet required standards easier. Audit readiness improves because environments are built consistently, governed by policy, and tracked through code. This reduces the burden on compliance teams and accelerates responses to regulatory demands.

Finally, as automation and guardrails take hold, teams become less dependent on specialized security expertise. Security best practices are enforced through code, allowing generalist engineers to operate safely within defined boundaries. This eases hiring pressure in a tight talent market and helps scale secure development across more teams.



COMMON BLOCKERS AND HOW TO OVERCOME THEM

As organizations work to mature their Infrastructure as Code practices, they often encounter familiar challenges. These blockers don't stem from a lack of motivation, but from gaps in capacity, alignment, or tooling that slow progress and increase risk. Recognizing these obstacles is the first step toward overcoming them.

Here are four of the most common blockers teams face on the road to IaC maturity:

ALERT FATIGUE AND TICKET OVERLOAD

Security tools often flood teams with alerts, leaving engineers to sort through hundreds—or thousands—of issues without clear prioritization or resolution paths. This creates a backlog that rarely gets cleared, leading to burnout, missed risks, or both.

FRICTION BETWEEN SECURITY AND ENGINEERING

Collaboration breaks down when security is viewed as a blocker rather than a partner. Developers are asked to fix issues they don't fully understand, while security teams struggle to enforce standards without disrupting delivery timelines.

LACK OF CLOUD SECURITY EXPERTISE

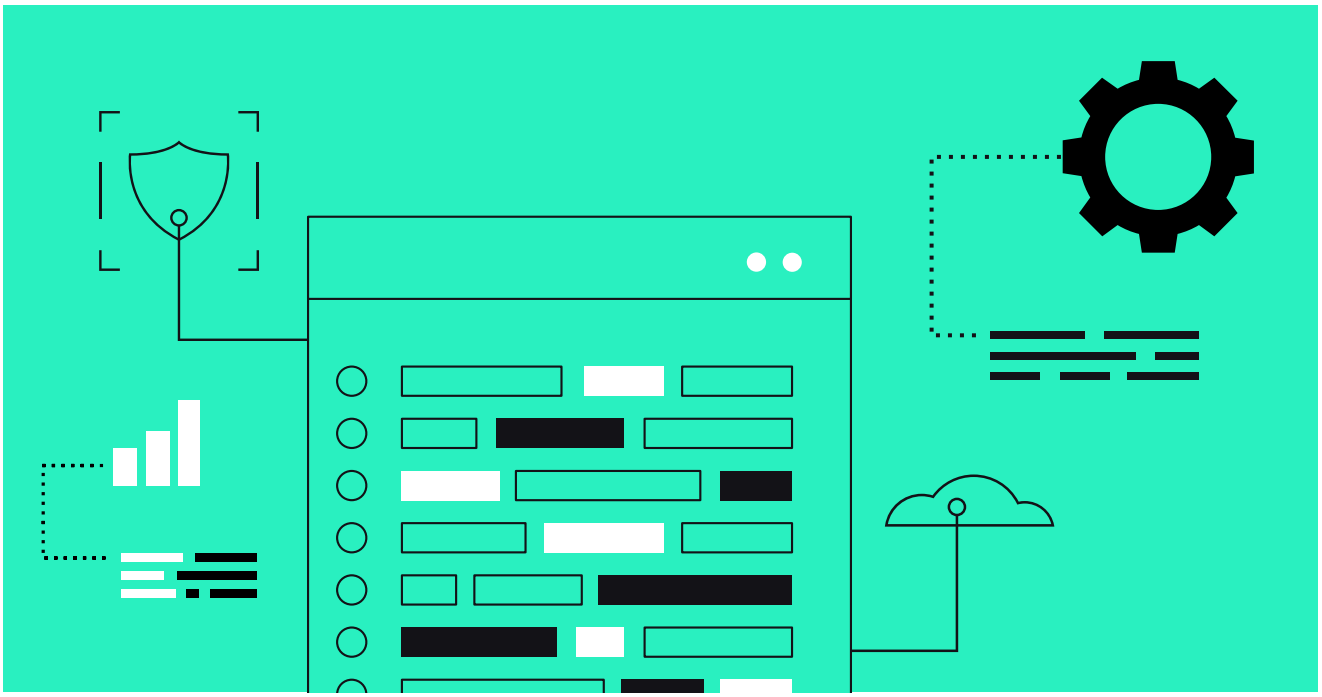
Secure infrastructure requires a deep understanding of cloud provider configurations, best practices, and evolving risks. However, hiring and retaining cloud security experts is increasingly difficult, leaving generalist teams unable to navigate complex issues without enough support.

GOVERNANCE WITHOUT ENFORCEMENT

Many teams define policies and best practices but lack mechanisms to ensure they are actually followed. Even well-documented standards can be ignored or inconsistently applied without enforcement across teams and environments.

These blockers are not unique to any one team or industry. They show up in nearly every organization trying to scale IaC securely. Addressing them requires a combination of automation, integration, and cultural alignment—solutions that move beyond visibility and into action.

The good news is that these challenges can be resolved with the right systems. Automation, particularly in policy-driven, code-based remediation, can turn these blockers into opportunities for lasting improvement.



GOMBOC IN ACTION: ACCELERATING MATURITY WITH AUTOMATED REMEDIATION

For many organizations, the biggest gap in their Infrastructure as Code journey is not visibility—it's action. Security tools may surface misconfigurations, but turning those alerts into safe, reviewable code changes is where progress often stalls. That's where Gomboc steps in Gomboc was built to remove the friction between security enforcement and engineering velocity. Rather than relying on manual tickets or vague recommendations, it translates policy violations into ready-to-merge pull requests. These fixes are context-aware, align with existing infrastructure such as code tooling, and fit directly into the workflows engineers already use.

This approach eliminates the common tradeoff between security and speed. Developers don't need to decipher alerts or write fixes from scratch. Instead, they receive clear, auditable changes within their version control system—just like any other code review. Remediation becomes as simple as approving a PR.

Gomboc also addresses the problem of infrastructure drift. It continuously scans both the intended state (in code) and the deployed state (in the cloud) to identify discrepancies. When drift is detected, Gomboc generates a fix that realigns the environment with defined policies, helping organizations maintain compliance without manual intervention.

Every change Gomboc proposes includes audit-ready documentation. This ensures teams can track what was fixed, why it was necessary, and how it aligns with organizational policies or external frameworks. Over time, this creates a living history of security improvements embedded in the codebase itself.

Every change Gomboc proposes includes audit-ready documentation. This ensures teams can track what was fixed, why it was necessary, and how it aligns with organizational policies or external frameworks.

Over time, this creates a living history of security improvements embedded in the codebase itself.

ABOUT GOMBOC AI

Gomboc AI is a developer-first platform that helps teams fix cloud infrastructure misconfigurations fast before they break production. Founded in 2022 and based in New York, Gomboc replaces alert fatigue and manual security backlog triage with automated, ready-to-merge pull requests that align with your infrastructure, policies, and team conventions.

By integrating directly into Git workflows, Gomboc enables platform and DevOps teams to reduce mean time to remediate from months to minutes, cut misconfiguration-related costs by up to $100K per cloud workload, and resolve 20% of the CSPM backlog in just two days, a process that traditionally takes DevOps teams 20 days. With Gomboc, security becomes an integral part of the development process, enhancing efficiency without compromising speed.

SCHEDULE A DEMO

Got Questions?
TALK TO AN EXPERT

GOMBOC IS CERTIFIED AND COMPLIANT WITH