Lab 5.1: A Puppet Module to Manage MySQL
IN719 Systems Administration

Introduction

In this lab we will build a more complex module to manage our MySQL server. This module will use a collection of related *classes*. We've used classes already, but you may not have paid much attention to them. Note that this lab is based on an example from *Pro Puppet* by James Turnbull. In this module we will handle not just installation and configuration of a service, but also preinstallation tasks, operation, and ongoing maintenance of the service.

1 Module setup

Create a standard module structure in the `/etc/puppet/modules` directory of your puppetmaster.

```
mysql
mysql/files/my.cnf
mysql/files/debian-start
mysql/manifests/init.pp
mysql/manifests/install.pp
mysql/manifests/config.pp
mysql/manifests/service.pp
mysql/templates
```

2 mysql::install

The **mysql::install** class includes the resources needed to install MySQL. Put the following in your `install.pp` file

```
class mysql::install {
  package { "mysql-server" :
            ensure => present,
            require => User["mysql"],
  }
  user { "mysql":
         ensure => present,
         comment => "MySQL user",
         gid => "mysql",
         shell => "/bin/false",
         require => Group["mysql"],
  }
  group { "mysql" :
          ensure => present,
  }
 }
```

Note how we use `require` directives to make sure that things are set up in the correct order.

3 mysql::config

Place the following classes in your `config.pp` file.

```
class mysql::config {
  file { "/etc/mysql/my.cnf":
    ensure => present,
    source => "puppet:///modules/mysql/my.cnf",
    mode => 0444,
    owner => "mysql",
    group => "mysql",
    require => Class["mysql::install"],
    notify => Class["mysql::service"],
  }
  file { "/etc/mysql/debian-start":
    ensure => present,
    source => "puppet:///modules/mysql/debian-start",
    owner => "root",
    group => "root",
    mode => 0555,
    require => Class["mysql::install"],
    notify => Class["mysql::service"],
  }
}
```

Notice how these resources require mysql::install, and they also *notify* mysql::service. This means that the mysql daemon will be restarted whenever its configuration changes.

You'll need to obtain copies of the `my.cnf` and `debian-start` files. You can use `scp` to copy them from your db server.

4 mysql::service

The mysql::service class is brief. Place it in your `service.pp` file.

```
class mysql::service {
  service { "mysql" :
    ensure => running,
    hasstatus => true,
    hasrestart => true,
    enable => true,
    require => Class["mysql::config"],
  }
}
```

This class will make sure that MySQL is running and will restart it if necessary when its configuration is changed by Puppet.

5 mysql class

Finally we just combine our classes in the `init.pp` file.

```
class mysql {
  include mysql::install, mysql::config, mysql::service
}
```

Now you can apply the module to your db server by placing `include mysql` in the node definition for your db server.

6 Follow up

With the ability to create modules like this, you are now prepared to start fully managing your Debian servers with Puppet. You should set nodes for your other Debian servers and get those servers connected to Puppet. There are a few Windows specific issues that we will sort out next week, so hold off on working on teh Windws server for now.