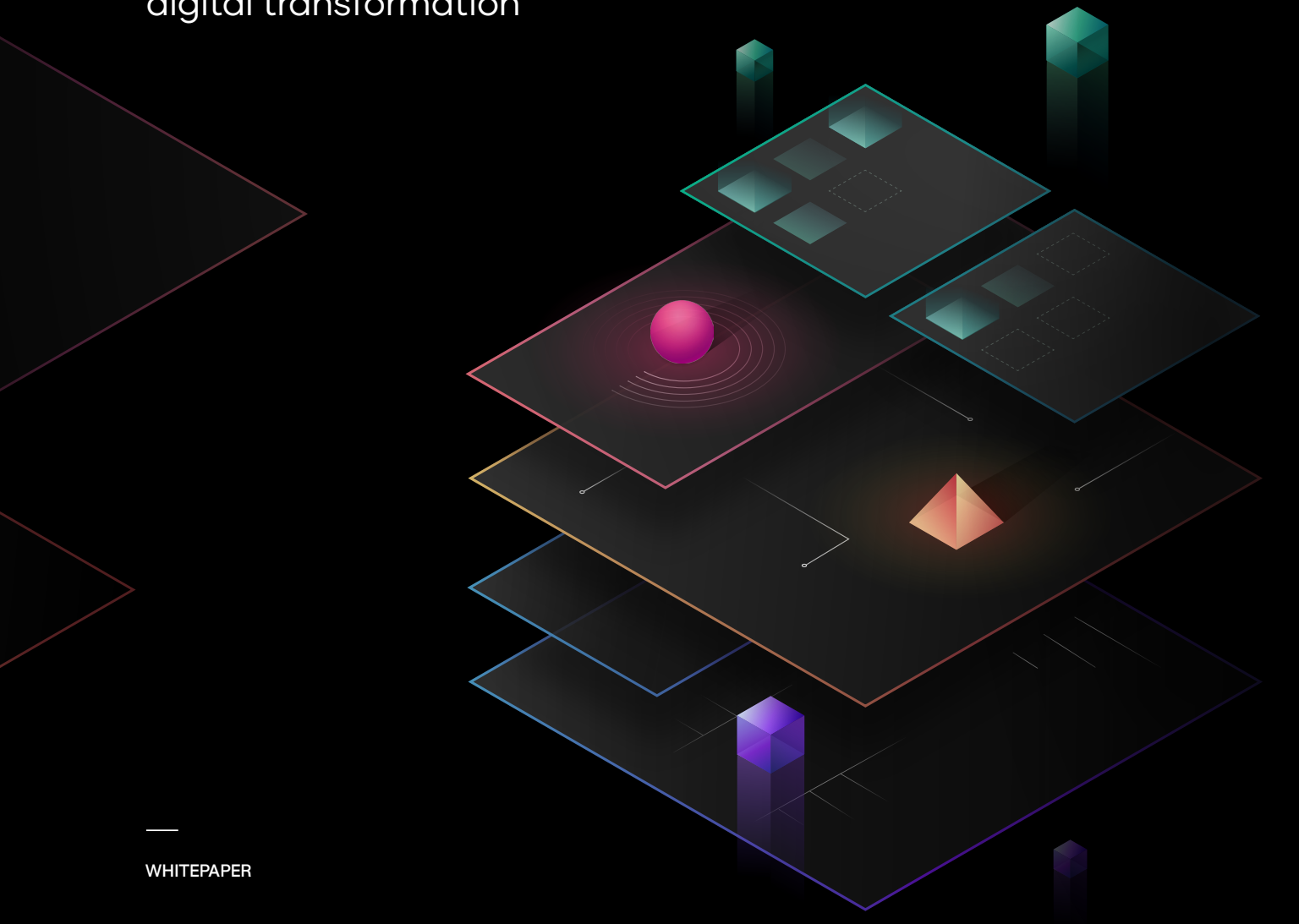
HashiCorp

Scale Your Cloud Operating Model with a Platform Team

Developing shared IT services to accelerate digital transformation



WHITEPAPER

Contents

Executive Summary 03

Introduction 04

Multi-Cloud Adoption Leads to Platform Teams 05

A Shift to Platform Teams and a Platform Mindset 05

Platform-as-a-Product Practices 06

Standardize Platform Team Workflows with a Cloud Operating Model 09

- Standardize Infrastructure Provisioning with HashiCorp Terraform 09
- Build Automated Images with HashiCorp Packer 12
- Manage Secrets and Protect Data with HashiCorp Vault 13
- Secure and Manage Access with HashiCorp Boundary 16
- Securely Connect Applications with HashiCorp Consul 19
- Standardize Application Deployment with HashiCorp Waypoint 23
- Standardize Workload Orchestration with HashiCorp Nomad 25

Conclusion: People, Process, and Tools 28

Getting Started Checklist 30

Executive Summary

Cloud is now the default choice for organizations delivering new value to their customers. Successful companies use a cloud operating model — a framework for adopting cloud services — to maximize agility, reliability, and security and deliver superior business outcomes.

But that's only the first step. The most mature organizations are further tuning their people, processes, and tools to create centralized platform teams that help scale their cloud adoption enterprise-wide.

Platform teams are instrumental in achieving the maximum benefits from a cloud operating model. The teams' engineers deliver shared infrastructure, runtimes, and other services that are consumed by developers across the organization. Effective platform teams enable a cloud operating model that delivers standardized workflows, compliant golden images, and a system of record for cloud adoption. This leads to greater productivity, more frequent releases, increased stability, lower risk, and optimized costs.

Introduction

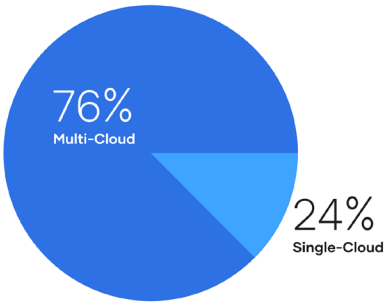
A cloud operating model is essential for organizations to succeed with cloud adoption and thrive in multi-cloud environments. This white paper explains the components of that approach and the key role of centralized platform teams in industrializing application delivery. It focuses on proven patterns for standardizing how people work, the processes they follow, and the tools they consume.

Platform teams include engineers who provision, run, and manage cloud infrastructure and other shared services. They create and operate highly automated platforms available on-demand across the organization. Developers can access the platform capabilities via self-service processes, making it easy to quickly create new environments and new service instances. The platform team's responsibility is to keep the platform stable, resilient, performant, and secure. Critically, bolstered by a reliable baseline of services from the platform team, application development teams across the organization can create and release new capabilities to users more quickly.

Over time, high-performing platform teams use feedback from developers, security engineers, and business leaders to refine processes, improve reliability, and add desirable new features.

Multi-Cloud Adoption Leads to Platform Teams

According to HashiCorp's 2021 State of Cloud Strategy Survey, 76% of survey respondents use more than one cloud. And the larger the organization, the more likely they are to use multiple clouds.



Not surprisingly, multi-cloud adoption skews heavily toward larger enterprises, but that gap closes over time:

Multi-cloud adoption by organization size

60%	76%	90%
Small businesses	**Midsize companies**	**Large enterprises**
(<100 EMPLOYEES)	(101 - 5,000 EMPLOYEES)	(>5,000 EMPLOYEES)

Source: 2021 State of the Cloud Strategy Survey

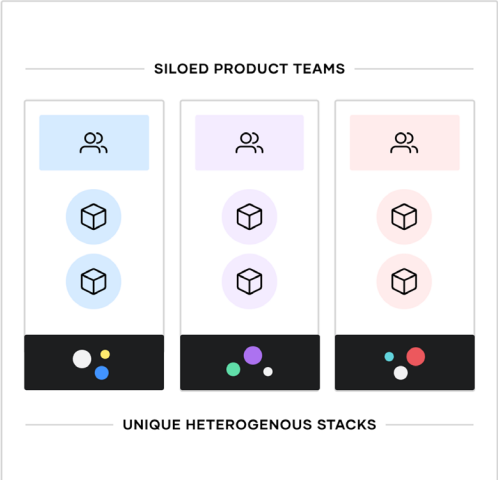
Multiple clouds have become the dominant pattern for many reasons: organic adoption, deliberate diversification, or mergers and acquisitions, to name a few. On the back of the move to multi-cloud, two new motions have become prevalent in large organizations:

1. **A shift to platform teams and a platform mindset.** This shift standardizes on a set of infrastructure services to reduce friction for developers and operations teams. It empowers a small, centralized group of platform engineers with the right tools to improve the developer experience with APIs, documentation, and advocacy.
2. **"Platform-as-a-product" practices** Heritage IT projects have a finite start and end date. That's not the case with your cloud platform; it's a product and never "finished." Ongoing tasks include backlog management, regular feature releases, and roadmap updates to stakeholders.

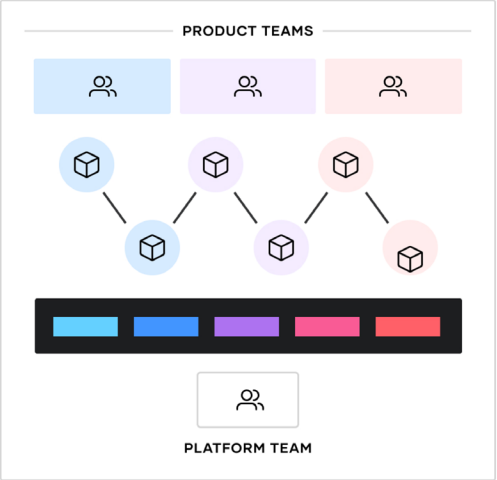
A Shift to Platform Teams and a Platform Mindset

A platform team abstracts the complexity of multi-cloud architectures away from development teams with a curated set of standardized APIs. This way, each team can "plug in" to the approved services, and focus their attention on creating custom code.

Platform teams can be a significant force multiplier for the organization; the goal is for an organization's collective cloud best practices — including security and compliance requirements — to be "baked in" to the shared platform. This model yields greater operational efficiency and developer productivity.



Organizations often develop silos of cloud usage. Here, each product team uses its own set of processes, tools, and architecture. The ongoing cost and complexity of maintaining unique technology stacks is substantial.



Organizations that use a platform team enjoy efficiencies of scale. Product teams can focus on their own applications, while the platform team provides standard services for provisioning, security, networking, and more.

Platform-as-a-Product Practices

Organizations should run their cloud platform as a product, a key principle of user-centered design. When building your cloud platform "product," the goal is to understand the needs of the teams that are building services to run atop the platform. A platform needs to have demonstrable value to promote adoption and success.

Building your platform as a product is unlike running a traditional IT project. Rather than discrete project start and end dates, platforms need iterative product development. The platform team should be constantly ingesting feedback from business and technical stakeholders, and regularly shipping new features and improvements based on this feedback.

For maximum impact, platform teams should focus on pragmatic approaches to nine key areas:

1. **Define and measure reliability:** It is crucial for developers to trust the platform team, and trust is built over time. A reliable and predictable set of services gives developers — and executives — confidence that the platform will be reliable and scalable during "moments of truth" for the business. Desired outcomes include meeting and exceeding application uptime and performance targets. Traditional service level agreements (SLAs) can be a useful measurement. Over time, more

. advanced site reliability engineering practices such as service level objectives and error budgets can yield even greater benefits.

2. **Continuously reduce toil and increase automation:** Seek to automate as much as possible. Automation increases consistency, reduces toil, and improves developer productivity. The first step to automation is to understand the steps required to complete a given workflow. Discussions with engineering teams can yield the blueprint for a given process. From there, platform teams can proceed to automate the task. The result: increased deployment frequency and faster time-to-market. Each automated workflow reduces the labor needed to achieve a new outcome later.

3. **Enable education and encourage platform adoption:** Successful enterprises tend to build and manage functional platforms that handle networking, security, infrastructure, backing services, and much more. A self-service portal and documentation for the platform and its components are crucial yet overlooked parts of this effort. Platform engineers who provide a thoughtful "developer experience" will win mindshare and encourage the use of standard patterns that speed the delivery of new applications and services.

4. **Build in security and compliance:** Security, compliance, encryption, auditing, and other InfoSec concerns should be addressed deep within the platform via thoughtful implementation of automation and reference architectures. For example, platform teams can deploy standardized golden images with the latest security patches in response to a critical vulnerability. Engineers can configure secrets management and credential rotation workflows to be performed automatically on behalf of the developer. This empowers individual teams to focus their security and compliance efforts on their particular service, rather than the entire stack. Developer productivity increases as a result.

5. **Build internal communities through advocacy:** It's not enough to mandate the use of a given platform or technology. To be effective in driving a cloud operating model, platform teams need to champion and promote the use of the platform for the right workloads and use cases. What's more, it's important to document new capabilities and share roadmap updates. Internal townhalls and hackathons can help platform teams engage with developers to learn about their goals and challenges. These efforts build empathy and trust between teams, driving platform adoption.

6. **Provide a delightful developer experience:** Developers may need to be coaxed into deploying to your organization's platform instead of competing internal options. You can attract developers by giving them what they crave most: rapid deployment options with minimal resistance. Lower your platform's barrier to adoption and create a great developer experience with sensible defaults, prescriptive guidance for common use cases, a "paved path" to production, tutorials, and code samples. In complex organizations, relief from administrative hurdles is a powerful draw for

developers to use approved platform services. Once developers learn the benefits for themselves, they'll help convince other teams around the organization.

7. **Employ pragmatic standardization:** There are many cloud services your organization can consume. Platform teams should evaluate these products and maintain a pragmatic "service catalog" exposed to developers. Platform teams can negotiate a reasonable list of supported services with business stakeholders to achieve a balance of innovation and sustainability. Done right, developers get the tools and services they need while platform teams avoid the operational overhead of managing unnecessary services.
8. **Optimize costs with "chargebacks" and "showbacks":** Platform teams should help create a more efficient model for cloud consumption. Extensive use of shared resources, automation, and standardization, for example, helps contain "server sprawl." Cost optimization comes when individual consumers understand how much they are actually spending. Platform teams can implement resource tagging to associate consumption services with a given team or business unit. This practice can provide cost transparency ("showback") or be used to draw down internal budget allocations ("chargeback"). When platform teams provide clear, transparent usage reports, budget allocation and cost optimization discussions are much easier.
9. **Align goals to business outcomes:** Finally, because a platform team is a service provider, the organization's platform approach should be aligned with business outcomes. Its success or failure should be measured against outcome-oriented metrics such as release frequency, platform stability and reliability, security-workflow efficiency, and cost optimization.

Standardize Platform Team Workflows with a Cloud Operating Model

A cloud operating model impacts teams across all layers of the stack: infrastructure, security, networking, and applications. The greater cloud maturity an organization has achieved, the faster its velocity. That's why enterprises are establishing platform teams: to deliver the dynamic services necessary at each layer for rapid application delivery, a strong security posture, and operational efficiency.

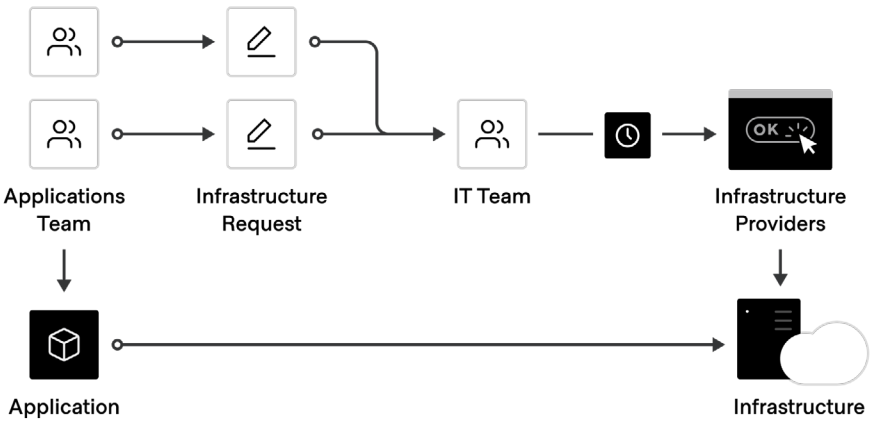
Based on what we've seen at successful organizations, here are some best-practice suggestions to adopt a cloud operating model and apply it to your platform at the infrastructure, security, networking, and application layers.

Standardize Infrastructure Provisioning with HashiCorp Terraform

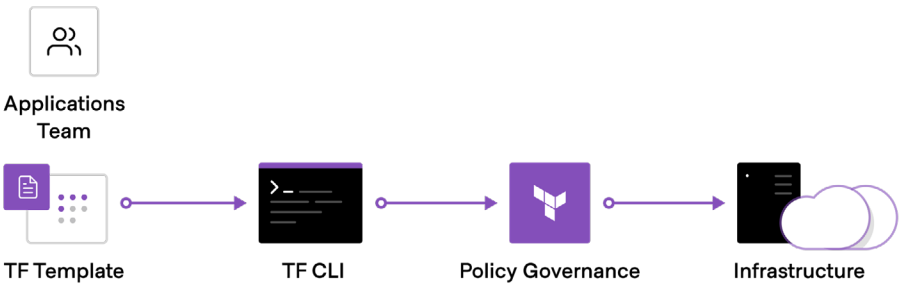
The foundation for running a cloud platform is infrastructure provisioning. HashiCorp Terraform is the world's most popular cloud provisioning product. Terraform is used to provision infrastructure for any service using an array of providers for any target cloud or popular software application.

To achieve shared services for infrastructure provisioning, platform teams should start by implementing reproducible infrastructure as code practices, and then layering compliance and governance workflows to ensure appropriate controls.

Before Terraform



After Terraform

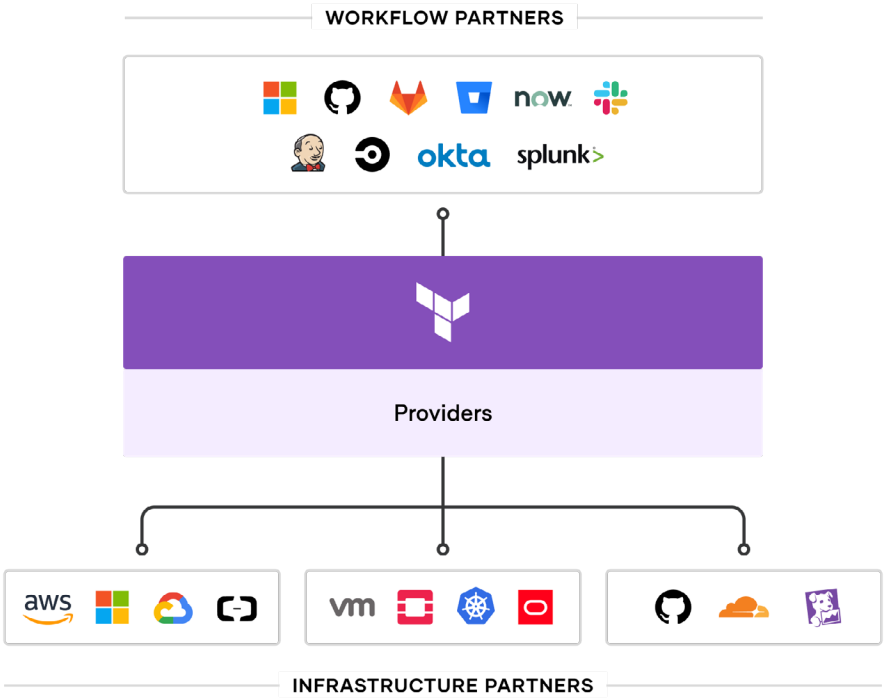


Without an automated provisioning workflow, requests for resources must be reviewed and approved manually. With HashiCorp Terraform, platform teams can automatically route all requests through pre-approved templates to deliver on-demand access to infrastructure for application teams.

Reproducible Infrastructure as Code

The first goal of a shared service for infrastructure provisioning is to enable the delivery of reproducible infrastructure as code, providing product teams a way to plan and provision resources inside CI/CD workflows using familiar tools. Ideally, this provisioning "just works" and is abstracted away from development teams.

Platform teams can create Terraform modules, which act as templates that express the configuration of services from one or more cloud platforms. Terraform integrates with all major configuration management tools to allow fine-grained provisioning to be handled following the provisioning of the underlying resources. Finally, templates can be extended with services from many other software vendors to include monitoring agents, application performance monitoring (APM) systems, security tooling, DNS, databases, and more. Once defined, the templates can be provisioned as required in an automated way. In doing so, Terraform becomes the lingua franca and common workflow for teams provisioning resources to help the platform scale, and extend the platform's capabilities.



The cloud ecosystem has standardized on HashiCorp Terraform: more than 2,000 Terraform providers help platform teams adopt consistent provisioning workflows.

For platform teams intent on delivering self-service infrastructure, the decoupling of the template-creation process and the provisioning process reduces the time taken for any application to go live, since developers no longer need to wait for operations approval as long as they use a pre-approved template.

Compliance and Management

Most platform teams also need to enforce policies on the type of infrastructure created, how it is used, and which teams get to use it. HashiCorp's Sentinel policy as code framework provides compliance and governance without requiring a shift in the overall team workflow. Sentinel is defined as code too, enabling collaboration and comprehension for the platform model.

Without policy as code, organizations resort to using a ticket-based review process to approve changes. This can become a bottleneck, making developers wait weeks or longer to provision infrastructure. Policy as code allows platform teams to solve this by abstracting the definition of the policy from the execution of the policy.

Platform teams should codify policies enforcing security, compliance, and operational best practices across all service provisioning. A "shift left" approach automates enforcement, assuring that changes are in compliance without creating a manual review bottleneck.

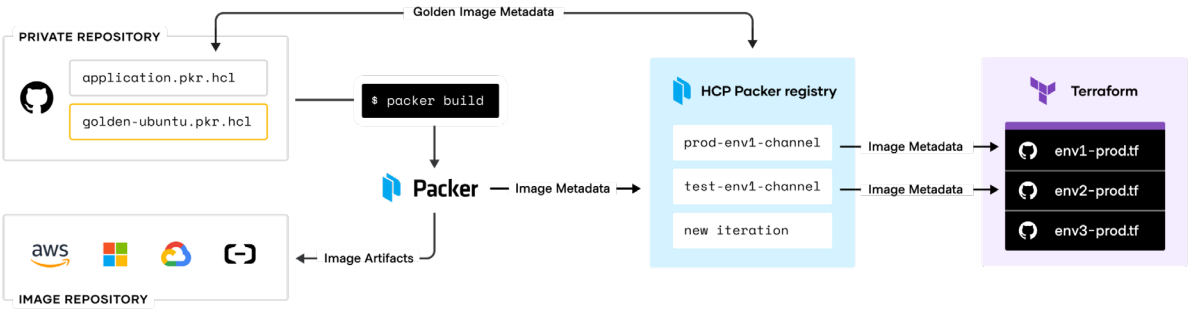
Build Automated Images with HashiCorp Packer

HashiCorp Packer is an open source tool that enables platform teams to create identical machine images for multiple clouds from a single source template. A common use case is creating "golden images" that platforms teams use to help standardize cloud infrastructure.

Packer automates the creation of any type of machine image, including Docker images, and images for use with cloud service providers. Often, images created with Packer are inputs that start a provisioning workflow with Terraform.

Platform teams that desire more automation can use the HCP Packer registry. HCP Packer is a cloud service that bridges the gap between Packer's image factories and Terraform's image deployments. This toolchain lets security and platform teams work together to create, manage, and consume images in a centralized way.

The HCP Packer registry stores metadata about the organization's golden images, including when they were created, where the image exists in the cloud, and what (if any) git commit is associated with the image build. The registry tracks information about the golden images the Packer builds produce, and clearly designates which images are appropriate for test and production environments. From there, the golden images can be used in infrastructure provisioning workflows.



Manage Secrets and Protect Data with HashiCorp Vault

Dynamic cloud infrastructure means a shift from host-based identity to application-based identity, with low or zero trust networks across multiple clouds without a clear network perimeter.

In the traditional security world, we assumed high trust internal networks, which resulted in a hard shell and soft interior. With the modern "zero trust" approach, we work to harden the inside as well. This requires that both humans and applications be explicitly authenticated, specifically authorized to fetch secrets and perform sensitive operations, and tightly audited.

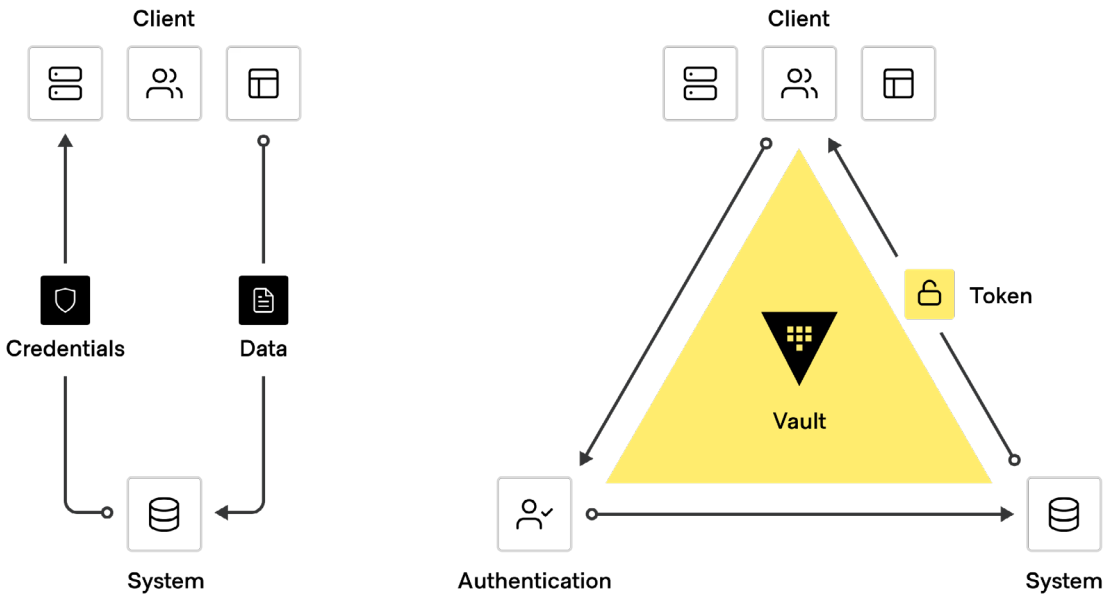
HashiCorp Vault enables platform teams to securely store and tightly control access to tokens, passwords, certificates, and encryption keys. This provides a comprehensive secrets management solution for machines and applications. Beyond that, Vault helps protect data at rest and data in transit. Vault exposes a high-level API for cryptography for developers to secure sensitive data without exposing encryption keys. Vault also can act as a certificate authority, to provide dynamic, short-lived certificates to secure communications with SSL/TLS. Lastly, Vault enables a brokering of identity between different platforms, such as Azure Active Directory (AAD) and AWS Identity and Access Management (AWS IAM) to allow applications to work across platform boundaries.

Vault is widely used by platform teams across industries from stock exchanges to large financial organizations and hotel chains to provide security in a cloud operating model.

To achieve shared services for security, platform teams should enable centralized secrets management services, and then use that foundation to deliver more advanced Encryption-as-a-Service use cases such as certificate and key rotations, and encryption of data in transit and at rest. This embeds security considerations within the platform, so product teams need only "plug in" to the provided APIs to ensure their service meets corporate security standards.

Before Vault

After Vault

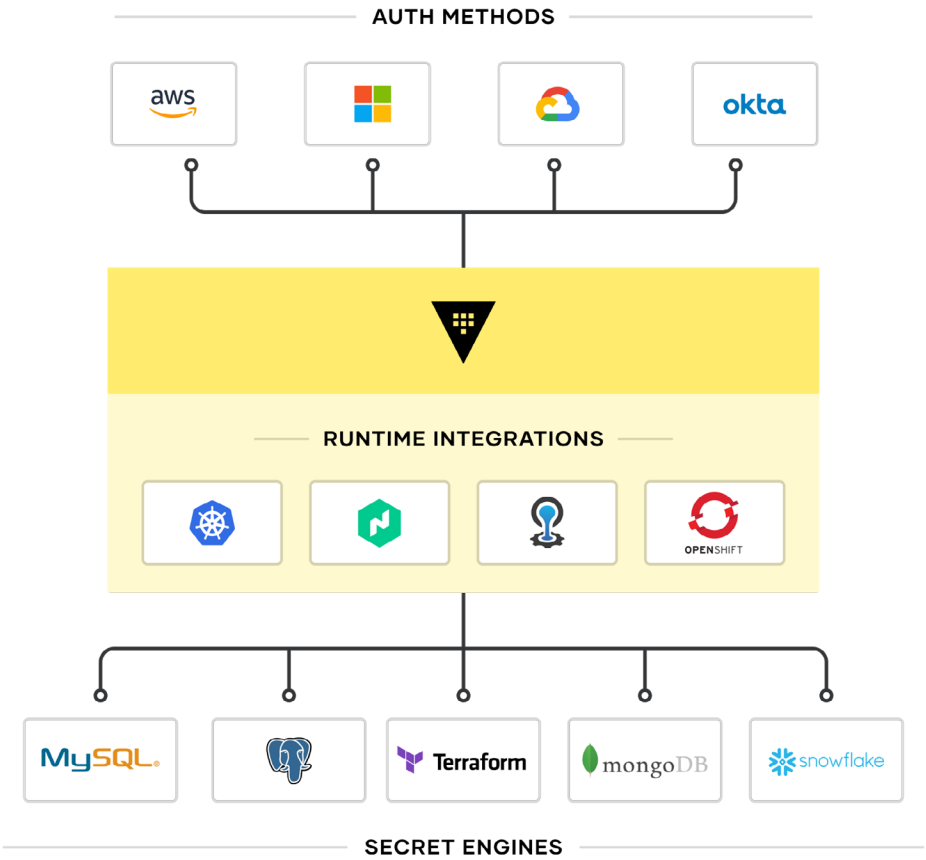


Traditional high trust deployments do not have a programmatic way to protect passwords and other sensitive information. With HashiCorp Vault, platform teams implement an automated workflow for both people and machines to centrally manage access to credentials.

Secrets Management

The first step in cloud security is secrets management: the central storage, access control, and distribution of dynamic secrets. But instead of depending on static IP addresses, it's vital to integrate with identity-based access systems such as AAD and AWS IAM to authenticate and access services and resources.

Vault uses policies to codify how applications authenticate, which credentials they are authorized to use, and how auditing should be performed. It can integrate with an array of trusted identity providers such as cloud identity and access management platforms, Kubernetes, Active Directory, and other SAML-based systems for authentication. Vault then centrally manages and enforces access to secrets and systems based on trusted sources of application and user identity.



HashiCorp Vault works with common sources of identity to be a trusted identity broker at scale.

Platform teams should build a shared service that enables the request of secrets for any system through a consistent, audited, and secured workflow. This kind of shared service not only improves security, it also improves developer productivity. Developers are spared the pain of manually searching for references to a compromised secret. Instead, Vault remediates the secret with an automated workflow of renewal and revocation.

Encryption-as-a-Service

Additionally, enterprises need to encrypt application data at rest and in transit. Vault can provide Encryption-as-a-Service to provide a consistent API for key management and cryptography. This allows platform teams to perform a single integration and then protect data across multiple environments.

]Using Vault as a basis for Encryption-as-a-Service solves complex problems faced by platform teams and security engineers, such as certificate and key rotation. Vault enables centralized key management to simplify encrypting data in transit and at rest across clouds and datacenters. This helps reduce costs around expensive hardware security modules (HSMs) and increases productivity with consistent security workflows and cryptographic standards across consumers of the platform.

While many organizations provide a mandate for developers to encrypt data, they don't often supply the "how", which leaves developers to build custom solutions without an adequate understanding of cryptography. Platform teams use Vault to provide developers a simple API, while adjacent security teams can use policy controls and lifecycle management APIs as needed.

Advanced Data Protection

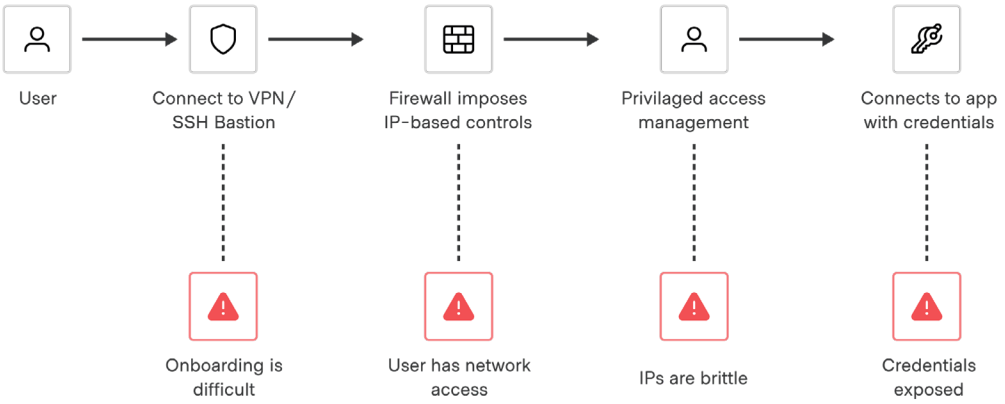
Often, platform teams must still maintain and support on-premises services and applications that need to perform cryptographic operations, such as data encryption for storage at rest. There may not be a business case to improve these applications in this fashion; the custom engineering required to manage these cryptographic keys can be substantial. The solution: delegate the task of key management to external providers. Vault's Advanced Data Protection capability allows organizations to securely connect, control, and integrate advanced encryption keys, operations, and management between infrastructure and Vault, including protecting data in MySQL, MongoDB, PostgreSQL, and other databases using transparent data encryption (TDE).

Advanced Data Protection provides platform teams with functionality for data tokenization, such as data masking, to protect sensitive data like credit card numbers and banking details. As such, the capability is popular with organizations that have high security requirements for data compliance (PCI DSS, HIPAA, etc.).

Secure and Manage Access with HashiCorp Boundary

The modern security principle of identity extends beyond the machine-to-machine access workflows addressed by Vault. Identity is equally central to securing human-to-machine interactions.

Traditional solutions for safeguarding user access require distributing and managing SSH keys, VPN credentials, and bastion hosts. This approach creates risks around credential sprawl, and requires substantial manual effort to maintain. For example, it's all too common for organizations to reduce toil by setting their SSH keys to never expire, which gives users access to entire networks and systems indefinitely.



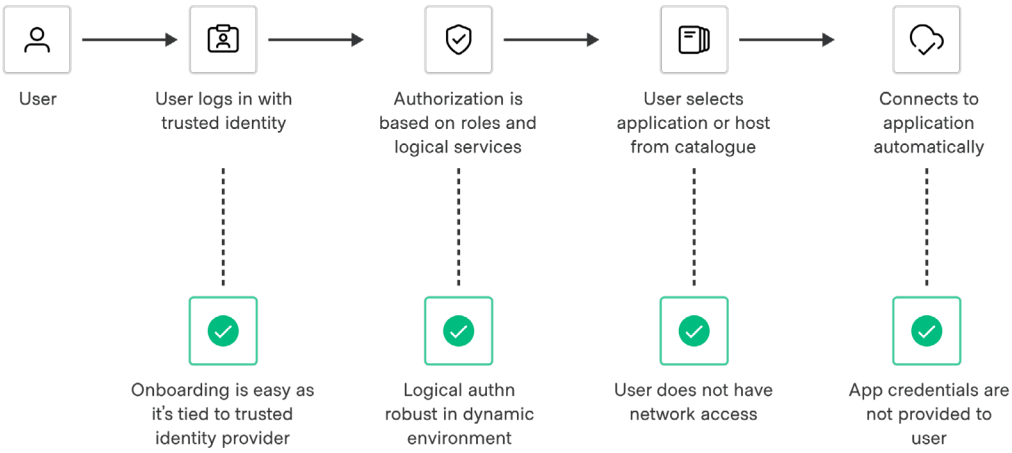
Traditional access workflows are manual and introduce multiple points of vulnerability to the system.

Consider this common scenario: a developer trying to access a production app to troubleshoot an issue. In a traditional access model, the app is likely deployed to a private network (or VPC) on a set of well-known virtual IP addresses. Access is configured at the IP level.

To configure access to the application in question, administrators need to go through many access layers, each requiring its own configuration:

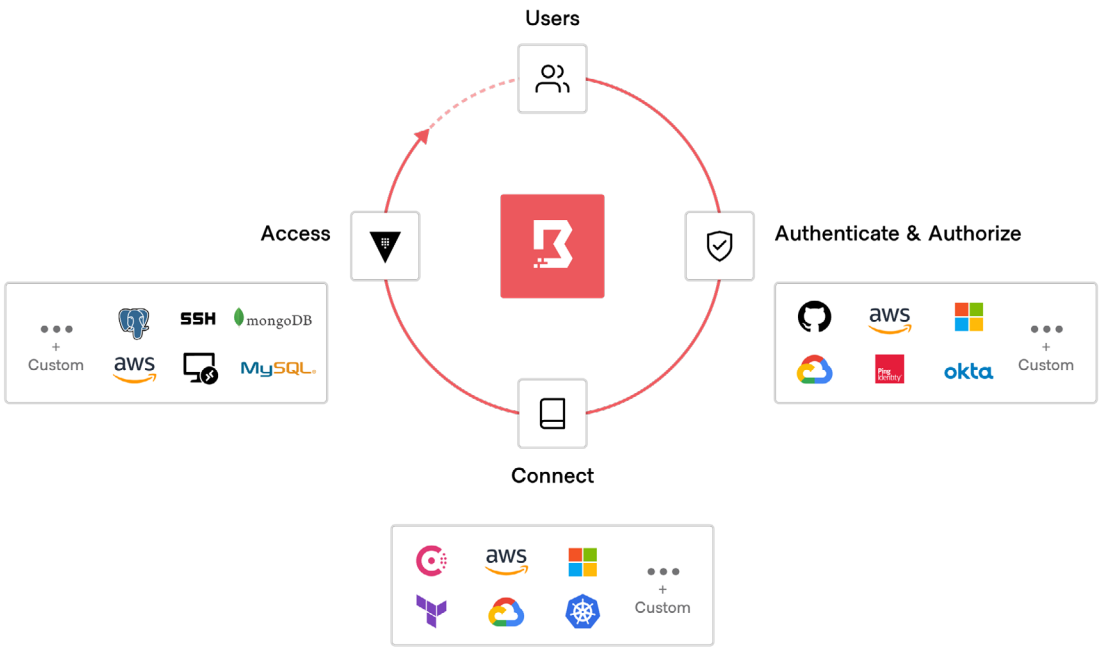
- The corporate VPN/private network layer that lacks granular access controls
- The bastion host/jumphosts layer that straddles the corporate network and datacenter network
- The network firewall layer (or other location-based controls) that enforces permissions at the client/target IP level, rather than based on logical identity

This philosophy breaks down in the world of cloud infrastructure, due to the nature of ephemeral cloud resources and dynamic IP addresses. A modern approach for human-to-machine access should rely on verifying the identity of the user. Further, access and provisioning of permissions to the end system should be automated.



Modern access workflows are automated and use identity as their foundation.

HashiCorp Boundary solves modern access challenges in a cloud operating model. Boundary is a secure remote access solution that provides an easy way to safeguard access to applications and critical systems with fine-grained authorizations based on trusted identities. The product governs access across clouds, local datacenters, and low-trust networks, without exposing the underlying network.



HashiCorp Boundary delivers simple and secure remote access to any system anywhere, based on user identity.

Users can authenticate to Boundary using their identity provider of choice (AAD, Okta, AWS IAM, etc.). From there, each user can be tightly authorized to perform actions on a dynamic set of targets, and be granted just-in-time access to connect to those targets via credentials provided by Vault or another credential-management solution.

This model fits the platform team approach of shared services, because it provides secure access to dynamic infrastructure with:

- **Identity-based access controls:** Platform engineers can streamline just-in-time access to privileged sessions (e.g. TCP, SSH, RDP) for users and applications. Teams can control access permissions with extensible role-based access controls.
- **Access automation:** Platform teams can define the perimeter of resources, identities, and access controls as code through Boundary's fully instrumented Terraform provider, REST API, CLI, and SDK. They can automate the discovery of new resources and enforcement of existing policies as resources are provisioned.
- **Session visibility:** Security engineers can monitor and manage each privileged session established with Boundary. Session logs can be exported to a wide variety of analytics tools.

No matter how the platform's capabilities evolve, Boundary can support secure remote access to any number of systems and applications.

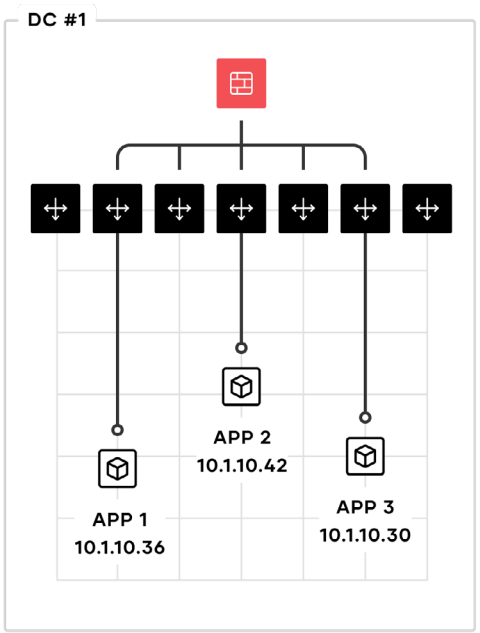
Securely Connect Applications with HashiCorp Consul

Networking in the cloud is one of the most difficult challenges for platform teams adopting a cloud operating model. Engineers must navigate dynamic IP addresses, account for a significant growth in east-west traffic in microservices implementations, and adjust to the lack of a clear network perimeter.

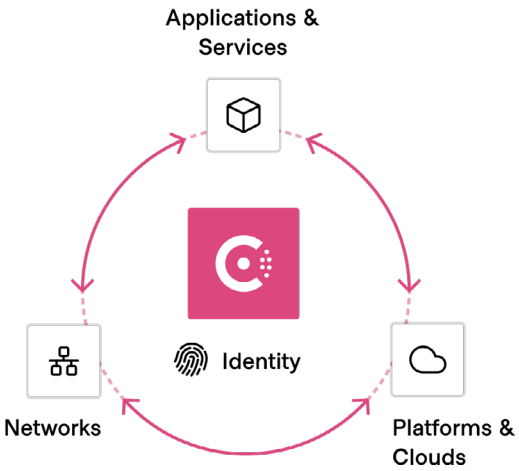
HashiCorp Consul enables platform teams to manage multi-cloud networks by discovering and securely connecting services. Consul is widely deployed to run service networks at a global scale.

Networking services should be based on service identity and provided centrally, allowing platform teams to create a centralized service registry for discovery purposes. The common registry provides a "map" of what services are running, where they are, and their current health. The registry can be queried programmatically to enable service discovery or drive network automation of API gateways, load balancers, firewalls, and other critical middleware components. Service mesh approaches can simplify the network topology, especially in multi-cloud and multi-datacenter environments.

Before Consul



After Consul



Historically, networking involved manual activities that connect hosts and static IP addresses. HashiCorp Consul gives modern cloud architectures and platform teams an automated, services-centric approach.

Service Discovery

The starting point for networking in a cloud operating model is a common service registry, which provides a real-time directory of what services are running, where they are on the network, and their current health. Traditional approaches to networking assume a static IP for long-lived applications, using a combination of load balancers and firewalls to regulate communication. Tracking the network location of services often requires disjointed manual processes and tools such as spreadsheets, load balancer dashboards, or configuration files.

With Consul, each service is programmatically registered and DNS and API interfaces are provided to enable any service to be discovered by other services. Consul's integrated health check monitors each service instance's health status so the platform team can triage the availability of each instance and Consul can help avoid routing traffic to unhealthy service instances.

Consul can be integrated with other services that manage existing north-south traffic, such as traditional load balancers, and distributed container orchestrators such as Kubernetes, to provide a consistent registry and discovery service across cloud platforms.

Network Infrastructure Automation

The next step is to reduce the operational complexity of existing networking infrastructure through network automation. Instead of a manual, ticket-based process to reconfigure load balancers and apply firewall rules every time there is a change in service network locations or configurations, Consul can automate these network operations using Terraform to execute changes based on predefined tasks. This is achieved by enabling network infrastructure devices to subscribe to service changes from the service registry, creating highly dynamic infrastructure that can scale significantly higher than static-based approaches.

This configuration — whereby Terraform executes configurations based on event changes detected by Consul — removes dependencies and obstacles for common tasks. Product teams can independently deploy applications while platform teams can rely on Consul to handle the downstream automation those product teams require. The benefits persist throughout the lifecycle of the service: automation can properly close firewall ports as services are retired.

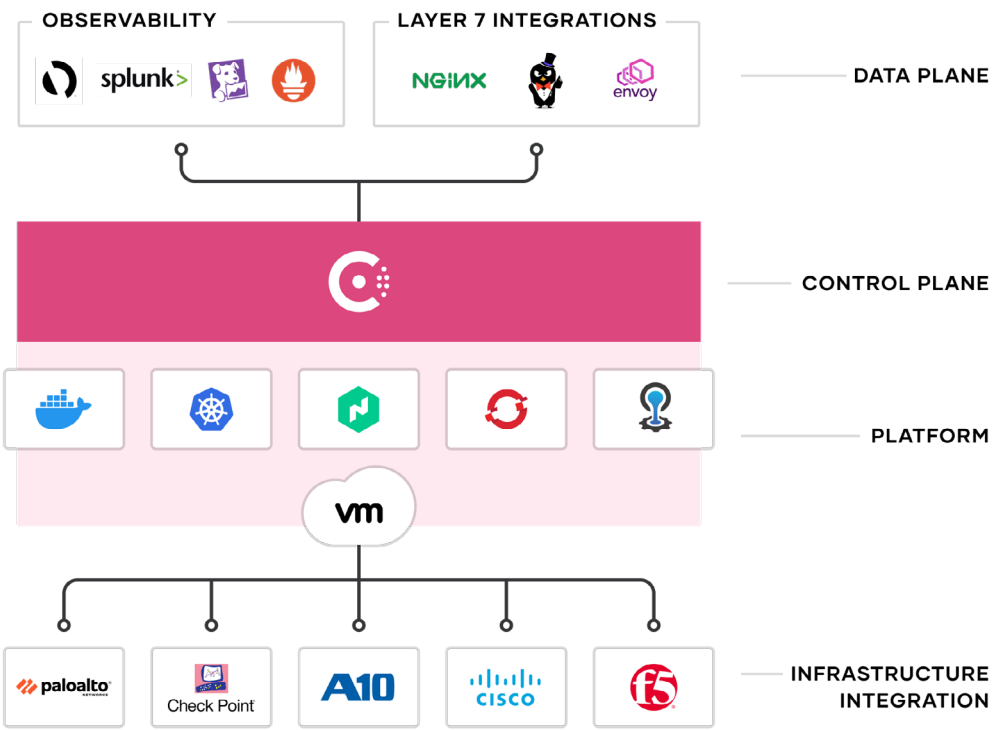
Zero Trust Networking with Service Mesh and API Gateway

As organizations scale with microservices-based and cloud-native applications, the underlying infrastructure becomes larger and more dynamic, with an explosion of east-west traffic.
Also, controlling external client access to these microservices ratchets up the volume of north-south traffic. This requires an unwanted proliferation of expensive network middleware devices that bring single points of failure and significant operational overhead for platform teams.

Consul provides a distributed service mesh that pushes routing, authorization, and other networking functions to the endpoints in the network, rather than imposing them through middleware. This makes the network topology simpler and easier to manage, reduces the need for expensive middleware devices within east-west traffic paths, and makes service-to-service communication more reliable and scalable. Adding Consul's API Gateway provides consistent control and security for how north-south traffic is handled through a single, centralized control plane.

Consul is an API-driven control plane that integrates with sidecar proxies alongside each service instance (proxies such as Envoy). These proxies provide the distributed data plane. Together, these two planes enable a zero trust network model that ensures all service-to-service communication is

authenticated, authorized, and encrypted. This security posture is achieved with automatic mutual TLS encryption and identity-based authorization. Platform teams can work with the appropriate stakeholders to define security policies with logical services (rather than IP addresses) and provide least-privilege access to developers.



HashiCorp Consul provides a single control plane to enable a broad ecosystem for service networking.

Consul can be integrated with Vault for centralized PKI and certificate management with automatic certificate rotation on both the control plane and data plane. Consul's integration also extends to Kubernetes deployments, including the storage of sensitive data such as keys and tokens in Vault. This approach reduces risk compared to relying on native Kubernetes secrets.

Consul Service Mesh secures service connections across any cloud environment, and on any runtime. This consistent dataplane allows developers and platform teams to connect their services across heterogeneous environments and abstractions. Furthermore, Consul supports multi-tenancy with

Administrative Partitions. With this feature, multiple deployments can remain under a single control plane allowing for consistent management and governance while maintaining autonomy and isolation for different tenants.

Standardize Application Deployment with HashiCorp Waypoint

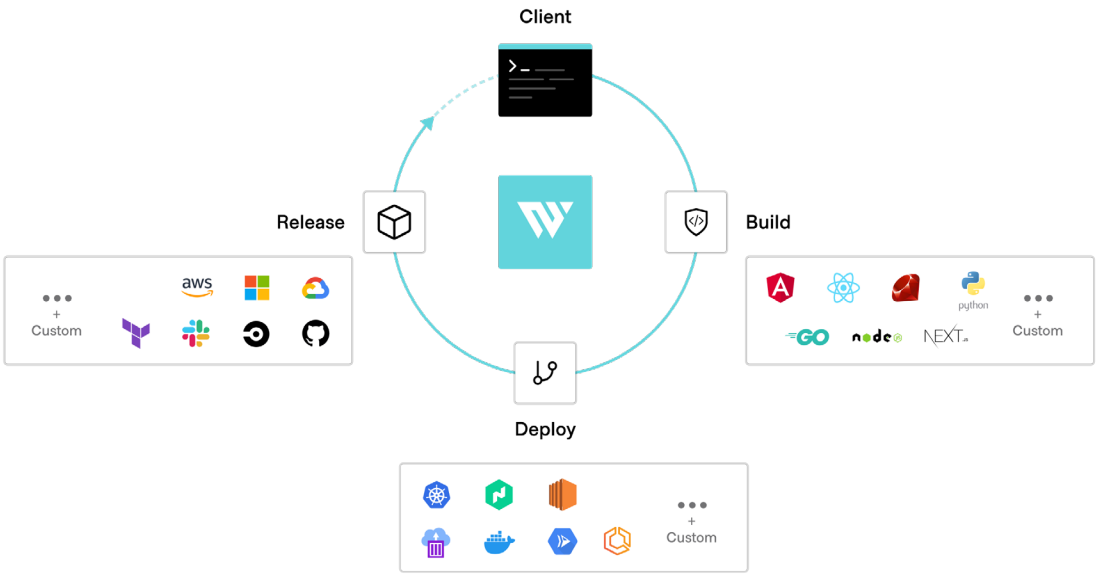
Organizations are creating new "systems of engagement" and modernizing large swaths of their application estate. Applications slated for modernization span a range of architectural styles, development frameworks, and business importance.

This process can flood developers with complexity: containers, schedulers, YAML files, serverless, and more. This complexity brings some benefits: microservices architectures can yield transformative outcomes. But the cost can be seen in the learning curve required to get a single service deployed to production.

Another challenge common in large organizations is that different tools deploy to different environments. Developers use Docker and kubectl for Kubernetes, HashiCorp Packer and Terraform for VMs, custom CLIs for each serverless platform, and so on. For individual developers and operators, this fragmentation poses a learning curve challenge. For teams, it's a challenge around consistency, complexity, and security.

Build, Deploy, Release

A cloud operating model unifies this fragmentation by establishing a golden workflow for production deployments with HashiCorp Waypoint. Waypoint provides a modern workflow to build, deploy, and release across runtimes.



HashiCorp Waypoint is a centralized developer experience for building, deploying, and releasing applications.

Waypoint provides one easy-to-use command to deploy any application: `waypoint up`. This workflow is consistent across all platforms, including Kubernetes, HashiCorp Nomad, Amazon EC2, and Google Cloud Run. Waypoint can be extended with plugins to target any build/deploy/release logic. Platform teams can add Waypoint to continuous deployment workflows like GitHub Actions, GitLab CI/CD, Jenkins, and CircleCI.

After deployment, Waypoint provides features such as logs, exec, and more to validate and debug any deployments.

Waypoint's Implications for Platform Teams

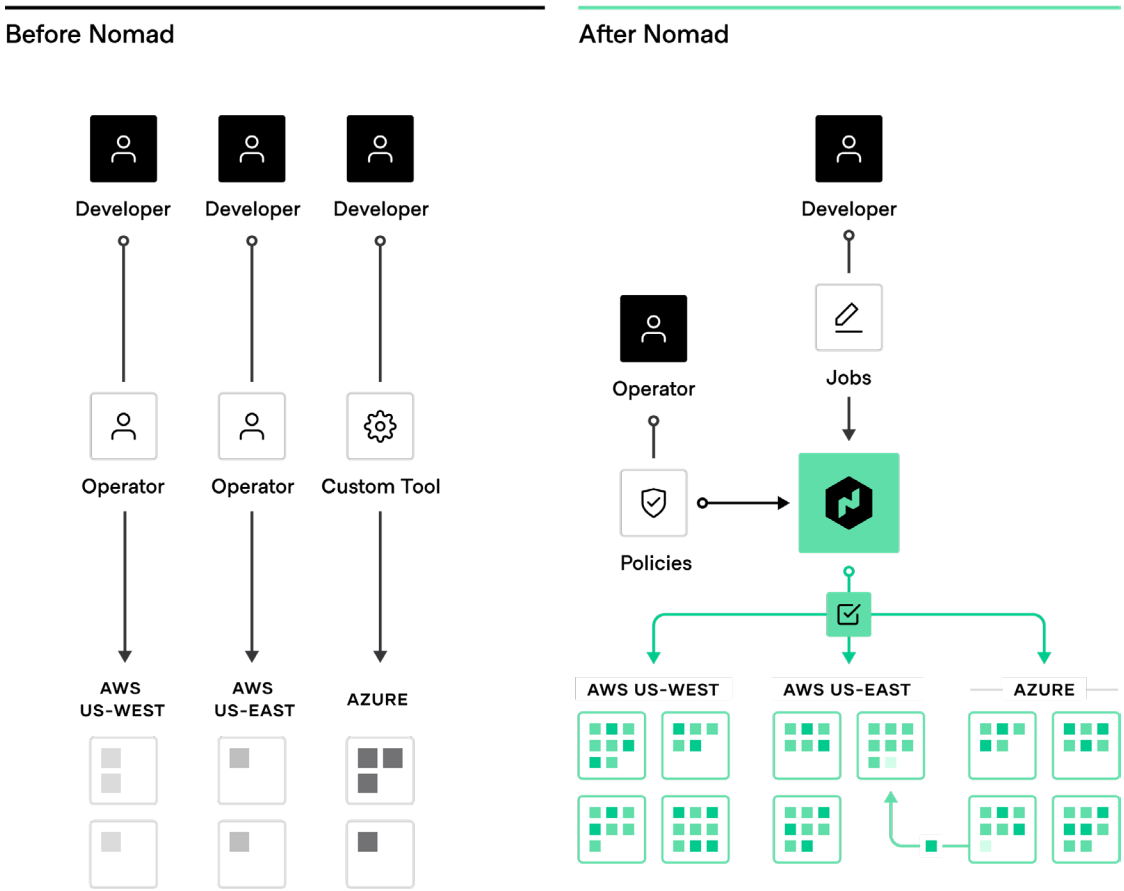
One of the most significant challenges for a platform team is delivering a single path to production that suits the needs of many development teams, InfoSec professionals, and compliance audits. Each development team builds services with a distinct blend of frameworks, backing services, target runtimes, deployment frequencies, and operational considerations. Security and compliance requirements must also be met.

Platform teams can use Waypoint in concert with Terraform, Vault, Boundary, and Consul to enable the consistent delivery of applications on cloud infrastructure, incorporating necessary compliance, security, and networking requirements.

Standardize Workload Orchestration with HashiCorp Nomad

At the application layer, new apps are increasingly distributed while heritage apps still need to be managed more flexibly. HashiCorp Nomad is a flexible orchestrator that can deploy and manage both traditional and modern applications for all types of workloads: from long-running services to short-lived batch jobs to system agents.

To get the benefits of shared services for application delivery, platform teams should use Nomad in concert with Terraform, Vault, and Consul. This combination enables the consistent delivery of applications on cloud infrastructure, while meeting necessary compliance, security, and networking requirements.

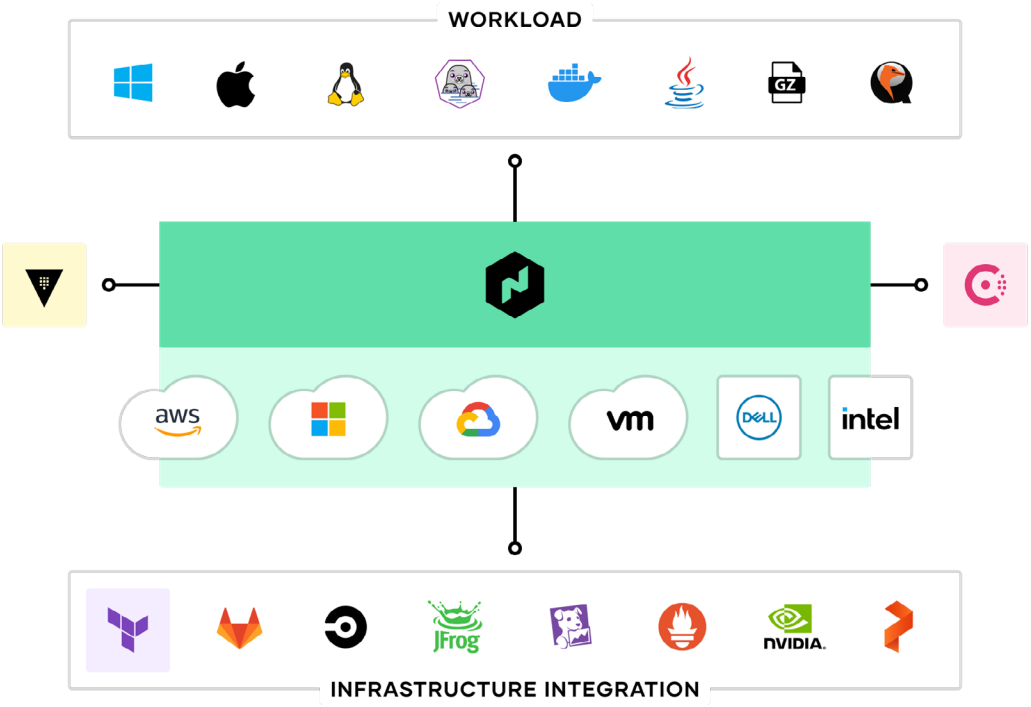


Development teams often build their own toolchain and deployment processes, which can result in operational inefficiencies. With Nomad, organizations can standardize on Nomad's single job spec and use a single workflow to run different types of applications across clouds.

Mixed Workload Orchestration

Today, many new workloads are developed with container packaging to be deployed to Kubernetes or other runtimes. But many legacy workloads will not be moved onto those platforms, nor will future serverless applications. Nomad provides a consistent process for deployment of all workloads from virtual machines through standalone binaries and containers. It provides core orchestration benefits across all those workloads, such as release automation, upgrade strategies, bin packing, and resilience.

For modern applications — typically built in containers — Nomad provides a consistent workflow at scale in any environment. Nomad is focused on simplicity and effectiveness at orchestration and scheduling, and avoids the complexity of schedulers such as Kubernetes that require specialized skills to operate and solve only for container workloads.



HashiCorp Nomad integrates with workflows and technologies already in use.

Nomad integrates into existing CI/CD workflows to provide fast, automatic application deployments for heritage and modern workloads.

High-Performance Compute

Nomad is designed to schedule applications with low latency across very large clusters. This is critical for platform teams that process large batch jobs, as is common with high performance computing (HPC) workloads. In the 2 Million Container Challenge, Nomad was able to schedule 2 million Docker containers in 22 minutes across 10 AWS regions globally, at an average rate of nearly 1,500 containers per second. Several enterprise Nomad deployments run at even larger scales.

Nomad makes it easy for high-performance applications to use an API to consume capacity dynamically, enabling efficient sharing of resources for data analytics applications like Apache Spark. The low-latency scheduling ensures results are available quickly and minimizes wasted idle resources.

Multi-Datacenter Workload Orchestration

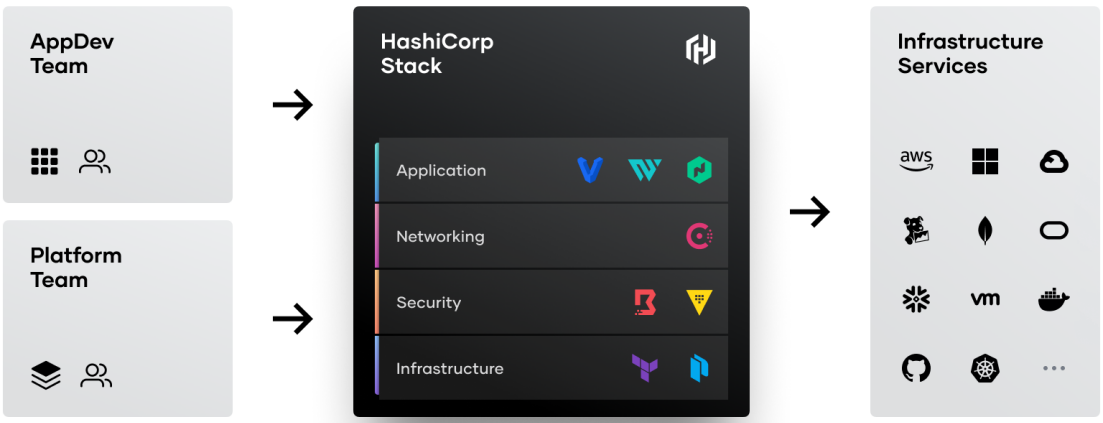
Nomad is multi-region and hybrid cloud by design, with a consistent workflow for deploying any workload. As platform teams roll out global applications in multiple datacenters or across cloud boundaries, Nomad provides orchestration and scheduling. The product is supported by infrastructure, security, and networking resources to help ensure the application is successfully deployed.

Conclusion: People, Processes, and Tools

Unlocking the fastest path to value in a modern multi-cloud environment requires coupling a common cloud operating model with the power of platform teams across three dimensions — people, processes, and tools. Each dimension requires its own shifts for optimal results:

- **People: Shift to site reliability engineering (SRE) practices**
  - Reuse expertise from internal datacenter management and single cloud vendors and apply them consistently in any environment.
  - Embrace DevSecOps and other agile practices to continuously deliver increasingly ephemeral and distributed systems.
  - Enable IT staff to automate with code and treat operations as if it is a software problem.
- **Processes: Shift to self-service**
  - Set up the platform team as an enabling shared service focused on application delivery velocity — shipping software ever more rapidly with minimal risk.
  - Establish centers of excellence for infrastructure, security, networking, and other functional areas for self-service delivery of capabilities.
  - Standardize on a "service catalog" with a set of common services for development teams to use, and add new capabilities according to feedback over time.
- **Tools: Shift to dynamic environments**
  - Use tools that support the increasing ephemerality and distribution of infrastructure and applications and that support critical workflows rather than being tied to specific technologies.
  - Provide policy and governance tooling to match the speed of delivery with compliance to manage risk in a self-service environment.
  - Embed security and compliance requirements within the platform itself to accelerate production deployments.

Unlocking the Cloud Operating Model



Put it all together, and it's clear that adopting a common cloud operating model is an inevitable shift for enterprises aiming to maximize their digital transformation efforts. Enterprise IT is evolving away from ITIL-based control points — focused on cost optimization — toward becoming self-service enablers focused on speed optimization.

Platform teams support this by delivering shared services across each layer of the cloud, helping product teams to deliver new business and customer value at speed. The HashiCorp suite of tools provides solutions for each layer of the cloud infrastructure to enable platform teams to successfully lead this shift to a cloud operating model.

About HashiCorp

HashiCorp is a leader in multi-cloud infrastructure automation software. The HashiCorp software suite enables organizations to adopt consistent workflows and create a system of record for automating the cloud: infrastructure provisioning, security, networking, and application deployment. HashiCorp's portfolio of products includes Vagrant™, Packer™, Terraform®, Vault™, Consul, Nomad™, Boundary, and Waypoint™. HashiCorp offers products as open source, enterprise, and as managed cloud services. The company is headquartered in San Francisco, though most of HashiCorp employees work remotely, strategically distributed around the globe. For more information, visit hashicorp.com or follow HashiCorp on Twitter @HashiCorp.

Getting Started Checklists

Standardize Platform Team Workflows with a Cloud Operating Model

- [ ] Sign up for Terraform Cloud
- [ ] Sign up for HashiCorp Cloud Platform
- [ ] Standardize infrastructure provisioning with HashiCorp Terraform
  - [ ] Establish reproducible infrastructure as code practices
  - [ ] Automate compliance and management workflows
- [ ] Build automated images with HashiCorp Packer
- [ ] Manage secrets and protect data with HashiCorp Vault
  - [ ] Establish secrets management automation
  - [ ] Implement Encryption-as-a-Service
  - [ ] Adopt Advanced Data Protection
- [ ] Secure and manage access with HashiCorp Boundary
- [ ] Securely connect applications with HashiCorp Consul
  - [ ] Establish service discovery patterns
  - [ ] Implement network infrastructure automation
  - [ ] Adopt zero trust networking with a service mesh and an API Gateway
- [ ] Standardize application deployment with HashiCorp Waypoint
- [ ] Standardize workload orchestration with HashiCorp Nomad

Platform-as-a-Product Practices

- [ ] Define and measure reliability
- [ ] Continuously reduce toil and increase automation
- [ ] Enable education and encourage platform adoption
- [ ] Build in security and compliance
- [ ] Build internal communities through advocacy
- [ ] Provide a delightful developer experience
- [ ] Employ pragmatic standardization
- [ ] Incentivize cost optimization with ”chargebacks” and “showbacks”
- [ ] Align goals to business outcomes

www.hashicorp.com