Each provider claims a specific prefix and exposes one or more resources and data sources whose names include that prefix: e.g., all the resources and data sources from the AWS Provider use the `aws_` prefix (e.g., `aws_instance`, `aws_autoscaling_group`, `aws_ami`), all the resources and data sources from the Azure provider use the `azurerm_` prefix (e.g., `azurerm_virtual_machine`, `azurerm_virtual_machine_scale_set`, `azurerm_image`), and so on.

How Do You Install Providers?

For official Terraform providers, such as the ones for AWS, Azure, and Google Cloud, it's enough to just add a `provider` block to your code:[1]

```
provider "aws" {
  region = "us-east-2"
}
```

As soon as you run `terraform init`, Terraform automatically downloads the code for the provider:

```
$ terraform init

Initializing provider plugins...
- Finding hashicorp/aws versions matching "4.19.0"...
- Installing hashicorp/aws v4.19.0...
- Installed hashicorp/aws v4.19.0 (signed by HashiCorp)
```

This is a bit magical, isn't it? How does Terraform know what provider you want? Or which version you want? Or where to download it from? Although it's OK to rely on this sort of magic for learning and experimenting, when writing production code, you'll probably want a bit more control over how Terraform installs providers. Do this by adding a `required_providers` block, which has the following syntax:

```
terraform {
  required_providers {
```

```
      <LOCAL_NAME> = {
        source  = "<URL>"
        version = "<VERSION>"
      }
    }
  }
```

where:

LOCAL_NAME

This is the *local name* to use for the provider in this module. You must give each provider a unique name, and you use that name in the `provider` block configuration. In almost all cases, you'll use the *preferred local name* of that provider: e.g., for the AWS Provider, the preferred local name is `aws`, which is why you write the provider block as `provider "aws" { … }`. However, in rare cases, you may end up with two providers that have the same preferred local name—e.g., two providers that both deal with HTTP requests and have a preferred local name of `http`—so you can use this local name to disambiguate between them.

URL

This is the URL from where Terraform should download the provider, in the format `[<HOSTNAME>/]<NAMESPACE>/<TYPE>`, where `HOSTNAME` is the hostname of a Terraform Registry that distributes the provider, `NAMESPACE` is the organizational namespace (typically, a company name), and `TYPE` is the name of the platform this provider manages (typically, `TYPE` is the preferred local name). For example, the full URL for the AWS Provider, which is hosted in the public Terraform Registry, is `registry.terraform.io/hashicorp/aws`. However, note that `HOSTNAME` is optional, and if you omit it, Terraform will by default download the provider from the public Terraform Registry, so the shorter and more common way to specify the exact same AWS Provider URL is `hashicorp/aws`. You typically only include `HOSTNAME` for custom providers that you're downloading

from private Terraform Registries (e.g., a private Registry you're running in Terraform Cloud or Terraform Enterprise).

VERSION

This is a version constraint. For example, you could set it to a specific version, such as `4.19.0`, or to a version range, such as `> 4.0, < 4.3`. You'll learn more about how to handle versioning in Chapter 8.

For example, to install version 4.x of the AWS Provider, you can use the following code:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
```

So now you can finally understand the magical provider installation behavior you saw earlier. If you add a new `provider` block named `foo` to your code, and you don't specify a `required_providers` block, when you run `terraform init`, Terraform will automatically do the following:

- Try to download provider `foo` with the assumption that the `HOSTNAME` is the public Terraform Registry and that the `NAMESPACE` is `hashicorp`, so the download URL is `registry.terraform.io/hashicorp/foo`.
- If that's a valid URL, install the latest version of the `foo` provider available at that URL.

If you want to install any provider not in the `hashicorp` namespace (e.g., if you want to use providers from Datadog, Cloudflare, or Confluent, or a custom provider you built yourself), or you want to control the version of