- Documentation
- Automated tests
- File layout
- Style guide

Documentation

In some sense, Terraform code is, in and of itself, a form of documentation. It describes in a simple language exactly what infrastructure you deployed and how that infrastructure is configured. However, there is no such thing as self-documenting code. Although well-written code can tell you *what* it does, no programming language that I'm aware of (including Terraform) can tell you *why* it does it.

This is why all software, including IaC, needs documentation beyond the code itself. There are several types of documentation that you can consider and have your team members require as part of code reviews:

Written documentation

> Most Terraform modules should have a README that explains what the module does, why it exists, how to use it, and how to modify it. In fact, you may want to write the README first, before any of the actual Terraform code, because that will force you to consider *what* you're building and *why* you're building it before you dive into the code and get lost in the details of *how* to build it.[4] Spending 20 minutes writing a README can often save you hours of writing code that solves the wrong problem. Beyond the basic README, you might also want to have tutorials, API documentation, wiki pages, and design documents that go deeper into how the code works and why it was built this way.

Code documentation

> Within the code itself, you can use comments as a form of documentation. Terraform treats any text that begins with a hash (#) as a comment. Don't use comments to explain what the code does; the code