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 Homogeneous blueprints of infrastructure owned by IT have grown inefficient and outdated.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.

---

1. 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.

Terraform can also connect directly to a version control system (VCS) to add additional features and improve workflows. For example, Terraform Cloud can automatically initiate runs when changes are committed to a specific branch or simplify code review by predicting how pull requests will affect infrastructure. Terraform runs can also be directly managed by Terraform Cloud via remote operations. These runs can be initiated by webhooks from your VCS provider, by UI controls within Terraform Cloud, by API calls, or through the Terraform CLI.

Robust authentication methods

Credential management plays a key role in ensuring a secure provisioning workflow with Terraform. The days when static passwords and IP-based security were a viable security strategy are long gone. Organizations must adapt their authentication workflows to support a multi-cloud environment. Integrating a proven secrets management solution with automated secrets generation and rotation is a good start. HashiCorp Vault is a popular choice that integrates well with Terraform.

Users can also leverage single sign-on (SSO) and role-based access control (RBAC) to govern access to their Terraform projects, workspaces, and managed resources. These workflows help centralize the management of Terraform Cloud users and other Software-as-a-Service (SaaS) vendors with supported providers including Okta, Microsoft Azure AD, and SAML.

Platform teams also need secure authentication to the providers Terraform interacts with, which can be achieved by implementing just-in-time (JIT) access. Terraform can help with its native dynamic provider credentials, which provide short-lived, JIT access to official cloud providers through the industry standard OpenID Connect (OIDC) protocol. These credentials are unique to each Terraform workload and can be generated on demand for Amazon Web Services (AWS), Microsoft Azure, Google Cloud, and the Vault provider, reducing the risk of potential exposure and reuse.

Terraform also offers Vault-backed dynamic credentials, which combines dynamic provider credentials with Vault secrets engines to offer a consolidated workflow. This approach authenticates Terraform runs to Vault using workload identity tokens generated by Terraform Cloud, then uses Vault secrets engines to generate dynamic credentials for the AWS, Azure, and Google Cloud providers. This authentication method is a significant enhancement for users already using Vault for on-demand cloud access and for any organization seeking to reduce the risks of managing its credentials.

2. Building and reusing secure modules

Writing infrastructure configurations from scratch can be time-consuming, error-prone, and difficult to scale in a multi-cloud environment. To alleviate this, Terraform provides the ability to codify infrastructure in reusable "modules" that contain your organization's security requirements and best practices.

Early in cloud migrations or Terraform adoption, operations teams often deploy Terraform in separate silos across the organization. This isolation can create issues such as duplication of code, non-secure or non-compliant configurations, and inconsistent processes for module creation and consumption. For many organizations, the solution is to set up a central internal Terraform module registry. This lets operators enable developers while limiting security risks by creating golden configurations that have been reviewed, tested, and validated. New Terraform users can also use the public registry to reference and deploy common infrastructure configurations. This registry contains more than 14,000 pre-written modules, providing foundational resources for a wide variety of provisioning efforts.

Artifact governance

Many organizations find their image creation and management processes face similar issues surrounding manual, time-consuming, and siloed workflows. These inconsistencies pose security risks as images lay the foundation for modern infrastructure, networking, security, and applications. HashiCorp Packer helps users mitigate these risks by codifying security and compliance requirements directly into their golden images. Similarly to Terraform, image versions can be reviewed, approved, and published to the HCP Packer artifact registry for reference and consumption.

3. Creating policy guardrails

Rapid provisioning opens up tremendous possibilities, but organizations need to maintain compliant infrastructure and prevent over-provisioning. In the past, these security, compliance, and cost policies required manual validation and enforcement. This process was error-prone and challenging to scale, resulting in bottlenecks in provisioning workflows.

Similar to HashiCorp's approach to provisioning with IaC, policy as code can be used to reduce manual errors, enable scaling, and avoid bottlenecks. HashiCorp's policy as code framework, Sentinel, helps you to write custom policies automatically enforced in the provisioning workflow. Terraform also natively supports the open source policy engine Open Policy Agent (OPA), allowing users to migrate their existing Rego-based policies.

Users getting started can take inspiration from pre-written policy sets by trusted experts in the policy libraries section of the official Terraform Registry. Reusing these pre-written policies streamlines your provisioning workflows and reduces the chance of misconfiguration. Users can also leverage the more than 20 run task partners to directly integrate third-party tools and context into their Terraform workflows such as code scanning, cost control, and regulatory compliance. For example, with the HCP provider's Packer data source, you can easily reference HCP Packer to pull in the most recent version of an image.

4. Enforcing guardrails at the time of provisioning

Terraform enables users to move security and compliance efforts upstream by enforcing guardrails during the provisioning process and automatically validating them against the code. For example, policies might validate that an end user is consuming published modules rather than creating custom code, the infrastructure is tagged for visibility, the data storage location adheres to GDPR, or that storage buckets are not accessible by externally facing IP addresses.

This automatic policy integration into your provisioning workflows can be customized with different enforcement levels:

- **Advisory**, to warn users when a policy breaks
- **Soft mandatory**, where users provisioning need to override the policy to break it
- **Hard mandatory**, where users provisioning are not allowed to break the policy

Terraform users can also use preconditions and postconditions; granular tests that validate your configuration before and after the terraform apply phase. These allow practitioners and module authors to enforce custom conditions, such as checking for correctness during the plan or apply phase or creating a precise software contract for modules. Defining conditions specific to your organization helps capture assumptions for future maintainers and returns useful information on errors to help consumers more easily diagnose and troubleshoot configuration issues.

As organizations scale their multi-cloud infrastructure, they often see an accumulation of resources that are no longer relevant or in use, particularly in testing and development environments. These unused or forgotten resources may be outdated and contain vulnerabilities that pose security risks if not managed properly. Coming soon with Terraform's ephemeral workspaces, you'll be able to define time-to-live policies and automate the cleanup of resources. Once your defined date is reached, Terraform will automatically queue and apply a destroy plan on the workspace, helping to mitigate the risk of outdated resources accumulating in your infrastructure.

5. Continuously enforcing guardrails

As organizations grow in size and complexity, it gets increasingly difficult to maintain consistent infrastructure. Even with a secure initial provisioning process, settings on infrastructure can still be undone or circumvented. This can open your infrastructure up to the possibility of configuration drift. To minimize outages, unnecessary costs, and emergent security holes, teams should have a system in place to monitor this drift. Organizations can try to build this into their processes, or they can use Terraform's native drift detection and health assessments. These continuous checks help you detect and respond to unexpected changes in provisioned resources on Day 2 and beyond.

Terraform's drift detection notifies you if your infrastructure has changed from its original state, so you can make sure security and compliance measures remain in place. With these infrastructure change alerts you can quickly get to the root reason for that change, understand if it is necessary, and complete the change, or automatically remediate if not.

You can also schedule regular automated health checks using assertions defined in your Terraform code with Terraform Cloud's continuous validation. Users can monitor whether these conditions continue to pass after Terraform provisions the infrastructure. These checks give customers flexible options to validate their infrastructure uptime, health, and security — all in one place without requiring additional tools.

6. Observability and visibility

The final step to ensure a secure Day 2 is having general observability of your entire infrastructure estate. In a Terraform environment, this means visibility into your workspaces, with a clear audit trail covering all changes made.

Audit logs are exposed in Terraform Cloud via the audit trails API, and Terraform Enterprise provides log forwarding. These logs give you visibility into important events such as user logins, changes to organization and workspace settings, run logs, approvals, policy violations, and policy overrides.

Explorer for Terraform Cloud provides a consolidated view of workspace data across your organization, including information on providers, modules, Terraform versions, and health checks from drift detection and continuous validation. This consolidated view helps teams ensure their environments have the necessary up-to-date versions for Terraform, modules, and providers while tracking health checks to ensure security, reliability, and compliance.

The ability to find and drill into workspaces on your Terraform dashboard is key to speedy debugging and health checking. Terraform's filtering and tagging capabilities enable users to quickly discover and access their workspaces. Workspaces also act as the system of record for provisioned infrastructure by maintaining secure storage and versioning of Terraform state files.

You can check workspace activity, gaining insight into its users and usage. This allows you to answer questions like:

- Which users are accessing which workspaces?
- What configurations are they changing, at what time, and from where?
- Who is accessing, modifying, and removing sensitive variables?
- Which users are changing or attempting to change your policy sets?

This organization-wide audit trail gives platform teams visibility into their entire infrastructure estate, helping them keep security and compliance top of mind.

Looking forward

While transitioning to cloud infrastructure can be difficult, it also presents an opportunity to get a fresh start in standardizing infrastructure workflows. While these six fundamentals were primarily focused on infrastructure security, these steps include fundamental practices that can help with speed, efficiency, cost savings, and reliability. Secure infrastructure automation enables innovation and provides a solid foundation for your multi-cloud estate, enabling success across other parts of your organization such as networking and applications.

A secure infrastructure foundation helps organizations continue their cloud journey to include zero trust security. Visit our zero trust security page to learn more about how HashiCorp can help your organization incorporate zero trust security principles throughout your cloud infrastructure.

Get started with Terraform Cloud for free to begin provisioning and managing your infrastructure in any environment.