Figure 10-6. Promoting an immutable, versioned artifact of Terraform code from environment to environment.

Conclusion

If you've made it to this point in the book, you now know just about everything you need to use Terraform in the real world, including how to write Terraform code; how to manage Terraform state; how to create reusable modules with Terraform; how to do loops, if-statements, and deployments; how to manage secrets; how to work with multiple regions, accounts, and clouds; how to write production-grade Terraform code; how to test your Terraform code; and how to use Terraform as a team. You've worked through examples of deploying and managing servers, clusters of servers, load balancers, databases, scheduled actions, CloudWatch alarms, IAM users, reusable modules, zero-downtime deployment, AWS Secrets Manager, Kubernetes clusters, automated tests, and more. Phew! Just don't forget to run `terraform destroy` in each module when you're all done.

The power of Terraform, and more generally, IaC, is that you can manage all the operational concerns around an application using the same coding principles as the application itself. This allows you to apply the full power of software engineering to your infrastructure, including modules, code reviews, version control, and automated testing.

If you use Terraform correctly, your team will be able to deploy faster and respond to changes more quickly. Hopefully, deployments will become routine and boring—and in the world of operations, boring is a very good thing. And if you really do your job right, rather than spending all your time managing infrastructure by hand, your team will be able to spend more and more time improving that infrastructure, allowing you to go even faster.

This is the end of the book but just the beginning of your journey with Terraform. To learn more about Terraform, IaC, and DevOps, head over to Appendix A for a list of recommended reading. And if you've got feedback

or questions, I'd love to hear from you at jim@ybrikman.com. Thank you for reading!

---

1 The Standish Group, "CHAOS Manifesto 2013: Think Big, Act Small," 2013, *https://oreil.ly/ydaWQ*.

2 Dan Milstein, "How to Survive a Ground-Up Rewrite Without Losing Your Sanity," OnStartups.com, April 8, 2013, *https://oreil.ly/nOGrU*.

3 See the Gruntwork Infrastructure as Code Library.

4 Writing the README first is called Readme-Driven Development.

5 See 10 real-world stories of how we've compromised CI/CD pipelines for some eye-opening examples.

6 Check out Gruntwork Pipelines for a real-world example of this worker pattern.

7 Credit for how to promote Terraform code across environments goes to Kief Morris: Using Pipelines to Manage Environments with Infrastructure as Code.