Chapter 9. How to Test Terraform Code

The DevOps world is full of fear: fear of downtime, fear of data loss, fear of security breaches. Every time you go to make a change, you're always wondering, what will this affect? Will it work the same way in every environment? Will this cause another outage? And if there is an outage, how late into the night will you need to stay up to fix it this time? As companies grow, there is more and more at stake, which makes the deployment process even scarier, and even more error prone. Many companies try to mitigate this risk by doing deployments less frequently, but the result is that each deployment is larger and actually more prone to breakage.

If you manage your infrastructure as code, you have a better way to mitigate risk: tests. The goal of testing is to give you the confidence to make changes. The key word here is *confidence*: no form of testing can guarantee that your code is free of bugs, so it's more of a game of probability. If you can capture all of your infrastructure and deployment processes as code, you can test that code in a pre-production environment, and if it works there, there's a high probability that when you use the exact same code in production, it will work there, too. And in a world of fear and uncertainty, high probability and high confidence go a long way.

In this chapter, I'll go over the process of testing infrastructure code, including both manual testing and automated testing, with the bulk of the chapter spent on the latter:

- Manual tests
    - Manual testing basics
    - Cleaning up after tests