command: `terraform apply`. In fact, you might want to commit your latest changes to Git so that you can keep track of the history of your infrastructure.

Conclusion

You now have a basic grasp of how to use Terraform. The declarative language makes it easy to describe exactly the infrastructure you want to create. The `plan` command allows you to verify your changes and catch bugs before deploying them. Variables, references, and dependencies allow you to remove duplication from your code and make it highly configurable.

However, you've only scratched the surface. In Chapter 3, you'll learn how Terraform keeps track of what infrastructure it has already created, and the profound impact that has on how you should structure your Terraform code. In Chapter 4, you'll see how to create reusable infrastructure with Terraform modules.

---

1 Source: "Global Cloud Services Spend Hits Record US$49.4 Billion in Q3 2021", Canalys, October 28, 2021.

2 If you find the AWS terminology confusing, be sure to check out Amazon Web Services in Plain English.

3 Check out the AWS Free Tier documentation for details.

4 For more details on AWS user management best practices, see the documentation.

5 You can learn more about IAM Policies on the AWS website.

6 I'm assuming that you're running the examples in this book in an AWS account dedicated solely to learning and testing so that the broad permissions of the `AdministratorAccess` Managed Policy are not a big risk. If you are running these examples in a more sensitive environment—which, for the record, I don't recommend!—and you're comfortable with creating custom IAM Policies, you can find a more pared-down set of permissions in this book's code examples repo.

7 You can also write Terraform code in pure JSON in files with the extension *.tf.json*. You can learn more about Terraform's HCL and JSON syntax in the Terraform documentation.

8 You can learn more about AWS regions and Availability Zones on the AWS website.

9 Finding AMI IDs is surprisingly complicated, as documented in this Gruntwork blog post.

10 You can find a handy list of HTTP server one-liners on GitHub.

11 To learn more about how CIDR works, see its Wikipedia page. For a handy calculator that converts between IP address ranges and CIDR notation, use either *https://cidr.xyz/* in your browser or install the `ipcalc` command in your terminal.

12 Note that while the `graph` command can be useful for visualizing the relationships between a small number of resources, with dozens or hundreds of resources, the graphs tend to become too large and messy to be useful.

13 From *The Pragmatic Programmer* by Andy Hunt and Dave Thomas (Addison-Wesley Professional).

14 For a deeper look at how to build highly available and scalable systems on AWS, see "A Comprehensive Guide to Building a Scalable Web App on Amazon Web Services - Part 1" by Josh Padnick.

15 These days, you should actually be using a *launch template* (and the `aws_launch_template` resource) with ASGs rather than a launch configuration. However, I've stuck with the launch configuration in the examples in this book as it is convenient for teaching some of the concepts in the zero-downtime deployment section of Chapter 5.

16 To keep these examples simple, the EC2 Instances and ALB are running in the same subnets. In production usage, you'd most likely run them in different subnets, with the EC2 Instances in private subnets (so they aren't directly accessible from the public internet) and the ALBs in public subnets (so users can access them directly).