we'll learn how to template configuration files out with Puppet, filling in variables with the managed node's facts.

Often we may want to maintain configuration files for applications that are different between servers. If we have a couple of configurations, it's not touch to maintain multiple files, but what if we have a very large number of differing configurations? We can manage this situation by writing ERB templates and populating the templates with node-specific information. This can be done in Puppet with the **template()** function:

Puppet supports templates written in the **ERB** templating and it can be used to specify the contents of files.

Now we want to manage many different websites, it would quickly become tedious to supply an almost identical virtual host file for each site, altering only the name and domain of the site.

The best way of doing this is to give Puppet master a template file into which it could just insert these variables for each different site. The **template()** function serves just this purpose. Anywhere we have multiple files that differ only slightly, or files that need to contain dynamic information, we just use a template.

template evaluation

Templates are evaluated via a simple function as show in the following example:

$value = template("my\_module/my\_template.erb")

Template files should be stored in the **templates** directory of a Puppet module, which allows the template function to locate them with the simplified path format shown above. For example, the file referenced by **template("my\_module/my\_template.erb")** would be found on disk at **/etc/puppet/modules/my\_module/templates/my\_template.erb**.

Templates are always evaluated **by the parser**, **not by the client**. This means that if we're using a puppet master server, then the templates only need to be on the server, and we never need to download them to the client. The client sees no difference between using a template and specifying all of the text of the file as a string. - [Docs: Using Puppet Templates](https://docs.puppetlabs.com/guides/templating.html)

conf file - deploying a virtual host

Suppose new we want to build three new sites: test1.com, test2.com, and test3.com. To prepare for this, we need to change the Puppet config for test.com to use a template so that we can later use the same template for the new sites.

1. Here is our new **modules/nginx/manifests/init.pp** file:
2. class nginx {
3. package { 'nginx':
4. ensure => installed,
5. }
6. service { 'nginx':
7. ensure => running,
8. enable => true,
9. require => Package['nginx'],
10. }
11. file { '/etc/nginx/sites-enabled/default':
12. ensure => 'absent',
13. }
14. }

Since we previously used the file **/etc/nginx/sites-enabled/ default** as the virtual host for test.com ([Puppet packages, services, and files II with nginx](http://www.bogotobogo.com/DevOps/Puppet/puppet_packages_files_services2_nginx.php)), we need to remove that now:

file { '/etc/nginx/sites-enabled/default':

ensure => absent,

}

1. We want to reate a new templates directory in the nginx module:
2. ubuntu@ip-172-31-45-62:/etc/puppet$ sudo mkdir -p modules/nginx/templates
3. Create the file **modules/nginx/templates/vhost.conf.erb** with the following contents. The template file is for the virtual host definition:
4. server {
5. listen 80;
6. root /var/www/<%= @site\_name %>;
7. server\_name <%= @site\_domain %>;
8. }

The <%= %> signs mark where parameters will go; we will supply **site\_name** and**site\_domain** later, when we use the template. Puppet will replace <%= @site\_name %> with the value of the site\_name variable.

1. Then in the **site.pp** file, we include the nginx module on the node:
2. node 'puppet-agent' {
3. include nginx
4. $site\_name = 'test'
5. $site\_domain = 'test.com'
6. file { '/etc/nginx/sites-enabled/test.conf':
7. content => template('nginx/vhost.conf.erb'),
8. notify => Service['nginx'],
9. }
10. ...
11. }

Note that before using the template, we need to set values for the variables **site\_name**and **site\_domain**:

$site\_name = 'test'

$site\_domain = 'test.com'

Also note that when we refer to these variables in Puppet code, we use a **$ prefix ($site\_name)**, but in the template it's an **@ prefix (@site\_name)**. This is because in templates we're actually writing Ruby, not Puppet!

Now we can use the template to generate the nginx virtual host file:

file { '/etc/nginx/sites-enabled/test.conf':

content => template('nginx/vhost.conf.erb'),

notify => Service['nginx'],

}

This looks just like any other file resource, with a content attribute. Though we can give the contents of the file as a literal string:

content => "Hello, world\n",

However, here we call the template function:

content => template('nginx/vhost.conf.erb'),

The argument to template tells Puppet where to find the template file. The path

nginx/vhost.conf.erb

Translates to

modules/nginx/templates/vhost.conf.erb

1. Run puppet:
2. ubuntu@ip-172-31-45-62:/etc/puppet$ sudo puppet apply modules/nginx/manifests/init.pp
3. Notice: Compiled catalog for ip-172-31-45-62.ec2.internal in environment production in 0.02 seconds
4. Notice: Finished catalog run in 0.02 seconds
5. Check the resulting virtual host file on agent node:
6. ubuntu@puppet-agent:~$ cat /etc/nginx/sites-enabled/test.conf
7. server {
8. listen 80;
9. root /var/www/test;
10. server\_name test.com;
11. }

Puppet now evaluated the template, inserted the values of any variables referenced in**<%= %>** signs in **modules/nginx/templates/vhost.conf.erb**, and generated the final output as we can see in the **test.conf**.