Create the Apache Module

1. From the Puppet Master, navigate to Puppet’s module directory and create the apachedirectory:
2. cd /etc/puppetlabs/code/environments/production/modules/

sudo mkdir apache

1. From within the apache directory, create manifests, templates, files, and examples directories:
2. cd apache
3. sudo mkdir {manifests,templates,files,examples}
4. Navigate to the manifests directory:

cd manifests

1. From here, the module will be separated into classes, based on the goals of that particular section of code. In this instance, there will be an init.pp class for downloading the Apache package, a params.pp file to define any variables and parameters, config.pp to manage any configuration files for the Apache service itself, and a vhosts.pp file to define the virtual hosts. This module will also make use of Hiera data to store variables for each node.

### Create the Initial Apache Class and Parameters

1. From within the manifests directory, create an init.pp file to hold the apache class. This class should share its name with the module name. This file will be used to install the Apache package. Since Ubuntu 18.04 and CentOS 7 use different package names for Apache, a variable will be used:

… **modules/apache/manifests/init.pp**

**class** apache {

**package** { 'apache':

name => $apachename,

ensure => **present**,

}

}

1. The package resource allows for the management of a package. This is used to add, remove, or ensure a package is present. In most cases, the name of the resource (apache, above) should be the name of the package being managed. Because of the difference in naming conventions, however, this resource is simply called apache, while the actual name of the package is called upon with the name reference. name, in this instance, calls for the yet-undefined variable $apachename. The ensure reference ensures that the package is present.
2. The params.pp file will be used to define the needed variables. While these variables could be defined within the init.pp file, since more variables will need to be used outside of the resource type itself, using a params.pp file allows for variables to be defined in if statements and used across multiple classes.

Create a params.pp and add the following code:

.. **modules/apache/manifests/params.pp**

**class** apache::params {

**if** $::osfamily == 'RedHat' {

$apachename = 'httpd'

}

**elsif** $::osfamily == 'Debian' {

$apachename = 'apache2'

}

**else** {

**fail** ( 'this is not a supported distro.')

}

}

Outside of the original init.pp class, each class name needs to branch off of apache. As such, this class is called apache::params. The name after the double colon should share a name with the file.

With the parameters finally defined, we need to call the params.pp file and the parameters into init.pp. To do this, the parameters need to be added after the class name, but before the opening curly bracket ({):

…. **modules/apache/manifests/init.pp**

**class** apache (

$apachename = $::apache::params::apachename,

) **inherits** ::apache::params {

The value string $::apache::params::value tells Puppet to pull the values from the apachemodules, params class, followed by the parameter name. The fragment inherits ::apache::params allows for init.pp to inherit these values.

### Manage Configuration Files

The Apache configuration file will be different depending on whether you are working on a Red Hat- or Debian-based system.

1. Copy the content of httpd.conf and apache2.conf in separate files and save them in the filesdirectory located at /etc/puppetlabs/code/environments/production/modules/apache/files.
2. Both files need to be edited to disable keepalive. You will need to add the line KeepAlive Offthe httpd.conf file. If you do not want to change this setting, a comment should be added to the top of each file:

… **modules/apache/files/httpd.conf**

Add these files to the init.pp file, so Puppet will know where they are located on both the master server and agent nodes. To do this, the file resource is used:

**…/modules/apache/manifests/init.pp**

**file** { 'configuration-file':

path => $conffile,

ensure => **file**,

source => $confsource,

}

1. Because the configuration file is found in two different locations, the resource is given the generic name configuration-file with the file path defined as a parameter with the pathattribute. ensure ensures that it is a file. source provides the location on the Puppet master of the files created above.
2. Open the params.pp file. The $conffile and $confsource variables need to be defined within the if statement:

…/ **modules/apache/manifests/params.pp**

**if** $::osfamily == 'RedHat' {

...

$conffile = '/etc/httpd/conf/httpd.conf'

$confsource = 'puppet:///modules/apache/httpd.conf'

}

**elsif** $::osfamily == 'Debian' {

...

$conffile = '/etc/apache2/apache2.conf'

$confsource = 'puppet:///modules/apache/apache2.conf'

}

**else** {

...

1. These parameters will also need to be added to the beginning of the apache class declaration in the init.pp file, similar to the previous example. A complete copy of the init.pp file can be seen here for reference.
2. When the configuration file is changed, Apache needs to restart. To automate this, the serviceresource can be used in combination with the notify attribute, which will call the resource to run whenever the configuration file is changed:

…/ **modules/apache/manifests/init.pp**

**file** { 'configuration-file':

path => $conffile,

ensure => **file**,

source => $confsource,

**notify** => **Service**['apache-service'],

}

**service** { 'apache-service':

name => $apachename,

hasrestart => **true**,

}

1. The service resource uses the already-created parameter that defined the Apache name on Red Hat and Debian systems. The hasrestart attribute will trigger a restart of the defined service.

### Create the Virtual Hosts Files

Depending on your systems distribution the virtual hosts files will be managed differently. Because of this, the code for virtual hosts will be encased in an if statement, similar to the one used in the params.pp class but containing actual Puppet resources. This will provide an example of an alternate use of if statements within Puppet’s code.

1. From within the apache/manifests/ directory, create and open a vhosts.pp file. Add the skeleton of the if statement:

… **modules/apache/manifests/vhosts.pp**

**class** apache::vhosts {

**if** $::osfamily == 'RedHat' {

} **elsif** $::osfamily == 'Debian' {

} **else** {

}

}

The location of the virtual hosts file on our CentOS 7 server is /etc/httpd/conf.d/vhost.conf. This file will need to be created as a template on the Puppet master. The same needs to be done for the Ubuntu virtual hosts file, which is located at /etc/apache2/sites-available/example.com.conf, replacing example.com with the server’s FQDN. Navigate to the templates file within the apachemodule, and then create two files for your virtual hosts:

For Red Hat systems:

… **modules/apache/templates/vhosts-rh.conf.erb**

**<VirtualHost** \*:80**>**

ServerAdmin <%= @adminemail %>

ServerName <%= @servername %>

ServerAlias www.<%= @servername %>

DocumentRoot /var/www/<%= @servername -%>/public\_html/

ErrorLog /var/www/<%- @servername -%>/logs/error.log

CustomLog /var/www/<%= @servername -%>/logs/access.log combined

**</Virtual** Host**>**

For Debian systems:

**/etc/puppet/modules/apache/templates/vhosts-deb.conf.erb**

**<VirtualHost** \*:80**>**

ServerAdmin <%= @adminemail %>

ServerName <%= @servername %>

ServerAlias www.<%= @servername %>

DocumentRoot /var/www/html/<%= @servername -%>/public\_html/

ErrorLog /var/www/html/<%- @servername -%>/logs/error.log

CustomLog /var/www/html/<%= @servername -%>/logs/access.log combined

**<Directory** /var/www/html/<%= @servername -%**>**/public\_html>

Require **all** granted

**</Directory>**

**</Virtual** Host**>**

1. Only two variables are used in these files: adminemail and servername. These will be defined on a node-by-node basis, within the site.pp file.
2. Return to the vhosts.pp file. The templates created can now be referenced in the code:

**/etc/puppetlabs/code/environments/production/modules/apache/manifests/vhosts.pp**

**class** apache::vhosts {

**if** $::osfamily == 'RedHat' {

**file** { '/etc/httpd/conf.d/vhost.conf':

ensure => **file**,

content => **template**('apache/vhosts-rh.conf.erb'),

}

} **elsif** $::osfamily == 'Debian' {

**file** { "/etc/apache2/sites-available/$servername.conf":

ensure => **file**,

content => **template**('apache/vhosts-deb.conf.erb'),

}

} **else** {

**fail**('This is not a supported distro.')

}

}

Both distribution families call to the file resource and take on the title of the virtual host’s location on the respective distribution. For Debian, this once more means referencing the $servername value. The content attribute calls to the respective templates.

Both virtual hosts files reference two directories that are not on the systems by default. These can be created through the use of the file resource, each located within the if statement. The complete vhosts.conf file should resemble

**/etc/puppetlabs/code/environments/production/modules/apache/manifests/vhosts.pp**

**class** apache::vhosts {

**if** $::osfamily == 'RedHat' {

**file** { '/etc/httpd/conf.d/vhost.conf':

ensure => **file**,

content => **template**('apache/vhosts-rh.conf.erb'),

}

**file** { [ '/var/www/$servername',

'/var/www/$servername/public\_html',

'/var/www/$servername/log', ]:

ensure => **directory**,

}

} **elsif** $::osfamily == 'Debian' {

**file** { "/etc/apache2/sites-available/$servername.conf":

ensure => **file**,

content => **template**('apache/vhosts-deb.conf.erb'),

}

**file** { [ '/var/www/$servername',

'/var/www/$servername/public\_html',

'/var/www/$servername/logs', ]:

ensure => **directory**,

}

} **else** {

**fail** ( 'This is not a supported distro.')

}

}

### Test and Run the Module

1. From within the apache/manifests/ directory, run the puppet parser on all files to ensure the Puppet coding is without error:
2. sudo /opt/puppetlabs/bin/puppet parser validate init.pp params.pp vhosts.pp

It should return empty, barring any issues.

1. Navigate to the examples directory within the apache module. Create an init.pp file and include the created classes. Replace the values for $servername and $adminemail with your own:

**/etc/puppetlabs/code/environments/production/modules/apache/examples/init.pp**

$serveremail = 'webmaster@example.com'

$servername = 'example.com'

**include** apache

**include** apache::vhosts

1. Test the module by running puppet apply with the --noop tag:
2. sudo /opt/puppetlabs/bin/puppet apply --noop init.pp

It should return no errors, and output that it will trigger refreshes from events. To install and configure apache on the Puppet master, this can be run again without --noop , if so desired.

1. Navigate back to the main Puppet directory and then to the manifests folder (**not** the one located in the Apache module).

cd /etc/puppetlabs/code/environments/production/manifests

Open site.pp and include the Apache module for each agent node. Also input the variables for the adminemail and servername parameters.

**/etc/puppetlabs/code/environments/production/manifests/site.pp**

**node** 'ubuntuhost.example.com' {

$adminemail = 'webmaster@example.com'

$servername = 'hostname.example.com'

**include** accounts

**include** apache

**include** apache::vhosts

**resources** { 'firewall':

purge => **true**,

}

Firewall {

**before** => **Class**['firewall::post'],

require => **Class**['firewall::pre'],

}

**class** { ['firewall::pre', 'firewall::post']: }

}

**node** 'centoshost.example.com' {

$adminemail = 'webmaster@example.com'

$servername = 'hostname.example.com'

**include** accounts

**include** apache

**include** apache::vhosts

**resources** { 'firewall':

purge => **true**,

}

Firewall {

**before** => **Class**['firewall::post'],

require => **Class**['firewall::pre'],

}

**class** { ['firewall::pre', 'firewall::post']: }

}

**/etc/puppetlabs/code/environments/production/manifests/site.pp**

**node** 'ubupuppet.members.linode.com' {

$adminemail = 'webmaster@example.com'

$servername = 'hostname.example.com'

**include** apache

**include** apache::vhosts

}

**node** 'centospuppet.members.linode.com' {

$adminemail = 'webmaster@example.com'

$servername = 'hostname.example.com'

**include** apache

**include** apache::vhosts

}

By default, the Puppet agent service on your managed nodes will automatically check with the master once every 30 minutes and apply any new configurations from the master. You can also manually invoke the Puppet agent process in-between automatic agent runs. To manually run the new module on your agent nodes, log in to the nodes and run:

sudo /opt/puppetlabs/bin/puppet agent -t

## Using the MySQL Module

Many modules needed to run a server already exist within Puppet Labs’ Puppet Forge. These can be configured just as extensively as a module that you created and can save time since the module need not be created from scratch.

**/etc/puppetlabs/code/environments/production/hiera.yaml**

---

version: 5

hierarchy:

- name: Common

path: common.yaml

defaults:

data\_hash: yaml\_data

datadir: data

Create the file common.yaml. It will be used to define the default root password for MySQL:

**/etc/puppetlabs/code/environments/production/common.yaml**

|  |  |
| --- | --- |
| 1 | mysql::server::root\_password: 'password' |

1. The common.yaml file is used when a variable is not defined elsewhere. This means all servers will share the same MySQL root password. These passwords can also be hashed to increase security.
2. To use the MySQL module’s defaults you can simply add an include '::mysql::server' line to the site.pp file. However, in this example, you will override some of the module’s defaults to create a database for each of your nodes. Edit the site.pp file with the following values:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | node 'ubupuppet.members.linode.com' {  $adminemail = 'webmaster@example.com'  $servername = 'hostname.example.com'  include apache  include apache::vhosts  include php  mysql::db { "mydb\_${fqdn}":  user => 'myuser',  password => 'mypass',  dbname => 'mydb',  host => $::fqdn,  grant => ['SELECT', 'UPDATE'],  tag => $domain,  }  }  node 'centospuppet.members.linode.com' {  $adminemail = 'webmaster@example.com'  $servername = 'hostname.example.com'  include apache  include apache::vhosts  include mysql::server  include php  mysql::db { "mydb\_${fqdn}":  user => 'myuser',  password => 'mypass',  dbname => 'mydb',  host => $::fqdn,  grant => ['SELECT', 'UPDATE'],  tag => $domain,  }  } |

1. You can run these updates manually on each node by SSHing into each node and issuing the following command:

sudo /opt/puppetlabs/bin/puppet agent -t

## Create the PHP Module

1. Create the php directory in the /etc/puppetlabs/code/environments/production/modules path, and generate the files, manifests, templates, and examples directories afterward:
2. sudo mkdir php
3. cd php
4. sudo mkdir {files,manifests,examples,templates}
5. Create the init.pp. This file will use the package resource to install PHP. Two packages will be installed: The PHP package and the PHP extension and application repository. Add the following contents to your file:

**/etc/puppetlabs/code/environments/production/modules/php/manifests/init.pp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **class** php {  **package** { 'php':  name: $phpname,  ensure: **present**,  }  **package** { 'php-pear':  ensure: **present**,  }  } |

1. Add include php to the hosts in your sites.pp file:

**/etc/puppetlabs/code/environments/production/manifests/site.pp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **node** 'ubupuppet.members.linode.com' {  $adminemail = 'webmaster@example.com'  $servername = 'hostname.example.com'  **include** apache  **include** apache::vhosts  **include** mysql::database  **include** php  }  **node** 'centospuppet.members.linode.com' {  $adminemail = 'webmaster@example.com'  $servername = 'hostname.example.com'  **include** apache  **include** apache::vhosts  **include** mysql::database  **include** php  } |

1. Run the following command on your agent nodes to pull in any changes to your servers.

sudo /opt/puppetlabs/bin/puppet agent -t