Puppet, by itself, provides a large number of features and functionality. As you’ve learned so far in this book, Puppet enables you to manage the configuration state of a wide variety of resources. Files, users, groups, software packages and running services are prime examples. Configuration management is an extremely complex and multi-faceted problem, however, and as result we cannot expect Puppet alone to address every problem. In this chapter, we cover a number of additional tools that work extremely well with Puppet. These tools address many of the problems Puppet alone does not address.

The first problem is concerned with de-duplicating effort. The Puppet Forge provides a central place for members of the Puppet community to publish and download re-usable modules. The Puppet Module tool works with the Forge, providing a convenient command line interface, much like the yum and apt-get packaging commands provide. This chapter demonstrates how to download, install, and use modules from the forge.

In addition, you’ll learn how puppet-module can be used to generate a skeleton module structure and package modules. Even if the modules will never be published outside of your organization, these features provide a way to track module versions and distribute them to other groups internally.

While not an external tool, the Ruby DSL in Puppet 2.6 provides an alternative to declaring configuration resources using the Puppet language. The declarative nature of the Puppet language is a great way to express configuration state, but you may run across a configuration that is awkward or impossible to express using the Puppet language itself. In these situations, Puppet allows you to declare classes and resources using the Ruby programming language, providing additional functionality. You’ll see how the Example.com developer uses the Ruby DSL to transform data external to Puppet into resources and their parameter values in the configuration catalog. One example of a problem that’s difficult to solve with the Puppet language is the management of login accounts. As people join and leave Example.com, the developer would have to add and remove resource declarations in the Puppet manifests. A more ideal solution would be if Puppet could automatically declare resources based on information from an outside data source like LDAP. The Ruby DSL is ideally suited to the task of iterating over an arbitrary amount of external data, then declaring resources using the data.

As Puppet configurations change, testing the change is always a good idea before pushing to the production infrastructure. Puppet is designed to model the desired state of a system, which is closely related to how that system behaves. With the idea of desired behavior in mind, the natural language specifications of Cucumber inspired Nikolay Sturm to develop cucumber-puppet. Cucumber-puppet allows you to describe the desired behavior of Puppet infrastructure and test the configuration model stored in the catalog.

Puppet Forge and Module Tool

The Puppet Forge, located at http://forge.puppetlabs.com/, provides an online repository of Puppet modules. This service provides the means to publish and locate modules for commonly managed services like iptables, apache, and NTP. In addition, there are modules targeted for specific use cases, such as Hadoop.
If you find yourself needing to quickly deploy a complex infrastructure like Hadoop, the Puppet Forge will save you much time and effort. Modules on the Forge provide a reference configuration that may be easily modified if necessary. The Forge strives to become to Puppet what CPAN is to Perl hackers. Puppet modules may be manually downloaded from the Forge using a standard web browser, but the process is made much easier through the use of the Puppet Module tool, called puppet-module.

The puppet-module command provides an interface to the Forge API. This command line interface allows you to create skeleton Puppet Modules for your own work, search the forge for existing modules, and install them into your configuration. In this section, we cover the process of downloading an already-existing module and publishing a new module to the forge.

**Installing the Puppet Module Tool**

Unlike Puppet, which is distributed in many package repositories for various operating systems, the Puppet Module Tool is primarily distributed through the RubyGems package repository. This has the advantage of making installation straightforward and easy on all platforms with RubyGems installed. Listing 8-1 shows how the Example.com operator installs the Puppet Module tool.

**Listing 8-1. Installing Puppet Module using Gems**

```
$ gem install puppet-module

Thank you for installing puppet-module from Puppet Labs!

* Usage instructions: read "README.markdown" or run `puppet-module usage`
* Changelog: read "CHANGES.markdown" or run `puppet-module changelog`
* Puppet Forge: visit http://forge.puppetlabs.com/

Successfully installed puppet-module-0.3.2
1 gem installed
Installing ri documentation for puppet-module-0.3.2...
Installing RDoc documentation for puppet-module-0.3.2...

# puppet-module version
0.3.2
```

The operator first installs puppet-module using the `gem` command, then he checks to make sure the command is executable and at the correct version.

---

**Note** The Puppet Module tool project page and source code are hosted on GitHub at https://github.com/puppetlabs/puppet-module-tool. An alternative to installing the software using RubyGems is to clone a copy of the source and use the `install.rb` script included in the source. This also gives you the ability to easily modify and contribute to the project.