HashiCorp

WHITEPAPER

Enabling a Cloud Operating Model

at the infrastructure layer

Contents

Introduction

The cloud demands a new operating model. Datacenter primitives are changing: dynamic instead of static infrastructure, security and networking policies based on identity rather than IP addresses, and an automated self-service platform as opposed to manual ticketing systems.

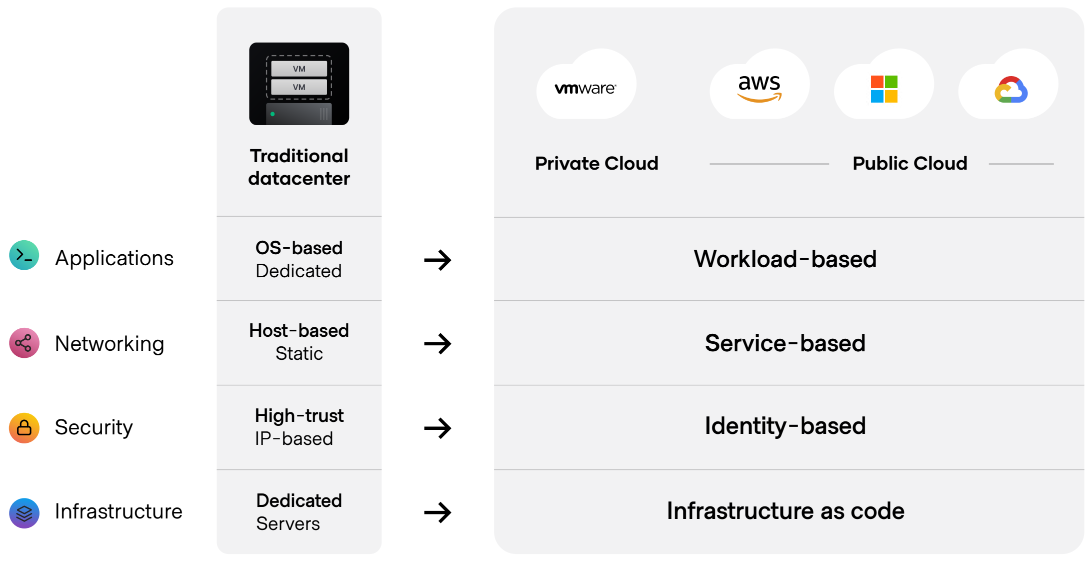
While this transformation speeds development and shrinks go-to-market times, it renders much of the previous generation of tools and processes obsolete. To fully realize the benefits of cloud computing, enterprises must transition to scalable dynamic workflows and management at every cloud layer. This is especially true at the infrastructure layer, where the shift takes the form of establishing infrastructure as code provisioning workflows, creating a system of record, and then establishing a system to manage the lifecycle of that infrastructure. These establish some of the core shared cloud services that platform teams deploy across their organizations to enable development teams to reduce costs, decrease risk, and increase speed and efficiency.

This paper focuses on the first layer of a cloud operating model, the infrastructure layer, and how best to gain value by deploying infrastructure as code based upon a cloud operating model and roadmapping this entire process according to the stages of a maturity model. This maturity model should be applied to three key use case areas: creating a provisioning workflow, building a system of record, and implementing an approach to Lifecycle Management HashiCorp has observed a number of common best practices among organizations that have successfully leveraged a platform team to build and manage a holistic cloud platform.

A new cloud operating model

Cloud computing is a generational transition, shifting from largely static, dedicated servers in private datacenters to a pool of service capacity available on demand from a variety of different providers. Successfully adopting the cloud requires enabling a cloud operating model to address the transitional concerns at all four operational layers: infrastructure, security, networking, and applications.

1. For infrastructure, provisioning and management is done using infrastructure as code
2. For security, brokering access and management of sensitive data is based on identity
3. For networking, access and connections are based on service identity
4. For applications, deployment and management is workload-based



As infrastructure, security, networking, and applications teams shift their focus from private datacenters to the cloud, the foundations of each layer change.

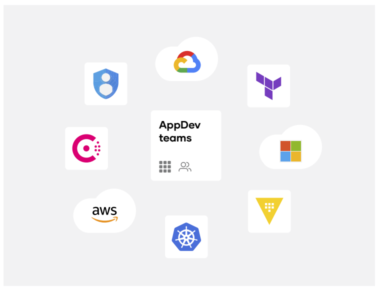
Successfully enabling a cloud operating model

Organizations that successfully enable a cloud operating model follow a typical blueprint, called a maturity model, and rely on centralized cloud platform teams.

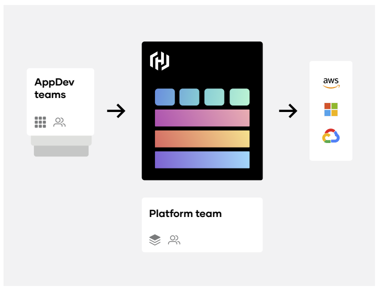
Cloud maturity model

Cloud adoption journeys typically follow an established pattern that flows through three stages:

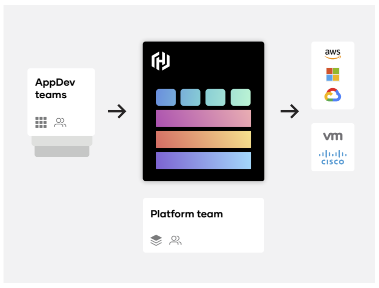
- **Stage 1: Adopting** — This is the beginning of cloud experimentation and usage, defined by individual teams engaging with cloud providers in silos to deliver applications and services. This stage is often defined by quicker adoption due to fewer process roadblocks. But because there is no standard platform, and limited knowledge shared between teams, and minimal cloud operating strategy, it can lead to many varied systems and processes in the same organization.
- **Stage 2: Standardizing** — As cloud usage increases, the organization shifts to strategic planning by a platform team to standardize the way developers interface with the cloud. The platform team is tasked with creating central services around provisioning, security, networking, and application deployment.
- **Stage 3: Scaling** — Once established in a single cloud environment, platform teams can extend these workflows to other cloud vendors and across an organization's private estate and SaaS applications, creating a consistent platform and system across all development and deployment areas.



Stage 1: Adopting



Stage 2: Standardizing



Stage 3: Scaling

This same maturity model should also be applied to each individual use case organizations apply across their cloud estate. Organizations will build out their cloud estate and add in new solutions separately over time as they build their cloud platform. With this in mind, different aspects of your cloud operating model may be in different stages of maturity at the same time. For infrastructure, we see 3 key areas that organizations should focus on to build out this layer of their cloud estate: **Provisioning Workflow, System of Record, and Lifecycle Management**. We will now discuss how to implement each of these in more detail.

Building the infrastructure layer of the cloud operating model

Common challenges at the infrastructure layer

As organizations move from on-premises infrastructure to the cloud, operators must deal with new requirements:

- **Scale:** Teams want to quickly scale their infrastructure usage up and down with no errors despite potentially extensive configuration changes.
- **Variety:** Teams want unified provisioning workflows on a variety of platforms.
- **Dependencies:** As part of this provisioning workflow, teams want to include and automate existing services and dependencies into configurations.

The foundation for running a cloud operating model is infrastructure provisioning leveraging **infrastructure as code (IaC)**. By converting infrastructure into code, teams can declaratively define the desired end state of a given deployment, ensuring consistency in every deployment while also allowing them to track and audit changes to that code.

Standardize infrastructure with HashiCorp

HashiCorp offers multiple products to help establish the infrastructure layer of a cloud operating model. HashiCorp Terraform enables infrastructure provisioning on any cloud services through IaC. As a fully extensible engine, Terraform enables thousands of easy integrations with cloud providers and popular software applications. By creating a shared service for infrastructure provisioning, Terraform lets product teams plan and provision resources inside CI/CD workflows using familiar tools. In doing so, Terraform becomes the lingua franca and common workflow for teams provisioning resources.

At its core, Terraform uses the declarative HashiCorp configuration language (HCL) to assert the desired state of a set of infrastructure and can be used to deliver and manage that state. From here, Terraform facilitates all infrastructure across an organization's varied cloud estate. It is used to deploy common workflows with built-in policy as code. This way compliance, networking, security, and other requirements are baked into the resulting infrastructure, abstracting these concerns away from development teams that will use it.

Build automated images with HashiCorp Packer

Organizations leverage HashiCorp Packer in tandem with Terraform to standardize cloud infrastructure through golden images — identical machine images for multiple clouds — from a single source template.

Teams that desire more automation can use HCP Packer, a HashiCorp-managed cloud service that builds a registry of images to be easily shared and deployed by Terraform. This toolchain lets security and platform teams work together to create, manage, and consume images in a centralized way.

The maturity model stages of the infrastructure cloud layer

As organizations implement their infrastructure layer, they must build out three main areas:

1. A best-in-class provisioning workflow
2. A system of record to provide a high level of visibility to support Day 2 operations
3. A resource management system to manage infrastructure lifecycles

For each of these areas, organizations will want to follow the stages of the maturity model — adopting, standardizing, and scaling — to build an implementation roadmap.

Infrastructure & Image Automation with Terraform & Packer

- Terraform
- Packer
- Both

	Stage 1: Adopting	Stage 2: Standardizing	Stage 3: Scaling
Provisioning **Workflow**	**Compose** Infrastructure as code Images as code Secure variables, dynamic credentials Vault integration **Collaborate** Version control, CI/CD Role-based access Remote operations	**Publish and discover** Public registry Private registry Image artifact registry **Policy & enforcement** Sentinel, OPA, run tasks Image channels	**Enable self-service** No-code provisioning **Golden Image pipeline** **Health checks** Continous validation
Resource **System of Record**	**Provision cloud** Public clouds (AWS, Azure, Google) App schedulers (K8's Nomad) Config-driven import Build Container images	**Provision networking & SaaS** Network & SaaS providers **Integrations/Extensibility** Native integrations and run tasks Boundary for remote access Tech partner & community plugins **Reference** Image channels Channel assignment history	**Provision private data center** On-premises infrastructure Self-hosted agents **Self-service integrations** ServiceNow, CI/CD HCP Waypoint Kubernetes operator **Better together** Image promotion via Terraform Dynamic image updates
Resource **Lifecycle Mgmt**	**Visibility** Terraform state Audit logging Image metadata Image ancestry	**Monitoring & compliance** Drift detection Workspace explorer Image revocation Image validation checks	**Event notifications** Email, Slack, Teams Webhooks **Fix & Optimize** Ephemeral Workspaces Channel rollback Inherited revocation

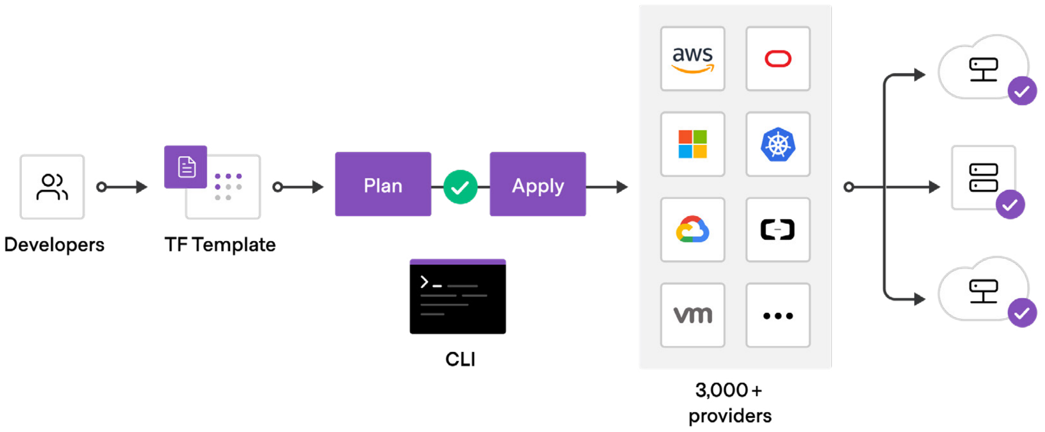
Provisioning workflow

Organizations begin their cloud adoption journey by establishing a provisioning toolset. Manual provisioning, however, is slow, error-prone, and difficult to use at scale. The provisioning workflows offered by Terraform and HCP Packer support composition, collaboration, and reuse of infrastructure as code to support each stage of the maturity model.

Adopting a provisioning workflow

The first step to establishing a provisioning workflow is integrating the right tools to provision and manage your cloud infrastructure. The cloud maturity journey begins by composing infrastructure as code with Terraform.

Organizations have two choices when beginning this IaC journey. Our recommendation is to use HCL, a human- and machine-friendly declarative configuration language used across multiple HashiCorp products. Teams with a strong preference for a particular programming language and familiar toolchains may choose to use the Cloud Development Kit for Terraform (CDKTF), which offers support for TypeScript, Python, Java, C#, and Go. Additionally, HashiCorp Packer codifies machine and container images and automates the image build process for use in provisioning workflows, also using HCL.



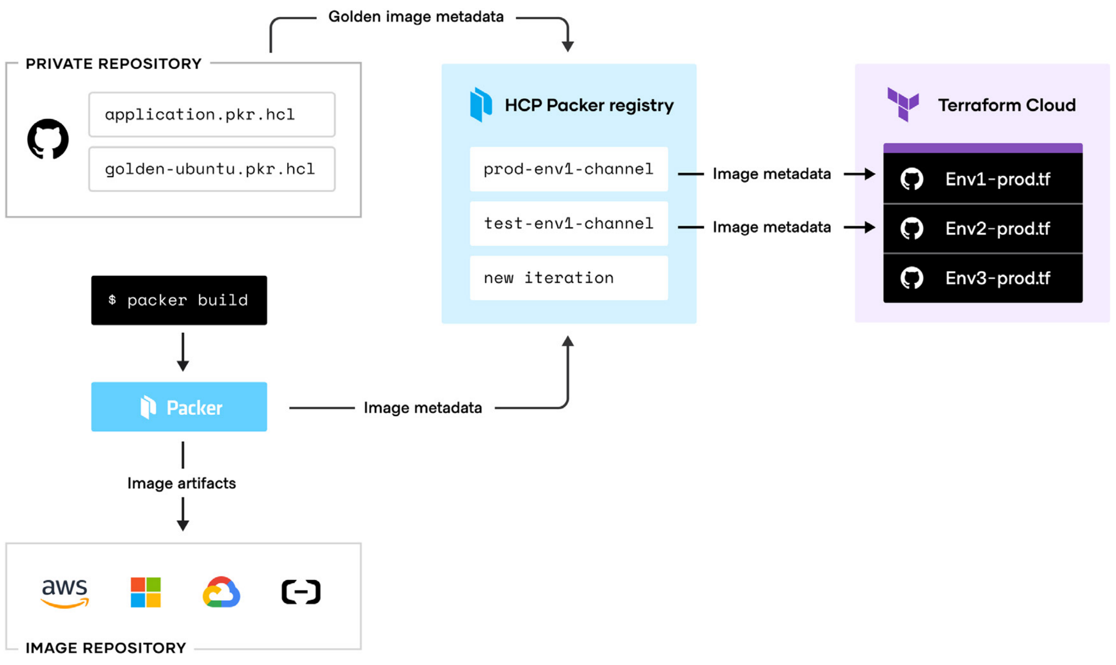
As individual adoption gives way to teams, codification lends itself to a collaborative model. Configurations are stored in version control tools, so whole teams can access, iterate on configurations, and provision resources without conflicts. Additionally, Terraform Cloud includes advanced team-management capabilities that establish role-based access controls (RBACs). The combination of predefined and customizable permissions provides granularity and flexibility to allow teams to work efficiently while preserving the principle of least privilege. Terraform also offers a consistent runtime environment for provisioning with secure state storage and versioning, disposable agents for executing runs, and multiple workflow options to let teams work the way that suits them best.

Adopting an infrastructure as code workflow also requires secure handling of sensitive secrets such as cloud credentials, passwords, certificates, and other operational data involved in the provisioning process. Terraform includes secure variable handling options, including an OIDC-based dynamic provider credentials feature to eliminate the need for static cloud credentials. Tight integration with HashiCorp Vault allows organizations to unify their secrets management across all cloud providers and workflows.

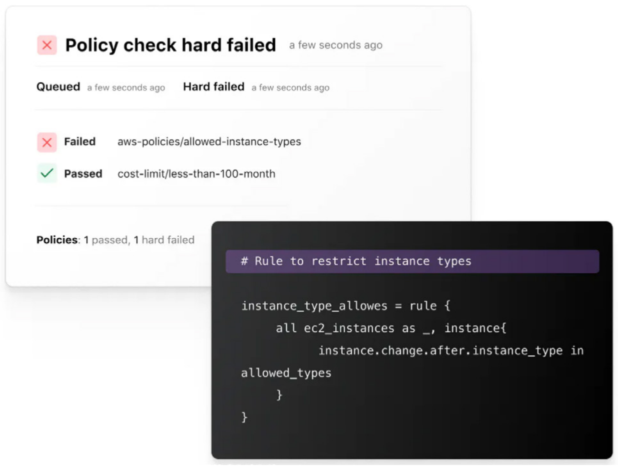
Once these tools are in place, platform teams must begin the process of standardizing provisioning tools and systems or risk the creation of inconsistent processes and policies that greatly increase cloud costs and security risks.

Standardizing a provisioning workflow

As an organization's provisioning workflows mature, reusability and policy become the key tenets of the standardization phase. For Terraform, this means abstracting frequently used configurations into reusable modules. Platform teams build up a library of approved modules and policies with security, compliance, and organizational best practices baked in — for example, dictating the size of virtual machines that can be deployed or confirming that new infrastructure to be provisioned is within allocated budgets. Terraform Cloud enables teams to quickly discover these approved modules through the private registry. Similarly, teams need a way to discover and consume the approved golden images built with Packer. HCP Packer, part of the HashiCorp Cloud Platform (HCP), is a multi-cloud image artifact registry that stores image metadata and makes it discoverable to downstream provisioning processes.



Policy enforcement also becomes crucial at this phase of workflow maturity. Cloud providers follow a shared security model, where the security of the cloud platform is the provider's responsibility, but the security of infrastructure and applications provisioned in the cloud is the customer's responsibility. Also, organizations in many industries face an array of compliance and legal requirements. Poorly configured or misconfigured cloud resources represent a security, compliance, and reliability risk. By enforcing policies as a native part of the provisioning workflow, such misconfigurations can be detected and prevented before infrastructure is provisioned. These policy guardrails often include a combination of compliance requirements, cloud provider recommendations, and organizational best practices for security, performance, and cost control.


Policy check hard failed a few seconds ago

Queued a few seconds ago Hard failed a few seconds ago

✗ Failed aws-policies/allowed-instance-types

✓ Passed cost-limit/less-than-100-month

Policies: 1 passed, 1 hard failed

```
# Rule to restrict instance types

instance_type_allowes = rule {
     all ec2_instances as _, instance{
           instance.change.after.instance_type in
allowed_types
     }
}
```

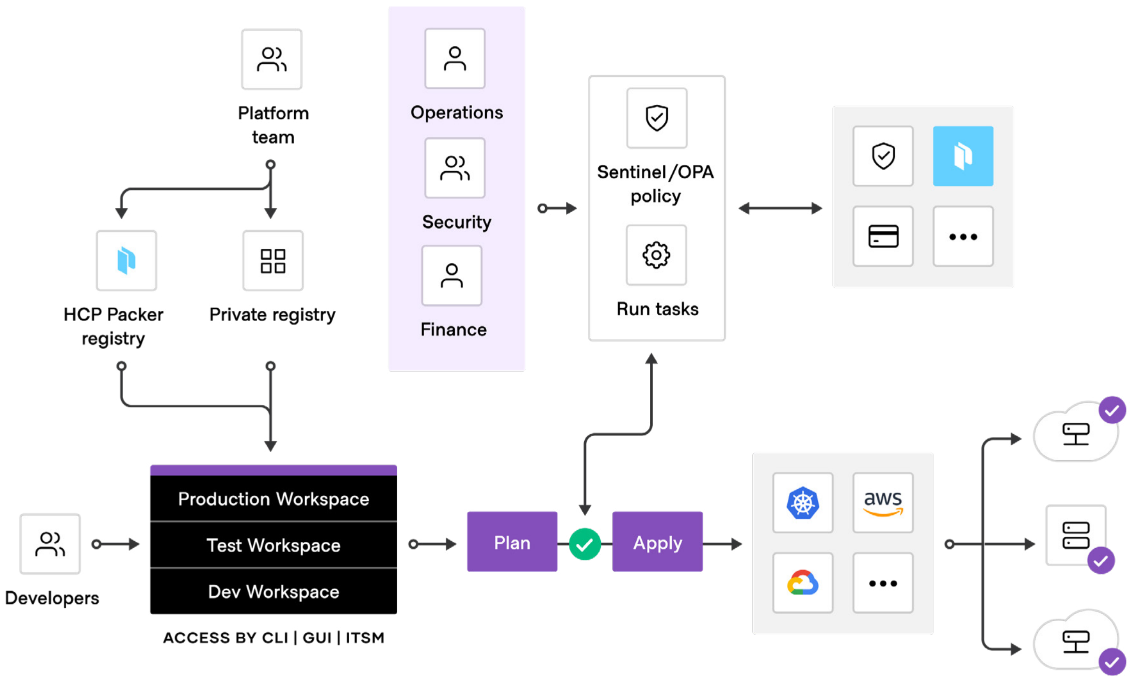
Terraform offers a choice of two powerful integrated policy enforcement capabilities: the HashiCorp Sentinel framework, and Open Policy Agent (OPA). Policies are defined and versioned separately from infrastructure code, providing a clear separation of duties between security teams and cloud operations or development teams. Flexible enforcement options allow policy maintainers to decide which rules should be blocking, overridable, or advisory. Provisioning workflows can be further extended with a wide range of third-party partner services using Terraform Cloud run tasks to provide additional security, compliance, and visibility during runs.

This gives platform teams a system of reusable modules and workflows that can be easily implemented by development teams while ensuring consistent processes and security policies are in place and enforced. Additionally, these teams have easy tools to access and deploy these modules, allowing for self-service infrastructure provisioning that frees up engineering and development resources for more business-critical tasks. Now platform teams have a blueprint to begin scaling these workflows beyond just the cloud.

Scaling a provisioning workflow

The final, and ongoing, stage of infrastructure as code workflow maturity is to scale these workflows across the organization's larger estate. Because platform teams are typically small compared to the number of downstream users they support, enabling self-service for users of all skill levels is required. Terraform paves the way with a no-code provisioning workflow that allows end users to provision infrastructure without writing any Terraform code, while operating within the guardrails established in the previous stages.

This workflow journey does not end with the Day 1 provisioning of infrastructure resources. To ensure the reliability of cloud resources on Day 2 and beyond, teams need ongoing visibility into their provisioned infrastructure's health. Continuous validation in Terraform provides automated health checks using assertions defined in Terraform code. Customizable notifications alert infrastructure owners as soon as a check fails, helping avoid costly downtime due to preventable issues such as expiring certificates or security issues from outdated images.



Once this is all in place, teams have a complete infrastructure provision workflow that integrates the requirements of core teams like operations, security, and finance with the needs of developers and the tools they need to do their jobs. This approach abstracts away the complexity of provisioning infrastructure into a core reusable provisioning workflow that can be leveraged at all levels of the organization.

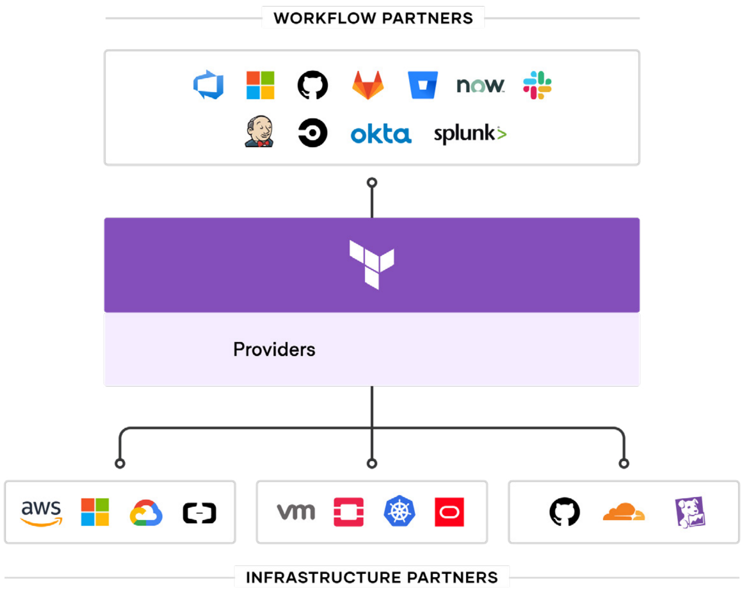
System of record

Organizations moving to a cloud operating model are faced with the reality of cobbling together multiple interfaces and tools to manage the many types of infrastructure with which they now must interact. The solution to this is to leverage a small set of tools to create a unifying system of record for the cloud.

Organizations can leverage Terraform and Packer for an infrastructure provisioning and management solution that works across infrastructure providers, developer workflows, and operator tools. The reach of this system of record is extended by HashiCorp's vast ecosystem of partnerships and the extensibility offered by Terraform and Packer to integrate the tools your development teams need or already use.

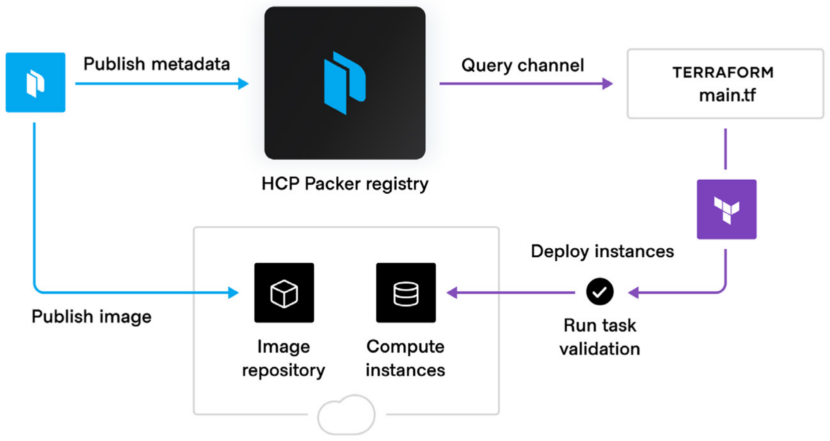
Adopting a system of record

To start, organizations need to properly interface with their cloud service providers. Terraform provides a consistent interface to interact with infrastructure provider APIs using plugins called providers. Similarly, Packer leverages plugins to build and provision customized images across multiple platforms. For most organizations, adoption begins with infrastructure as a service (IaaS) offerings from one or more of the major public cloud providers — Amazon Web Services (AWS), Microsoft Azure, Google Cloud, and others — and common platforms like Kubernetes. The dynamic, on-demand nature of public cloud infrastructure services provides a natural starting point for infrastructure automation. And with the config-driven import workflow with automatic code generation, even pre-existing resources can be brought under management by Terraform.



Standardizing a system of record

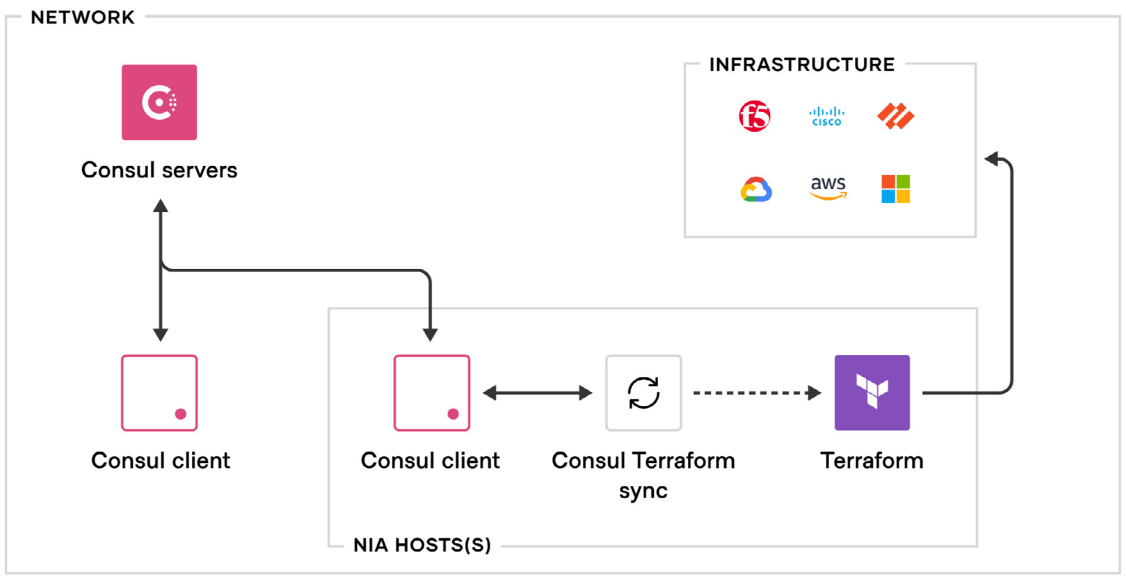
The next stage of establishing a system of record is to expand beyond basic cloud infrastructure and move into networking and Software-as-a-service (SaaS). The massive Terraform ecosystem of more than 3,000 providers in the public Terraform Registry enables teams to manage heterogeneous stacks of infrastructure and applications. The same workflow used to provision an Amazon EC2 instance or Azure VM can also manage firewall rules, load balancer configurations, Git repositories, and performance monitoring services. Packer builds can be extended with builder and provisioner plugins for dozens of platforms and configuration management tools.



Networking changes can also be standardized through HashiCorp's Network Infrastructure Automation (NIA) integration with HashiCorp Consul. Consul allows organizations to discover and automate network services across any runtime, while NIA lets platform teams build integrations that automatically apply network and security infrastructure changes to the Consul service mesh.

Network changes occurring in the Consul service mesh are automatically applied to adjacent networking services using Consul-Terraform-Sync (CTS), a multi-platform tool that connects to the Consul catalog and monitors changes in the services' state and health. CTS leverages Terraform as the underlying automation tool and uses the Terraform provider ecosystem to drive relevant change to network infrastructure.

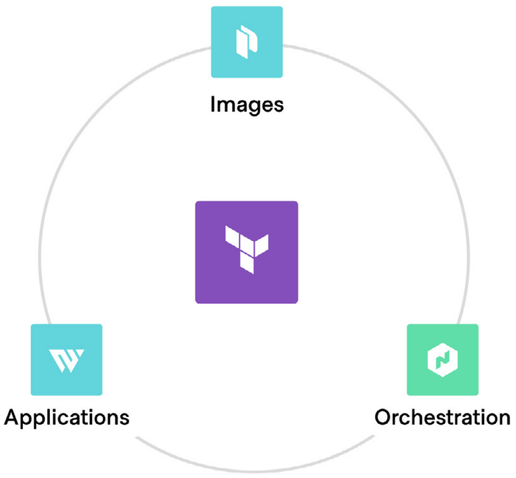
Additionally, Terraform enables simple integrations for enterprise tools such as SAML single sign-on with Okta or Azure Active Directory and audit logging with Splunk. At this stage, HCP Packer becomes the system of record for golden image artifacts. Image channels facilitate the promotion of image versions aligned with the right environment and use case, so downstream provisioning processes always reference the latest approved image. Access to these systems can be further controlled through the use of integrations with HashiCorp Vault for identity-based security and HashiCorp Boundary for privileged access management (PAM), which facilitate administrative access to systems through short-lived, just-in-time credentials.



Scaling a system of record

The third stage of maturity extends infrastructure as code provisioning into the private datacenter, managing on-premises computing, network, and storage environments. Self-hosted Terraform agents extend provisioning capabilities into restricted networks, requiring no inbound firewall rules.The HCP Packer provider can further automate the golden image pipeline, managing the HCP Packer artifact registry and image promotion process via Terraform. At this stage, Terraform truly becomes the system of record across all of an organization's infrastructure.

Extending the Terraform Cloud workflow, self-service platforms like ServiceNow or CI/CD pipeline tooling provide familiar interfaces for end users and developers. Development teams can also use HCP Waypoint (HashiCorp's applications deployment product) to deploy their application code while transparently provisioning the underlying infrastructure in accordance with the organization's existing Terraform standards.

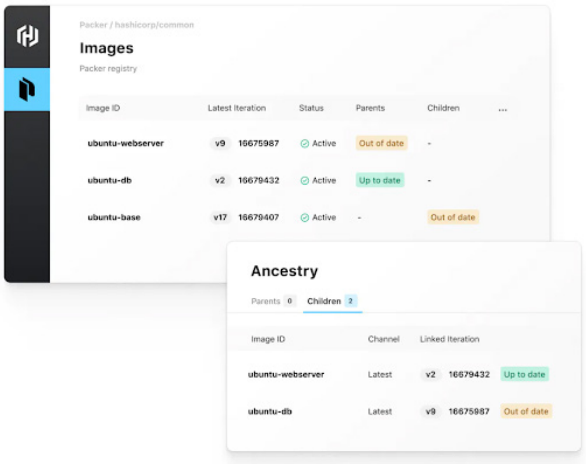


Lifecycle management

The third use case of the infrastructure maturity model is lifecycle management, where the Terraform workflow and system of record combine to provide visibility and optimization across the entire infrastructure lifecycle.

Adopting lifecycle management

Terraform maintains its view of the world in state files. State is used to map real-world resources to the expected configuration, determine what changes need to be made during an apply run, and to track metadata such as resource dependencies. With Terraform, state files are centrally managed with secure storage and automatic versioning. Audit logs supply a historical trail of all activities occurring in the Terraform organization — user logins, changes to workspace settings, run logs, policy violations, and so on.



Packer / hashicorp/common

Images

Packer registry

Image ID	Latest Iteration	Status	Parents	Children
ubuntu-webserver	v9 16675987	✓ Active	Out of date	-
ubuntu-db	v2 16679432	✓ Active	Up to date	-
ubuntu-base	v17 16679407	✓ Active	-	Out of date

Ancestry

Parents 0 Children 2

Image ID	Channel	Linked Iteration	
ubuntu-webserver	Latest	v2 16679432	Up to date
ubuntu-db	Latest	v9 16675987	Out of date

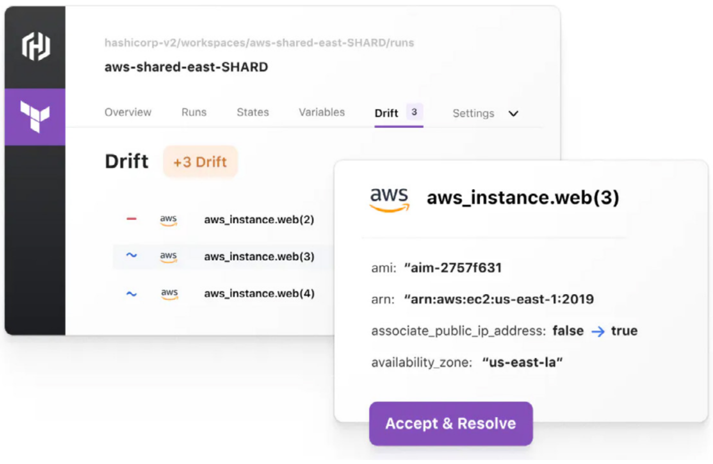
The HCP Packer artifact registry provides a consolidated view of metadata across all image versions, including ancestry tracking to understand the provenance of parent/child image relationships. Audit log streaming produces a near real-time feed of activity occurring in the registry for operational and security monitoring.

This is the beginning of infrastructure lifecycles and the key to managing this across an entire organization as their cloud programs mature.

Standardizing lifecycle management

Standardizing an infrastructure lifecycle approach makes Terraform the authoritative source of truth for resource management. Changes to infrastructure are managed and tracked within Terraform according to tools and policies established by the platform team.

But despite all these guardrails being in place, sometimes changes can still occur outside of the normal workflow. This can be due to accidental changes made directly through the cloud console, emergency actions taken during an outage or incident response, or even changes in provider defaults. As organizations standardize their infrastructure lifecycle management, they also need to implement tools to watch for these unexpected changes often referred to as infrastructure drift. Drift detection, part of the health assessments in Terraform, regularly scans for infrastructure drift by comparing the actual state of resources to the last saved state. Teams can leverage this visibility to ensure unexpected changes aren't occurring, and identify potential policy or process issues that need to be addressed.


hashicorp-v2/workspaces/aws-shared-east-SHARD/runs

aws-shared-east-SHARD

aws_instance.web(2)
aws_instance.web(3)
aws_instance.web(4)

aws_instance.web(3)

ami: "aim-2757f631
arn: "arn:aws:ec2:us-east-1:2019
associate_public_ip_address: false → true
availability_zone: "us-east-la"

Similarly, HCP Packer becomes the standard for image lifecycle management. Golden images have a finite shelf life after which they must be refreshed with the most recent patches and hardening specs. Using out-of-date images can lead to security and operational incidents. To prevent this, old or insecure images can be automatically revoked to prevent their use by downstream builds or provisioning pipelines.

The HCP Packer run task for Terraform validates that managed images are in use and blocks the provisioning of revoked image versions. By centrally managing the state of an organization's cloud infrastructure and image pipeline, platform teams — and by association, the developer leveraging the platform — can be sure they are in compliance and operating in the best possible environment for their business requirements.

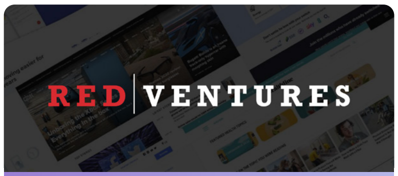
Scaling lifecycle management

The final maturity stage of lifecycle management puts platform teams on a proactive footing. Customizable notifications alert administrators and operators when run errors occur or health issues like drift and failed checks are detected in a workspace. When a Terraform plan action needs approval, the system sends an email or Slack message to alert the responsible team. When drift detection discovers a managed resource has been manually changed, send a webhook to open a ticket in a service management system.

Reaching this stage allows organizations to leverage their fully matured infrastructure automation platform to respond to issues using the same workflow to accept, repair, or roll back changes to managed resources. Terraform facilitates this through its standard apply or refresh workflows and even offers a state rollback mechanism for severe events. HCP Packer provides automated channel rollback and inherited revocation so an entire parent/child image chain can be revoked in a single operation.

Conclusion

With these three cloud infrastructure use cases fully matured and deployed, organizations now have a dedicated platform to manage all aspects of their provisioning and infrastructure systems both in the cloud and across their private estate.

		
5x	$20k	96%
Faster time to market with a 9% decrease in time spent provisioning infrastructure	**Saved** each month in infrastructure operating costs	**Faster** provisioning without security or compliance issues with modules

That's critical to optimizing the most fundamental shift in computing over the last 20 years. Because while the cloud promises dramatic advances in how organizations innovate, respond to market trends, and connect with their customers and employees, it also requires significant changes in how applications are built, deployed, and managed.

This may seem daunting, which is why successful organizations create platform teams and apply a maturity model to manage and benchmark this process. At the infrastructure layer, platform teams create a provisioning center of excellence that supports cloud adoption by delivering shared infrastructure services. They then apply this into a system of record that is easily delivered to development teams. Finally they formalize the lifecycle management process for all infrastructure. Employing a maturity model helps these teams establish a detailed roadmap laying out where the organization currently stands and what needs to be done to reach its goals. This roadmap helps teams throughout the organization make better decisions faster. Just as important, it enables benchmarking progress along the way. Without a thoughtful roadmap and a clearly articulated process, cloud adoption can be significantly slower and more expensive. When done properly, platform teams will deliver a standard system of infrastructure provisioning, recording, and management to development teams that they can leverage in a self service manner to better focus on their own business critical projects.

Adopting a cloud operating model is a critical step for enterprises aiming to maximize their digital transformation efforts. The cloud strategy/program office is evolving away from ITIL-based control points — focused on cost — toward becoming self-service enablers focused on speed. They enable product teams to deliver new business and customer value more quickly, decrease spending, and reduce risk.

The HashiCorp platform of products provides solutions for each layer of cloud infrastructure to enable platform teams to successfully lead the shift to a cloud operating model. For platform engineers focused on building out the infrastructure layer of the cloud, Terraform and Packer are critical tools that will drive continued organizational success.

For more than a decade, HashiCorp has partnered with thousands of organizations around the world to make their cloud operating model a reality. Our portfolio of community and commercial products has been downloaded more than 450 million times in the last year alone, and our ecosystem includes hundreds of partners and more than 3,000 integrations.

Learn more about how HashiCorp can make the cloud operating model a reality for you at **www.hashicorp.com**

HashiCorp

www.hashicorp.com