What is Terraform?

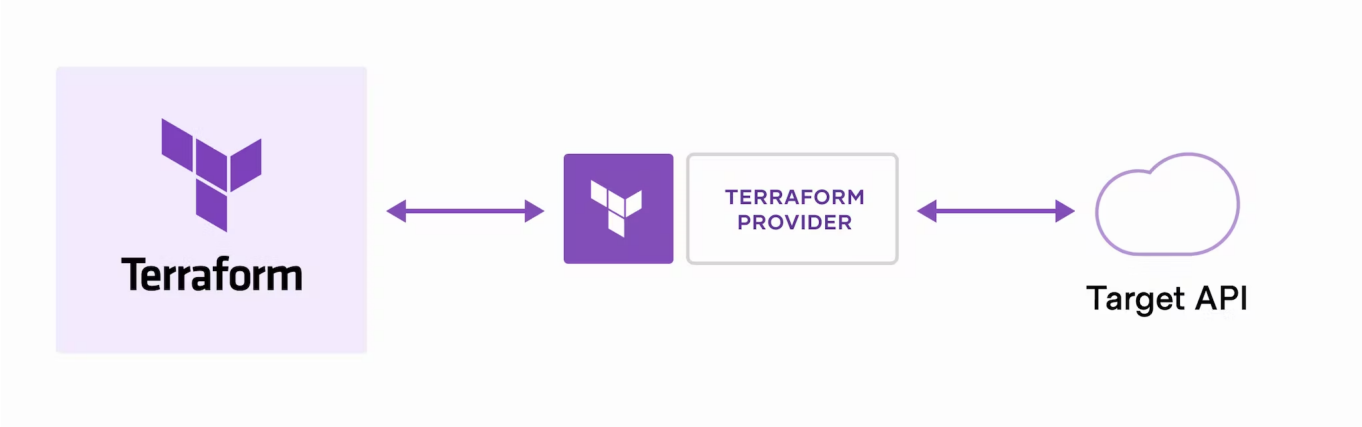
Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

> **Hands On:** Try the Get Started tutorials to start managing infrastructure on popular cloud providers: Amazon Web Services, Azure, Google Cloud Platform, Oracle Cloud Infrastructure, and Docker.

How does Terraform work?

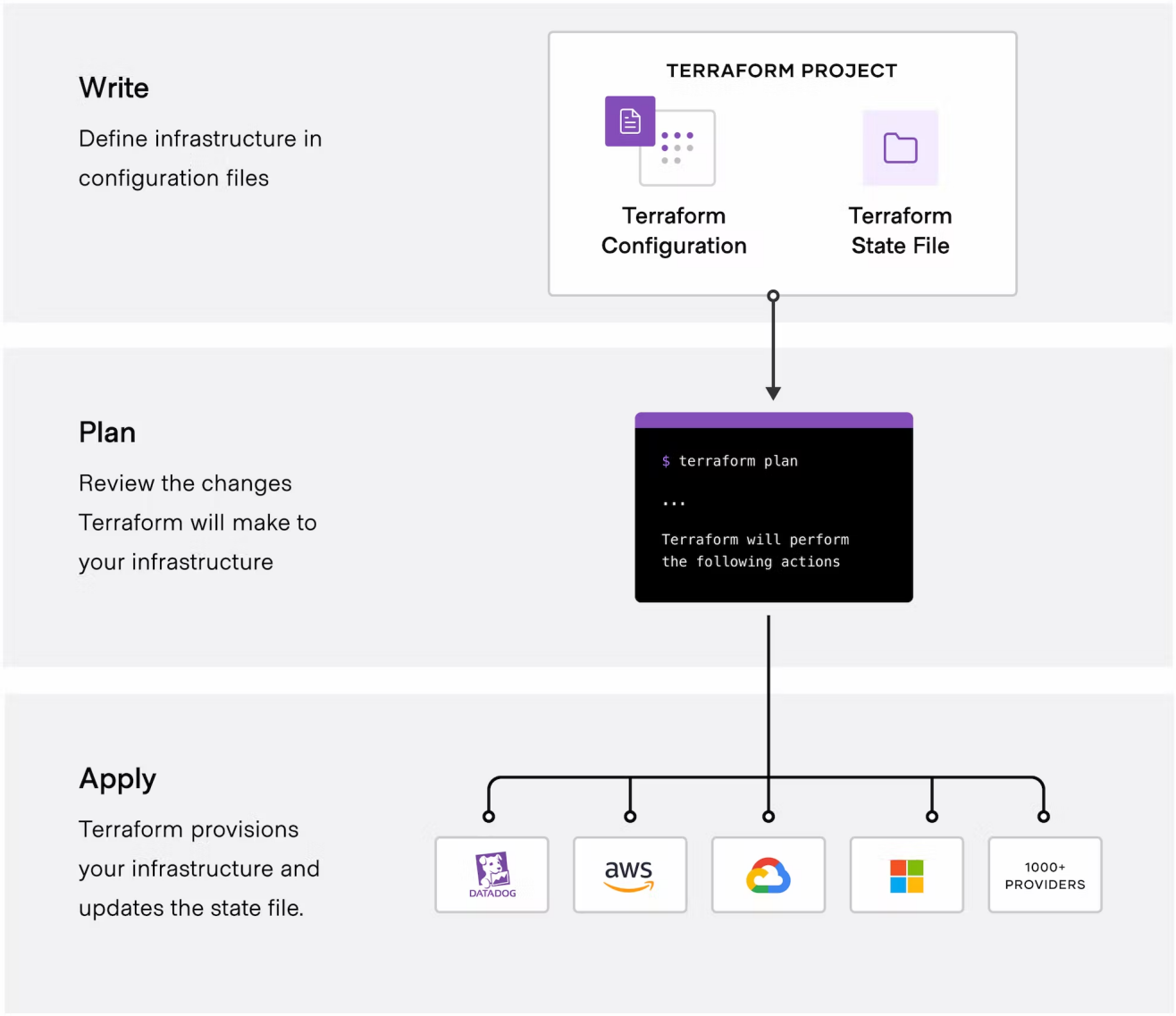
Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.



HashiCorp and the Terraform community have already written **thousands of providers** to manage many different types of resources and services. You can find all publicly available providers on the Terraform Registry, including Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog, and many more.

The core Terraform workflow consists of three stages:

- **Write:** You define resources, which may be across multiple cloud providers and services. For example, you might create a configuration to deploy an application on virtual machines in a Virtual Private Cloud (VPC) network with security groups and a load balancer.
- **Plan:** Terraform creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and your configuration.
- **Apply:** On approval, Terraform performs the proposed operations in the correct order, respecting any resource dependencies. For example, if you update the properties of a VPC and change the number of virtual machines in that VPC, Terraform will recreate the VPC before scaling the virtual machines.



Why Terraform?

HashiCorp co-founder and CTO Armon Dadgar explains how Terraform solves infrastructure challenges.

Manage any infrastructure

Find providers for many of the platforms and services you already use in the Terraform Registry. You can also write your own. Terraform takes an immutable approach to infrastructure, reducing the complexity of upgrading or modifying your services and infrastructure.

Track your infrastructure

Terraform generates a plan and prompts you for your approval before modifying your infrastructure. It also keeps track of your real infrastructure in a state file, which acts as a source of truth for your environment. Terraform uses the state file to determine the changes to make to your infrastructure so that it will match your configuration.

Automate changes

Terraform configuration files are declarative, meaning that they describe the end state of your infrastructure. You do not need to write step-by-step instructions to create resources because Terraform handles the underlying logic. Terraform builds a resource graph to determine resource dependencies and creates or modifies non-dependent resources in parallel. This allows Terraform to provision resources efficiently.

Standardize configurations

Terraform supports reusable configuration components called modules that define configurable collections of infrastructure, saving time and encouraging best practices. You can use publicly available modules from the Terraform Registry, or write your own.

Collaborate

Since your configuration is written in a file, you can commit it to a Version Control System (VCS) and use Terraform Cloud to efficiently manage Terraform workflows across teams. Terraform Cloud runs Terraform in a consistent, reliable environment and provides secure access to shared state and secret data, role-based access controls, a private registry for sharing both modules and providers, and more.

Tip: Learn more about Terraform use cases and how Terraform compares to alternatives.

Community

We welcome questions, suggestions, and contributions from the community.

- Ask questions in HashiCorp Discuss.
- Read our contributing guide.
- Submit an issue for bugs and feature requests.

Edit this page on GitHub