Lab 4.3: A Puppet Module for Hosts Files
IN719 Systems Administration

Introduction

Last time we made a Puppet module for `sudo`. That module wasn't very flexible, however. It only worked because we configure `sudo` in the exact same way on every system. Today we'll make a more flexible module to manage our hosts files. To accomplish this, we'll make use of Puppet's *variables*, *conditionals*, and *templates*.

1 Module setup

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

```
hosts_file
hosts_file/manifests
hosts_file/files
hosts_file/templates
```

Create an `init.pp` file in your `manifests` subdirectory.

2 Module manifest

The following code is the basis for your `init.pp` file. At the end of this document you will find sample host files, but you will need to adjust the IP addresses to match your systems. Have a look at the explanation below before starting to work on it.

```
class hosts_file {
  if $osfamily == 'Debian' {
    include deb_hosts
  }
  elsif $osfamily == 'windows' {
    include win_hosts
  }
}

class hosts_file::deb_hosts {
  file { "/etc/hosts" :
    ensure => present,
    owner => 'root',
    group => 'root',
    mode => 0444,
    content => template('hosts_file/debhosts.erb'),
  }
}

class hosts_file::win_hosts {
  file {"C:/windows/System32/drivers/etc/hosts" :
    ensure => present,
    content => template('hosts_file/winhosts.erb'),
  }
}
```

There are a few new things happening in this manifest.

- We're using a *variable*, `$osfamily`. We can define and use our own variables, but many variables are populated for us by a utility called *Facter*. You can see a list of the core facts produced by Facter at `http://docs.puppetlabs.com/facter/1.6/core_facts.html`.
  - Hint: Use `facter -p` to find out the variable values for your systems. Try it out!
- We are using an `if/elsif` structure to conditionally select which Puppet class to use based in the operating system of the agent.
- Instead of copying over static files, we are using *templates*. The template files are to be placed in the `templates` subdirectory of the module. Puppet's templates use the erb (Embedded Ruby) templating system.

Right now, we can not just copy and past those files. Depending on your configuration, you may need to deal with the different hosts files on Debian and Ubuntu systems.

Do you need to deal with it?

If so, which variable can you use instead of `$osfamily` to differentiate between Debian and Ubuntu systems in puppet?

3 Template files

Finally, we need to write our template files in the `templates` subdirectory of our module. The text of those files is below. Again, you will need to modify those to capture our context.

debhosts.erb

```
127.0.0.1       localhost <%= hostname %>
10.26.1.50      ad ad.micro-agents.net
10.26.1.51      app app.micro-agents.net
10.26.1.52      db db.micro-agents.net
10.26.1.53      mgmt mgmt.micro-agents.net
10.26.1.54      backup backup.micro-agents.net

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

winhosts.erb

```
127.0.0.1       localhost <%= hostname %>
10.26.1.50      ad ad.micro-agents.net
10.26.1.51      app app.micro-agents.net
10.26.1.52      db db.micro-agents.net
10.26.1.53      mgmt mgmt.micro-agents.net
10.26.1.54      backup backup.micro-agents.net
```

In these templates we are inserting the correct value for the local host name with the `hostname` variable that is defined by Facter.

4 Follow up

You can, and should, read more about Puppet templates at http://docs.puppetlabs.com/learning/templates.html.