The only difference from application code is that infrastructure code tests typically take longer, so you'll want to put more thought into how you can shorten the test cycle so that you can get feedback on your changes as quickly as possible. In "Test stages", you saw that you can use these test stages to rerun only specific stages of a test suite, dramatically shortening the feedback loop.

As you make changes, be sure to regularly commit your work:

```
$ git commit -m "Updated Hello, World text"
```

Submit Changes for Review

After your code is working the way you expect, you can create a pull request to get your code reviewed, just as you would with application code. Your team will review your code changes, looking for bugs as well as enforcing *coding guidelines*. Whenever you're writing code as a team, regardless of what type of code you're writing, you should define guidelines for everyone to follow. One of my favorite definitions of "clean code" comes from an interview I did with Nick Dellamaggiore for my earlier book, *Hello, Startup*:

> *If I look at a single file and it's written by 10 different engineers, it should be almost indistinguishable which part was written by which person. To me, that is clean code.*
>
> *The way you do that is through code reviews and publishing your style guide, your patterns, and your language idioms. Once you learn them, everybody is way more productive because you all know how to write code the same way. At that point, it's more about what you're writing and not how you write it.*
>
> —Nick Dellamaggiore, Infrastructure Lead at Coursera

The Terraform coding guidelines that make sense for each team will be different, so here, I'll list a few of the common ones that are useful for most teams: