HashiCorp | GitHub

Increasing Developer Velocity in the Cloud Operating Model

HashiCorp

Executive summary

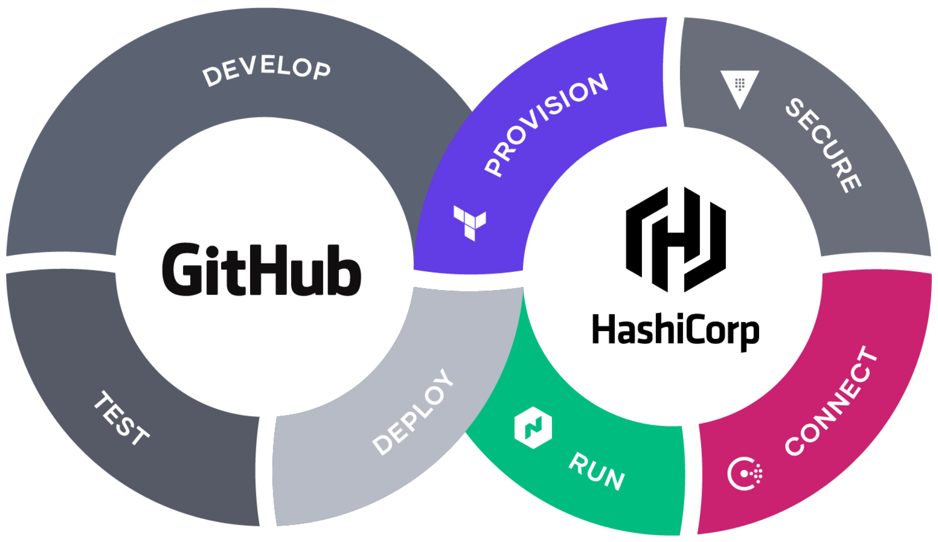
Success for modern enterprise development teams can be measured in the time it takes to develop and ship. The objective of any modern enterprise development team is to ship more value to more customers more quickly than before.

To maximize the scale and availability of experiences to customers, trends such as microservices and cloud nativism has increased the complexity of delivering apps owing to the highly distributed and service-based architectures those trends enforce. This in turn has increased the need for developers to automate infrastructure operations to increase velocity with which they can deploy new applications.

Throughout this, development and operations teams desire one consistent workflow to achieve the delivery of any application in the portfolio: delivering value more rapidly, while maintaining process cohesion and efficiency, with less risk of failure.

As enterprises start to scale in a multi-cloud environment, the Continuous Integration and Continuous Deployment (CI/CD) process is no longer just a developer specific workflow. It is a DevOps workflow that involves both developers and operators. Developers are the inner loop of the process and are able to modify and iterate on code multiple times a day while operators are the outer loop, pushing software on demand at any time of the day.

In order to properly implement this workflow, operators need to provision and maintain infrastructure at the same speed as developers. In this paper, we explore how GitHub and Terraform combine to provide a powerful CI/CD solution that makes this a reality.



The powerful combination of GitHub and HashiCorp are working together to enable organizations to unlock the Cloud Operating Model for enterprise development teams to accelerate value delivery to their users.

Application deployment in the multi-cloud world

In the past, application deployments were a fairly straightforward process. Developers would create the code on their own machines and test it using pre-configured virtual machines (VMs) provided to them by the operations team. These VMs would adhere to organizational requirements for operating system, size, and other specifications. Once the code was finalized, it was packaged and then sent to production. Teams would work through standard - albeit slow - processes such as ITIL.

Cloud's on-demand capabilities have created significant opportunities for development teams to create applications faster and better than ever before. Now environments can be created in a matter of seconds and are accessible from any machine. Application and infrastructure code is stored in a shared repository and pushed to cloud-based environments that can be located anywhere.

This world of self-service for development teams creates new challenges for operations teams and organizational controls. As a result, organizations need to have a centralized way to efficiently and securely manage cloud infrastructure that scales with developer demand.

The multi-cloud reality

According to a July 2018 study by Forrester and Virtustream, 86% of the companies surveyed described their current cloud strategy as multi-cloud. When asked further the reasons for this, the answer was simple; there is not one cloud platform that meets all of their enterprise requirements. Combine this with a desire to use multiple clouds for disaster recovery and that companies may inherit additional cloud environments through mergers and acquisitions, having a multi-cloud strategy is logical, and inevitable, for the majority of large-scale enterprises.

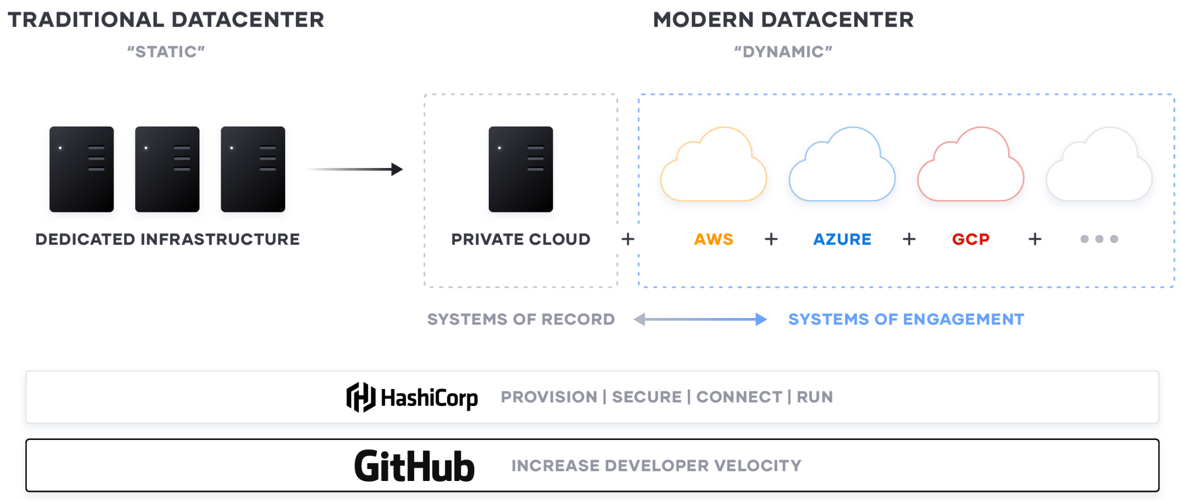
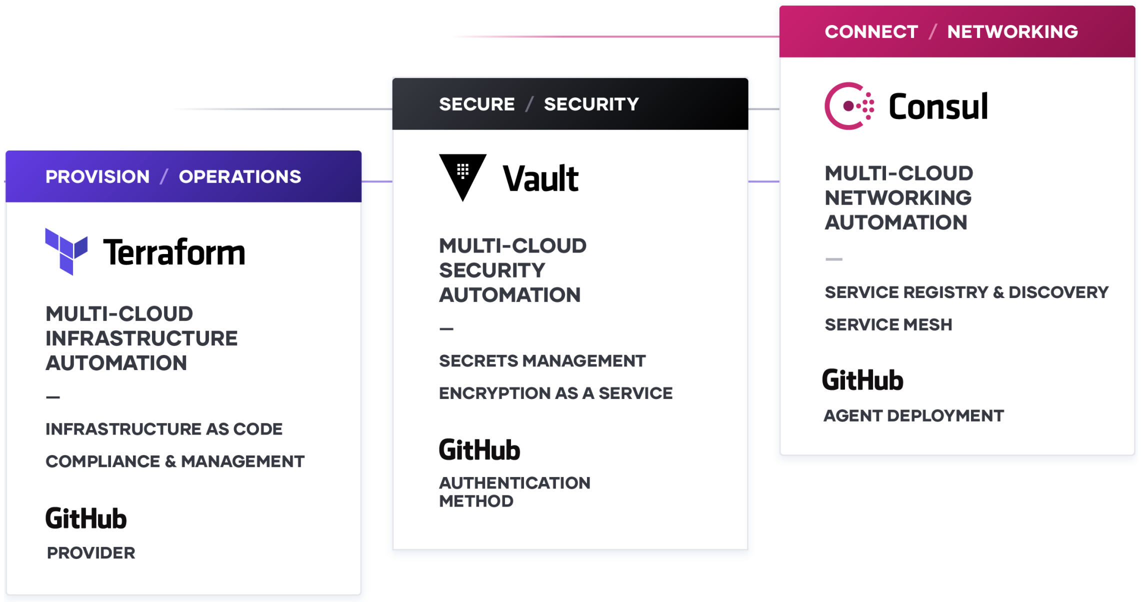


Figure 1: Static to dynamic shift

The HashiCorp Cloud Operating Model is a blueprint for how organizations migrate to, and address the challenges of, a multi-cloud reality, to take advantage of a computing model that scales dynamically, on demand. Each cloud vendor has specific offerings to assist with application delivery, but none match the need for a single consistent workflow and the advantages that brings to people and processes. Enterprises should plan and build workflows for consistent automation at every layer infrastructure, security, networking, and runtime.

Details of the process and challenges can be seen in HashiCorp's 'Unlocking the Cloud Operating Model' white paper, but here we instead drill down into the layers of provisioning infrastructure, secrets management, and service-based networking and examine how they align to the application lifecycle. Managing this lifecycle is critical to ensure consistent and secure deployments across multiple clouds, to deliver on the Cloud Operating Model.



Increasing developer velocity

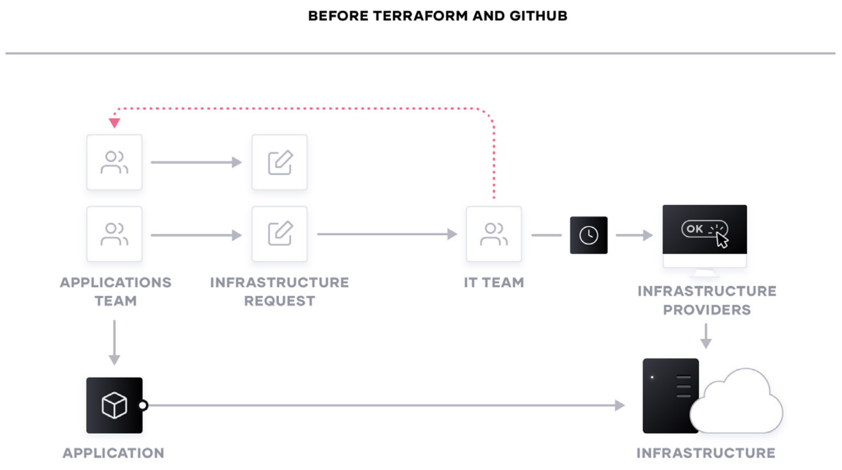
Development teams are capitalizing on the ability to self-serve environments in the cloud, and shifting to increasingly rapid deployment cycles. This shift enables teams to more quickly address customer feedback, apply new innovation and constantly improve the customer experience in a space where expectations are growing quickly. Core to supporting this approach to software development is a strong DevOps foundation that enables developer collaboration and limits the manual work that is required to support developer workflows. Workflow automation across the software development lifecycle reduces the transactional cost of software development and reduces the time between ships as developers are able to spend more time focusing on writing code than managing process.

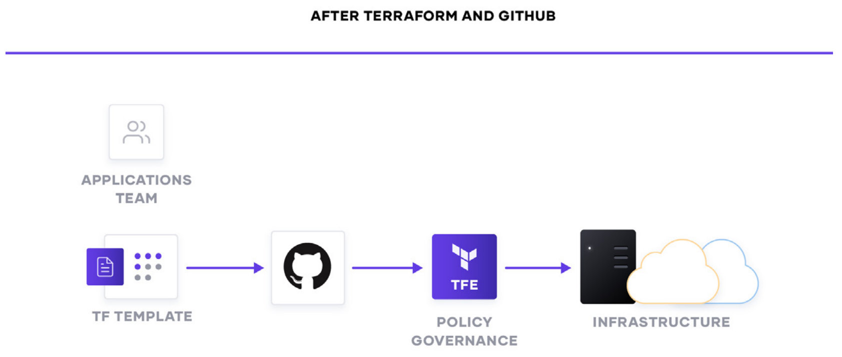
GitHub, through GitHub Actions with built in CI/CD, has developed a powerful automation solution that allows for collaboration across the SDLC harnessing many of the same collaboration principles that have helped drive innovation in the Open Source Community. Through the ability to share actions and workflows teams can quickly build on industry and internal best practices to establish workflows built around accelerating velocity and deliver code to production faster. Seamless integration of environment provisioning as part of workflow automation is a critical component of enabling rapid iteration cycles. According to a recent Forrester Total Economic Impact study, organizations leveraging GitHub for DevOps save 45 minutes of time per developer per day allowing them to spend more time coding.

HashiCorp and GitHub manage infrastructure and application lifecycles

Provision: HashiCorp Terraform and GitHub

Terraform and GitHub's integrations form the foundation for aligning the goals of operations and developers to create a DevOps workflow that moves at the speed organizations desire. As we mentioned earlier, GitHub increases developer velocity by enabling them to automatically integrate changes to code from anywhere and then deploy those changes in any environment on demand. This presents a challenge for operations teams to have the infrastructure ready at the time that it's needed. One possible solution is to leave instances active at all times. While this solves the challenge of having environments available, it introduces new concerns like cost and resource pressures. The better solution is to make the infrastructure a part of the versioning process and be able to add or remove resources on-demand. By tying application development to underlying infrastructure, GitHub and Terraform solve the provisioning challenge to keep developer velocity up and cloud costs down.





Both Terraform Cloud and Enterprise support GitHub as a first class software development platform. This means that as changes are committed to a repository that affect the attached Terraform organization, Terraform will initiate a run and make the necessary changes to accommodate the change. Operators can also use this integration to version their various environments. If more VMs are required for a specific event or test, these changes can be made to the Terraform configurations stored in GitHub. After the event has completed or if an issue arises from the deployment, operators can roll back the infrastructure to a previous, stable version.

Terraform can also be used for maintaining GitHub itself. The Terraform GitHub provider enables operators to create and manage repositories, teams, and organizations. For example, if a new development team is being onboarded and needs access to certain environments, Terraform can be used to create the new organization, assign new team members, and enable access to their infrastructure environment. This ensures there is a consistent workflow for operators to enable developers at the pace they are looking for.

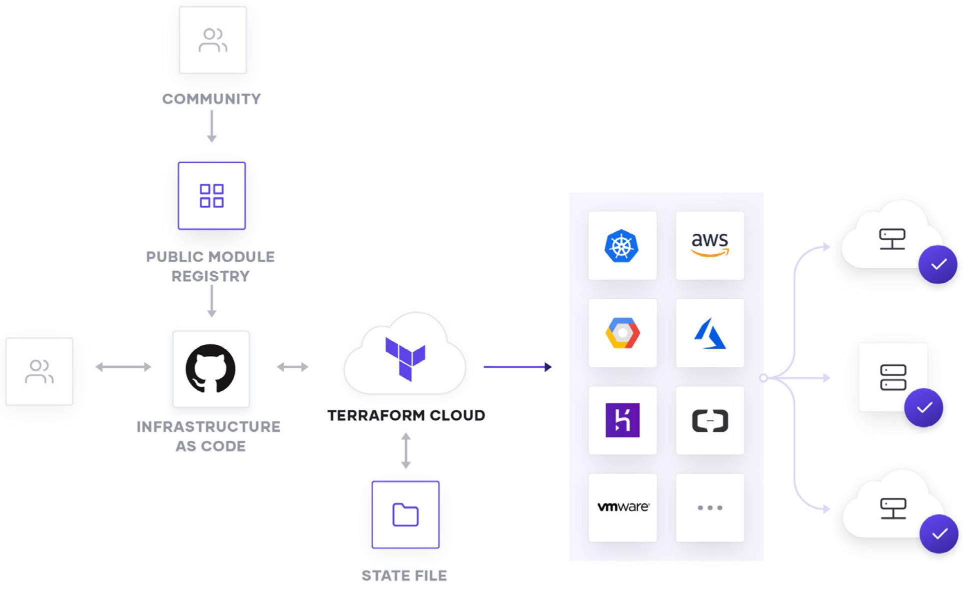


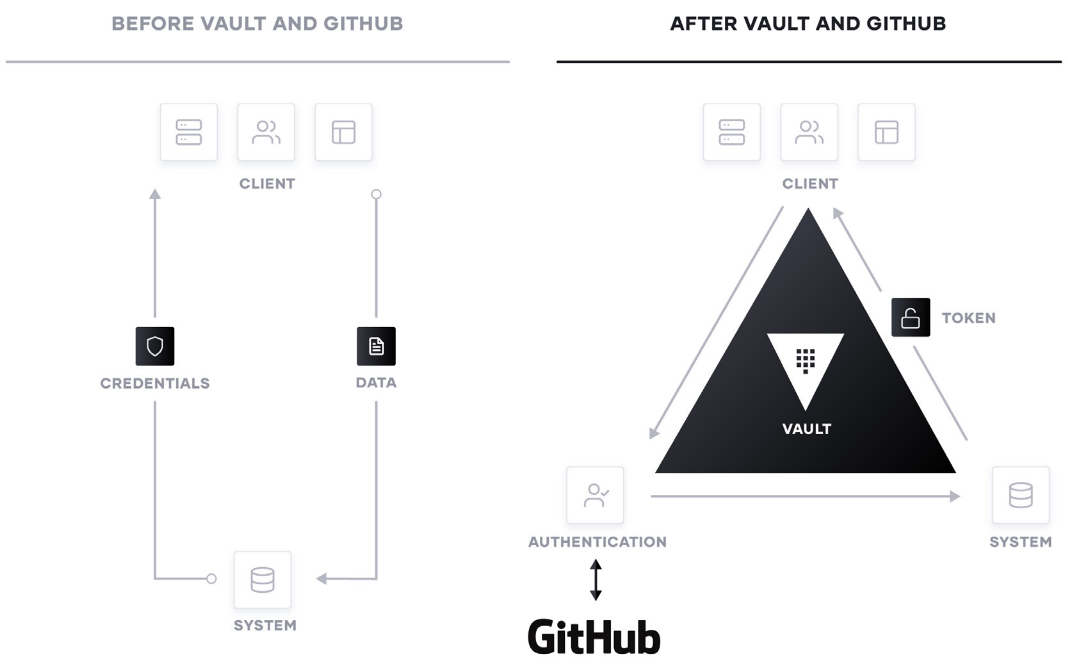
Figure 2: How it works

Ensuring compliance and governance with policy as code.

Utilizing Terraform Cloud or Terraform Enterprise and GitHub together also enables organizational governance for managing infrastructure. Sentinel, HashiCorp's policy as code framework, can be used to ensure that any infrastructure that is created follows organizational policy. As an example, an organization might require that any new cloud instance has tags indicating whether it is a development or a test instance for compliance purposes. Using Sentinel, an administrator could set a hard fail policy that prevents the instance from deploying if a user commits a change without the proper tags. This could be extended to ensuring that the instance type, region or even cloud provider are consistent across all teams.

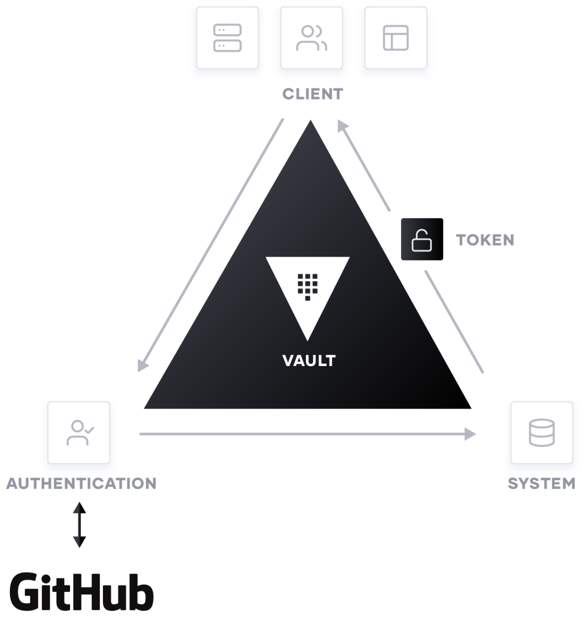
Secure: HashiCorp Vault and GitHub

Once an organization has established the proper provisioning workflow that matches the speed organizations desire, the next major consideration is how is the organization controlling access to these systems and protecting credentials. Using static credentials creates vulnerabilities. It's very difficult to keep track of how old a password is, where it is being used, and who has access to it. This becomes increasingly problematic in the cloud landscape because traditional security methods become harder to enforce, and there are many more credentials across distributed services. Infrastructure no longer has a defined perimeter and protecting each individual service inhibits the scalability of that service. If organizations treat these environments as a "zero-trust" network and instead use identity-based security for authentication coupled with well-defined policies, this challenge of system access lessens. Through an integration with Vault, GitHub can be used as that source of identity. GitHub's native token scanning capabilities helps organizations keep track of credentials that may have been exposed in a public repo and eliminate them.



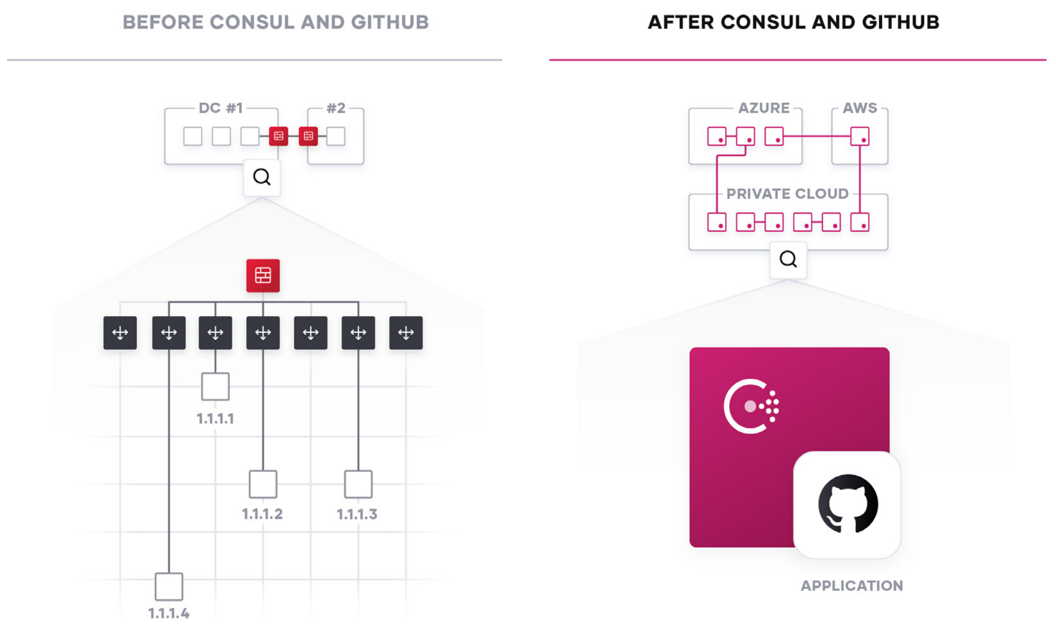
The GitHub auth method enables developers to use their GitHub ID to authenticate access to Vault and generate tokens for accessing systems. What systems these developers are able to access is based on policies and defined by the Vault administrator. The policies are uploaded directly to Vault and can apply to either individual users or entire paths. For example, imagine a developer is writing a service that requires access to a specific database. They need the application to be able to access the information, but do not need to have the ability to alter the information that is stored in this database. Rather than providing the users with a username and password, the security team could instead enable access via their GitHub ID. Now when that developer requires access to the database, they make the request to Vault directly. Vault then ensures it is a valid request and returns a token for accessing the database. These tokens will have a preset Time to Live (TTL) and will be revoked once that time period has elapsed. Using this workflow, the developer is able to access the database information they require on demand without creating a potential vulnerability in the process.

This workflow is the foundation for how GitHub and HashiCorp are enabling developers to move at the speed they require while providing peace of mind to the security team. Looking forward, there are more integration points between Vault and GitHub which will further enable organizations to protect their multi-cloud environments. In the future, GitHub and HashiCorp will look towards implementing easier ways to integrate a GitHub organization with Vault. Some of the possible solutions include the ability to attach a new or existing Vault server to an organization upon creation. This would provide more control to what credentials are being created or used by developers and restricting system access.



Connect: HashiCorp Consul and GitHub

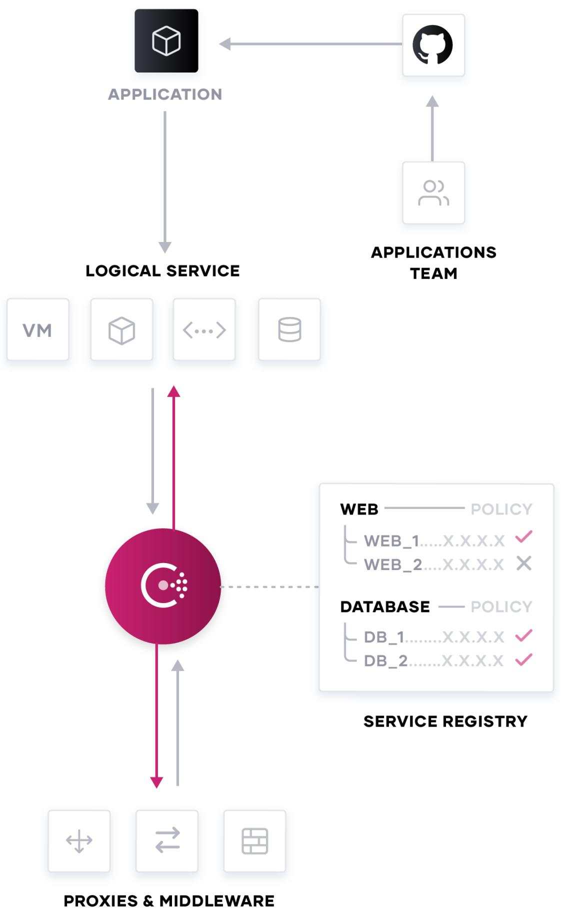
Having defined solutions for Infrastructure provisioning and identity-based security, the last piece to focus on is in the service-based networking layer. A difficult challenge for organizations to overcome when making the leap to cloud is managing the connections between services. A modern application can easily consist of dozens, perhaps hundreds of individual services, each working at scale with a need to authenticate and connect with each other. The previous system for managing traffic between these services was to assign each a specific IP and manually make the connections. This method is untenable in a world of continuous delivery where IP addresses change as often as code is released. Developers need a single, centralized registry for keeping track of all the existing services by their identity and as the organization shifts more towards a microservice environment, automate connections between new and existing services.



HashiCorp Consul is a service-based networking tool capable of keeping a centralized registry for any new or existing services in any environment. Developers can add a Consul configuration as part of their deployments without having to alter their existing GitHub workflow. When the changes are pushed live, a Consul agent will be deployed as well. If configured properly, the Consul agent will do two things. First, it will seek to join an existing Consul cluster. Once it has done that, the Consul agent will then register any of the services within its environment and then communicate its existence to the rest of the cluster. Now any of the other agents, and by extension their services, are capable of discovering that new service.

This shared registry enables a crucial aspect of modern application delivery: Service Discovery. In order for developers to be able to move quickly in cloud environments, they need to be able to identify the necessary services for their applications on-demand.

Filing a ticket with a networking team and waiting for updated firewall rules slows down deployments. By deploying Consul with each application, developers can trust Consul to both identify and automate the process of connecting those services. Similarly services can have rules to allow or deny these connections, to maintain integrity of the portfolio design.



With these agents up and running, Consul is also capable of providing telemetry data (health checks, network data, and updates on the cluster, etc.) to Application Performance Monitoring (APM) solutions like Datadog, AppDynamics, and SignalFX. This data enables developers to get greater granularity into application performance and overall network health.

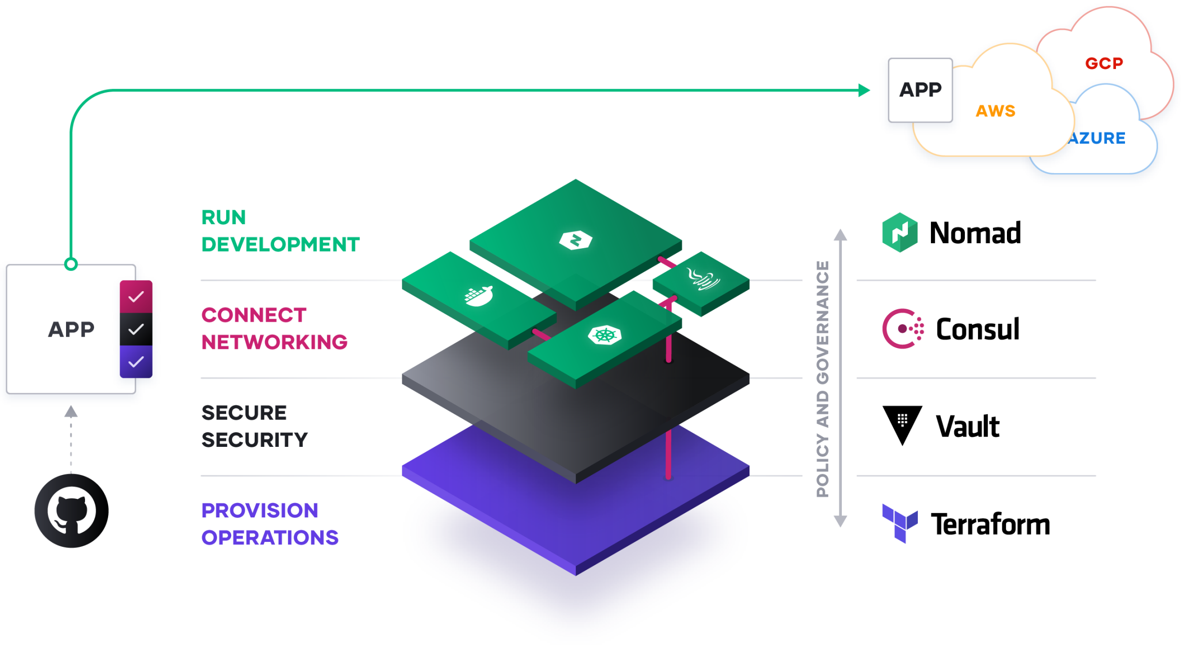
Beyond just discovering new services, automating the connecting process for microservices using a service mesh is quickly becoming a goal for organizations. As is the case above, the ideal workflow for a developer would be to push the changes to their code, deploy the application, and have it automatically connect to the services that it needs access to. Consul Connect's service mesh capabilities can do exactly that. Developers can set the idea of intentions and enable any instance of a specific service to talk to another type of service and encrypt that communication using mTLS. This is an emerging space for developers and is further aligning the goals of the application teams with the networking teams.

Conclusion

Operating in multi-cloud operating model offers enterprises a broader reach and scale than traditional datacenters. The potential to reduce standard lead time for delivering applications and capabilities can go from weeks to a matter of minutes with the right automation across every layer: from the inner loop of development, to the outer loop of infrastructure provisioning, identity-based security, and service-based networking.

Developers and operators need to collaborate to enable a centralized workflow for both adopting the cloud and increasing delivery speed.

GitHub and HashiCorp have created a common blueprint for consistent workflows for CI/CD in a multi-cloud environment that are adaptable to the needs of any enterprise, as well as being integrated with other core technology that enterprise development teams rely upon.



HashiCorp