HashiCorp Terraform

# 6 ways Terraform can help secure your infrastructure

Secure your infrastructure by bridging skills gaps, enabling standard workflows, and enforcing policy guardrails with Terraform.

Mitchell Ross, Melar Chen, Mitch Pronschinske, Dan Barr

The way organizations provision infrastructure has significantly changed as they move from dedicated servers to capacity on-demand in the cloud. Homogeneous blueprints of infrastructure owned by IT have grown inefficient and outdated. In the cloud, infrastructure resources must be readily available across a variety of providers.

While infrastructure automation underpins the move to the cloud and the overall modernization of application delivery, this shift exposes organizations to a diverse new set of security challenges. According to the 2023 HashiCorp State of the Cloud Strategy Survey, security ranked as the #1 enabler of multi-cloud success.

Security is important — everyone knows how crucial it is to stay on top of the rapidly changing cloud security landscape. But to do so, organizations must address shortcomings in their traditional provisioning processes, including:

- Slow, error-prone, manual workflows and ticketing systems
- A lack of built-in security controls or secure templates for infrastructure code reuse
- Inconsistent or non-existent policy enforcement processes
- No system to detect non-compliant or drifted infrastructure
- Insufficient auditing and observability

To successfully tackle these issues, teams must first think about how infrastructure and applications interact with two core audiences with very different priorities:

Developers and application teams who consume the infrastructure to deploy and manage their applications. Their priority is to work with infrastructure in an easily consumable way that makes the deployment process easier.

Operators or platform teams who provide the infrastructure in a self-service way for their end developers. These teams solve problems such as, “How do I make the provisioning process repeatable?” “How do I

implement proper policy guardrails?" and "How do I ensure security and compliance for the duration of the resource lifecycle?"

As organizations progress in their cloud journey, it can quickly become challenging to maintain a balance between the needs and wants of these two groups of stakeholders. How can teams preserve productivity for developers while ensuring best practices set by security, compliance, and finance are met across the entire infrastructure estate?

The answer is infrastructure as code (IaC). Tools like HashiCorp Terraform codify infrastructure to make it versionable, scannable, and reusable, ensuring security and compliance are always at the forefront of the provisioning processes. This post offers six fundamental practices for your Terraform workflow that can help ensure secure infrastructure from the first provision to months and years in the future.

---

# Bridging the provisioning skills gap

According to the HashiCorp State of Cloud Strategy Survey, organizations rank skills gaps as the most common barrier to multi-cloud adoption. Platform teams must design workflows with the needs of junior developers in mind. Not only does this help devs get up to speed sooner, but it also protects the organization's infrastructure from security and compliance issues caused by inexperience or a lack of standard processes.

## Simplified workflows

The first way to leverage Terraform is to build your infrastructure using HashiCorp Configuration Language (HCL). The gentle learning curve for HCL is a big reason for Terraform's popularity in the IaC world. Its simple syntax lets you describe the desired state of infrastructure resources using a declarative approach, defining an intended end-state rather than the individual steps to reach that goal. If you'd rather use another programming language, such as TypeScript, Python, Java, C#, or Go, Terraform's CDK (Cloud Development Kit) lets you do just that.

Terraform provides unified provisioning for multi-cloud to reduce many workflows into a single golden provisioning workflow for any type of infrastructure. This allows operators and developers to limit their focus to one segment of the workflow, and reduces misconfiguration from lack of expertise. For instance, once an operator builds, validates, and approves a module, a developer can utilize Terraform's no-code provisioning to provision infrastructure from this module without writing a single line of HCL.

While Terraform provides a single workflow for all infrastructure, we understand that not all infrastructure resources are provisioned through Terraform today. Config-driven import provides an automated and secure way to plan multiple imports with existing workflows (VCS, UI, and CLI) within Terraform Cloud. Developers can import during the Terraform plan and apply stages without needing to access the state file or credentials, enabling a self-serve workflow while keeping resources secure.