The private keys, in turn, are protected by a password that the developer either memorizes or stores in a personal secrets manager.

Centralized secret stores are typically web services that you talk to over the network that encrypt your secrets and store them in a data store such as MySQL, PostgreSQL, DynamoDB, etc. To encrypt these secrets, these centralized secret stores need an encryption key. Typically, the encryption key is managed by the service itself, or the service relies on a cloud provider's KMS.

The Interface You Use to Access Secrets

Most secret management tools can be accessed via an API, CLI, and/or UI.

Just about all centralized secret stores expose an API that you can consume via network requests: e.g., a REST API you access over HTTP. The API is convenient for when your code needs to programmatically read secrets. For example, when an app is booting up, it can make an API call to your centralized secret store to retrieve a database password. Also, as you'll see later in this chapter, you can write Terraform code that, under the hood, uses a centralized secret store's API to retrieve secrets.

All the file-based secret stores work via a *command-line interface (CLI)*. Many of the centralized secret stores also provide CLI tools that, under the hood, make API calls to the service. CLI tools are a convenient way for developers to access secrets (e.g., using a few CLI commands to encrypt a file) and for scripting (e.g., writing a script to encrypt secrets).

Some of the centralized secret stores also expose a *user interface (UI)* via the web, desktop, or mobile. This is potentially an even more convenient way for everyone on your team to access secrets.

A Comparison of Secret Management Tools

Table 6-1 shows a comparison of popular secret management tools, broken down by the three considerations defined in the previous sections.