Every team should enforce a set of conventions about code style, including the use of whitespace, newlines, indentation, curly braces, variable naming, and so on. Although programmers love to debate spaces versus tabs and where the curly brace should go, the more important thing is that you are consistent throughout your codebase.

Terraform has a built-in `fmt` command that can reformat code to a consistent style automatically:

```
$ terraform fmt
```

I recommend running this command as part of a commit hook to ensure that all code committed to version control uses a consistent style.

Run Automated Tests

Just as with application code, your infrastructure code should have commit hooks that kick off automated tests in a CI server after every commit and show the results of those tests in the pull request. You already saw how to write unit tests, integration tests, and end-to-end tests for your Terraform code in Chapter 9. There's one other critical type of test you should run: `terraform plan`. The rule here is simple:

Always run `plan` before `apply`.

Terraform shows the `plan` output automatically when you run `apply`, so what this rule really means is that you should always pause and read the `plan` output! You'd be amazed at the type of errors you can catch by taking 30 seconds to scan the "diff" you get as an output. A great way to encourage this behavior is by integrating `plan` into your code review flow. For example, Atlantis is an open source tool that automatically runs `terraform plan` on commits and adds the `plan` output to pull requests as a comment, as shown in Figure 10-3.

atlantisbot commented

Ran Plan in dir: `.` workspace: `default`

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

+ null_resource.demo
      id: <computed>
Plan: 1 to add, 0 to change, 0 to destroy.
```

- ▶️ To **apply** this plan, comment:
  - `atlantis apply -d .`
- 🚮 To **delete** this plan click here
- 🔁 To **plan** this project again, comment:
  - `atlantis plan -d .`

- ⏩ To **apply** all unapplied plans, comment:
  - `atlantis apply`