TRANSPARENT PORTABILITY BETWEEN CLOUD PROVIDERS

Because Terraform supports many different cloud providers, a common question that arises is whether it supports *transparent portability* between them. For example, if you used Terraform to define a bunch of servers, databases, load balancers, and other infrastructure in AWS, could you instruct Terraform to deploy exactly the same infrastructure in another cloud provider, such as Azure or Google Cloud, in just a few commands?

This question turns out to be a bit of a red herring. The reality is that you can't deploy "exactly the same infrastructure" in a different cloud provider because the cloud providers don't offer the same types of infrastructure! The servers, load balancers, and databases offered by AWS are very different from those in Azure and Google Cloud in terms of features, configuration, management, security, scalability, availability, observability, and so on. There is no easy way to "transparently" paper over these differences, especially as functionality in one cloud provider often doesn't exist at all in the others. Terraform's approach is to allow you to write code that is specific to each provider, taking advantage of that provider's unique functionality, but to use the same language, toolset, and IaC practices under the hood for all providers.

How Does Terraform Compare to Other IaC Tools?

Infrastructure as code is wonderful, but the process of picking an IaC tool is not. Many of the IaC tools overlap in what they do. Many of them are open source. Many of them offer commercial support. Unless you've used each one yourself, it's not clear what criteria you should use to pick one or the other.

What makes this even more difficult is that most of the comparisons you find between these tools do little more than list the general properties of each one and make it sound as if you could be equally successful with any of them. And although that's technically true, it's not helpful. It's a bit like telling a programming newbie that you could be equally successful building a website with PHP, C, or assembly—a statement that's technically true but one that omits a huge amount of information that is essential for making a good decision.