



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Mastering Puppet

Pull the strings of Puppet to configure enterprise-grade environments for performance optimization

Thomas Uphill

[PACKT] open source*
PUBLISHING community experience distilled

Mastering Puppet

Pull the strings of Puppet to configure enterprise-grade environments for performance optimization

Thomas Uphill



BIRMINGHAM - MUMBAI

Mastering Puppet

Copyright © 2014 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: July 2014

Production reference: 1090714

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78398-218-9

www.packtpub.com

Cover image by Gagandeep Sharma (er.gagansharma@gmail.com)

Credits

Author

Thomas Uphill

Project Coordinator

Danuta Jones

Reviewers

Ugo Bellavance

C. N. A. Corrêa

Jeroen Hooyberghs

Johan De Wit

Proofreaders

Faye Coulman

Maria Gould

Indexers

Mariammal Chettiyar

Tejal Soni

Priya Subramani

Commissioning Editor

Edward Gordon

Acquisition Editor

Meeta Rajani

Graphics

Sheetal Aute

Ronak Dhruv

Content Development Editor

Sharvari Tawde

Production Coordinator

Shantanu Zagade

Technical Editors

Veena Pagare

Anand Singh

Cover Work

Shantanu Zagade

Copy Editors

Sarang Chari

Mradula Hegde

About the Author

Thomas Uphill is an RHCA who has been using Puppet since version 0.24. He has been a system administrator for nearly 20 years, more than 10 of which have been with Red Hat Linux and its derivatives. He has presented tutorials on Puppet at LOPSA-East and has spoken at PuppetConf 2013. He enjoys teaching others how to use Puppet to automate as much system administration tasks as possible. When he's not at the Seattle Puppet Meetup, you can find him at <http://ramblings.narrabilis.com>.

I am very thankful to my friend and colleague Joško Plazonić for introducing me to Puppet and getting me started on this journey. I would like to thank my wife Priya Fernandes for putting up with the long nights and weekends it took to finish this book. Thanks to Nate Tade for his encouragement while I worked on this book, the rest of my team for trying my crazy ideas, and Shawn Foley for a few not-so-crazy ideas. Thanks to Theresa, David, and Ben for their support.

About the Reviewers

Ugo Bellavance has done most of his studies in e-commerce. He started using Linux from RedHat 5.2, got Linux training from Savoir-faire Linux at age 20, and got his RHCE on RHEL 6 in 2011. He's been a consultant in the past, but he's now an employee for a provincial government agency for which he manages the IT infrastructure (servers, workstations, network, security, virtualization, SAN/NAS, and PBX). He's a big fan of open source software and its underlying philosophy. He has worked with Debian, Ubuntu, and SUSE, but what he knows best is RHEL-based distributions. He's known for his contributions to the MailScanner project (he has been a technical reviewer for *MailScanner User Guide and Training Manual*, Julian Field), but he has also given time to different open source projects such as Mondo Rescue, OTRS, SpamAssassin, pfSense, and a few others. He's been a technical reviewer for *Centos 6 Linux Server Cookbook*, Jonathan Hobson, Packt Publishing and *Puppet 3 Beginner's Guide*, John Arundel, Packt Publishing.

I thank my lover, Lysanne, who accepted to allow me some free time slots for this review even with two dynamic children to take care of. The presence of these three human beings in my life is simply invaluable.

I must also thank my friend Sébastien, whose generosity is only matched by his knowledge and kindness. I would never have reached this high in my career if it wasn't for him.

C. N. A. Corrêa (@cnacorreia) is an IT operations manager and consultant. He is also a Puppet enthusiast and an old-school Linux hacker. He has a master's degree in Systems Virtualization and holds the CISSP and RHCE certifications. Backed by a 15-year career on systems administration, Carlos leads IT operations teams for companies in Brazil, Africa, and the USA. He is also a part-time professor for graduate and undergraduate courses in Brazil. Carlos co-authored several research papers on network virtualization and OpenFlow, presented on peer-reviewed IEEE and ACM conferences worldwide.

I thank God for all the opportunities of hard work and all the lovely people I always find on my way. I thank the sweetest of them all, my wife Nanda, for all her loving care and support that pushes me forward. I would also like to thank my parents, Nilton and Zélia, for being such a big inspiration for all the things I do.

Jeroen Hooyberghs has eight years of professional experience in many different Linux environments. Currently, he's employed as an Open Source and Linux Consultant at Open-Future in Belgium. Since the past year, a lot of his time has been going into implementing and maintaining Puppet installations for clients.

I would like to thank my two girls, Eveline and Tess, for understanding that a passion for open source requires evenings and weekends spent on it.

Johan De Wit was an early Linux user, and he still remembers the day he built a 0.9x Linux kernel on his brand new 486 computer that took an entire night. His love for the UNIX operating systems existed before Linux was announced. It is not surprising that he started a career as a UNIX system administrator.

He doesn't remember precisely when he started working with open source software, but since 2009, he is working as an Open Source Consultant at Open-Future, where he got the opportunity to work with Puppet. Right now, Puppet has become Johan's biggest interest. He also loves to teach Puppet as one of the few official Puppet trainers in Belgium.

Johan started the Belgian Puppet User Group a year ago, where he tries to bring some Puppeteers together having great and interesting meetups. When he takes time writing some Puppet-related blogs, he mostly does that at <http://puppet-be.github.io/>, the BPUG website. Also, from time to time, he tries to spread some hopefully wise Puppet words by presenting talks at Puppet camps across in Europe.

Besides having fun at work, he spends a lot of his free time with his two lovely kids, his two Belgian draft horses, and if time and the weather permits, he likes to (re)build and drive his old-school chopper.

www.PacktPub.com

Support files, eBooks, discount offers, and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	1
Chapter 1: Dealing with Load/Scale	7
Divide and conquer	7
Puppet with passenger	8
Splitting up the workload	15
Certificate signing	15
Reporting	15
Storeconfigs	15
Catalog compilation	16
Keeping the code consistent	22
One more split	23
One last split or maybe a few more	27
Conquer by dividing	28
Creating an rpm	30
Creating the YUM repository	33
Summary	35
Chapter 2: Organizing Your Nodes and Data	37
Getting started	37
Organizing the nodes with ENC	38
A simple example	39
Hostname strategy	42
Modified ENC using hostname strategy	43
LDAP backend	47
OpenLDAP configuration	47
Hiera	53
Configuring hiera	53
Using hiera_include	56
Summary	62

Chapter 3: Git and Environments	63
Environments	63
Environments and hiera	66
Multiple hierarchies	67
Single hierarchy for all environments	68
Dynamic environments	70
Git	71
Why Git?	72
A simple Git workflow	73
Git Hooks	81
Using post-receive to set up environments	82
Puppet-sync	85
Playing nice with other developers	86
Not playing nice with others	89
Git for everyone	93
Summary	95
Chapter 4: Public Modules	97
Getting modules	97
Using GitHub for public modules	97
Modules from the Forge	101
Using librarian	102
Using r10k	105
Using modules	110
concat	110
inifile	114
firewall	119
lvm	123
stdlib	127
Summary	130
Chapter 5: Custom Facts and Modules	131
Module manifest files	132
Module files and templates	135
Naming a module	136
Creating modules with a Puppet module	137
Comments in modules	137
Multiple definitions	139
Custom facts	142
Creating custom facts	143
Creating a custom fact for use in hiera	150
Summary	153

Chapter 6: Custom Types	155
Parameterized classes	155
Defined types	156
Types and providers	167
Creating a new type	168
Summary	174
Chapter 7: Reporting and Orchestration	175
Turning on reporting	175
Syslog	176
Store	177
IRC	177
Foreman	181
Installing Foreman	182
Attaching Foreman to Puppet	182
Using Foreman	185
Puppet Dashboard	187
Using passenger with Dashboard	189
Linking Dashboard to Puppet	192
Processing reports	193
mcollective	194
Installing activemq	195
Configuring nodes to use activemq	198
Connecting a client to activemq	201
Using mcollective	203
Summary	205
Chapter 8: Exported Resources	207
Configuring puppetdb – using the forge module	208
Manually installing puppetdb	210
Installing Puppet and puppetdb	211
Installing and configuring PostgreSQL	211
Configuring puppetdb to use PostgreSQL	213
Configuring Puppet to use puppetdb	214
Exported resource concepts	215
Declaring exported resources	215
Collecting exported resources	216
Simple example: a host entry	216
Resource tags	218
Exported SSH keys	219
sshkey collection for laptops	220
Putting it all together	222
Summary	231

Chapter 9: Roles and Profiles	233
Design pattern	233
Creating an example CDN role	234
Creating a sub-CDN role	238
Dealing with exceptions	240
Summary	241
Chapter 10: Troubleshooting	243
Connectivity issues	243
Catalog failures	247
Full trace of a catalog compile	251
The classes.txt file	252
Debugging	253
Personal and bugfix branches	254
Echo statements	255
Scope	255
Profiling and summarizing	257
Summary	258
Index	259

Preface

Every project changes when you scale it out. Puppet is no different. Working on a small number of nodes with a small team of developers is a completely different task than working with thousands of nodes with a large group of developers.

Mastering Puppet deals with the issues faced with larger deployments, such as scaling and duplicate resource definitions. It will show you how to fit Puppet into your organization and keep everyone working. The concepts presented can be adopted to suit organizations of any size.

What this book covers

Chapter 1, Dealing with Load/Scale, deals with scaling out your Puppet infrastructure to handle a large number of nodes. Using proxying techniques, a sample deployment is presented.

Chapter 2, Organizing Your Nodes and Data, is where we examine different methods of applying modules to nodes. In addition to ENCs (external node classifiers), we use `hiera` and `hiera_include` to apply modules to nodes.

Chapter 3, Git and Environments, shows you how to use Git hooks to deploy your code to your Puppet masters and enforce access control for your modules.

Chapter 4, Public Modules, presents several supported modules from the Puppet Forge and has real-world example use cases.

Chapter 5, Custom Facts and Modules, is all about extending `facter` with custom facts and rolling your own modules to solve problems.

Chapter 6, Custom Types, covers how to implement defined types and create your own custom types where appropriate.

Chapter 7, Reporting and Orchestration, says that without reporting you'll never know when everything is broken. We explore two popular options for reporting, Foreman and Puppet Dashboard. We then configure and use the marionette collective (mcollective or mco) to perform orchestration tasks.

Chapter 8, Exported Resources, is an advanced topic where we have resource definitions on one node applying to another node. We start by configuring puppetdb and more onto real-world exported resources examples with Forge modules.

Chapter 9, Roles and Profiles, is a popular design paradigm used by many large installations. We show how this design can be implemented using all of the knowledge from the previous chapters.

Chapter 10, Troubleshooting, is a necessity. Things will always break, and we will always need to fix them. This chapter shows some common techniques for troubleshooting.

What you need for this book

All the examples in this book were written and tested using an Enterprise Linux 6.5 derived installation such as CentOS 6.5, Scientific Linux 6.5, or Springdale Linux 6.5. Additional repositories used were EPEL (Extra Packages for Enterprise Linux), the Software Collections (SCL) Repository, the Foreman repository, and Puppet Labs repository. The version of Puppet used was the latest 3.4 series at the time of writing.

Who this book is for

This book is for system administrators and Puppeteers writing Puppet code in an enterprise setting. Puppet masters will appreciate the scaling and troubleshooting chapters and Puppet implementers will find useful tips in the customization chapters.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Puppet code words in text, module names, folder names, filenames, dummy URLs, and user input are shown as follows: "The file `/var/lib/puppet/classes.txt` contains a list of the classes applied to the machine."

A block of code is set as follows:

```
class base {  
  file {'one':  
    path => '/tmp/one',  
    ensure => 'directory',  
  }  
  file {"two":  
    path => "/tmp/one$one",  
    ensure => 'file',  
  }  
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
service {'nginx':  
  require => Package['nginx'],  
  ensure => true,  
  enable => true,  
}
```

Any command-line input or output is written as follows:

```
$ mco ping  
worker1.example.com           time=86.03 ms  
node2.example.com             time=96.21 ms  
node1.example.com             time=97.64 ms  
---- ping statistics ----  
3 replies max: 97.64 min: 86.03 avg: 93.29
```

New **terms** and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Then navigate to the settings section and update the **trusted_puppetmaster_hosts** setting."



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata is verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Dealing with Load/Scale

A large deployment will have a large number of nodes. If you are growing your installation from scratch, you may have started with a single Puppet master running the built-in WEBrick server and moved up to a passenger installation. At a certain point in your deployment, a single Puppet master just won't cut it – the load will become too great. In my experience, this limit was around 600 nodes. Puppet agent runs begin to fail on the nodes, and catalogs fail to compile. There are two ways to deal with this problem: divide and conquer or conquer by dividing.

That is, we can either split up our Puppet master and divide the workload among several machines or we can make each of our nodes apply our code directly using Puppet agent (this is known as a masterless configuration). We'll examine each of these solutions separately.

Divide and conquer

When you start to think about dividing up your Puppet server, the main thing to realize is that many parts of Puppet are simply HTTP SSL transactions. If you treat those things as you would a web service, you can scale out to any size required using HTTP load balancing techniques.

The first step in splitting up the Puppet master is to configure the Puppet master to run under passenger. To ensure we all have the same infrastructure, we'll install a stock passenger configuration together and then start tweaking the configuration. We'll begin building on an x86_64 Enterprise 6 rpm-based Linux; the examples in this book were built using CentOS 6.5 and Springdale Linux 6.5 distributions. Once we have passenger running, we'll look at splitting up the workload.

Puppet with passenger

In our example installation, we will be using the name `puppet.example.com` for our Puppet server. Starting with a server installation of Enterprise Linux version 6, we install `httpd` and `mod_ssl` using the following code:

```
# yum install httpd mod_ssl
```

Installed:

```
httpd-2.2.15-29.el6_4.x86_64
mod_ssl-2.2.15-29.el6_4.x86_64
```



Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.packtpub.com>. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.



In each example, I will install the latest available version for Enterprise Linux 6.5 and display the version for the package requested (some packages may pull in dependencies – those versions are not shown).

To install `mod_passenger`, we pull in the **Extra Packages for Enterprise Linux (EPEL)** repository available at <https://fedoraproject.org/wiki/EPEL>. Install the EPEL repository by downloading the rpm file from http://download.fedoraproject.org/pub/epel/6/x86_64/repoview/epel-release.html or use the following code:

```
# yum install http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Installed:

```
epel-release-6-8.noarch
```

Once EPEL is installed, we install `mod_passenger` from that repository using the following code:

```
# yum install mod_passenger
```

Installed:

```
mod_passenger-3.0.21-5.el6.x86_64
```

Next, we will pull in Puppet from the puppetlabs repository available at http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html#for-red-hat-enterprise-linux-and-derivatives using the following code:

```
# yum install http://yum.puppetlabs.com/el/6/products/x86_64/puppetlabs-release-6-7.noarch.rpm
```

Installed:

```
puppetlabs-release-6-7.noarch
```

With the puppetlabs repository installed, we can then install Puppet using the following command:

```
# yum install puppet
```

Installed:

```
puppet-3.3.2-1.el6.noarch
```

The Puppet rpm will create the `/etc/puppet` and `/var/lib/puppet` directories. In `/etc/puppet`, there will be a template `puppet.conf`; we begin by editing that file to set the name of our Puppet server (`puppet.example.com`) in the `certname` setting using the following code:

```
[main]
logdir = /var/log/puppet
rundir = /var/run/puppet
vardir = /var/lib/puppet
ssldir = $vardir/ssl
certname = puppet.example.com
[agent]
server = puppet.example.com
classfile = $vardir/classes.txt
localconfig = $vardir/localconfig
```

The other lines in this file are defaults. At this point, we would expect `puppet.example.com` to be resolved with a DNS query correctly, but if you do not control DNS at your organization or cannot have this name resolved properly at this point, edit `/etc/hosts`, and put in an entry for your host pointing to `puppet.example.com`. In all the examples, you would substitute `example.com` for your own domain name.

```
127.0.0.1    localhost localhost.localdomain puppet
            puppet.example.com
```

We now need to create certificates for our master; to ensure the Certificate Authority (CA) certificates are created, run Puppet cert list using the following command:

```
# puppet cert list
```

Notice: Signed certificate request for ca

In your enterprise, you may have to answer requests from multiple DNS names, for example, `puppet.example.com`, `puppet`, and `puppet.devel.example.com`. To make sure our certificate is valid for all those DNS names, we will pass the `dns-alt-names` option to `puppet certificate generate`; we also need to specify that the certificates are to be signed by the local machine using the following command:


```
puppet# puppet certificate generate --ca-location local --dns-alt-names
puppet,puppet.prod.example.com,puppet.dev.example.com puppet.example.com
Notice: puppet.example.com has a waiting certificate request
true
```

Now, to sign the certificate request, first verify the certificate list using the following commands:

```
puppet# puppet cert list

"puppet.example.com" (SHA256) E5:F7:26:0A:6C:41:26:FA:80:02:E5:A6:A1
:DB:F4:E0:9D:9C:5B:2D:A5:BF:EC:D1:FA:84:51:F4:8C:FD:9B:AF (alt names:
"DNS:puppet", "DNS:puppet.dev.example.com", "DNS:puppet.example.com",
"DNS:puppet.prod.example.com")

puppet# puppet cert sign puppet.example.com
Notice: Signed certificate request for puppet.example.com
Notice: Removing file Puppet::SSL::CertificateRequest puppet.example.com
at '/var/lib/puppet/ssl/ca/requests/puppet.example.com.pem'
```

 We specified the `ssldir` directive in our configuration. To interactively determine where the certificates will be stored using the following command line:

```
$ puppet config print ssldir
```

One last task is to copy the certificate that you just signed into `certs` by navigating to `/var/lib/puppet/ssl/certs`. You can use Puppet `certificate find` to do this using the following command:

```
# puppet certificate find puppet.example.com --ca-location local
-----BEGIN CERTIFICATE-----
MIIF1TCCA72gAwIBAgIBAjANBgkqhkiG9w0BAQsFADAoMSYwJAYDVQQDDDB1QdXBw
...
-----END CERTIFICATE-----
```

When you install Puppet from the `puppetlabs` repository, the `rpm` will create an Apache configuration file called `apache2.conf`. Locate this file and copy it into your Apache configuration directory using the following command:

```
# cp /usr/share/puppet/ext/rack/example-passenger-vhost.conf /etc/httpd/
conf.d/puppet.conf
```

We will now show the Apache config file and point out the important settings using the following configuration:

```

PassengerHighPerformance on
PassengerMaxPoolSize 12
PassengerPoolIdleTime 1500
# PassengerMaxRequests 1000
PassengerStatThrottleRate 120
RackAutoDetect Off
RailsAutoDetect Off

```

The preceding lines of code configure passenger for performance. `PassengerHighPerformance` turns off some compatibility that isn't required. The other options are tuning parameters. For more information on these settings, see <http://www.modrails.com/documentation/Users%20guide%20Apache.html>.

Next we will need to modify the file to ensure it points to the newly created certificates. We will need to edit the lines for `SSLCertificateFile` and `SSLCertificateKeyFile`. The other SSL file settings should point to the correct certificate, chain, and revocation list files as shown in the following code:

```

Listen 8140
<VirtualHost *:8140>
    ServerName puppet.example.com
    SSLEngine on
    SSLProtocol -ALL +SSLv3 +TLSv1
    SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:-LOW:-SSLv2:-EXP

    SSLCertificateFile /var/lib/puppet/ssl/certs/puppet.example.com.pem
    SSLCertificateKeyFile /var/lib/puppet/ssl/private_keys/puppet.
example.com.pem
    SSLCertificateChainFile /var/lib/puppet/ssl/ca/ca.crt.pem
    SSLCACertificateFile /var/lib/puppet/ssl/ca/ca.crt.pem
    # If Apache complains about invalid signatures on the CRL, you can
try disabling
    # CRL checking by commenting the next line, but this is not
recommended.
    SSLCARevocationFile /var/lib/puppet/ssl/ca/ca.crl.pem
    SSLVerifyClient optional
    SSLVerifyDepth 1
    # The `ExportCertData` option is needed for agent certificate
expiration warnings
    SSLOptions +StdEnvVars +ExportCertData
    RequestHeader set X-SSL-Subject %{SSL_CLIENT_S_DN}e
    RequestHeader set X-Client-DN %{SSL_CLIENT_S_DN}e
    RequestHeader set X-Client-Verify %{SSL_CLIENT_VERIFY}e

    DocumentRoot /etc/puppet/rack/public/

```



```
RackBaseURI /
<Directory /etc/puppet/rack/>
  Options None
  AllowOverride None
  Order allow,deny
  allow from all
</Directory>
</VirtualHost>
```

In this VirtualHost we listen on 8140 and configure the SSL certificates in the SSL lines. The RequestHeader lines are used to pass certificate information to the Puppet process spawned by passenger. The DocumentRoot and RackBaseURI settings are used to tell passenger where to find its configuration file `config.ru`. We create `/etc/puppet/rack` and its subdirectories and then copy the example `config.ru` into that directory using the following commands:

```
# mkdir -p /etc/puppet/rack/{public,tmp}
# cp /usr/share/puppet/ext/rack/files/config.ru /etc/puppet/rack
# chown puppet:puppet /etc/puppet/rack/config.ru
```

We change the owner of `config.ru` to `puppet:puppet` as the passenger process will run as the owner of `config.ru`. Our `config.ru` will contain the following code:

```
$0 = "master"

# if you want debugging:
# ARGV << "--debug"

ARGV << "--rack"
ARGV << "--confdir" << "/etc/puppet"
ARGV << "--vardir" << "/var/lib/puppet"

require 'puppet/util/command_line'
run Puppet::Util::CommandLine.new.execute
```



In this example, we have used the repository rpms supplied by Puppet and EPEL. In a production installation, you would use `reposync` to copy these repositories locally so that your Puppet machines do not need to access the Internet directly.

The `config.ru` file sets the command-line arguments for Puppet. The ARGV lines are used to set additional parameters to the puppet process. As noted in the Puppet master main page, any valid configuration parameter from `puppet.conf` can be specified as an argument here. Only the options that affect where Puppet will look for files should be specified here. Once puppet knows where to find `puppet.conf`, adding arguments here could be confusing.

With this configuration in place, we are ready to start Apache as our Puppet master. Simply start Apache with a service `httpd` start.

SELinux

Security Enhanced Linux (SELinux) is a system for Linux that provides support for **mandatory access controls (MAC)**. If your servers are running with SELinux enabled, great! You will need to make some policy changes to allow Puppet to work within passenger. The easiest way to build up your policy is to use *audit2allow*, which is provided in `policycoreutils-python`. Rotate the audit logs to get a clean log file, and then start a Puppet run. After the Puppet run, get *audit2allow* to build a policy module for you and insert it. Then turn SELinux back on. Refer to https://bugzilla.redhat.com/show_bug.cgi?id=1051461 for more information.



```
# setenforce 0
# service auditd rotate
# service httpd restart
(start a puppet run remotely)
# audit2allow -i /var/log/audit/audit.log -M puppet_
passenger
# semodule -i puppet_passenger.pp
# setenforce 1
```

If necessary, repeat the process until everything runs cleanly. `semodule` will sometimes suggest enabling the `allow_yppbind` Boolean; this is a very bad idea. The `allow_yppbind` Boolean allows so many things that it is almost as bad as turning SELinux off.

Now that Puppet is running, you'll need to open the local firewall (`iptables`) on port 8140 to allow your nodes to connect. Then you'll need an example `site.pp` to get started. For testing we will create a basic `site.pp` that defines a default node with a single class attached to the default node as shown in the following code:

```
node default {
  include example
}

class example {
  notify {"This is an example": }
}
```

You can start a practice node or two and run their agent against the Puppet server either using `--server puppet.example.com` or editing the agents `puppet.conf` file to point at your server. Agents will by default look for an unqualified host called Puppet. Then search based on your DNS configuration (search in `/etc/resolv.conf`), and if you do not control DNS, you may have to edit the local `/etc/hosts` file to specify the IP address of your Puppet master. A sample run, for a node called `node1`, should look something like the following commands:

```
[root@node1 ~]# puppet agent -t
Info: Creating a new SSL key for node1
Info: Caching certificate for ca
Info: Creating a new SSL certificate request for node1
Info: Certificate Request fingerprint (SHA256): C4:0D:7A:54:ED:C8:E8:CC:6
8:D0:A6:13:C4:91:28:3D:B1:66:71:48:57:85:D8:99:AF:D0:81:54:B9:64:AB:F2
Exiting; no certificate found and waitforcert is disabled
```

Sign the certificate on the Puppet master and run again; the run should look like the following commands:

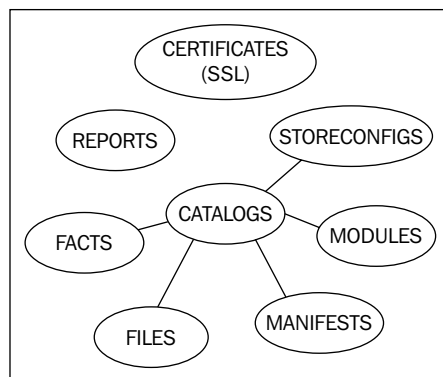
```
[root@puppet ~]# puppet cert sign node1
Notice: Signed certificate request for node1
Notice: Removing file Puppet::SSL::CertificateRequest node1 at '/var/lib/
puppet/ssl/ca/requests/node1.pem'
```

```
[root@node1 ~]# puppet agent -t
Info: Caching certificate for node1
Info: Caching certificate_revocation_list for ca
Info: Retrieving plugin
Info: Caching catalog for node1
Info: Applying configuration version '1386310193'
Notice: This is an example
Notice: /Stage[main]/Example/Notify[This is an example]/message: defined
'message' as 'This is an example'
Notice: Finished catalog run in 0.03 seconds
```

You now have a working passenger configuration. This configuration can handle a much larger load than the default WEBrick server provided with puppet. Puppet Labs suggests the WEBrick server is appropriate for *small* installations; in my experience that number is much less than 100 nodes, maybe even less than 50. You can tune the passenger configuration and handle a large number of nodes, but to handle a very large installation (1000s of nodes), you'll need to start splitting up the workload.

Splitting up the workload

Puppet is a web service. But there are several different components supporting that web service, as shown in the following diagram:



Each of the different components in your Puppet infrastructure: SSL CA, Reporting, Storeconfigs, and Catalog compilation can be split up into their own server or servers.

Certificate signing

Unless you are having issues with certificate signing consuming too many resources, it's simpler to keep the signing machine a single instance, possibly with a hot spare. Having multiple certificate signing machines means that you have to keep certificate revocation lists synchronized.

Reporting

Reporting should be done on a single instance if possible. Reporting options will be shown in *Chapter 7, Reporting and Orchestration*.

Storeconfigs

Storeconfigs should be run on a single server, storeconfigs allows for exported resources and is optional. The recommended configuration for storeconfigs is puppetdb, which can handle several thousand nodes in a single installation.