VALIDATING INFRASTRUCTURE

The examples in this chapter use `curl` and HTTP requests to validate that the infrastructure is working, because the infrastructure you're testing includes a load balancer that responds to HTTP requests. For other types of infrastructure, you'll need to replace `curl` and HTTP requests with a different form of validation. For example, if your infrastructure code deploys a MySQL database, you'll need to use a MySQL client to validate it; if your infrastructure code deploys a VPN server, you'll need to use a VPN client to validate it; if your infrastructure code deploys a server that isn't listening for requests at all, you might need to SSH to the server and execute some commands locally to test it; and so on. So although you can use the same basic test structure described in this chapter with any type of infrastructure, the validation steps will change depending on what you're testing.

In short, when working with Terraform, every developer needs good example code to test and a real deployment environment (e.g., an AWS account) to use as an equivalent to localhost for running those tests. In the process of manual testing, you're likely to bring up and tear down a lot of infrastructure, and likely make lots of mistakes along the way, so this environment should be completely isolated from your other, more stable environments, such as staging, and especially production.

Therefore, I strongly recommend that every team sets up an isolated *sandbox environment*, in which developers can bring up and tear down any infrastructure they want without worrying about affecting others. In fact, to reduce the chances of conflicts between multiple developers (e.g., two developers trying to create a load balancer with the same name), the gold standard is that each developer gets their own completely isolated sandbox environment. For example, if you're using Terraform with AWS, the gold standard is for each developer to have their own AWS account that they can use to test anything they want.[1]

Cleaning Up After Tests

Having many sandbox environments is essential for developer productivity, but if you're not careful, you can end up with infrastructure running all over