

SECURE BY DESIGN:

A PLAYBOOK FOR
AI-ASSISTED
CODING

snyk

AI coding assistants like GitHub Copilot and Google Gemini Code Assist transform how development teams write software. These tools can significantly accelerate delivery by reducing boilerplate, suggesting functions, and even scaffolding entire components. But that speed comes with a new class of risks. **Recent studies show that nearly 30 to 55%** of AI-generated code contains vulnerabilities, reflecting a trade-off between productivity and secure coding practices. As organizations race to embrace these tools, the challenge becomes clear: how do you unlock the promise of AI-assisted development without introducing unnecessary security debt? This new, high-velocity challenge is compounded by the persistent, massive security backlog that already slows down innovation and increases risk. A complete strategy must address both the future and the past.

This playbook is designed to answer that question. It offers a practical, step-by-step approach to deploying AI development tools securely and scalably, ensuring that innovation doesn't come at the expense of trust. Whether your teams are experimenting with AI assistants or already deploying them across the organization, this guide will help you implement the right guardrails in the right places and at the right time.

With the Snyk AI Trust Platform, organizations can move beyond policy documents and training reminders. Snyk translates trust policies into enforceable, real-time protections embedded directly into developer workflows from the IDE to pull requests and CI/CD pipelines. These guardrails work with developers, not against them, helping to surface risks earlier, reduce rework, and maintain delivery velocity.

Here's what you can expect inside this guide:

- A maturity model to benchmark and grow your AI security posture.
- Tactical strategies for integrating guardrails into IDE, PR, and CI/CD workflows.
- Approaches for developer enablement and access control that support adoption.
- Real-world use cases for Snyk Guard, Agent, Assist, and Studio.
- Actionable metrics to measure effectiveness without slowing teams down.

Whether you're a security leader trying to scale governance or a platform engineer supporting developer workflows, this playbook will help you build a secure-by-default foundation for the next generation of software development.

WHY GUARDRAILS MATTER NOW

AI coding assistants are no longer experimental but a staple in modern software development. Tools like GitHub Copilot and Google Gemini Code Assist are becoming default extensions in developers' workflows, dramatically increasing speed and productivity. But that speed comes with a catch. A growing body of research has found that over 30% of AI-generated code contains vulnerabilities, many of which stem from insecure patterns that developers may not even recognize as risky.

The problem isn't that AI is inherently unsafe. It's that the speed and volume of AI-generated code leave little room for manual review or catch-up security. Vulnerabilities introduced early can slip through into pull requests, pipelines, and production environments before traditional scanners even run, forcing costly rework and delays.

This is why guardrails matter. With Snyk's IDE plugins and MCP Server, AI-generated code is scanned and validated as soon as it reaches the developer. This ensures it is secure at inception before it ever enters a pull request or pipeline, rather than relying on later-stage checks to catch up.

Guardrails aren't about slowing teams down; they're about embedding security into the development experience so that issues can surface before they become problems. They work with AI tools, not against them, adding just enough friction in the right places to help developers course-correct in real time. For developers, this means fast, helpful AI assistance with built-in safety checks. For enterprises, it means traceability, compliance, and control without sacrificing velocity.

Snyk's AI Trust Platform brings this vision to life. By embedding policy-driven protections directly into IDEs, pull request flows, and CI/CD pipelines, Snyk ensures that AI-assisted development remains scalable and secure. Guardrails reconcile the needs of builders and defenders, empowering developers to move fast while giving security teams the visibility and enforcement they need to stay ahead of risk.

MAPPING GUARDRAIL MATURITY

Securing AI-assisted development is a progression. As teams evolve their DevSecOps practices over time, their approach to governing AI-generated code must mature as adoption grows. That's why it's critical to understand where your organization is today, and what actionable steps will move you toward a more scalable, resilient AI security posture.

This maturity model provides a clear framework for doing exactly that. Each level represents a distinct stage in AI governance, from ad-hoc experimentation to dynamic, policy-driven enforcement. No matter where your teams start, this roadmap helps identify gaps, set goals, and apply the right guardrails at the right time.

Level 0 — Unmonitored AI usage. At this stage, developers use tools like Copilot or Gemini without formal oversight. There are no policies in place, no visibility into how or where AI is being used, and no guardrails to prevent insecure code from slipping through. Risk is high and untracked.

Level 1 — Visibility and passive scanning. Teams begin to get a sense of the landscape. AI usage is monitored, and passive scanning is introduced through pull request checks or CLI tools. These scans may catch vulnerabilities post-hoc, but developers aren't yet required to adopt security tooling in their local environments. It's a start, but still reactive.

Level 2 — Inline guidance and soft enforcement. This is where the shift from passive observation to active guidance begins. Developers use IDE plugins like Snyk's to receive real-time feedback as they code. Pull request checks are enforced more consistently, and conditional access rules or audit logging begin to nudge behavior. Guardrails become part of the workflow, not a bolt-on. Unlike IDE plugins, which require voluntary developer adoption, pull request and CI/CD guardrails act as hard enforcement gates, ensuring security standards are met before code proceeds.

Level 3 — Full lifecycle policy enforcement. Guardrails are now embedded across the entire SDLC, from code generation to deployment. AI Bills of Materials (AI-BOMs) are tracked to ensure traceability. Organizational policies are enforced automatically through CI/CD, and network-level restrictions prevent unsanctioned use. Security is no longer optional but built into every step.

Level 4 — Adaptive, agentic governance. At the most advanced stage, AI governance becomes dynamic. Policies are centrally defined but enforced contextually, adapting to AI-generated code's source, intent, or content. Tools like Snyk Studio and Guard provide full auditability, override tracking, and autonomous enforcement across environments. Governance keeps pace with AI's speed without slowing development down.

This model is more than a scorecard. It's a playbook. It gives teams a structured way to build maturity over time, ensuring that security scales with adoption and never becomes a bottleneck.

THE AI GUARDRAILS FRAMEWORK AT A GLANCE

Once you understand where your organization falls on the maturity curve, the next step is knowing how to implement guardrails that drive both security and speed. That's where the AI Guardrails Framework comes in, a holistic architecture that embeds trust and control into every phase of the software development lifecycle.

Consider a flow that begins the moment an AI assistant suggests code. From there, the code moves through pull requests, into CI/CD pipelines, and ultimately toward deployment. At each point, Snyk's AI Trust Stack, which comprises Assist, Agent, Guard, and Studio, applies targeted protections, enforcement logic, and developer support.

What makes this framework uniquely powerful is its comprehensive coverage:

- **Breadth:** Snyk integrates with all major AI coding assistants and IDEs, ensuring developers can work in the tools they already use while benefiting from real-time security insights.
- **Depth:** Guardrails are woven throughout rather than being layered on top. From code suggestion to merge to deploy, Snyk enforces security policies where they matter most.
- **Adaptability:** Whether you're a startup testing Copilot in a single team or an enterprise rolling out Gemini org-wide, policies are fully configurable and designed to grow with your needs.

The result is a framework that meets developers where they are while giving security teams the visibility and control to scale AI confidently. It bridges the gap between speed and safety, so you don't have to choose.

AI GUARDRAILS IN ACTION

With the framework and maturity model in place, the next step is implementing these principles. Guardrails are only effective when embedded directly into the tools and workflows developers use daily. In the following sections, we'll explore how these protections can be applied seamlessly across the development lifecycle, meeting teams where they work, without introducing friction or slowing momentum

IDE GUARDRAILS: INLINE FEEDBACK WHERE CODE IS WRITTEN

Effective guardrails start where developers interact with code. The Snyk IDE plugin integrates directly into environments like VS Code, IntelliJ, or Cursor, scanning AI-generated code as it appears. Developers see issues flagged in real time, with context-aware guidance and fix suggestions before a commit or pull request happens.

This developer-first approach keeps security visible without breaking flow. It builds secure coding habits while ensuring vulnerabilities introduced by AI assistants like Copilot or Gemini are caught immediately. With IDE plugins, security becomes a natural part of coding, inline, lightweight, and frictionless.

SNYK MCP SERVER: ENABLING SECURITY AT INCEPTION

The Snyk MCP server extends guardrails beyond the IDE. With the implementation we guide customers toward, the MCP server can be incorporated into AI-native workflows that ensure AI-generated code is secure before the AI task is even complete. Unlike the IDE plugin, it does not rely on the developer to run scans or apply fixes; the MCP server ensures that insecure patterns in AI-generated code are automatically detected and remediated before code is generated and with little developer involvement.

This enables true security at inception: AI code is validated and fixed as it's created, before reaching version control or pipelines. The MCP server delivers enforceable guardrails that help organizations adopt AI responsibly, ensuring only secure, policy-compliant code ever enters the SDLC.

PR CHECKS AND MERGE-TIME ENFORCEMENT

The pull request is a natural checkpoint for catching security issues before code is merged. Snyk integrates with GitHub, GitLab, and Bitbucket to run automated PR checks, flagging vulnerabilities without disrupting workflows.

With Snyk Agent Fix and IDE Plugins, developers get pre-validated fix suggestions and secure-by-default options that speed up remediation and reduce rework. Instead of rewriting large sections, teams can resolve issues quickly and keep momentum.

Organizations maintain delivery speed by enforcing guardrails at merge time while ensuring only secure code moves forward.

BUILD PIPELINE AND CI/CD DEFENSE

Even with strong IDE and PR checks, risks can still surface during build and deployment. Integrating the Snyk CLI into your CI/CD pipeline provides a final enforcement point, validating code, containers, and infrastructure as code before anything reaches production.

These gates are intentionally designed to create hard stops when a security issue violates organizational policy. If earlier guardrails miss something, whether due to dependency updates, misconfigurations, or drift, CI/CD enforcement ensures non-compliant builds are blocked. This last line of defense protects production environments while giving teams the confidence that what ships is secure and aligned with enterprise standards.

This multi-layer testing helps catch issues introduced late in the process, like dependency changes or config drift, and supports automated policy enforcement to block non-compliant builds.

DRIVING DEVELOPER ADOPTION AT SCALE

Implementing guardrails is only half the equation; ensuring they're adopted is what makes them effective. Even the most well-designed security controls can fall short if they create friction or feel disconnected from how developers work. The following section focuses on how to drive meaningful adoption at scale, making security a natural part of the development process rather than an afterthought. When done right, security doesn't interrupt momentum; it empowers developers to adopt AI tools confidently and safely at speed.

CARROT-FIRST ROLLOUTS

Developer adoption hinges on trust and your rollout strategy, which can make or break it. Rather than leading with restrictions, a carrot-first approach encourages secure habits by pairing access with simple, proactive steps.

One effective tactic is to require lightweight evidence of plugin installation before granting access to AI coding assistants like Copilot or Gemini. This could be as simple as a screenshot showing the Snyk IDE plugin installed or a short attestation completed during onboarding.

By setting this expectation upfront, teams signal that security is part of the process, not a separate concern. It's a low-friction way to establish baseline protections and build shared accountability from day one. Developers get the tools they want, and security teams gain confidence that those tools are being used responsibly. Everyone starts on the same page.

TRUST BUT VERIFY

While a carrot-first approach helps establish early adoption, long-term success depends on visibility and accountability. Once access is granted, teams need a way to ensure secure practices stick without resorting to heavy-handed enforcement.

This is where a "trust but verify" model comes in. Security and platform teams can use IDE and CLI usage reports to monitor whether developers actively use required tools like the Snyk plugin or the Snyk MCP server. These reports can be cross-referenced with AI assistant access logs to identify gaps, such as developers generating code with Copilot or Gemini but skipping local scans.

Rather than blocking access outright, organizations can use soft nudges, such as friendly reminders, alerts, or auto-generated Slack messages, to prompt action. If needed, enforcement can be escalated progressively, pausing access only when repeated gaps remain unresolved.

This approach reinforces positive behavior while preserving developer autonomy, helping teams scale guardrail adoption without creating unnecessary friction.

SECURE-BY-DEFAULT ENVIRONMENTS

One of the most effective ways to drive adoption at scale is to reduce the setup burden entirely. Instead of asking developers to configure security tools themselves, give them environments where secure defaults are already baked in.

Teams can use preconfigured dev containers complete with Snyk security plugins and AI coding assistants like Copilot or Gemini to ensure that every new project starts with the right tools. These environments streamline onboarding, minimize setup errors, and close early compliance gaps.

Tools like GitHub Codespaces make this even easier, allowing platform teams to distribute ready-to-go development environments with a single template. The result is consistent tooling, faster ramp-up, and less support overhead.

Secure-by-default environments shift the question from “Did the developer install it?” to “*How soon can they start building?*” without sacrificing governance.

POLICY-DRIVEN ACCESS AND GOVERNANCE AUTOMATION

With the right policies in place, the next challenge is enforcement at scale and without disrupting developer workflows. Rather than relying on manual checks or after-the-fact audits, teams can use automation to embed compliance directly into access decisions. The following sections explore how to make that possible using tools and infrastructure that many organizations already have in place.

While many teams will naturally meet policy requirements through secure defaults, these automation techniques exist to support scale and consistency, not to block progress. We aim to build confidence in AI adoption, not barriers to it, wherever possible.

CONDITIONAL ACCESS CONTROLS

Using enterprise tools like Intune, Jamf, or Citrix, organizations can automatically gate access to AI tools. If the required configurations aren't present, such as missing security plugins or out-of-date environments, access can be paused until the issue is resolved.

This kind of enforcement happens quietly in the background, creating a seamless experience for compliant users while ensuring consistent guardrail adoption across the board. It's a scalable way to align access with accountability. No tickets or manual approvals required.

Enterprise distribution tools can also be used to deploy the Snyk security plugins, including the Snyk MCP Server and the necessary configuration to ensure AI-generated code is secure at inception.

NETWORK-LEVEL ENFORCEMENT

While conditional access controls tie tool usage to compliance, network-level enforcement takes things further, ensuring that AI tools can't be used unless security requirements are met. This approach moves beyond monitoring behavior and instead builds security into the infrastructure. By configuring your environment to block API calls to AI assistants like Copilot or Gemini unless local security tooling such as the Snyk IDE plugin or MCP server is detected, you create a technical backstop that's difficult to bypass. If the required tools aren't running, the assistant simply doesn't work.

This method is especially valuable in environments where consistency and control are critical. It removes the need to rely on developer behavior or manual audits and ensures that guardrails are enforced automatically, at the network level.

It makes compliance invisible by baking security into connectivity and ensuring that every AI-assisted workflow starts with protection already in place.

AI-BOM AND AGENTIC GOVERNANCE

As AI-generated code becomes a routine part of development, visibility and traceability become essential. Security teams need to understand what code was written, how, by whom, and under what conditions. Without that context, assessing risk, enforcing policy, or meeting compliance obligations is nearly impossible.

This is where the AI Bill of Materials (AI-BOM) comes into play. Much like a software BOM tracks open source components, an AI-BOM is a comprehensive list that shows all the AI models, tools, and data used in a software application. This can include an AI system's data pipelines, models, training procedures, and operational performance. With this data in hand, organizations can confidently audit, trace, and govern AI contributions.

But visibility alone isn't enough. AI tools operate autonomously and generate code at scale, so organizations need governance that adapts to this new reality. That's the role of agentic governance: a model that treats AI tools as non-human contributors and automatically applies policies to their output without manual intervention.

Snyk Studio makes this possible by enabling AI-BOM generation, prompt inspection, and policy-based decision-making at the organizational level. It allows teams to define, enforce, and audit trust policies across all AI-generated code without slowing down developers or introducing bottlenecks.

THE FULL AI TRUST STACK IN PRACTICE

With governance strategies defined and policy automation in place, the next step is understanding how these guardrails come to life through Snyk's platform. The following section breaks down the components of the AI Trust Stack and shows how each contributes to secure, scalable AI-assisted development in practice.



SNYK GUARD

Snyk Guard acts as the policy enforcement layer for AI guardrails, applying security controls across key stages of the development workflow. Today, it supports enforcement during pull requests and CI/CD pipelines, adding firm security gates that ensure insecure code cannot progress unnoticed, with ongoing expansion toward IDE-level integrations. By embedding policy logic into the tools teams already use, Snyk Guard helps organizations apply consistent standards and scale governance with minimal disruption.

It integrates directly with the tools developers already use, validating posture in real time without adding unnecessary friction. This enables teams to confidently scale secure development, knowing that compliance isn't just encouraged; it's enforced automatically where it matters most.



SNYK AGENT

Snyk Agent brings security closer to the developer by delivering autonomous, in-context remediation. When a vulnerability is introduced, whether during code generation or in a pull request, Snyk Agent offers secure fix suggestions immediately, tailored to the specific issue and environment. Beyond these in-context fixes, Snyk Agent is the engine for AI-Accelerated Remediation, capable of systematically working through an organization's entire security backlog to eliminate existing risk at scale.

By addressing problems early and in place, teams can avoid costly refactors later in the process. It's a smarter, faster way to resolve issues, keeping code secure without slowing developers down.



SNYK ASSIST

Snyk Assist acts as an in-IDE guide, offering contextual education and recommendations exactly when and where developers need them. When a vulnerability is flagged, Assist explains the issue, why it matters, and how to fix it, tailored to the code and environment.

Snyk Assist helps developers make informed decisions without leaving their workflow by delivering support in real time, turning every security event into a learning opportunity.



SNYK STUDIO

Snyk Studio provides a centralized interface for managing AI governance workflows within the Snyk MCP Server. It enables teams to define and test custom security rules for AI-generated code, track metadata like AI Bill of Materials (AI-BOM), and begin implementing prompt inspection and policy logic. While still evolving, Studio lays the foundation for scalable, policy-driven oversight of AI coding assistants across secure development environments.

It empowers security leaders to define trust boundaries, automate enforcement, and maintain audit-ready visibility across all AI-assisted development.

METRICS THAT MATTER — MEASURING GUARDRAIL MATURITY

Having the right guardrails in place is an important step, but it means little if you can't verify they're working. Without clear metrics, even the most well-designed programs can stall, leaving teams unsure of where to improve or how to scale.

Organizations need to track key performance indicators that reflect both adoption and impact to measure progress effectively. For example, monitoring plugin installation and scan frequency helps confirm that security tooling is used consistently in local environments. PR check coverage and the rate of skipped or bypassed checks offer insight into how thoroughly policies are applied during code review.

Remediation speed remains one of the most useful signals for gauging effectiveness. By tracking how quickly vulnerabilities are addressed after discovery, teams can spot areas where developers are resolving issues promptly versus places that may need additional support or enablement. Snyk's telemetry provides visibility into scan activity and fix adoption, allowing organizations to measure whether guardrails are being used consistently and whether issues are being remediated promptly.

To bring this data together, Snyk Analytics offers a centralized view into guardrail adoption and developer behavior. Teams can surface policy gaps, identify risk hotspots, and make informed decisions about where to strengthen their security posture.

These metrics don't just measure maturity; they accelerate it. They turn assumptions into insights, helping teams refine their strategy and scale secure development without guesswork.

VALUE FOR TECHNICAL STAKEHOLDERS

With guardrails in place and progress measurable, it's important to connect these outcomes back to the people making them possible. Whether you're building workflows, writing code, or shaping policy, the impact of AI guardrails is felt differently across roles. The following section highlights how these efforts translate into real value for the developers, security teams, and platform engineers driving secure AI adoption forward.



SECURITY TEAMS

For security teams, the rise of AI-assisted development presents both an opportunity and a challenge. On one hand, it accelerates delivery and empowers developers. On the other hand, it introduces new risks at unprecedented speed and scale. Guardrails offer a way to meet both realities, embedding governance directly into developer workflows without becoming a blocker.

With Snyk Guard, security policies are enforced as active controls during pull requests and CI/CD pipelines, with expansion toward IDE-level integrations. This ensures standards are consistently applied at key development lifecycle enforcement points.

These guardrails also provide visibility through Snyk telemetry and reporting, helping security teams see when vulnerabilities were detected, when fixes were applied, and how remediation activity aligns with policy requirements. This creates audit-ready evidence without requiring manual tracking.

In addition to governing new AI workflows, Snyk empowers security teams to tackle the long-standing problem of security debt. With AI-Accelerated Remediation, they can finally report a measurable, rapid reduction in their organization's overall security backlog, turning a persistent source of risk into a story of success.



DEVELOPERS

For developers, the promise of AI is speed, but without support, that speed can collapse under the weight of security rework. The value of Snyk's guardrails is simple: they help developers fix issues immediately, so problems don't pile up in pull requests or production. The result is less time spent squashing bugs and more time for shipping features.

With Snyk Assist, security guidance appears in real time, right in the IDE. Developers can resolve issues as they code, without context-switching or waiting for reviews. When vulnerabilities surface, PR checks and automated fix suggestions ensure they're handled quickly, keeping work moving forward.

And with Snyk Learn integrated, every flagged issue doubles as a just-in-time learning moment, helping developers grow skills organically, while staying focused on delivery.



PLATFORM ENGINEERS

Platform engineers play a pivotal role in making security scalable and seamless. With the right tools and templates, they can bake compliance into the development environment, so developers don't have to consider it.

Platform teams can provide infrastructure templates pre-wired for security, ensuring every workspace starts with the right plugins, policies, and permissions. With support for the Snyk MCP server, organizations can enforce security in two complementary ways. In synchronous workflows, AI-generated code is evaluated in near real time, enabling secure at-inception guardrails during code generation. In asynchronous workflows, the MCP server can remediate issues automatically in the background, reducing backlog without requiring manual intervention. These modes ensure developers remain productive while security is consistently applied across immediate and deferred workflows.

Onboarding also becomes faster and more consistent. With GitHub Codespaces, dev containers, and MDM tools like Intune or Jamf, platform engineers can deploy secure, standardized environments across teams and geographies without manual setup. The result is governance at scale without slowing down innovation.



LET'S BUILD SECURE AI WORKFLOWS — TOGETHER

The future of software development isn't on the horizon; it's already here. AI-assisted coding is changing how teams build, test, and ship software, bringing new levels of speed and creativity to the development process. But with that speed comes risk, and without the right guardrails in place, organizations can quickly lose control over what's being generated, reviewed, and deployed.

At Snyk, we believe security shouldn't slow you down but enable you to confidently move faster. That's why we champion a secure at inception approach where protections are built into the earliest phase of development, ensuring AI-generated code is secure by the time it reaches the developer. Our AI Trust Platform was built to help teams scale AI adoption safely and seamlessly, embedding protections into the tools and workflows developers already use. From real-time IDE scanning to policy automation, auditability, and fix suggestions, we help you balance trust and velocity.

Whether you're just getting started or scaling AI usage across your organization, we're here to help.

Ready to see it in action?
Talk to us about implementing secure
AI development with Snyk.

