Red Hat Satellite 5 and 6 Puppet Guide

Using Puppet with Red Hat Satellite 5 and 6
Edition 1

Clifford Perry Bryan Kearney

Red Hat Satellite 5 and 6 Puppet Guide

Using Puppet with Red Hat Satellite 5 and 6
Edition 1

Clifford Perry
Red Hat Engineering
cperry@redhat.com

Bryan Kearney
Red Hat Engineering
bkearney@redhat.com

Abstract

A technical white paper for a successful delivery of Red Hat Satellite and Puppet within your infrastructure

Table of Contents

Chapter 1. Introduction

1.1. Introduction

Within the world of Linux Configuration Management there are many tools and technologies to choose from. One of the most popular vendor neutral choices is Puppet. Red Hat for its next generation of Systems Management products will be providing support for Puppet as a key component for the Red Hat Satellite 6 product. Today many Satellite 5 customers are already using Puppet to coincide with their Satellite 5 deployment or looking to bring Puppet into their environment.

Puppet Support

Puppet is currently not supported by Red Hat with Red Hat Satellite and this white paper is not providing support for Puppet with the current Satellite 5 releases.

1.2. Objective

The objective of this technical guide is to help provide information for Satellite 5 customers to have some recommendations for usage of Puppet and Satellite 5, in a manner which will allow them to move smoothly to the future Satellite 6 product. We will provide technical information and guidance for today and then provide insight on how Satellite 6 will be using Puppet, so that the transition from one to the other in the future is successfully completed. Additionally Satellite 6 will provide a supported Puppet environment for our Red Hat Enterprise Linux (RHEL) customers.

We will start with a detailed description of how to use Satellite 5 with Puppet today - allowing the reader, if already familiar with Puppet and/or Satellite 5, to quickly see how to implement a functioning deployment. We will then review the future Red Hat Satellite 6 products usage of Puppet and this will allow the reader to translate current usage into future usage.

1.3. Plans for Supported Versions

Currently the Satellite 6 product plans to provide Puppet 2.7 (or later) for RHEL 5 managed systems and Puppet 3.6 (or later) for RHEL 6 and RHEL 7 managed systems and the server pieces. These packages will be supported Red Hat versions of the upstream packages. Satellite customers can download community versions of these packages today from either EPEL (`https://fedoraproject.org/wiki/EPEL`) or from the Puppet Labs community site (`http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html`)

Red Hat Satellite 6 will not support integrating with Puppet packages which are not provided with the Satellite subscription.

Chapter 2. Red Hat Satellite 5 and Puppet

2.1. Overview

Objective: This section is aimed at current Satellite 5 customers or those about to deploy Satellite 5 and looking for guidance on how to use Satellite 5 and Puppet side-by-side in a manner that will allow a smooth transition to the future Satellite 6 product. This section is tested by Satellite Engineering using recommended practises. Usage of a modular puppet deployment will allow for smoothest migration and re-use of the modules even if the puppet masters and location providing node definitions changes when going to Satellite 6.

Currently the Satellite 6 product plans to provide Puppet 2.7 for RHEL 5 managed systems and Puppet 3.6 for RHEL 6 & 7 managed systems. This section will initially guide you through downloading community available versions of puppet. When Satellite 6 is released you will be able to upgrade or reinstall the puppet client to the Red Hat supported package versions. Additionally for a fully supported puppet deployment you would reconfigure the puppet client to register and pull content from the Satellite 6 puppet master verses a previously established puppet master environment.

This section is going to guide a customer for creating a new Satellite 5 deployment. If you already using Satellite you will see many familiar aspects of the product and can use this guidance as a basis for any modifications you may choose to implement to achieve similar end results. If you are new to Satellite 5, a lot of our customers have invested time, resources and customizations for their own Standard Operating Environments (SOE) using similar processes as outlined below. We have additional detailed documentation including a guide for using Satellite 5 to deploy SOE’s online.

2.2. Initial Satellite 5 Configuration

RHEL versions

This process assumes that you are using both RHEL 5 and 6 within your environment, but is also applicable to RHEL 7. Please make adjustments as required for your own needs.

Objective: Within this section we are going to outline the basic installation, backing up the database and updating the Satellite software. Additionally we show an example of how to import Red Hat content into Satellite 5. If you already familiar or have a Satellite 5 instance up and running you can skip to Import Puppet Packages.

Follow the Red Hat Satellite 5 Installation documentation and install Satellite 5.

Technical Side Note

It is always recommended to have backups of your Satellite database. We recommend to take an offline/cold backup after initial installation is complete, and then use the online backup capabilities to take updates at various points during this process. This will allow you to restore your database if needed, without losing too much data. Examples such as after syncing content and after cloning channels steps below. Please see the Satellite Install Guide for the section on performing backups. The db-control command is used to perform backups of your database.

Once installed it is recommended to perform a yum update to download any Satellite 5 errata that are available, typically important bugfixes. You can find details for Satellite errata (and all released errata) from our public site `https://rhn.redhat.com/errata`.

Once Installed and updated you will want to import the RHEL content, this is achieved by using the satellite-sync command line tool to import RHEL 5 and 6 content.

Import the RHEL 5 and RHEL 6 x86_64 channels, including the supplementary and optional channels, along with RHN Tools child channel. Example:

```
# satellite-sync --email -crhel-x86_64-server-5 -crhn-tools-rhel-x86_64-server-5 -
crhel-x86_64-server-6 -crhn-tools-rhel-x86_64-server-6 -crhel-x86_64-server-
optional-6 -crhel-x86_64-server-supplementary-5
```

This process can take several hours to complete. Once you have the channels you can move to getting Puppet content downloaded into your Satellite.

2.3. Import Puppet Packages

Objective: Within this section we will walk through the process of importing 3rd party RHEL 5 and 6 puppet RPM's into Satellite 5 and make them available via child channels of the main RHEL channels. We will show how to use 3rd party yum repos, configure them and import into Satellite.

Puppet Package Notes

- As with all 3rd party custom content Red Hat does not support the usage of the content. We do support the usage of Satellite as a means to centrally manage Red Hat, custom content and 3rd party content.
- This document describes how to import and use the Puppet Labs public GPG Key to verify the signature of their signed RPM's. The other main option available is to resign those RPMs with your own company/organizations GPG Key. This will then allow the puppet packages to be deployed and installed with the same GPG Key as used with your own custom RPM's.

Within this guide we will create custom child channels for the Puppet content based off the main RHEL base channels.

- Create Custom Child channel for RHEL 5 and RHEL 6 Puppet packages
  - Login to Satellite 5 WebUI
  - Click through - Channels -> Manage Software Channels -> Create New Channel ->
  - Enter:

```
    Channel Name: Puppet EL6 Server x86_64
    Channel Label: puppet-rhel6-server-x86_64
    Parent Channel: Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
    Architecture: x86_64
    Yum Repository Checksum Type: sha256
    Channel Summary: Public Puppet Packages for RHEL
    GPG key URL: http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
    GPG key ID: 4BD6EC30
    GPG key Fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

- Click onto Create channel

GPG Information

The GPG Key information is not required. Within this context it is informative for the Satellite Administrator as to where the GPG is found upstream.

- Next create a new repo to associate to the channels
- Go to Manage Repositories and Create New Repository
- Enter:

```
Repository Label*: puppet-repo-rhel6-server-x86_64
Repository URL*: http://yum.puppetlabs.com/el/6Server/products/x86_64/
```

- Repeat to create a second repo for dependencies.

```
Repository Label*: puppet-deps-repo-rhel6-server-x86_64
Repository URL*: http://yum.puppetlabs.com/el/6Server/dependencies/x86_64/
```

- Go to Manage Software Channels -> Puppet EL6 Server x86_64 -> Repositories
- Now click and associate the two new repos with this channel.
  - Click onto Sync tab and then hit 'Sync Now'.

Scheduling of external repositories

You can also schedule to have the Puppet Labs repository regularly imported into the Satellite from this page.

It will take a couple of minutes for the import to start and several more before it completes. You can monitor the status as it progresses by reviewing log file:

```
/var/log/rhn/reposync/puppet-rhel6-server-x86_64.log
```

Repeat the process for RHEL 5 Server packages to import from Puppet Labs.

For the EL5 content the details for the Channel are as follows.

```
Channel Name*: Puppet EL5 Server x86_64
Channel Label*: puppet-rhel5-server-x86_64
Parent Channel: Red Hat Enterprise Linux Server (v. 5 for 64-bit x86_64)
Architecture: x86_64
Yum Repository Checksum Type: sha1
Channel Summary*: Public Puppet Packages for RHEL
GPG key URL: http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
GPG key ID: 4BD6EC30
GPG key Fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

For the EL5 repos the details are:

```
    Repository Label*: puppet-repo-rhel5-server-x86_64
    Repository URL*: http://yum.puppetlabs.com/el/5Server/products/x86_64/

    Repository Label*: puppet-deps-repo-rhel5-server-x86_64
    Repository URL*: http://yum.puppetlabs.com/el/5Server/dependencies/x86_64/
```

Within the guide we are going to use mod_passenger to provide a more scalable puppet master server environment. For this we need to download a select few (from 1000's) of packages from the EPEL repo. If you have sufficient disk space you can download the entire repo similar to the Puppet Repo packages and selectively clone the listed packages.

Within Satellite, go to create a new repository and enter:

```
    Repository Label*: epel-repo-rhel6-server-x86_64
    Repository URL*: http://dl.fedoraproject.org/pub/epel/6/x86_64/
```

Next create a Custom child channel off the main RHEL 6 channel to use. On the Create Software Channel page enter:

```
    Channel Name*: EPEL Puppet Master Deps EL6 Server x86_64
    Channel Label*: epel-puppet-rhel6-server-x86_64
    Parent Channel: Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
    Architecture: x86_64
    Yum Repository Checksum Type: sha256
    Channel Summary*: Public EPEL Repo Packages for RHEL
    Channel Description: Selectively sync'd Public EPEL Repo Packages for RHEL -
 containing required dependencies for deploying Puppet Master within Apache Passenger
    GPG key URL: http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-6
```

Once created click onto Repositories and associate the new channel to the previously created EPEL repo.

Do Not Use WebUI To Initiate EPEL Repo Sync

Do not use the WebUI to initiate a sync of the EPEL repo as we will use the command line tool to selectively sync

The packages we want from the EL6 EPEL repo are:

1. mod_passenger
2. libev
3. rubygem-daemon_controller
4. rubygem-fastthread
5. rubygem-passenger
6. rubygem-passenger-native
7. rubygem-passenger-native-libs
8. rubygem-rack

Using the include only option for spacewalk-repo-sync we will specify the packages required. Example is:

```
# spacewalk-repo-sync -c epel-puppet-rhel6-server-x86_64 -i
mod_passenger,libev,rubygem-daemon_controller,rubygem-fastthread,rubygem-
passenger,rubygem-passenger-native,rubygem-passenger-native-libs,rubygem-rack
```

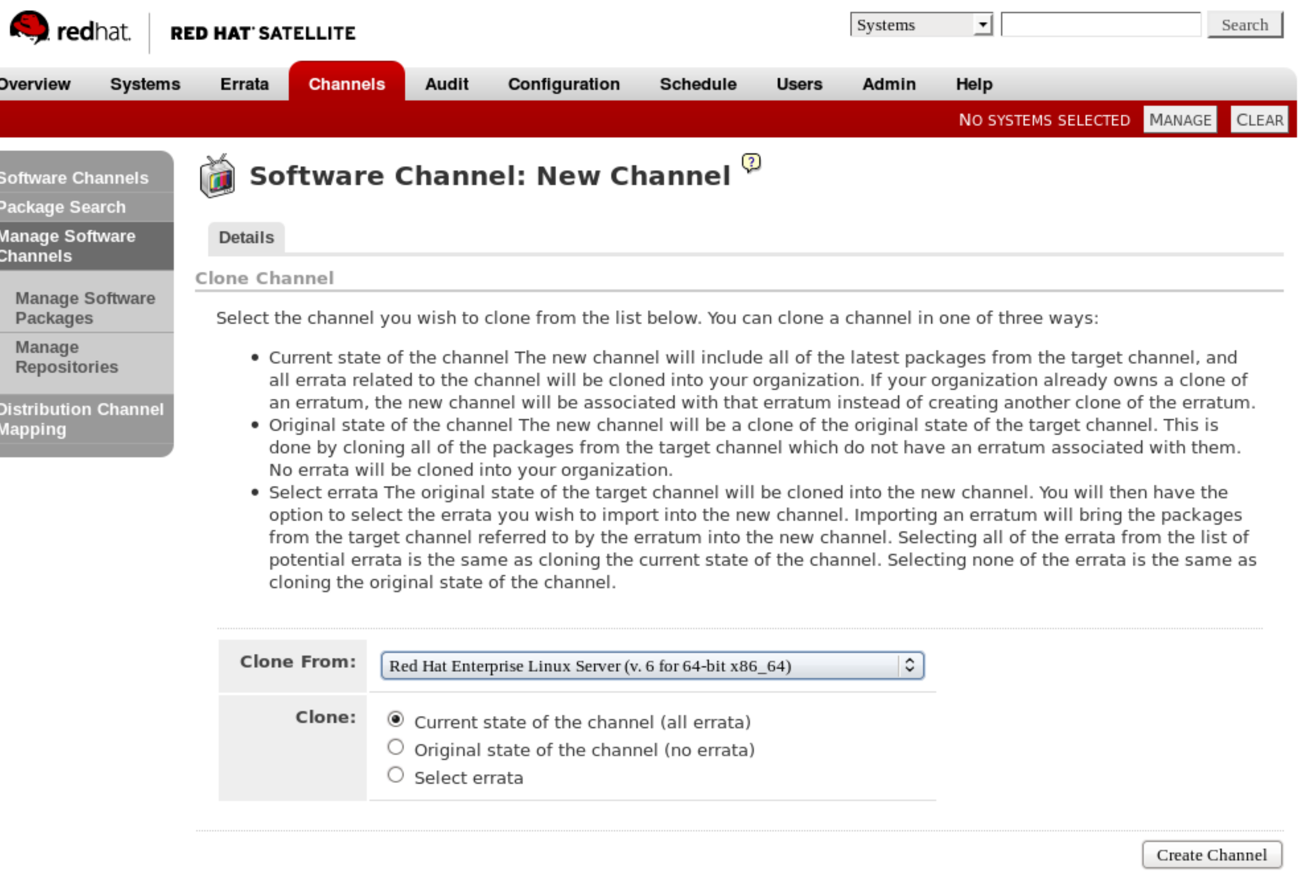
Once this is setup, we are going to use ability of Satellite 5 to clone and lock in content into custom cloned channels, this helps to prevent accidental upgrades or other packages being installed, which was not desired. We will clone en-mass the main RHEL channels, and then selectively clone the content from puppet. We will want the 3.x puppet server pieces to be available to the puppet master serving your environment. We will want the 3.x puppet client packages for the RHEL 6 managed systems, finally, we will want the 2.x puppet client packages available for the RHEL 5 managed systems. At the end of this section your Satellite will be ready to use.

2.4. Clone Channels

Objective: We will go through a standard process used by many customers to create cloned channels of content. Channel cloning is frequently used to provide a set snapshot in time of RPMs which are made available to systems. Often customers will clone another cloned channel, going through a DEVEL -> TEST -> PROD life cycle. Since we are interested in a subset of the 3rd party Puppet repo content we will use channel cloning to copy over only the packages we want. Channel cloning is being used to control what RPMs the managed systems see as available to them.

Within the Satellite browse to Channels -> Manage Software Channels -> Clone Channel. Select the options:

```
    Clone From: Red Hat Enterprise Linux Server (v.6 for 64-bit x86_64)
    Clone: Current state of the channel (all errata)
```



Software Channel: New Channel

Clone Channel

Select the channel you wish to clone from the list below. You can clone a channel in one of three ways:

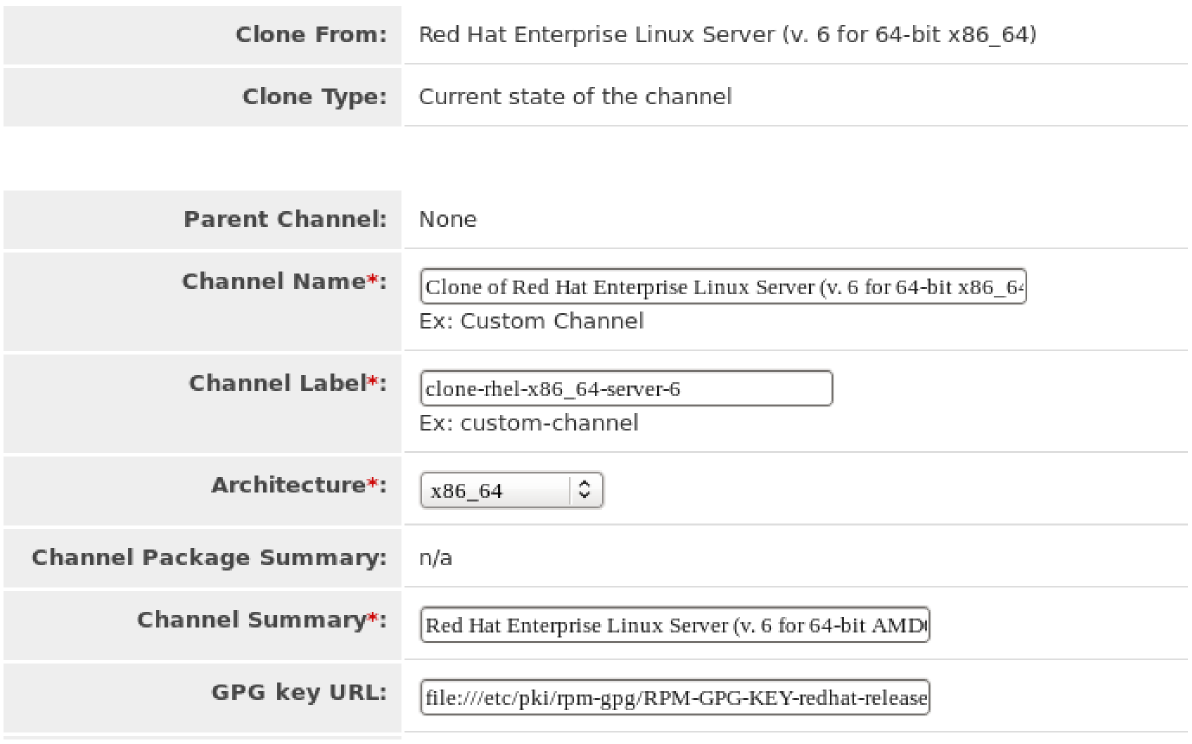
- Current state of the channel The new channel will include all of the latest packages from the target channel, and all errata related to the channel will be cloned into your organization. If your organization already owns a clone of an erratum, the new channel will be associated with that erratum instead of creating another clone of the erratum.
- Original state of the channel The new channel will be a clone of the original state of the target channel. This is done by cloning all of the packages from the target channel which do not have an erratum associated with them. No errata will be cloned into your organization.
- Select errata The original state of the target channel will be cloned into the new channel. You will then have the option to select the errata you wish to import into the new channel. Importing an erratum will bring the packages from the target channel referred to by the erratum into the new channel. Selecting all of the errata from the list of potential errata is the same as cloning the current state of the channel. Selecting none of the errata is the same as cloning the original state of the channel.

Clone From:	Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
Clone:	[✓ Current state of the channel (all errata)] Original state of the channel (no errata) Select errata

Create Channel

Figure 2.1. Satellite 5 Clone Channel

Click the Create Channel button leaving all defaults and click the Create Channel button. This process will take several minutes depending on hardware profile.



Clone From:	Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
Clone Type:	Current state of the channel
Parent Channel:	None
Channel Name*:	Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64 Ex: Custom Channel
Channel Label*:	clone-rhel-x86_64-server-6 Ex: custom-channel
Architecture*:	x86_64
Channel Package Summary:	n/a
Channel Summary*:	Red Hat Enterprise Linux Server (v. 6 for 64-bit AMD
GPG key URL:	file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

Figure 2.2. Satellite 5 Clone Channel Details

Initial Cloning of Errata Slowness

The main RHEL channels have 1000's of packages and errata which are being copied/cloned in the database. If this has never been done before then the first time will be slower still, due to no database statics having been previously gathered for this task. You can run in a terminal the db-control command with the gather-stats option before, during and after to help *prime the pump* as much as possible.

```
# db-control report-stats
Tables with empty statistics: 2
Tables with stale statistics: 392
# db-control gather-stats
Gathering statistics...
WARNING: this may be a very slow process.
```

Repeat the Channel Clone for all Red Hat channels:

- Red Hat Enterprise Linux Server (v.6 for 64-bit x86_64)
  - RHEL Server Optional (v.6 64-bit x86_64)
  - RHN Tools for RHEL (v.6 for 64-bit x86_64)
- Red Hat Enterprise Linux (v.5 for 64-bit x86_64)
  - RHEL Supplementary (v.5 for 64-bit x86_64)

- Red Hat Network Tools for RHEL Server (v.5 64-bit x86_64)

Once completed we will now create three child channels.

1. RHEL 6 child channel for Satellite + Puppet 3.6.x managed systems to subscribe to
2. RHEL 5 child channel for Satellite + Puppet 2.7.x managed systems to subscribe to
3. RHEL 6 child channel for puppet master to subscribe to

At time of writing this document EPEL only provided Puppet 2.6 for EL5 and 2.7 for EL6, while Puppet Labs repos provided the newest Puppet 3.x+ for both EL5 and EL6. As noted earlier it is Red Hat's plan to provide Puppet 2.7 for RHEL 5 and Puppet 3.6 for RHEL 6 and 7. We have imported the Puppet Labs Repo and will now selectively copy/clone the older versions and dependencies we need from repo for our requirements.

2.4.1. RHEL 6 Child Channel for Satellite + Puppet 3.6

Within Satellite click through to create a custom child channel, Channels -> Manage Software Channels -> create new channel

```
     Channel Name*: Custom Clone of Puppet EL6 Server x86_64
     Channel Label*: custom-clone-client-puppet-rhel6-server-x86_64
     Parent Channel: Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit
x86_64)
     Architecture: x86_64
     Yum Repository Checksum Type: sha256
     Channel Summary*: Client Side 3.x Puppet Packages for RHEL
     GPG key URL: http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
     GPG key ID: 4BD6EC30
     GPG key Fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

Click the Create Channel button. Next go to Packages -> Add -> Select:

```
     Channel: Puppet EL6 Server x86_64
```

Hit the View Packages button. Now search and select the *newest* following 7 packages:

1. facter
2. hiera
3. puppet-3.6.x
4. ruby-augeas
5. rubygem-json
6. ruby-rgen
7. ruby-shadow

Click the Confirm Addition button then confirm by clicking the Add Package(s) button.

2.4.2. RHEL 5 Child Channel for Satellite + Puppet 2.7

Repeat the same process as above but with this specific data and packages.

```
    Channel Name*: Custom Clone of Puppet EL5 Server x86_64
    Channel Label*: custom-clone-client-puppet-rhel5-server-x86_64
    Parent Channel: Clone of Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)
    Architecture: x86_64
    Yum Repository Checksum Type: sha1
    Channel Summary*: Client Side 2.x Puppet Packages for RHEL
    GPG key URL: http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
    GPG key ID: 4BD6EC30
    GPG key Fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

```
    Channel: Puppet EL5 Server x86_64
```

Now search and select *newest* following 7 packages:

1. augeas-libs
2. facter
3. puppet-2.7.x
4. ruby
5. ruby-augeas
6. ruby-libs
7. ruby-shadow

2.4.3. RHEL 6 child channel for puppet master

Repeat the same process as above but with this specific data and packages.

```
    Channel Name*: Custom Clone Puppet Master of Puppet EL6 Server x86_64
    Channel Label*: custom-clone-master-puppet-rhel6-server-x86_64
    Parent Channel: Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit
 x86_64)
    Architecture: x86_64
    Yum Repository Checksum Type: sha256
    Channel Summary*: Master 3.x Puppet Packages for RHEL
    GPG key URL: http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
    GPG key ID: 4BD6EC30
    GPG key Fingerprint: 47B3 20EB 4C7C 375A A9DA E1A0 1054 B7A2 4BD6 EC30
```

Click the Create Channel button. Next go to Packages -> Add -> Select:

```
    Channel: Puppet EL6 Server x86_64
```

Hit the View Packages button. Now search and select the *newest* following 8 packages:

1. facter
2. hiera
3. puppet-3.6.x
4. puppet-server-3.6.x
5. ruby-augeas

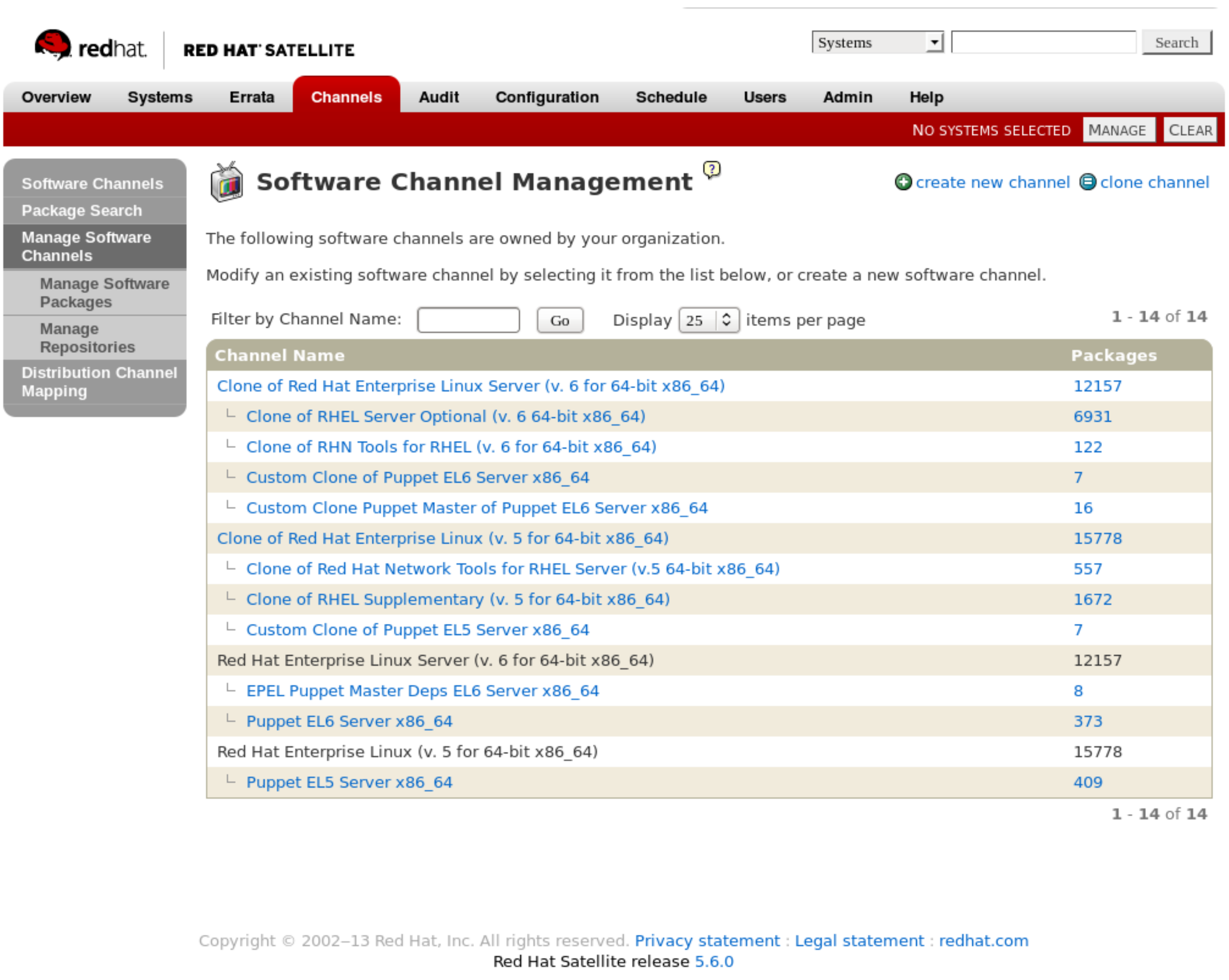
6. rubygem-json
7. ruby-rgen
8. ruby-shadow

Next we will add the mod_passenger and associated packages from EPEL into this channel. Go to Packages -> Add -> Select:

```
    Channel: EPEL Puppet Master Deps EL6 Server x86_64
```

Hit the View Packages button. Now click the Select All button and then confirm the addition of the 8 packages.

At this stage the custom channels will look something like:



Software Channel Management

The following software channels are owned by your organization.

Modify an existing software channel by selecting it from the list below, or create a new software channel.

Filter by Channel Name:

1 - 14 of 14

Channel Name	Packages
Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)	12157
└ Clone of RHEL Server Optional (v. 6 64-bit x86_64)	6931
└ Clone of RHN Tools for RHEL (v. 6 for 64-bit x86_64)	122
└ Custom Clone of Puppet EL6 Server x86_64	7
└ Custom Clone Puppet Master of Puppet EL6 Server x86_64	16
Clone of Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)	15778
└ Clone of Red Hat Network Tools for RHEL Server (v.5 64-bit x86_64)	557
└ Clone of RHEL Supplementary (v. 5 for 64-bit x86_64)	1672
└ Custom Clone of Puppet EL5 Server x86_64	7
Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)	12157
└ EPEL Puppet Master Deps EL6 Server x86_64	8
└ Puppet EL6 Server x86_64	373
Red Hat Enterprise Linux (v. 5 for 64-bit x86_64)	15778
└ Puppet EL5 Server x86_64	409

1 - 14 of 14

Copyright © 2002–13 Red Hat, Inc. All rights reserved. Privacy statement : Legal statement : redhat.com
Red Hat Satellite release 5.6.0

Figure 2.3. Satellite 5 Manage Software Channels Overview

2.5. System Groups

Objective: System Groups are used to help logically group systems with similar traits or commonality. You can then use System Groups to act on one group or do logical Unions and Intersections of different groups. Many customers create System Groups for location, function, OS type, Environment (Dev, Test, Prod) types. We are going to create one to group the systems managed by Puppet by each major RHEL version. This allows the administrator to quickly see which systems are managed by puppet. Satellite allows selecting systems on a per-group basis to then perform management tasks, such as scheduling package installations, reboots or provisioning events.

We are now going to create System Groups within Satellite, these we will create as an example two System Groups:

1. RHEL 6 Puppet
2. RHEL 5 Puppet

Within Satellite, go to Systems -> System Groups -> create new group, Enter:

```
Name*: RHEL 6 Puppet
Description*: RHEL 6 Puppet Managed Systems
```

Click the Create Group button. Repeat for RHEL 5.

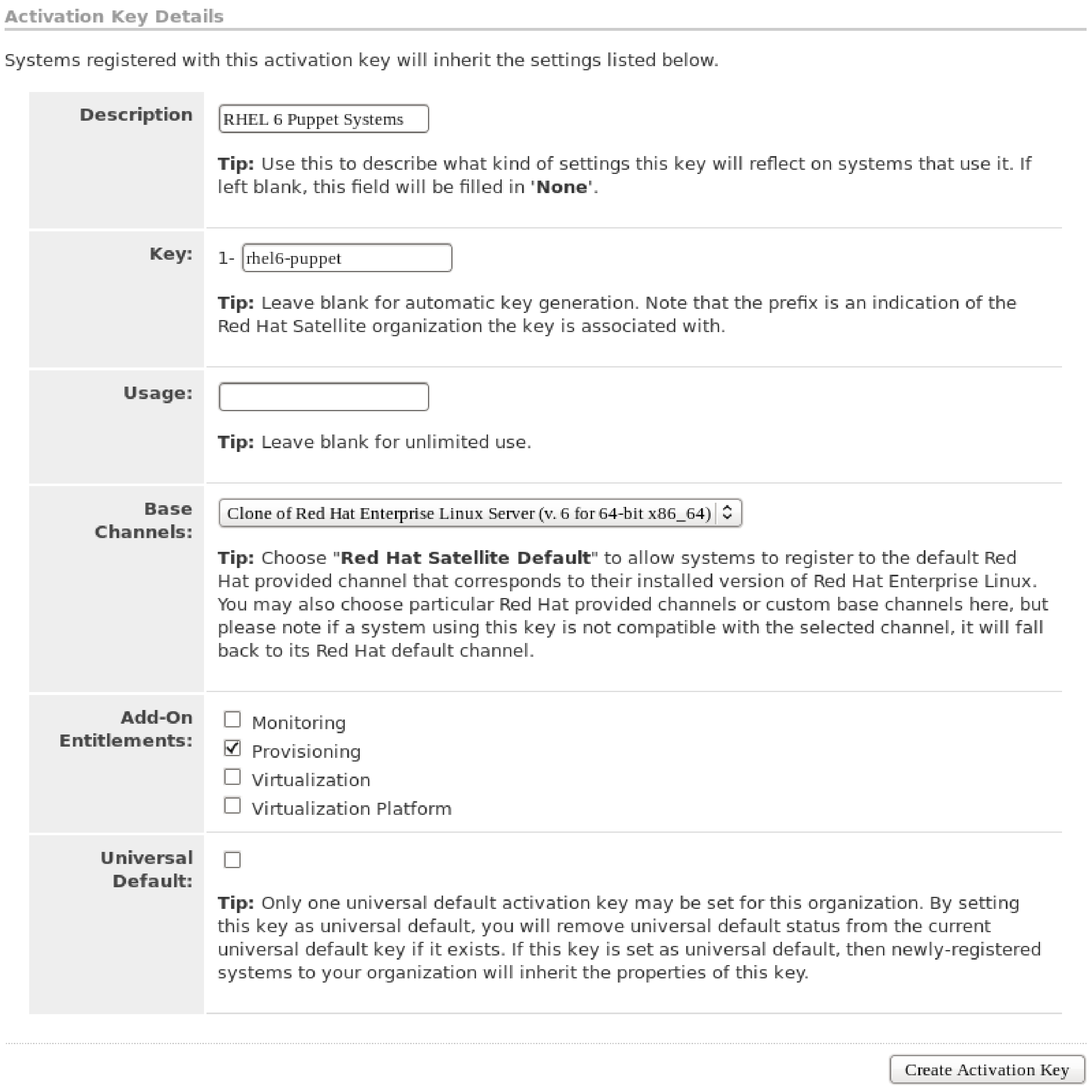
2.6. Activation Keys

Objective: Activation keys are a very powerful component of Satellite 5. They allow you to combine many aspects of Satellite and define what the initial state of a system will be with respect to subscribed channels as well as helping to ensure they have correct packages installed. We are going to create two activation keys one for RHEL 5 based Puppet Managed systems and one for RHEL 6 based.

Within Satellite go to Systems -> Activation Keys -> create new key. Enter options as:

```
Description: RHEL 6 Puppet Systems
Key: rhel6-puppet
Base Channels: Clone of Red Hat Enterprise Linux Server (v.6 for 64-bit x86_64)
Add-On Entitlements: Provisioning
```

and click the Create Activation Key button.



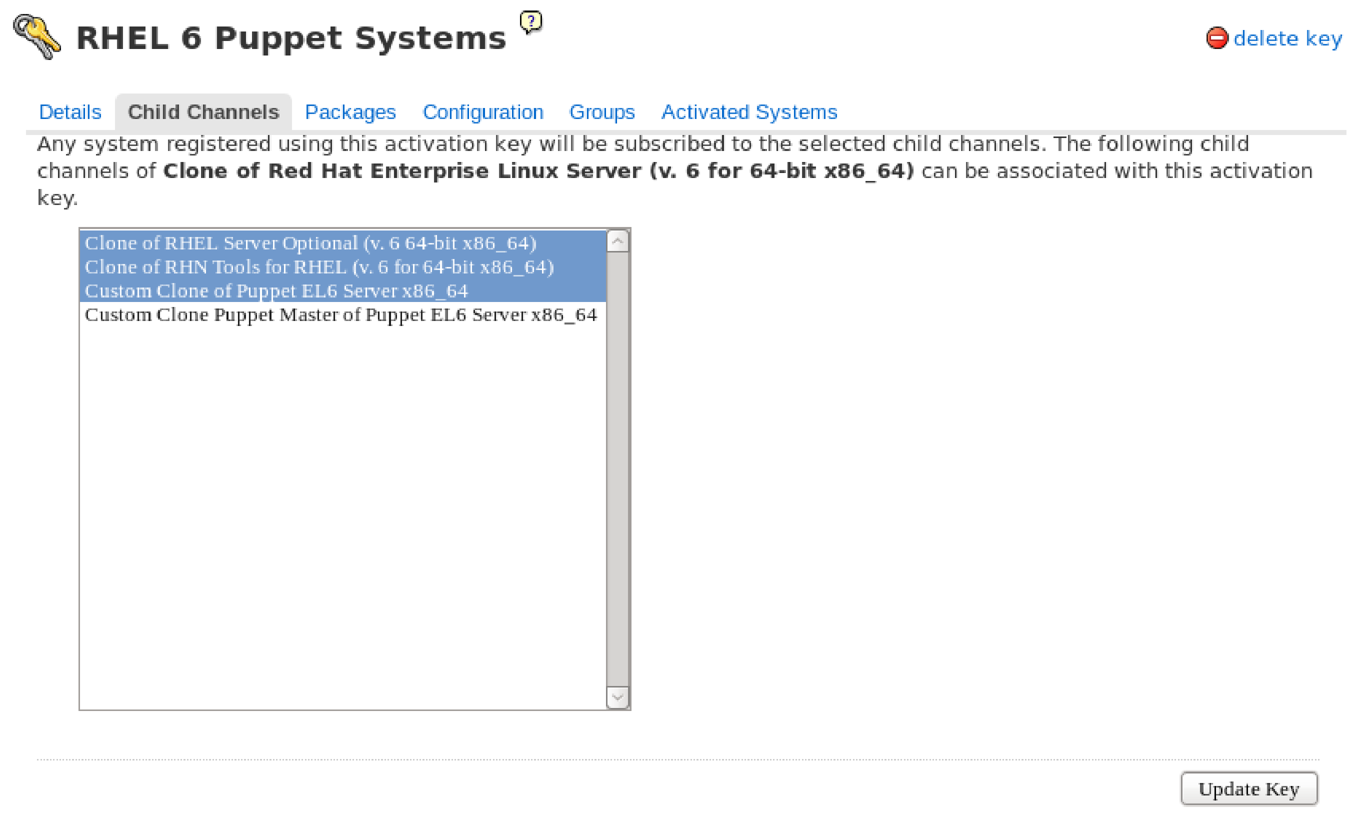
Activation Key Details

Systems registered with this activation key will inherit the settings listed below.

Description	RHEL 6 Puppet Systems **Tip:** Use this to describe what kind of settings this key will reflect on systems that use it. If left blank, this field will be filled in '**None**'.
Key:	1- rhel6-puppet **Tip:** Leave blank for automatic key generation. Note that the prefix is an indication of the Red Hat Satellite organization the key is associated with.
Usage:	**Tip:** Leave blank for unlimited use.
Base Channels:	Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64) **Tip:** Choose "**Red Hat Satellite Default**" to allow systems to register to the default Red Hat provided channel that corresponds to their installed version of Red Hat Enterprise Linux. You may also choose particular Red Hat provided channels or custom base channels here, but please note if a system using this key is not compatible with the selected channel, it will fall back to its Red Hat default channel.
Add-On Entitlements:	☐ Monitoring ☑ Provisioning ☐ Virtualization ☐ Virtualization Platform
Universal Default:	☐ **Tip:** Only one universal default activation key may be set for this organization. By setting this key as universal default, you will remove universal default status from the current universal default key if it exists. If this key is set as universal default, then newly-registered systems to your organization will inherit the properties of this key.

Figure 2.4. Satellite 5 Activation Key Details

Next go to the Child Channels tab and select all but the Custom Clone Puppet Master of Puppet EL6 Server x86_64 child channel and click the Update Key button.



Any system registered using this activation key will be subscribed to the selected child channels. The following child channels of **Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)** can be associated with this activation key.

Clone of RHEL Server Optional (v. 6 64-bit x86_64)
Clone of RHN Tools for RHEL (v. 6 for 64-bit x86_64)
Custom Clone of Puppet EL6 Server x86_64
Custom Clone Puppet Master of Puppet EL6 Server x86_64

Figure 2.5. Satellite 5 Activation Key Channels

Next go to the Groups tab and click Join, select the RHEL 6 Puppet System Group, click Join Selected Groups.

Finally go to the Packages tab and enter within the main box **puppet** and click the Update Key button. We should now be finished with this Activation Key.

Repeat the above process to create a RHEL 5 based Activation Key and remember to select the correct Cloned/custom channels from the options.

At this point you can add the activation keys into any custom registration scripts or bootstrap.sh scripts. Additionally they can be used manually on the command line, along with the rhnreg_ks cmd line tool. If used at this stage you should successfully have in the end the system registered to Satellite to the correct System Groups, Channels and have the puppet package installed (but not configured). Example:

```
# rpm --import http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
# rpm -Uvh http://<Satellite>.example.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm
# rhnreg_ks --activationkey 1-rhel6-puppet --force --
serverUrl=https://<Satellite>.example.com/XMLRPC --sslCACert=/usr/share/rhn/RHN-ORG-
TRUSTED-SSL-CERT
```

2.7. GPG Keys

Objective: GPG keys are used to sign RPM content as a validation that it can be trusted. Red Hat ships our GPG keys as standard. For 3rd party content you have to establish what their published key is and then configure systems to know and trust it. Satellite cannot install or add GPG Keys after a system is installed and registered. If you have established systems you will need to manually (or scripted) deploy the

Puppet Labs GPG key to the systems or as noted previously if one is already in use GPG resign the specific packages to your own custom GPG key. We will show how to manually configure an individual system for the 3rd party GPG key and then a method to capture the GPG key into Satellite for Kickstart provisioned systems.

One method to manually install the Puppet Labs GPG key is using the following command as root:

```
# rpm --import http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
```

For systems being provisioned via Satellite you can copy the GPG key to Satellite and associate it with the kickstart profiles to ensure it is installed on systems at time of (re)installation. To do this download the GPG key to your local workstation from `http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs`. Next within Satellite go to Systems -> Kickstart -> GPG and SSL Keys -> create new stored key/cert

```
     Description*: Puppet GPG Key
     Type: GPG
     Select file to upload: * Browse to local copy to upload
```

2.8. Install & Configure Puppet Master

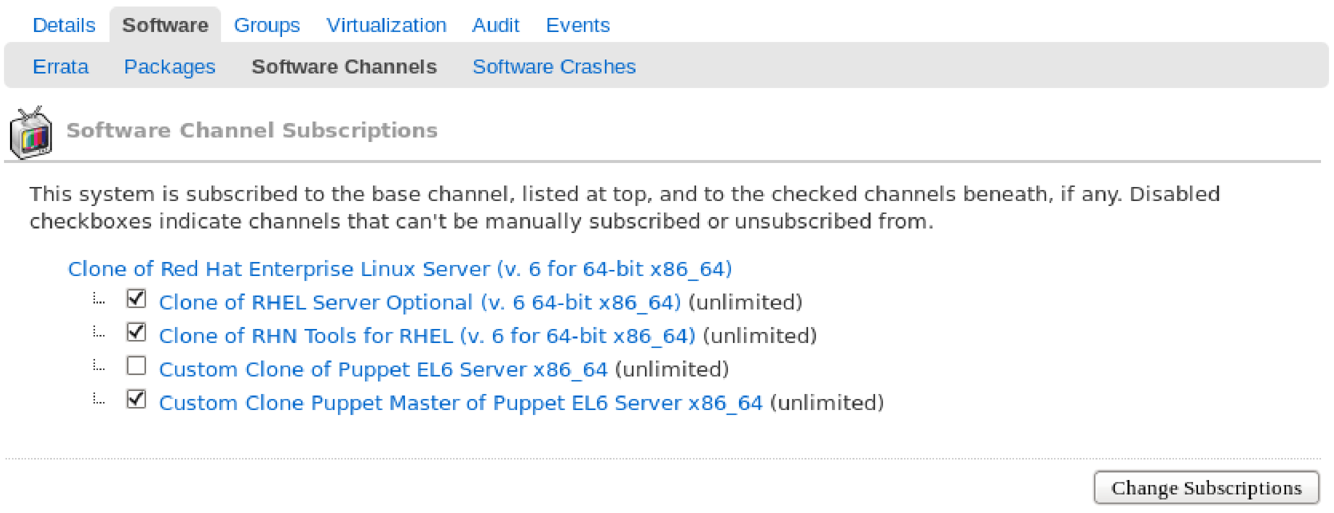
Objective: We are getting close to completing the initial tasks for infrastructure setup. We will walk through setting up a Puppet Master which is used in conjunction with Satellite 5. At the end we will show how to manually test with a client that communication to the puppet master is working. We will then tie the two together with Kickstart profile provisioning of Satellite.

We are going to guide you through getting the Puppet Master (server) components installed and running on a RHEL 6 based system. To start with have a freshly installed RHEL 6 Server x86_64.

Puppet Master Deployment Notes

- This system is a different RHEL instance from the one which Satellite runs on.
- We recommend to review official Puppet Labs documentation for requirements and recommendations for installation and configuration of a Puppet Master. This documentation guide is provided is an example but should be taken in consideration with official documentation which can change over time.
  - `http://docs.puppetlabs.com/puppet/`
  - `http://docs.puppetlabs.com/guides/installation.html`

Register the puppet master RHEL system to Satellite and ensure that it is fully up to date. Next subscribe it to the Puppet Master child channel, Optional and RHN Tools child channels.



This system is subscribed to the base channel, listed at top, and to the checked channels beneath, if any. Disabled checkboxes indicate channels that can't be manually subscribed or unsubscribed from.

Clone of Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)
- ✓ Clone of RHEL Server Optional (v. 6 64-bit x86_64) (unlimited)
- ✓ Clone of RHN Tools for RHEL (v. 6 for 64-bit x86_64) (unlimited)
- ☐ Custom Clone of Puppet EL6 Server x86_64 (unlimited)
- ✓ Custom Clone Puppet Master of Puppet EL6 Server x86_64 (unlimited)

Figure 2.6. Satellite 5 System Channel Selection

Now install the puppet-server packages:

```
# rpm --import http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
# yum install puppet-server
```

Reviewing the puppet documentation for installation perform any specific changes required within configuration, especially noting the dns_alt_names option, if you plan to have more than one valid cname or Fully Qualified Domain Name (FQDN) for your puppet master deployment.

You can use puppet to manage the puppet master itself. As such you may want to configure on your puppet master within the /etc/puppet/puppet.conf file within the [main] block the server option to read the same as your hostname.

Puppet and puppetmaster services by default are disabled. Use the following commands to enable puppet and confirm. We will configure puppetmaster next to run within apache web services.

```
# puppet resource service puppet ensure=running enable=true
Notice: /Service[puppet]/ensure: ensure changed 'stopped' to 'running'
service { 'puppet':
 ensure => 'running',
 enable => 'true',
}
# chkconfig --list | grep -i pupp
puppet           0:off    1:off    2:on    3:on    4:on    5:on    6:off
puppetmaster     0:off    1:off    2:off    3:off    4:off    5:off    6:off
# service puppet status
puppet (pid  15624) is running...
```

Now we will install and configure mod_passenger for usage by the Puppet Master. Please cross reference the main Puppet documentation - See `http://docs.puppetlabs.com/guides/passenger.html`

```
# rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-6
# yum install httpd httpd-devel mod_ssl ruby-devel rubygems gcc
# yum install mod_passenger
```

2.8.1. Installation of Puppet Master Apache Rack Application

Use the commands shown below to copy over a stock configuration file supplied via RPM into the default configuration location required for this Puppet Master setup.

```
# mkdir -p /usr/share/puppet/rack/puppetmasterd
# mkdir /usr/share/puppet/rack/puppetmasterd/public
/usr/share/puppet/rack/puppetmasterd/tmp
# cp /usr/share/puppet/ext/rack/files/config.ru
/usr/share/puppet/rack/puppetmasterd/
# chown puppet:puppet /usr/share/puppet/rack/puppetmasterd/config.ru
# service puppetmaster restart
Stopping puppetmaster: [FAILED]
Starting puppetmaster: [  OK  ]
```

2.8.2. Apache Configuration

Create an Apache Virtual Host file. Below is slightly modified version listed on Puppet Labs documentation. Create the file as /etc/httpd/conf.d/puppetmaster.conf. Please note to change the paths for Apache SSL Certs listed below to correct ones discovered on your own system, based on FQDN.

Contents:

```
# And the passenger performance tuning settings:
PassengerHighPerformance On
# Set this to about 1.5 times the number of CPU cores in your master:
PassengerMaxPoolSize 12
# Recycle master processes after they service 1000 requests
PassengerMaxRequests 1000
# Stop processes if they sit idle for 10 minutes
PassengerPoolIdleTime 600


Listen 8140
<VirtualHost *:8140>
    SSLEngine On

    # Only allow high security cryptography. Alter if needed for compatibility.
    SSLProtocol             All -SSLv2
    SSLCipherSuite          HIGH:!ADH:RC4+RSA:-MEDIUM:-LOW:-EXP
    SSLCertificateFile      /var/lib/puppet/ssl/certs/puppet-server.example.com.pem
    SSLCertificateKeyFile   /var/lib/puppet/ssl/private_keys/puppet-
server.example.pem
    SSLCertificateChainFile /var/lib/puppet/ssl/ca/ca_crt.pem
    SSLCACertificateFile    /var/lib/puppet/ssl/ca/ca_crt.pem
    SSLCARevocationFile     /var/lib/puppet/ssl/ca/ca_crl.pem
    SSLVerifyClient         optional
    SSLVerifyDepth          1
    SSLOptions              +StdEnvVars +ExportCertData

    # These request headers are used to pass the client certificate
    # authentication information on to the puppet master process
    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 /usr/share/puppet/rack/puppetmasterd/public

    <Directory /usr/share/puppet/rack/puppetmasterd/>
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
    </Directory>
    ErrorLog /var/log/httpd/puppet-server_ssl_error.log
    CustomLog /var/log/httpd/puppet-server_ssl_access.log combined
</VirtualHost>
```

By default SELinux will deny mod_passenger. This can be resolved by creating custom SELinux policy or using an alternative method of installing your puppet master.

If you wish to learn more about creating a simple SELinux policy review the Red Hat Documentation. The best source of information can be found within the Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide - https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html. Additionally there is other 3rd party sites which show how to create a simple SELinux policy for mod_passenger for puppet usage - one such article can be found here - http://linuxfollies.blogspot.com.es/2012/01/puppet-apache-modpassenger-and-selinux.html.

Once you have a working SELinux policy in place ensure that you are running within Enforcing mode. Additionally you will next want to ensure Apache is enabled by default. Please note that we have configured Apache to listen to port 8140 - ensure any firewall rules in place accommodate for incoming traffic to this port.

```
# setenforce 1
# chkconfig httpd on
```

2.8.3. x509 (SSL) Certificates

By default SSL Cert creation to allow client/server communication is manually done. For ease of use within this guide I'm using the autosigning ability - to trust any system which can communicate to the puppet master. This may not be ideal within your environment, please review the Puppet documentation to determine the best solution for you. Example autosign configurations can be found here http://docs.puppetlabs.com/guides/configuring.html#autosignconf.

1. Add on the puppet master within configuration file /etc/puppet/puppet.conf section

```
[puppetmasterd]
 ssl_client_header = SSL_CLIENT_S_DN
 ssl_client_verify_header = SSL_CLIENT_VERIFY
```

2. To enable autosigning for Certificates

```
# echo "*" > /etc/puppet/autosign.conf
```

3. We are now going to restart services, looking for any errors.

```
# service httpd stop
Stopping httpd: [  OK  ]
# service puppetmaster restart
Stopping puppetmaster: [  OK  ]
Starting puppetmaster: [  OK  ]
# service httpd restart
Stopping httpd: [FAILED]
Starting httpd: [  OK  ]
#
```

4. Create a place holder manifest file to initiate with. Create file /etc/puppet/manifests/site.pp with:

```
# CONTENTS OF /etc/puppet/manifests/site.pp
node "default" { }
```

We now have the basic framework for a functional puppet master environment. If you were to now install the puppet client package onto a system and then configure it to point to the PuppetMaster it should function and return no errors.

2.8.4. Puppet Clients

To manually test a puppet client system to the Satellite, use the following example commands:

```
# rpm --import http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
# rpm -Uvh http://<Satellite>.example.com/pub/rhn-org-trusted-ssl-cert-1.0-
1.noarch.rpm
# rhnreg_ks --activationkey 1-rhel6-puppet --force --
serverUrl=https://<Satellite>.example.com/XMLRPC --sslCACert=/usr/share/rhn/RHN-ORG-
TRUSTED-SSL-CERT
```

Then edit the /etc/puppet/puppet.conf file and add within the [main] section the server line to point to your puppet master

```
[main]
    server = puppet.example.com
```

You can now run the following test command on the puppet client

```
# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for puppet.example.com
Info: Applying configuration version '1390552206'
Notice: Finished catalog run in 0.02 seconds
#
```

Assuming that all is well we will now tie the puppet and Satellite environments together so that newly installed systems are correctly configured to communicate with both Satellite for content and general management and with the puppet master for configuration of systems via puppet modules.

Confirm Puppet Client Configuration

You can use the follow option of puppet to see all possible options and their current settings

```
# puppet config print
```

or to see the specific server value:

```
# puppet config print server
```

2.9. Kickstart Provisioning

Objective: We are now going to show how to configure and use kickstart provisioning to tie activation keys, system groups, puppet master location and GPG key all together for a system that is provisioned and talking to Satellite + puppet. This will be the default method to configure the puppet clients.

We will use the cobbler snippet capabilities to ensure that by default newly provisioned systems are correctly configured. Cobbler snippets are small scripts which can be re-used and associated easily with multiple kickstart profiles. So a change to the one snippet is automatically propagated out to all associated kickstart profiles vs manually editing %post scripts within each profile.

Other options are available to install the puppet RPMs and configured to talk to the puppet master but will not be detailed. These options include manually or scripting the process such as creating a simple bash script to:

- Manually import the GPG key, subscribing to the correct child channel and installation of the puppet client, then configuring the puppet.
- Re-using the bash scripting within %post of the kickstart profile
- Re-using the bash scripting within a bootstrap.sh script
- Using a re-activation key and activation key chained together
- Using Satellite's own configuration management system to lay down the puppet.conf file with the static puppet server url line within it

Below are two versions of the example snippets we will add. One for RHEL 5 and the other for RHEL 6

EL6

```
#raw
FILE=/etc/sysconfig/puppet
if [ -f $FILE ]
then
   echo "Puppet config file $FILE exists - proceeding."
   echo "PUPPET_SERVER=puppet.example.com" >> /etc/sysconfig/puppet
   perl -pi -e 's/\[main\]/\[main\]\n   server = puppet.example.com/g'
/etc/puppet/puppet.conf
   puppet agent --test --waitforcert 60 --no-splay --server=puppet.example.com --
onetime --verbose
   chkconfig puppet on
else
   echo "File $FILE does not exist - exiting."
fi
#end
```

EL5

```
#raw
FILE=/etc/sysconfig/puppet
if [ -f $FILE ]
then
   echo "PUPPET_SERVER=puppet.example.redhat.com" >> /etc/sysconfig/puppet
   perl -pi -e 's/\[main\]/\[main\]\n   server = puppet.example.com/g'
/etc/puppet/puppet.conf
   puppetd --test --waitforcert 60 --no-splay --server=puppet.example.com --onetime
--verbose
   chkconfig puppet on
else
   echo "File $FILE does not exist - exiting."
fi
#end
```

Within Satellite, go to Systems -> Kickstart -> Kickstart Snippets -> create new snippet, enter options:

```
    Snippet Name*    puppet-config-el6
    Contents*       <Enter above EL6 code, including the #raw and #end>
```

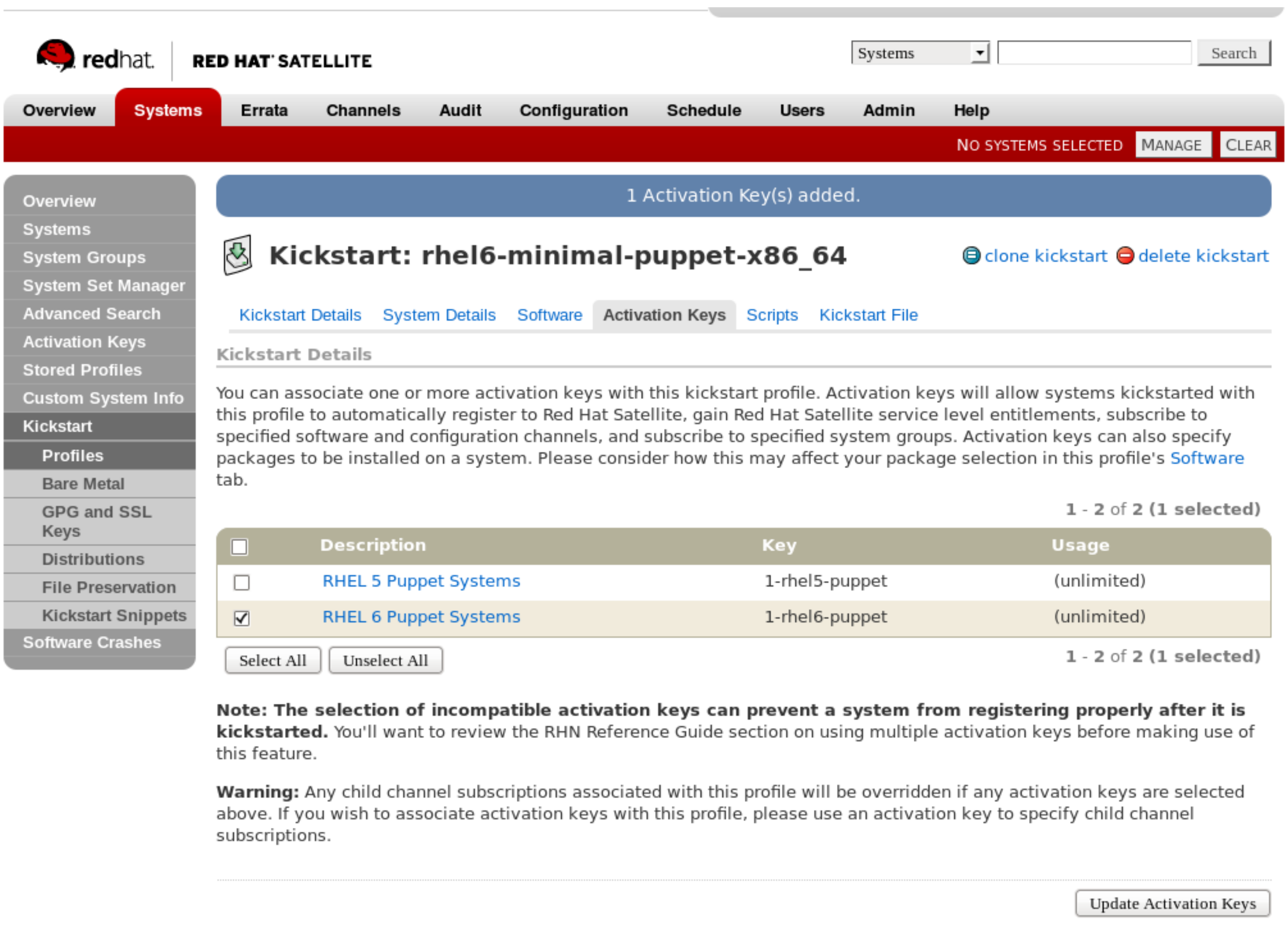
And click the Create Snippet button. Repeat for the RHEL 5 version.

We will now create a kickstart profile used for the kickstarting of physical systems for RHEL 6 and to use the puppet-config-el6 snippet along with the el6 puppet activation key.

Within Satellite Kickstart go to Profiles -> create new kickstart profile and within the Wizard select:

```
    Label*: rhel6-minimal-puppet-x86_64
    Base Channel*: Red Hat Enterprise Linux Server (v.6 for 64-bit x86_64)
    Kickstartable Tree*: ks-rhel-x86_64-server-6-6.5
```

- Click the Next button then Next again and then enter your default root password and click the Finish button. You will now be able to further edit the kickstart profile. I recommend to select the options for Logging of custom scripts - in case debugging is required.
- Click onto the System Details -> GPG & SSL tab, select the Puppet GPG Key and click the Update keys button.
- Click onto the Activation Keys tab for the Kickstart profile and select the RHEL 6 Puppet Systems key and click the Update Activation Keys button.



1 Activation Key(s) added.

Kickstart: rhel6-minimal-puppet-x86_64

Kickstart Details

You can associate one or more activation keys with this kickstart profile. Activation keys will allow systems kickstarted with this profile to automatically register to Red Hat Satellite, gain Red Hat Satellite service level entitlements, subscribe to specified software and configuration channels, and subscribe to specified system groups. Activation keys can also specify packages to be installed on a system. Please consider how this may affect your package selection in this profile's Software tab.

1 - 2 of 2 (1 selected)

☐	Description	Key	Usage
☐	RHEL 5 Puppet Systems	1-rhel5-puppet	(unlimited)
✓	RHEL 6 Puppet Systems	1-rhel6-puppet	(unlimited)

1 - 2 of 2 (1 selected)

Note: The selection of incompatible activation keys can prevent a system from registering properly after it is kickstarted. You'll want to review the RHN Reference Guide section on using multiple activation keys before making use of this feature.

Warning: Any child channel subscriptions associated with this profile will be overridden if any activation keys are selected above. If you wish to associate activation keys with this profile, please use an activation key to specify child channel subscriptions.

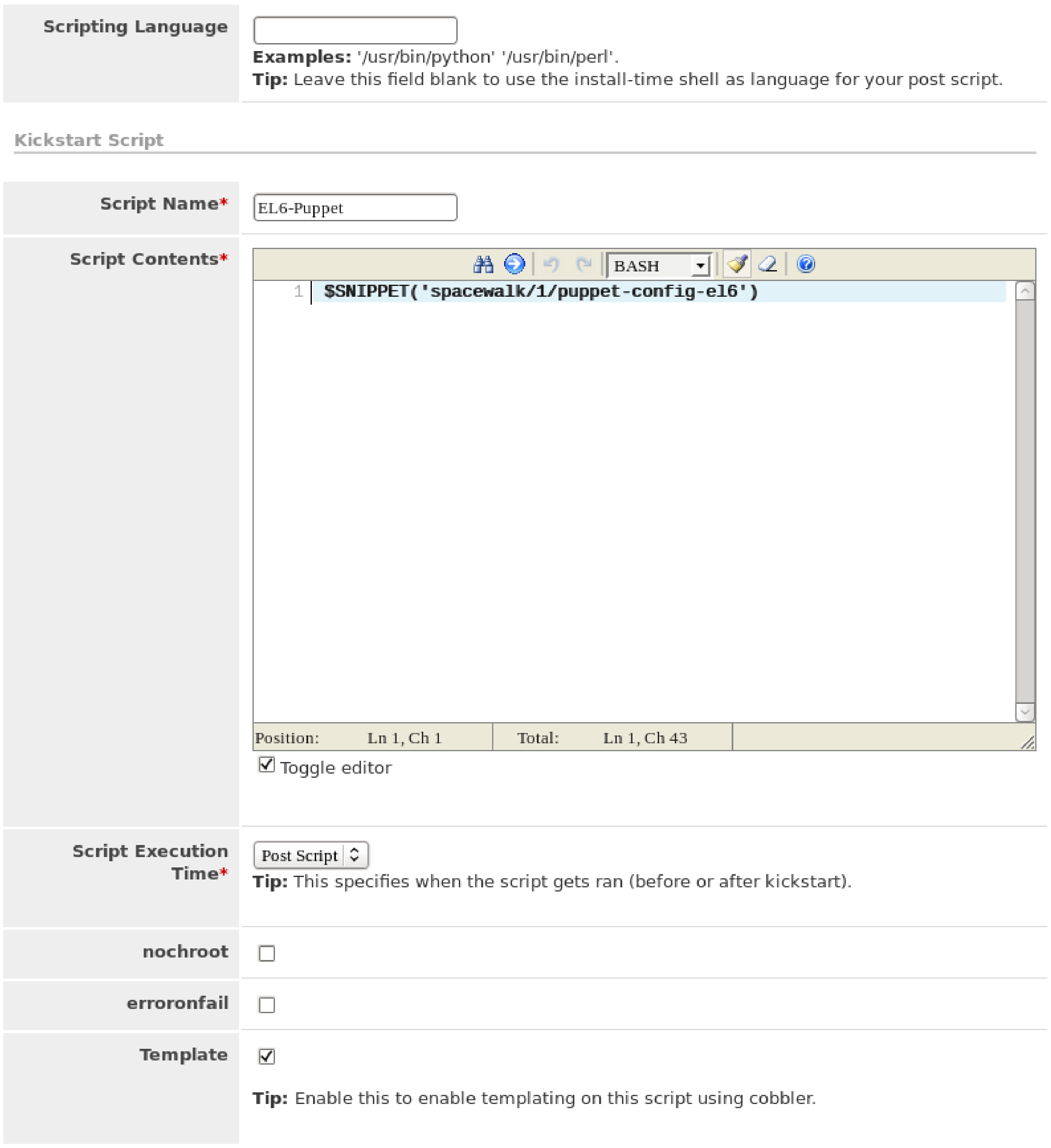
Figure 2.7. Satellite 5 Kickstart Profile Activation Keys

- Next click onto the Scripts tab -> add new kickstart script. Enter

```
    Script Name* EL6-Puppet
    Script Contents*

$SNIPPET('spacewalk/1/puppet-config-el6')

    Script Execution Time*  Post Script
    Template - selected
```



Scripting Language	Examples: '/usr/bin/python' '/usr/bin/perl'. Tip: Leave this field blank to use the install-time shell as language for your post script.

Kickstart Script

Script Name*	EL6-Puppet
Script Contents*	`$SNIPPET('spacewalk/1/puppet-config-el6')` Position: Ln 1, Ch 1 Total: Ln 1, Ch 43 ☑ Toggle editor
Script Execution Time*	Post Script. Tip: This specifies when the script gets ran (before or after kickstart).
nochroot	☐
erroronfail	☐
Template	☑ Tip: Enable this to enable templating on this script using cobbler.

Figure 2.8. Satellite 5 Kickstart Script

This kickstart profile can now be used to install or reinstall systems to RHEL 6 with a puppet client configured to communicate with Satellite 5 + Puppet Master.

Repeat the above process to create a RHEL 5 based profile.

2.10. Cobbler as External Node Classifier

Objective: One of the powers of puppet is the ability to dynamically make changes based on the specific system in question. Within the Satellite 6 section we discuss key considerations when creating puppet modules. This section outlines how you can use the cobbler daemon shipped within Satellite 5 as a means to store this additional meta-data about the system. We will walk through how to use Satellite 5 to create system profiles within cobbler and then using cobbler to add/change management values. We will then walk through configuring puppet to talk to cobbler for this information.

Please note that the older cobbler shipped with Satellite 5.x is not fully compatible with newer puppet parameterized classes for modules, but does support the basic usage of them as outlined in this document. There are many other options to use as an External Node Classified (ENC).

Further details on ENC's and Cobbler can be found online at the following locations

- http://docs.puppetlabs.com/guides/external_nodes.html
- https://fedorahosted.org/cobbler/wiki/UsingCobblerWithConfigManagementSystem

2.10.1. Puppet Master

On the Puppet Master we are going to do a minimal installation of cobbler reusing the version that is provided by Satellite. We will then configure the puppet master to call out to the cobbler server for node definitions.

The first step is to install cobbler - one way to achieve this is to copy the Satellite Installation ISO onto the Puppet Master and locally mount it. Once mounted change directory into where all the main RPMs are. We will then use yum to install cobbler and dependencies.

```
# mkdir sat56-iso
# mount -o loop satellite-5.6.0-20130927-rhel-6-x86_64.iso sat56-iso
# cd sat56-iso/Satellite/
# yum localinstall cobbler-2.0.7-37.el6sat.noarch.rpm libyaml-0.1.2-5.el6.x86_64.rpm
PyYAML-3.09-3.el6sat.x86_64.rpm
```

Next we have to configure the puppet installed cobbler to communicate with the Satellite's main cobbler. To do this, we will need to know the Satellite's IP address. Open within a text editor the /etc/cobbler/settings file and change line from

```
server: 127.0.0.1
```

to:

```
server: IP-address-of-Satellite-Server
```

To confirm that the cobbler section is functioning you can manually run cobbler-ext-nodes command with the option localhost. It will return empty data.

```
# cobbler-ext-nodes localhost
classes: []
parameters: {}
```

The final step on the Puppet Master is to configure it to use cobbler as an external node classifier. To do this we will edit the /etc/puppet/puppet.conf configuration file and add the following new section to it:

```
[master]
 node_terminus = exec
 external_nodes = /usr/bin/cobbler-ext-nodes
```

2.10.2. Satellite Server

Within Satellite for each system that you want to manage with puppet you will need to generate a system profile for it within cobbler. There is several ways to achieve this including usage of API (recommended for ease of automation and bulk change), or spacecmd tool from Spacewalk, or the cobbler command line or using the Satellite WebUI. If using the Satellite WebUI browse to Systems -> System Profile Name -> Provisioning -> Select the Puppet Kickstart Profile and then click the Create Cobbler System Record button, and then Continue to confirm.

As root we will now use the cobbler command line tool to list and see the system records for known systems and then show how you can set various management classes. These management classes are read by puppet when it calls out to the ENC to determine what puppet modules are available to the system being managed by puppet. These management classes can be either global or a per-system basis. Within the example below we will show how to enable/set the ntp class.

To list the system profiles known to cobbler and puppet use:

```
# cobbler system list
   example1.example.com:1
   example2.example.com:1
   example3.example.com:1
#
```

To get a lot more detailed information on a specific system use the system report option within cobbler:

```
# cobbler system report example1.example.com:1
```

If your environment is not setup correctly at this point you will see the DNS Name is incorrect on the profile - puppet requires the full hostname and correct DNS resolution. The above command will also list all the Management Classes specifically assigned to the system profile.

Alternatively, to see just the management classes exposed for a system you can call directly the cobbler-ext-nodes command which is what puppet calls to find details for a system:

```
# cobbler-ext-nodes example1.example.com
```

There is 4 distinct locations that you can enable a puppet module (class) within cobbler - globally within the cobbler configuration file, on a per kickstart tree distro level, on a per kickstart profile level, and finally on a per individual system basis. We will now walk through each of the four methods using dummy example puppet modules.

1. If you had example puppet modules called ntpfred and ntpsally and wanted to enable it globally for all systems, you will need to edit the /etc/cobbler/settings file and look for the line:

```
mgmt_classes: []
```

You would edit this to read:

```
mgmt_classes: [ntpfred, ntpsally]
```

Then restart the cobblerd daemon:

```
# service cobblerd restart
Stopping cobbler daemon: [  OK  ]
Starting cobbler daemon: [  OK  ]
#
```

2. If you had an example puppet module called ntpabc and wanted to enable it at a kickstart tree distribution level, you can use:

```
# cobbler distro edit --name=ks-rhel-x86_64-server-6-65 --mgmt-classes="ntpabc"
```

This will be inherited by any kickstart profile using this kickstart tree distribution.

3. If you had an example puppet module called ntpxzy and wanted to enable it at a kickstart profile level, you can use:

```
# cobbler profile edit --name=rhel6-minimal-puppet-
x86_64:1:GLOBALSUPPORTSERVIREDHATINC --mgmt-classes="ntpxyz"
```

This will be inherited by any system associated with this kickstart profile.

4. If you had an example ntp puppet module and wanted to enable it on only a specific system, or select few, you can use:

```
# cobbler system edit --name example1.example.com:1 --mgmt-classes="ntp" --dns-
name=example1.example.com
```

In the end, you can use cobbler-ext-nodes to list all enabled puppet modules for the system, which would look something similar to:

```
# cobbler-ext-nodes example1.example.com
classes: [ntpfred, ntpsally, ntpabc, ntpxzy, ntp]
parameters: {from_cobbler: 1, media_path: /ks/dist/ks-rhel-x86_64-server-6-6.5,
org: 1,
 use_ipv6_gateway: 'false'}

#
```

If you followed the above examples you will want to blank out and remove the bogus entries by re-running the commands but this time with --mgmt-classes="". Failure to remove the bogus modules will result in puppet attempting to look for the ntpfred and associated modules and throw an error when it fails to find them. Example error:

```
# puppet agent --test --noop
err: Could not retrieve catalog from remote server: Error 400 on SERVER: Could not
find class ntpfred for example1.example.com on node example1.example.com
```

We now have a functioning system for puppet using an External Node Classifier to lookup details about managed systems via cobbler running on the Satellite, and using cobbler to define what puppet modules are allowed for give system(s). Next we will show how the populate a puppet module into the puppet master.

Support of command line options listed

Since the usage of puppet with Satellite 5 is not supported by Red Hat and this white paper is written to provide a recommended guidance on such deployment we have listed several cobbler command line options that are not officially document nor supported by Red Hat. As we have shown, they are functional and work but not rigorously tested and supported by Red Hat. Specifically the usage of cobbler-ext-nodes and the setting of mgmt-classes are not documented within official product documents for Red Hat Satellite 5.

2.11. Puppet Modules

Objective: We know that customers will often do a mixture of using 3rd party puppet modules such as those from puppetforge or their own custom modules. We will now show in detail how to deploy a custom puppet module using the popular puppetforge ntp module.

The ntp module can be downloaded from `http://forge.puppetlabs.com/puppetlabs/ntp` - use the Download as a .tar.gz link and place into /root/ of your puppet master. You can also store approved modules within a git repo and then import them from that repo into your puppet master. This would allow you to version control different revisions over time of the same module as changes are made to it.

You can list what modules are available by using the following command, which initially is empty:

```
# puppet module list
/etc/puppet/modules (no modules installed)
/usr/share/puppet/modules (no modules installed)
#
```

We are now going to import the ntp module which we downloaded (this can work for custom as well as puppetforge content):

```
# puppet module install ~/puppetlabs-ntp-3.0.3.tar.gz
Notice: Preparing to install into /etc/puppet/modules ...
Notice: Downloading from https://forge.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/modules
└─┬ puppetlabs-ntp (v3.0.3)
  └── puppetlabs-stdlib (v4.1.0)
#
```

And now, if we list, we see it is installed and available:

```
# puppet module list
/etc/puppet/modules
├── puppetlabs-ntp (v3.0.3)
└── puppetlabs-stdlib (v4.1.0)
/usr/share/puppet/modules (no modules installed)
#
```

With the ntp module now uploaded within the puppet master you can now enable the ntp module for all system associated with your puppet kickstart profile used earlier to install systems:

```
# cobbler system edit --name example1.example.com:1 --mgmt-classes="ntp" --dns-
name=example1.example.com
```

On your test puppet client if you now run puppet agent with the noop option (to only show what would happen) you should see a lot of output similar to:

```
# puppet agent --test --noop
info: Caching catalog for example1.example.com
info: Applying configuration version '1399189234'
notice: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]/content:
--- /etc/ntp.conf    2014-05-08 03:43:02.000000000 -0400
+++ /tmp/puppet-file20140508-29183-13wlryo-0    2014-05-08 07:01:27.000000000 -
0400
@@ -1,51 +1,24 @@
-# Permit time synchronization with our time source, but do not
-# permit the source to query or modify the service on this system.
+# ntp.conf: Managed by puppet.
+#

[ … OUTUT REMOVED … ]

-server 127.127.1.0
-fudge    127.127.1.0 stratum 10

notice: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]/content: current_value
{md5}e0275b9f2a204940fe8d8192c1c2df9e, should be
{md5}60f05f47809aa69d4dddb9dae8db528f (noop)
notice: Class[Ntp::Config]: Would have triggered 'refresh' from 1 events
info: Class[Ntp::Config]: Scheduling refresh of Class[Ntp::Service]
notice: Class[Ntp::Service]: Would have triggered 'refresh' from 1 events
info: Class[Ntp::Service]: Scheduling refresh of Service[ntp]
notice: /Stage[main]/Ntp::Service/Service[ntp]: Would have triggered 'refresh' from
1 events
notice: Class[Ntp::Service]: Would have triggered 'refresh' from 1 events
notice: Stage[main]: Would have triggered 'refresh' from 2 events
notice: Finished catalog run in 1.03 seconds
#
```

We now have a puppet master with a module in it where cobbler running as the puppet masters ENC defines what modules are available to which systems. With puppet running as an active service on managed systems puppet will pull down and apply changes automatically over time to keep systems within puppet managed defined system state.

2.12. Summary

We have now shown end to end on how to setup Red Hat Satellite 5 and Puppet to work together to manage an environment of RHEL systems. When used in this manner, along with the additional notes and comments within the following section, you should find that moving from Satellite 5 + puppet to a fully supported puppet deployment within Satellite 6 will be smooth to achieve with minimal re-architecture of your modules.

We have shown how to import upstream puppet RPMs into Satellite. We then used cloned and custom channels to lock content into known good state. We used system groups and activation keys to logically group puppet managed systems together at time of registration. We installed and configured a puppet master next to Satellite, using the Satellite's cobbler daemon as its External Node Classifier. We have created kickstart profiles, with post installation scripts that will install required packages and then register the system at time of provisioning to the puppet master. Finally, we have shown how to manage and enable puppet modules for sets of managed systems, which will be applied to the systems every time the puppet daemon on the clients connects to the puppet master.

We hope this information provides useful insight on one example deployment of using an unsupported puppet today, with Satellite 5, in readiness for future Satellite 6.

Chapter 3. Red Hat Satellite 6 and Puppet

3.1. Satellite 6 Application Lifecycle

Satellite 6 has been designed based on numerous customer design sessions and community recommendations based on the usage of Spacewalk. The usage patterns which are most commonly reported is to deploy software into a series of “environments” which each environment is more locked down and hardened then the one before it. A typical scenario is that software is deployed first to “Test”, then to “Stage”, and then to “Production”

Satellite 6 will support this pattern by allowing customers to mirror all the software and configuration they use into a central Library. The software is then promoted out through a series of environments. Satellite 6 will allow users to combine Puppet configuration with these packages into single views of both configuration and software, which can be promoted together. During the promotion, Satellite 6 will ensure that all managed Puppetmasters in the enterprise are updated correctly with the appropriate Puppet modules for each content view.

3.2. Preparing for Satellite 6

The following sections outline recommendations for getting ready for Satellite 6.

3.2.1. Use Channel Cloning for Application Lifecycle

Channel Cloning in Satellite 5 is a great way to set up a Application Lifecycle environment like Satellite 6 delivers. Customers should begin to use channel cloning to support a lifecycle which works in their enterprise.

Thomas Cameron provides a very good overview of this in his Satellite Power User Tips and Tricks presentation which you can view - `http://www.redhat.com/about/events-webinars/webinars/2013-12-12-tips-and-tricks-red-hat-satellite` or get the slides - `http://people.redhat.com/tcameron/Summit2012/Satellite_System_Deployment/cameron_t_1040_RHNSatellite_Power_User_Tips-n-Tricks_Pt_1.pdf` for it.

Rich Jerrido provides another source of excellent guidance for channel cloning within this article Errata Management Guide for Satellite 5.x - `https://access.redhat.com/site/articles/469173`

3.2.2. Utilize Puppet Modules

Puppet Modules - `http://docs.puppetlabs.com/puppet/latest/reference/modules_fundamentals.html` - are a good way to package and manage complex Puppet deployments. While more complex than just writing scripts, the extra effort of modularizing the work will make scaling the Puppet environment that much easier in the future.

Customers should begin to use Puppet modules for their configuration recipes. Customers should keep Puppet modules as modular as possible - covering a single component of the system if possible. Using role and profile classes - `http://docs.puppetlabs.com/puppet/latest/reference/modules_fundamentals.html` is recommended. This will allow users to map the modules or role and profile classes to Satellite 6 host groups. Puppet Labs provides a good overview - `http://docs.puppetlabs.com/puppet/latest/reference/modules_fundamentals.html` for writing modules.

Customers should define Modulefiles for modules so dependencies are explicitly declared. They should build modules artifacts as archives as if using Puppet Forge. This will allow import of modules into Satellite 6 and for it to display details of the modules.

3.2.3. Store Puppet Modules in GIT

The modules which are created by the customer, or downloaded from the Forge `https://forge.puppetlabs.com/`, should be stored in git. Customers can then promote them to their environments using tools like r10k `https://github.com/adrienthebo/r10k` or Librarian `https://github.com/rodjek/librarian-puppet` which deploy the modules to the Puppet master.

Customers may choose to invest in using Git Dynamic Branches `http://terrarum.net/administration/puppet-infrastructure-with-r10k.html` to mirror the environments defined in the channel cloning above. However, note that Satellite 6 will provide similar features and will mirror content from your git branches into the Library and Content Views.

If a Customer wishes to get a preview of how Satellite 6 will manage puppet modules, checkout the Satellite 6.0 Beta program, alternatively look how to mirror the Puppet Forge `http://www.pulpproject.org/2012/08/29/mirroring-puppet-forge-with-pulp/` and creating modules from git `https://github.com/pulp/pulp_puppet/blob/puppet-tools/docs/user-guide/recipes.rst#building-and-importing-modules`.

3.2.4. Do not use Dynamic Scoping

Dynamic scoping will not be supported by Satellite 6. Dynamic scoping was deprecated in Puppet 2.7 and removed in Puppet 3. Please see this page - `https://docs.puppetlabs.com/guides/scope_and_puppet.html` for more information on Dynamic Scoping.

3.2.5. Do not use Node Definitions inside of Manifests

The use of Node definitions within manifests will not be supported in Satellite 6.

3.2.6. Limit the use of Hiera functions inside of Manifests

Satellite 6 provides smart parameter support which provides a similar solution to variable population to what is provided by Hiera. Hiera function calls will be supported within Satellite 6. However, the user experience with smart variables will be a superior experience

Customers should edit their modules to limit the use of Hiera function calls. These should be documented so that when they transition to Satellite 6 then can be exposed as smart variables.

Revision History

Revision 1.0-1	**Tue July 22 2014**	**Clifford Perry**
Fixed title and content formatting		
Revision 1.0-0	**Tue July 1 2014**	**Clifford Perry**
Formatting and fixes completed		
Revision 0.9-0	**Fri June 6 2014**	**Clifford Perry**
First Completed Document		
Revision 0.0-0	**Thu May 8 2014**	**Bryan Kearney**
Initial creation by publican		

Index