Lab 10.2: Puppet Modules for Bacula
IN719 Systems Administration

Preface

You must complete lab 10.1 before starting on this task.

Introduction

In the previous lab we performed a minimal installation of Bacula on our `mgmt` servers so that we could try out a backup/restore process. In this lab we're going to start preparing our Puppet modules to manage a complete Bacula installation across our systems. We will create three modules in this lab:

- `bacula-director` will install and manage the Bacula Director service. We will apply this module to our `mgmt` servers.
- `bacula-file` will install and manage the Bacula File Daemon (client) service. We will apply this module to every Linux server.
- `bacula-storage` will install and manage the Bacula Storage Daemon. We will apply this module to our `storage` server.

1 Module Structure

We will lay out each of our three modules using the same structure we have used in the past. Each module will be placed in a subdirectory of `/etc/puppet/modules`. Each module diectory will have three subdirectories: `files`, `manifests`, and `templates`. Inside the `manifests` subdirectory we will create the files `init.pp`, `install.pp`, `config.pp`, and `service.pp`. We will place copies of the appropriate configuration files in the `files` subdirectory.

2 Installation

Our three modules will need to install the following packages:

- The `bacula-director` module will install the packages `bacula-server` and `bacula-console`.
- The `bacula-file` module will install the `bacula-fd` package.
- The `bacula-storage` module will install the `bacula-sd` package.

The installation code will go in the `install.pp` file in each module. For example, the `bacula-file` module will have an install.pp file like this:

```
class bacula-file::install {
  package { 'bacula-fd',
      ensure => present,
  }
}
```

Then, put the following code in the module's `init.pp` file:

```
class bacula-file {
  include bacula-file::install,
}
```

Set up each of your modules and write the code to install the appropriate packages for them. Leave the config and service files empty for now. Apply your modules to their target servers and make sure they function correctly before proceeding.

3 Configuration

Every Bacula component has its own configuration file that we need to manage.

- The `bacula-director` module needs to manage the files `/etc/bacula/bacula-dir.conf` and `/etc/bacula/bconsole.conf`
- The `bacula-fd` module will manage the file `/etc/bacula/bacula-fd.conf`.
- The `bacula-storage` module will manage the file `/etc/bacula/bacula-sd.conf`.

You can get initial copies of each of these files from servers on which you performed the installations in the previous step. Set up the appropriate file resources in your modules' respective `config` classes, and add the config classes to the modules' `init.pp` file. For example, the `bacula-file` module's `init.pp` should look like this:

```
class bacula-file {
include bacula-file::install, bacula-file::config,
}
```

Test your module additions by applying them to to your servers before continuing.

4 Services

One you have the installation and configuration incorporated into your Puppet modules, you need to add service resources.

- The `bacula-director` module will ensure that the `bacula-director` service is running on `mgmt`
- The `bacula-storage` module will ensure that the `bacula-sd` module is running on `storage`
- The `bacula-file` module will ensure that the `bacula-fd` service is running on every Linux server.

Once you have the service classes implemented in your Puppet modules you can add `Notify` clauses to the configuration classes so that the services will be restarted when configuration changes.

Once these modules are prepared and applied we can begin modifying the configurations to support our backup and restore jobs.