# Creating a Git Repo with Terraform In Mind – Dustin DortchUnderstanding Terraform

Terraform is an open-source Infrastructure as a code (IAC) provisioning tool that allows to create, manage & deploy the production-ready environment.

Terraform codifies cloud APIs into declarative configuration files. Terraform infrastructure allow you to manage both existing service providers and custom in-house solutions.

Terraform(IaC) allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version, reuse, and share.

Using Terraform has several advantages over manually managing your infrastructure:

* Terraform can manage infrastructure on multiple cloud platforms.
* The human-readable configuration language helps you write infrastructure code quickly.
* Terraform's state allows you to track resource changes throughout your deployments.
* You can commit your configurations to version control to safely collaborate on infrastructure.

## Terraform Manage any infrastructure

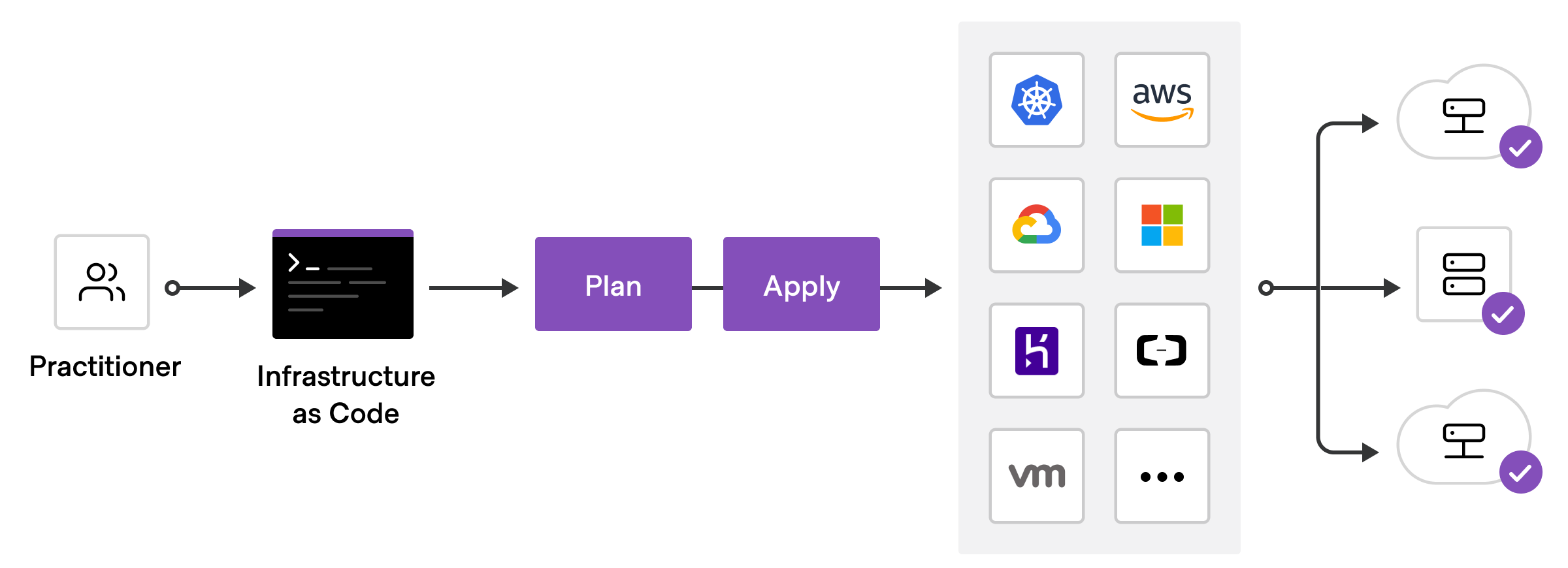
Terraform plugins called providers let Terraform interact with cloud platforms and other services via their application programming interfaces (APIs).

HashiCorp and the Terraform community have written over 1,000 providers to manage resources on Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, and DataDog, just to name a few.

## Standardize your deployment workflow

Providers define individual units of infrastructure, for example compute instances or private networks, as resources. You can compose resources from different providers into reusable Terraform configurations called modules, and manage them with a consistent language and workflow.

Terraform's configuration language is declarative, meaning that it describes the desired end-state for your infrastructure, in contrast to procedural programming languages that require step-by-step instructions to perform tasks. Terraform providers automatically calculate dependencies between resources to create or destroy them in the correct order.



To deploy infrastructure with Terraform:

* **Scope** - Identify the infrastructure for your project.
* **Author** - Write the configuration for your infrastructure.
* **Initialize** - Install the plugins Terraform needs to manage the infrastructure.
* **Plan** - Preview the changes Terraform will make to match your configuration.
* **Apply** - Make the planned changes.

## CONCLUDE

## Collaborate

Terraform allows you to collaborate on your infrastructure with its remote state backends. When you use Terraform Cloud (free for up to five users), you can securely share your state with your teammates, provide a stable environment for Terraform to run in, and prevent race conditions when multiple people make configuration changes at once.

You can also connect Terraform Cloud to version control systems (VCSs) like GitHub, GitLab, and others, allowing it to automatically propose infrastructure changes when you commit configuration changes to VCS. This lets you manage changes to your infrastructure through version control, as you would with application code.

Thank you - HashiCorp and Contributors