using `tfenv`), Terragrunt (including using `tgswitch`), and Terraform providers (including how to use the lock file).

Improved maturity

Terraform has been downloaded over 100 million times, has had over 1,500 open source contributors, and is in use at ~79% of Fortune 500 companies,[1] so it's safe to say that the ecosystem has grown and matured significantly over the last several years. There are now more developers, providers, reusable modules, tools, plugins, classes, books, and tutorials for Terraform than ever before. Moreover, HashiCorp, the company that created Terraform, had its IPO (initial public offering) in 2021, so Terraform is no longer backed by a small startup but by a large, stable, publicly traded company, for which Terraform is its biggest business line.

Many other changes

There were many other changes along the way, including the launch of Terraform Cloud (a web UI for using Terraform); the improved maturity of popular community tools such as Terragrunt, Terratest, and tfenv; the addition of many new provider features (including new ways to do zero-downtime deployment, such as instance refresh, which I've added to Chapter 5) and new functions (e.g., I added examples of how to use the `one` function in Chapter 5 and the `try` function in Chapter 7); the deprecation of many old features (e.g., `template_file` data source, many `aws_s3_bucket` parameters, `list` and `map`, support for external references on `destroy` provisioners); and much more.

Changes from the First Edition to the Second Edition

Going back in time even further, the second edition of the book added roughly 150 pages of new content on top of the first edition. Here is a

summary of those changes, which also covers how Terraform changed between 2017 and 2019:

Four major Terraform releases

Terraform was at version 0.8 when the first edition came out; between then and the time of the second edition, Terraform had four major releases, all the way up to version 0.12. These releases introduced some amazing new functionality, as I'll describe shortly, as well as a fair amount of upgrade work for users![2]

Automated testing improvements

The tooling and practices for writing automated tests for Terraform code evolved considerably between 2017 and 2019. In the second edition, I added Chapter 9, a completely new chapter dedicated to testing, covering topics such as unit tests, integration tests, end-to-end tests, dependency injection, test parallelism, static analysis, and more.

Module improvements

The tooling and practices for creating Terraform modules also evolved considerably. In the second edition, I added Chapter 8, a new chapter that contains a guide to building reusable, battle-tested, production-grade Terraform modules—the kind of modules you'd bet your company on.

Workflow improvements

Chapter 10 was completely rewritten in the second edition to reflect the changes in how teams integrate Terraform into their workflows, including a detailed guide on how to take application code and infrastructure code from development through testing and all the way to production.

HCL2

Terraform 0.12 overhauled the underlying language from HCL to HCL2. This included support for first-class expressions, rich type constraints, lazily evaluated conditional expressions, support for `null`, `for_each` and `for` expressions, dynamic inline blocks, and more. All the code examples in the second edition of the book were updated to use HCL2, and the new language features were covered extensively in Chapters 5 and 8.

Terraform state revamp

Terraform 0.9 introduced backends as a first-class way to store and share Terraform state, including built-in support for locking. Terraform 0.9 also introduced state environments as a way to manage deployments across multiple environments. In Terraform 0.10, state environments were replaced with Terraform workspaces. I cover all of these topics in Chapter 3.

Terraform providers split

In Terraform 0.10, the core Terraform code was split up from the code for all the providers (i.e., the code for AWS, GCP, Azure, etc.). This allowed providers to be developed in their own repositories, at their own cadence, with their own versioning. However, you now must run `terraform init` to download the provider code every time you start working with a new module, as discussed in Chapters 2 and 9.

Massive provider growth

From 2016 to 2019, Terraform grew from a handful of major cloud providers (the usual suspects, such as AWS, GCP, and Azure) to more than one hundred official providers and many more community providers.[3] This means that you can now use Terraform to not only manage many other types of clouds (e.g., there are now providers for Alicloud, Oracle Cloud Infrastructure, VMware vSphere, and others) but also to manage many other aspects of your world as code, including version control systems with the GitHub, GitLab, and Bitbucket

providers; data stores with the MySQL, PostgreSQL, and InfluxDB providers; monitoring and alerting systems with the Datadog, New Relic, and Grafana providers; platform tools with the Kubernetes, Helm, Heroku, Rundeck, and RightScale providers; and much more. Moreover, each provider has much better coverage these days: AWS now covers the majority of important AWS services and often adds support for new services even before CloudFormation does!

Terraform Registry

HashiCorp launched the Terraform Registry in 2017, a UI that made it easy to browse and consume open source, reusable Terraform modules contributed by the community. In 2018, HashiCorp added the ability to run a Private Terraform Registry within your own organization. Terraform 0.11 added first-class syntax support for consuming modules from a Terraform Registry. We look at the Registry in Chapter 8.

Better error handling

Terraform 0.9 updated state error handling: if there was an error writing state to a remote backend, the state would be saved locally in an *errored.tfstate* file. Terraform 0.12 completely overhauled error handling, by catching errors earlier, showing clearer error messages, and including the filepath, line number, and a code snippet in the error message.

Many other changes

There were many other changes along the way, including the introduction of local values (see "Module Locals"), new "escape hatches" for having Terraform interact with the outside world via scripts (see "Beyond Terraform Modules"), running `plan` as part of the `apply` command, fixes for the `create_before_destroy` cycle issues, major improvements to the `count` parameter so that it can include references to data sources and resources, dozens of new built-in functions, an overhaul in `provider` inheritance, and much more.