Writing your first puppet module:

In the previous two classes we did the initial installation and configuration of Puppet. Puppet is a very powerful configuration and management tool. If we start with a minimal Linux installation (in our case CentOS 7) with an enabled Puppet Client, we can then use Puppet to configure the Client for any desired functionality. This greatly simplifies the Cobbler installation since the only part that needs to change between kickstart files is the network configuration. For what we covered so far please see:

|  |
| --- |
| * Class 3 - Cobbler Installation Part 1 * Class 4 - Cobbler Installation Part 2 * Class 6 - Using Different Kickstart Files * Class 6 - Puppet Installation and Configuration |

To begin to use of the power of Puppet, we will write our first Puppet module. As we go along you will begin to understand how to use Puppet.

Understanding How Puppet Works:

When we installed Puppet Server, we created to the script:

|  |
| --- |
| /etc/puppetlabs/puppet/node.sh |

When the **puppet agent** runs on the **Puppet Client,** the above script is run on the **Puppet Server** with one command line argument (the **hostname** of the **Puppet Client**) and the output is used by the **Puppet Server** to create a **“catalog”** (list of classes) for the **Puppet Client.**

When you run this script it produces the output:

|  |
| --- |
| ---  classes:  role::htcxxx: {} |

This tells the **Puppet Server** to create the “**catalog**” containing the class below:

|  |
| --- |
| role:htcxxx:{} |

All the classes are stored in **manifest** files. **Manifest** files are in the **modules** subdirectory and have the extension **.pp**.

|  |
| --- |
| /etc/puppetlabs/code/environments/production/modules |

When we installed **Puppet Server**, we installed the modules **motd** and **role**. The **.pp** (**manifest**) files for each of these modules are in:

|  |
| --- |
| /etc/puppetlabs/code/environments/production/modules/role/manifests/init.pp  /etc/puppetlabs/code/environments/production/modules/motd/manifests/init.pp |

The code in **role/manifests/init.pp** is:

|  |
| --- |
| class role {  notify{'role':, message => "Class: role"}  }  #  # htcxxx role  #  class role::htcxxx inherits role {  notice("Top level role for htcxxx")  include motd  } |

From this we see that **Puppet** supports **inheritance** and **includes**. The class:

|  |
| --- |
| class role::htcxxx |

**inherits** from **role** (in this case role simply prints a message) and includes the class **motd**.

The class **motd** is in **motd/manifests/init.pp**. The first motd class we used was:

|  |
| --- |
| class motd {  file {'/etc/motd':,  ensure => file,  replace => true,  owner => 'root',  group => 'root',  mode => '0644',  source => "puppet:///modules/motd/motd"  }  } |

In this class:

* /etc/motd - File to update on the client
* ensure - Make sure this file exists
* replace - Replace the file when updating
* owner - The file is owned by root
* group - The file is in the group root
* mode - The file has permissions 0644.
* source - Get the file from the Puppet Server in modules/motd/**files**/motd  
   Notice the addition of the subdirectory **files**.

Summary:

**Puppet Server** constructed a **catalog** beginning with r**ole::htcxxx{}**. That class included **motd**. The **motd** class instructed the **Puppet Client** to check the contents of the file **/etc/motd** against the contents of the file on the puppet server (**modules/motd/files/motd**). If they are different replace the file on the client with the file on the server. If they are the same, do nothing.



Creating a module:

|  |
| --- |
| **All commands are run on the Puppet Server.** |

With the above information we will create a **module** that will install the packages **nfs-utils** and **autofs**, and the configuration files **auto.master** and **auto.nfs**. It will also make sure that **autofs** is running, **enable** it start at boot time and **restart** it if the configuration files change.

Create a module:  
The **puppet command** provides a means for creating a **module skelton**. Shown below is the command syntax and a sample output. You can either accept all the defaults or fill in the bolded responses in the sample output.

|  |
| --- |
| cd /etc/puppetlabs/code/environments/production/modules  puppet module generate htcclass-autofs  mkdir -p autofs/files autofs/templates |

Sample output from the above command:

|  |
| --- |
| root> puppet module generate htcclass-autofs  We need to create a metadata.json file for this module. Please answer the  following questions; if the question is not applicable to this module, feel free  to leave it blank.  Puppet uses Semantic Versioning (semver.org) to version modules.  What version is this module? [0.1.0]  -->  Who wrote this module? [htcclass]  --> **An Najah HTC Class**  What license does this module code fall under? [Apache-2.0]  -->  How would you describe this module in a single sentence?  --> **This module installs and manages autofs**  Where is this module's source code repository?  -->  Where can others go to learn more about this module?  -->  Where can others go to file issues about this module?  -->  ----------------------------------------  {  "name": "htcclass-autofs",  "version": "0.1.0",  "author": "An Najah HTC Class",  "summary": "This module installs and manages autofs",  "license": "Apache-2.0",  "source": "",  "project\_page": null,  "issues\_url": null,  "dependencies": [  {"name":"puppetlabs-stdlib","version\_requirement":">= 1.0.0"}  ],  "data\_provider": null  }  ----------------------------------------  About to generate this metadata; continue? [n/Y]  -->  Notice: Generating module at /batch/theforeman/autofs...  Notice: Populating templates...  Finished; module generated in autofs.  autofs/Gemfile  autofs/Rakefile  autofs/examples  autofs/examples/init.pp  autofs/manifests  autofs/manifests/init.pp  autofs/spec  autofs/spec/classes  autofs/spec/classes/init\_spec.rb  autofs/spec/spec\_helper.rb  autofs/README.md  autofs/metadata.json |

You have now created the module **autofs** with the files:

|  |
| --- |
| /etc/puppetlabs/code/environments/production/modules/  autofs/Gemfile  autofs/Rakefile  autofs/examples  autofs/examples/init.pp  autofs/manifests  autofs/manifests/init.pp  autofs/spec  autofs/spec/classes  autofs/spec/classes/init\_spec.rb  autofs/spec/spec\_helper.rb  autofs/README.md  autofs/metadata.json  autofs/files  autofs/templates |

The controlling puppet script for the module **autofs** is:

|  |
| --- |
| autofs/manifests/init.pp |

Let’s edit that file and add the contents shown below.

|  |
| --- |
| vi autofs/manifests/init.pp |

The code below is also available from:

|  |
| --- |
| /nfs/htc180/pub/AnNajah-Files/autofs/manifests/init.pp |

The code for modules/manifests/init.pp:

|  |
| --- |
| class autofs {  file { "/etc/auto.master":  notify => Service['autofs'],  require => Package['autofs', nfs-utils],  ensure => file,  replace => true,  owner => 'root',  group => 'root',  mode => "0644",  source => "puppet:///modules/autofs/auto.master",  }  file { "/etc/auto.nfs":  notify => Service['autofs'],  require => Package['autofs', 'nfs-utils'],  ensure => file,  replace => true,  owner => 'root',  group => 'root',  mode => "0644",  source => "puppet:///modules/autofs/auto.nfs",  }  package {"nfs-utils":  ensure => installed,  }  package {"autofs":  ensure => installed,  }  service {"autofs":  ensure => running,  enable => true,  }  } |

Analyze the contents of init.pp:  
The puppet **file** directive:

The first section of code used the puppet directive **file**. This defines a file on the client that puppet should monitor. This looks very similar to the **motd** **module** except for the use of:

|  |
| --- |
| notify => Service[‘autofs’]  require => Package['autofs', nfs-utils], |

The **first directive** means **notify** the service “**autofs**” if the file **/etc/auto.master** changes. The **second directive** means **require** that the packages **autofs** and **nfs-utils** are **installed** before acting on the file **/etc/auto.master**.

|  |
| --- |
| file { "/etc/auto.master":  notify => Service['autofs'],  require => Package['autofs', nfs-utils],  ensure => file,  replace => true,  owner => 'root',  group => 'root',  mode => "0644",  source => "puppet:///modules/autofs/auto.master",  } |

The puppet “**package**” command:  
The next part of the **manifest** defines the two packages, **autofs** and **nfs-utils**, and tells puppet to make sure (**ensure**) that they are **installed**.

|  |
| --- |
| package {"nfs-utils":  ensure => installed,  }  package {"autofs":  ensure => installed,  } |

The puppet “**service**” command:

The last part of the manifest defines the **service** “**autofs**”. This code tells **puppet** to make sure (**ensure**) that autofs is running and to **enable** it so that it is **started** on a **reboot**.

|  |
| --- |
| service {"autofs":  ensure => running,  enable => true,  } |

Install the configuration files in **Puppet repository**:

We can use the previously installed **autofs** configuration files and put them into the puppet repository, We simply have to copy them to the files subdirectory of the autofs module:

|  |
| --- |
| cp /etc/auto.master \  /etc/puppetlabs/code/environments/production/modules/autofs/files/  cp /etc/auto.nfs \  /etc/puppetlabs/code/environments/production/modules/autofs/files/ |

Add autofs to role/manifests/init.pp for client:

We now tell puppet to use the **autofs** module with the **class role:htcxx** by including it in the the **manifest** for **role**.

Modify **modules/role/manifests/init.pp** and add the line as shown below.

|  |
| --- |
| #  # htcxxx role  #  class role::htcxxx inherits role {  notice("Top level role for htcxxx")  include motd  **include autofs**  } |

Run agent on client:

Now go to the **Puppet Client** and run the **puppet agent command**:

|  |
| --- |
| puppet agent -t |

You should output similar to:

|  |
| --- |
| Info: Using configured environment 'production'  Info: Retrieving pluginfacts  Info: Retrieving plugin  Info: Caching catalog for htc192.najah.edu  Info: Applying configuration version '1522063314'  Notice: Class: role  Notice: /Stage[main]/Role/Notify[role]/message: defined 'message' as 'Class: role'  Notice: /Stage[main]/Autofs/Package[nfs-utils]/ensure: created  Notice: /Stage[main]/Autofs/Package[autofs]/ensure: created  Notice: /Stage[main]/Autofs/File[/etc/auto.master]/content:  --- /etc/auto.master 2017-11-30 21:17:52.000000000 +0200  +++ /tmp/puppet-file20180326-9713-dxog3s 2018-03-26 14:22:03.945567443 +0300  @@ -4,18 +4,18 @@  # mount-point [map-type[,format]:]map [options]  # For details of the format look at auto.master(5).  #  -/misc /etc/auto.misc  +# /misc /etc/auto.misc  #  # NOTE: mounts done from a hosts map will be mounted with the  # "nosuid" and "nodev" options unless the "suid" and "dev"  # options are explicitly given.  #  -/net -hosts  +# /net -hosts  #  # Include /etc/auto.master.d/\*.autofs  # The included files must conform to the format of this file.  #  -+dir:/etc/auto.master.d  +# +dir:/etc/auto.master.d  #  # Include central master map if it can be found using  # nsswitch sources.  @@ -25,4 +25,6 @@  # same will not be seen as the first read key seen takes  # precedence.  #  -+auto.master  +# +auto.master  +/nfs /etc/auto.nfs  +  Info: Computing checksum on file /etc/auto.master  Info: FileBucket got a duplicate file {md5}cbf1ddb411c80749876271e3078e6d7c  Info: /Stage[main]/Autofs/File[/etc/auto.master]: Filebucketed /etc/auto.master to puppet with sum cbf1ddb411c80749876271e3078e6d7c  Notice: /Stage[main]/Autofs/File[/etc/auto.master]/content:  Notice: /Stage[main]/Autofs/File[/etc/auto.master]/content: content changed '{md5}cbf1ddb411c80749876271e3078e6d7c' to '{md5}3e10fe35a25405e24c1c8a27857be10a'  Info: /Stage[main]/Autofs/File[/etc/auto.master]: Scheduling refresh of Service[autofs]  Notice: /Stage[main]/Autofs/File[/etc/auto.nfs]/ensure: defined content as '{md5}a147cc0949d3d7054a16ce3e87182bf5'  Info: /Stage[main]/Autofs/File[/etc/auto.nfs]: Scheduling refresh of Service[autofs]  Notice: /Stage[main]/Autofs/Service[autofs]/ensure: ensure changed 'stopped' to 'running'  Info: /Stage[main]/Autofs/Service[autofs]: Unscheduling refresh on Service[autofs]  Notice: Applied catalog in 8.60 seconds |

Check that /nfs/htc180 now works on the client:

Remaining on the **Puppet Client**, you should now have access to the **NFS disk**:

|  |
| --- |
| ls -al /nfs/htc180/ |