```
  assume_role {
    role_arn =
"arn:aws:iam::222222222222:role/OrganizationAccountAccessRole"
  }
}
```

And now you can pass them into the *modules/multi-account* module as follows:

```
module "multi_account_example" {
  source = "../../modules/multi-account"

  providers = {
    aws.parent = aws.parent
    aws.child  = aws.child
  }
}
```

The keys in the `providers` map must match the names of the configuration aliases within the module; if any of the names from configuration aliases are missing in the `providers` map, Terraform will show an error. This way, when you're building a reusable module, you can define what providers that module needs, and Terraform will ensure users pass those providers in; and when you're building a root module, you can define your `provider` blocks just once and pass around references to them to the reusable modules you depend on.

Working with Multiple Different Providers

You've now seen how to work with multiple providers when all of them are the same type of provider: e.g., multiple copies of the `aws` provider. This section talks about how to work with multiple different providers.

Readers of the first two editions of this book often asked for examples of using multiple clouds together (*multicloud*), but I couldn't find much useful to share. In part, this is because using multiple clouds is usually a bad practice,[2] but even if you're forced to do it (most large companies are

multicloud, whether they want to be or not), it's rare to manage multiple clouds in a single module for the same reason it's rare to manage multiple regions or accounts in a single module. If you're using multiple clouds, you're far better off managing each one in a separate module.

Moreover, translating every single AWS example in the book into the equivalent solutions for other clouds (Azure and Google Cloud) is impractical: the book would end up way too long, and while you would learn more about each cloud, you wouldn't learn any new Terraform concepts along the way, which is the real goal of the book. If you do want to see examples of what the Terraform code for similar infrastructure looks like across different clouds, have a look at the *examples* folder in the Terratest repo. As you'll see in Chapter 9, Terratest provides a set of tools for writing automated tests for different types of infrastructure code and different types of clouds, so in the *examples* folder you'll find Terraform code for similar infrastructure in AWS, Google Cloud, and Azure, including individual servers, groups of servers, databases, and more. You'll also find automated tests for all those examples in the *test* folder.

In this book, instead of an unrealistic multicloud example, I decided to instead show you how to use multiple providers together in a slightly more realistic scenario (one that was also requested by many readers of the first two editions): namely, how to use the AWS Provider with the Kubernetes provider to deploy Dockerized apps. Kubernetes is, in many ways, a cloud of its own—it can run applications, networks, data stores, load balancers, secret stores, and much more—so, in a sense, this is both a multiprovider and multicloud example. And because Kubernetes is a cloud, that means there is a lot to learn, so I'm going to have to build up to it one step at a time, starting with mini crash courses on Docker and Kubernetes, before finally moving on to the full multiprovider example that uses both AWS and Kubernetes:

- A crash course on Docker
- A crash course on Kubernetes