



Unlocking the Cloud Operating Model: Cloud Compliance and Management



Contents

Overview.....03

Implications of the Cloud Operating Model04

Cloud Operating Model: Provision.....05

Cloud Operating Model: Compliance & Management.....07

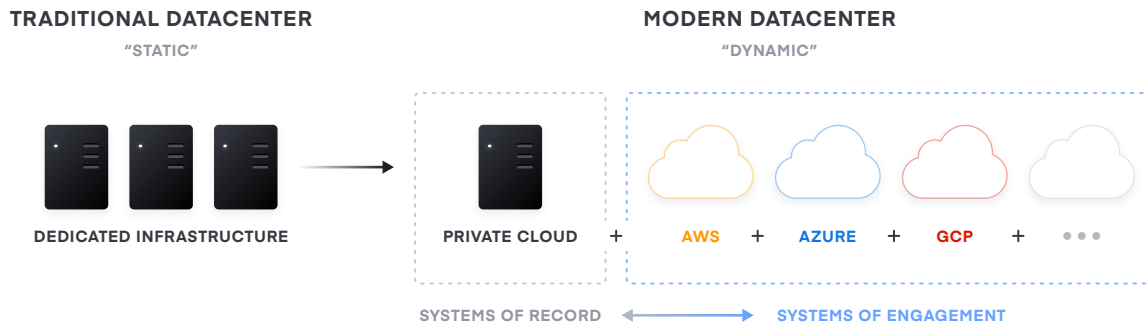
 Terraform Enterprise08

 Workflow Integrations.....10

Conclusion.....12

Overview

Cloud adoption is a secular trend. Organizations undergoing a digital transformation ultimately put pressure on teams delivering and supporting software applications. Digital experiences are the primary interface between customers and businesses; even businesses selling to other businesses. Modern digital interactions are responsively designed and built cloud-first to provide rich, personalized experiences informed by large scale data processing and intelligence as quickly as possible. This pattern of prioritizing digital interactions forces a change in the model for software delivery and is felt most by IT, with a strong dependency on that team to build an organization-wide operating model for delivering cloud based applications.











Implications of the Cloud Operating Model

The essential implication of the transition to cloud is the operating model to accommodate the shift from “static” infrastructure to “dynamic” infrastructure. The challenges IT teams are addressing with the Cloud Operating Model are:

- 1. Volume and distribution of services
- 2. Ephemerality and immutability
- 3. Deploying to multiple target environments

The impact of these changes is that IT teams will need to adjust their approaches for each of the four layers of operation:

	STATIC		DYNAMIC
 Run	Dedicated Infrastructure		Scheduled across the fleet
 Connect	Host-based Dynamic IP		Service-based Dynamic IP
 Secure	High trust IP-based		Low trust Identity-based
 Provision	Dedicated servers Homogeneous		Capacity on-demand Heterogeneous

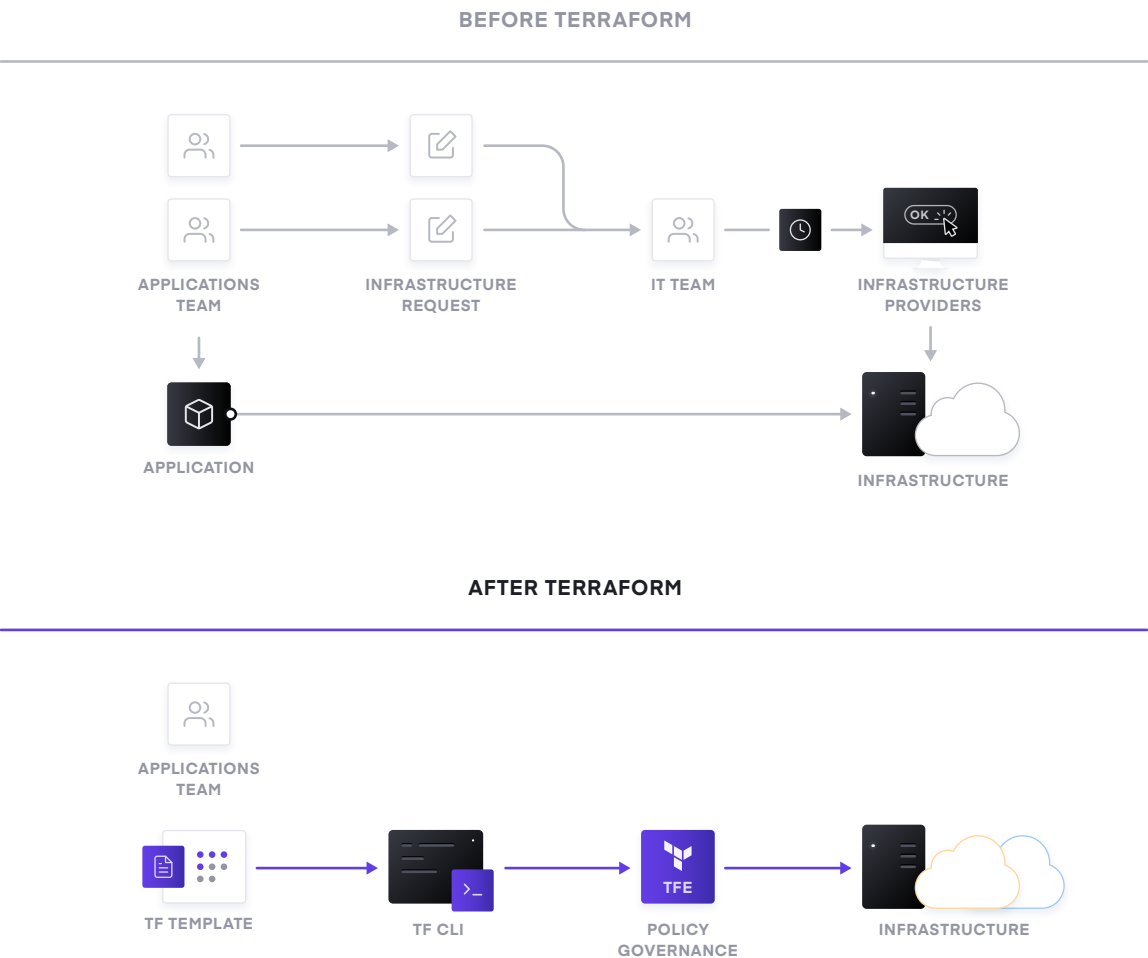
Cloud Operating Model: Provision

A common starting point for the Cloud Operating Model is to enable the operations team to shift their focus away from provisioning only dedicated servers based on homogenous sets of infrastructure to workflows that enable shift left IT with capacity on-demand from a variety of Cloud (and service) providers.

To address the core tenets affecting provisioning (volume and distribution of services, ephemerality and immutability, and deploying to multiple target environments), organizations are moving to an automation based operating model for cloud infrastructure.

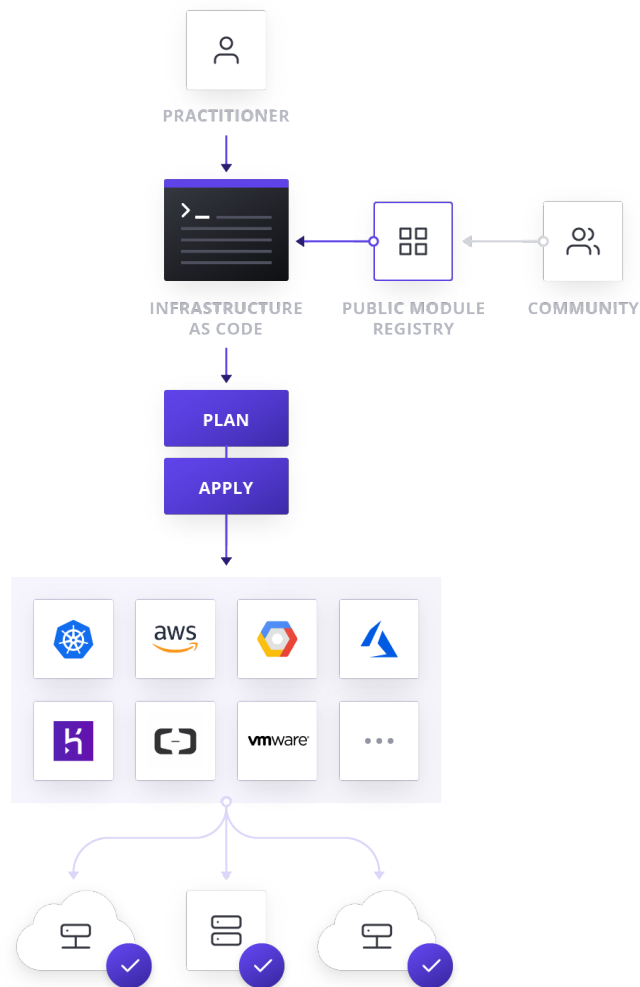
HashiCorp Terraform

HashiCorp Terraform provides the foundation for cloud and on-premises infrastructure automation using infrastructure as code for provisioning and compliance in the cloud operating model.



Reproducible infrastructure as code

Terraform automates through an infrastructure as code approach to provisioning cloud infrastructure and services. Users define a desired topology of infrastructure and services in a configuration file using version control. Terraform translates those configuration files into appropriate API calls to end providers automating resource provisioning to reduce human error and failed builds. Open source providers allow rapid creation and support for any infrastructure.



Cloud Operating Model: Compliance & Management

The infrastructure as code approach to cloud infrastructure automation enables organizations to extend Terraform for provisioning and management in the cloud operating model and reduces the risks organizations are faced with as they grow their footprint in the cloud. However, the lack of IT governance in the distributed operating model increases risks for security, regulatory compliance, and operational consistency.

Security Posture. How to create and enforce policies to prevent breaches? Example security concerns include:

- Restrict app versions with vulnerabilities
- Restrict resources with a public IP address
- Prevent security groups with egress 0.0.0.0
- Restrict use to only approved modules

Regulatory Compliance. How to create and enforce policies to prevent regulatory noncompliance? Example of regulatory compliance concerns include :

- GDPR regulations
- FedRamp regulations
- HIPAA regulations
- PCI standards

Operational Consistency. How to create and enforce policies to ensure operation consistency to manage costs? Examples of operational concerns include:

- Ensure infrastructure follows organization-based best practices
- Manage and control cloud and service expenditure and growth

Traditional approaches to preventing these types of things of infractions from occurring are anchored in the idea the IT is the gatekeeper to infrastructure. Policy isn't codified, but rather known tribally within the IT team. If policy is enforced, then it's enforced manually by that gatekeeping organization—this approach stymies the speed and developer self-service benefits cloud infrastructure offers organizations. Some teams will automate policy enforcement, but will do so through scans that occur after the infrastructure is provisioned opening up opportunities for risk while the out-of-compliance infrastructure exists before it's scanned. Finally, we see organizations that take a least common denominator approach to enforcement assuming that all infrastructure is only susceptible to certain risks and checking against those— leaving open unique vectors for bespoke parts of the organizations infrastructure topology.

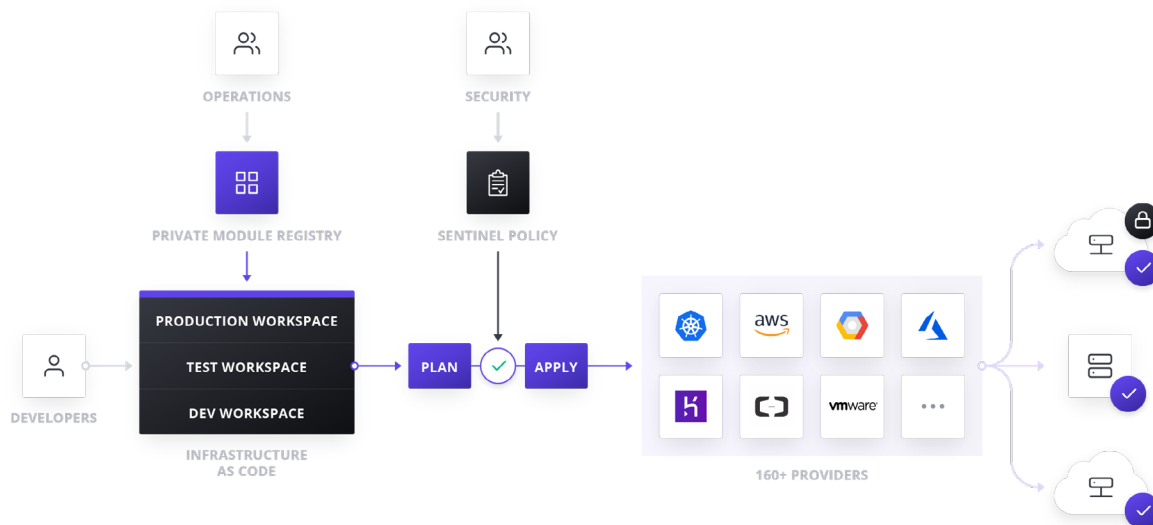
Terraform Enterprise

Terraform Enterprise provides the foundation for cloud infrastructure automation with infrastructure and policy as code for compliance and management in the cloud operating model. Organizations adopting the cloud operating model look for high agility and high control as they rapidly bring products to markets and internal customers. They seek a desired state of:

- Consistent, technology agnostic workflows
- Automation for infrastructure and service provisioning
- Security posture aligned to the speed and surface area of automation

To achieve that desired state organizations need to:

- Enable real-time control and proactive policy enforcement
- Eliminate manual processes and bottlenecks
- Centralize management and control across technologies



Cloud Compliance and Management

Terraform Enterprise addresses cloud compliance and management with an automation platform that enforces policies within the provisioning workflow to reduce risk through proactive policy enforcement, manage costs, and increase productivity through automation.

Sentinel Policy as Code. Using Terraform Enterprise, policy owners (security, compliance, audit, finance, operations) use Sentinel policy as code to define policies. Sentinel policies are then enforced against each Terraform plan prior to executing the provisioning. Because Sentinel offers preventative, proactive policy organizations can confidently instill best practices for production workloads that maintain compliance with necessary regulations.

Sentinel Policies can be enforced across a range levels from Advisory, which warns when a policy breaks, but doesn't prevent it from being provisioned, to Soft Mandatory, which requires an override to break policy, or Hard Mandatory, which prevents provisioning if a policy breaks.

Automated Policy Enforcement. Terraform enforces Sentinel policies on workspaces before it provisions them, meaning that once an organization defines a guardrail in Sentinel no infrastructure can be provisioned that breaks it— enforcement is automatic.

More information about writing and testing Sentinel policies can be found in the [Sentinel Guide](#) as well as [a repository](#) of example policies.

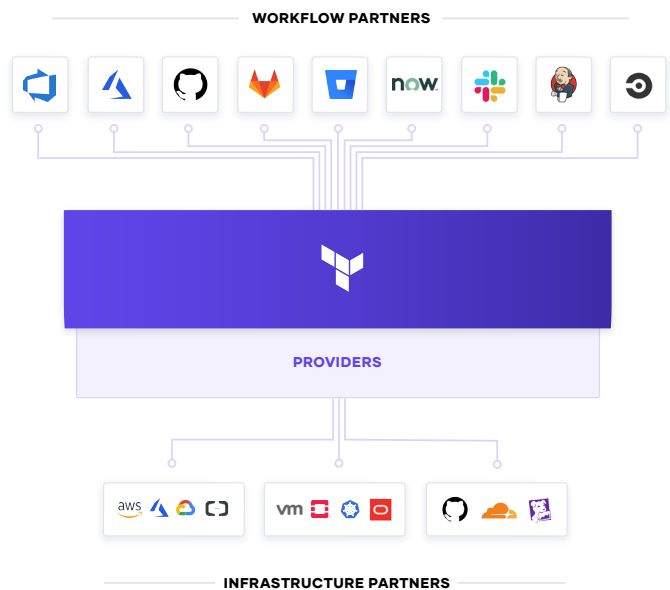
Cost Estimation. Cloud infrastructure provides compelling pay-as-you go pricing. When provisioning infrastructure it is challenging to understand the cost implications of new or changed infrastructure before it is applied. Most organizations rely on after-the-fact alerts from their cloud provider, using dedicated third party services that continually monitor changes in cost, or potentially waiting until they receive their end of month bill to understand the cost impact of their changes. Terraform provides a cost estimation capability that programmatic estimates cost for new cloud deployments or changes to existing ones, before applying those changes or actually incurring costs.

Cost Management via Policy Enforcement. Sentinel allows cost-centric policies to be created and then automatically enforced in the Terraform workflow. Administrators then have the ability to approve significant changes or to completely prevent specific workspaces from exceeding predetermined thresholds. For example, administrators could write policies that prevent large, expensive machines from being provisioned unnecessarily, or enforce overall budget restrictions for a teams' deployments without prescribing what machines to use. Even further, policies can be dynamically written such that a certain threshold of change is allowed month over month, but nothing exceeding a certain dollar limit.

Audit logging. Terraform Enterprise offers rich audit logging for organizations that need insight into the resources managed by terraform. Audit logs emit information whenever any resource managed by Terraform Enterprise is changed, so teams can understand who made changes and what changes were made. Many organizations leverage audit logging to achieve and ensure regulatory compliance.

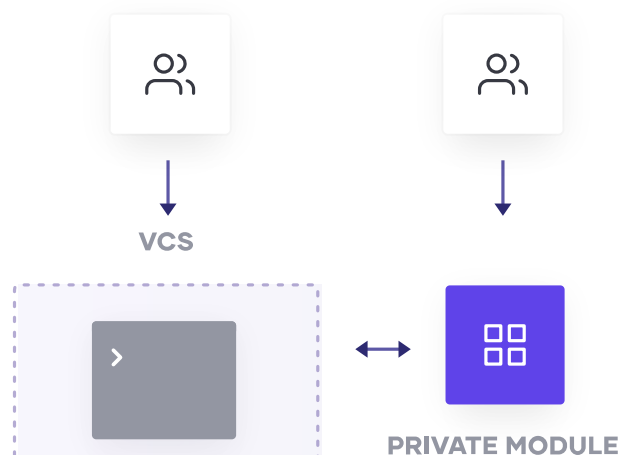
Workflow Integrations

Many customers leverage the cloud management features of Terraform from within an pre-existing workflow or tool chain. Terraform enables this through integrations with major VCS, CI/CD, and service management tooling as well as supporting a full REST API. These integrations allow organizations to drive operational consistency without impacting productivity.



Version Control Systems (VCS)

Terraform users define infrastructure in a simple, human-readable configuration language called HCL (HashiCorp Configuration Language). Users can write unique HCL configuration files or borrow existing templates from the public module registry. Most users will store these configuration files in a version control system (VCS) repository and connect that repository to a Terraform workspace. With that connection in place, users can borrow best practices from software engineering to version and iterate on infrastructure as code, using VCS and Terraform Cloud as a delivery pipeline for infrastructure. Terraform has integrations with Azure DevOps, BitBucket, Github, and Gitlab.



When you push changes to a connected VCS repository, Terraform will automatically trigger a plan in any workspace connected to that repository. This plan can be reviewed for safety and accuracy in the Terraform UI, then it can be applied to provision the specified infrastructure.

Continuous Integration, Continuous Delivery (CI/CD) Pipeline

Terraform can be called from within most CI/CD pipelines such as Jenkins, Circle, Travis, and Gitlab. Many users leverage the programmability of Terraform to automate as much of their provisioning workflow as possible, while enforcing guardrails through policy as code. Terraform's API-driven run provides flexible provisioning workflows using an infrastructure as code approach that any organization can manage. A continuous integration (CI) system monitors changes in Terraform code and drives provisioning using Terraform Cloud's REST API. This approach allows organizations to implement a range of actions in their CI pipeline as part of an infrastructure provisioning workflow and still benefit from Terraform Cloud's capabilities such as private modules, state management, policy as code (Sentinel) and more.

IT Service Management (ITSM)

Terraform Enterprise also includes a first class ServiceNow integration. ServiceNow provides digital workflow management, helping teams work quickly and efficiently with one another by offering a straightforward workflow for their interactions. The ServiceNow Service Catalog offers a storefront of services that can be ordered by different people in the organization. One common request between teams is for Cloud resources: a developer needs a fleet of machines to test out a codebase or the IT team in finance has a request for infrastructure to run their new accounting software. For organizations who use the ServiceNow Service Catalog, the requests can be submitted through ServiceNow and routed to the right team for Cloud Infrastructure. Terraform Enterprise provides provisioning automation through infrastructure as code and security, compliance, and cost-sensitive policy enforcement against all resources as they are provisioned.

Our newest integration connects the human workflow power of ServiceNow with the infrastructure workflow capabilities of Terraform Enterprise. This enables teams that are not code-centric to safely adopt best-in-class provisioning workflows and tooling, while still getting developing competency with infrastructure as code.

Conclusion

As organizations begin to adopt the cloud operating model, they first face the challenge of provisioning cloud infrastructure. Many organizations believe that even if the cloud unlocks new speed, it also opens up new security risks, which can only be solved through processes that reduce that unlocked speed. However, Terraform offers a powerful alternative to that legacy model by combining an infrastructure as code approach to provisioning with a policy as code approach to compliance and management. This enables organizations to have both high agility and high control as they develop competency in infrastructure provisioning, compliance and management. This paper provides an overview of how organizations can leverage Terraform Enterprise to implement multi-cloud compliance and management ensuring effective, unencumbered adoption of the cloud operating model.

