Stacki Tutorial #7: Stacki Tutorial for Puppet

Introduction

One common use case for Stacki is to use Stacki to do hardware/VM provisioning, and Puppet to perform application provisioning on top of the systems Stacki has spun up.

To make life easier, this tutorial will walk through the steps of making puppet agent installation a part of your organizations' standard Stacki installations, so when Stacki is finished installing/reinstalling, Puppet will be able to take over immediately.

This tutorial is focused on installing agents and pointing them at your existing puppet servers. For help on setting up a puppet server, see the Puppet server installation guide.

Prerequisites:

1. Have Stacki server installed and working. This tutorial is based upon the CentOS 6.X version of Stacki server.
2. Have the FQDN of your preferred puppet master available. This tutorial is based upon the 4.X versions of puppet.
3. Make certain that yumdownloader is installed.
   - yum install yumdownloader
4. Make certain you have network connectivity to the puppet master.

This tutorial also draws heavily on the Carts guide on the Stacki wiki. If you are using some other application provisioning tool than puppet, this example and that guide should be enough to get you going, but please do drop us a line if you would like to see similar tutorials for other application automation toolsets.

When installing puppet 4.X agents, there are basically three steps – install the RPMs, configure the puppet server address, and use the puppet command line to install/configure the service. This guide will walk you through all of these steps using Stacki, so that you can quickly configure puppet agent on any number of nodes.

Anatomy of a Stacki Cart.

For this tutorial, we can run with a quick overview of how a stacki cart works. A cart adds site-specific install items to a pallet. A pallet is a definition of what is required to create a working Linux server with a given setup. A pallet is a blueprint of what to install, how, and when, along with all of the ISOs and RPMS required to do the installation, also included are the config-

uration scripts to set things up for the environment.

When you first install Stacki, you will have a "backend" pallet that knows how to install Linux (in our case CentOS 6.7) and configures it to work in your environment. When you enter host information into stacki, things like hostname, IP address, partitioning, etc. are applied to the installation to give the configuration steps the data they need to make each server customized.

A Cart merely layers on top of a pallet. The point of a cart is to add applications that you need onto the default installation – or any pallet-based installation. So for this tutorial, we will be creating a cart named "puppet" that will be added on to the back-end pallet, yielding a final installation that has all of the backend functionality plus the puppet agent installed and running.

If this is at all confusing, think of a pallet as you sitting there configuring any RAID card, then installing the ISO and entering all of that data, then updating the system, then installing select RPMS that are commonly used. All of that is the "pallet" part, but Stacki completely automates it. The point at which you end basic server config and start to set up your apps is where carts normally come in. They are the application information that maybe every server in your group/department/datacenter need, but not every team uses them. Puppet is one such app. Important for some organizations, but others don't want it on their systems. So it's not in backend, and we're going to add it with a cart.

The first step is to create an empty cart. Stacki can do that for you if on the Stacki server you type the following:

```
stack add cart puppet
```

This will create a directory structure in /export/stack/carts/puppet where your RPMS and configuration XML will go, and initially create an XML file that does nothing. We'll have to fill it in to tell it how to install puppet.

Let's take a quick look at the directory structure:
graphs
nodes
RPMs

For our purposes (and most cart purposes), we'll skip over **graphs**, which can be used to gain greater control over the order of operations during an install.
nodes holds the XML that tells stacki how to install our cart.
RPMs holds any RPMS that our cart needs.

First, since we know we need the puppet agent RPM, lets get that. If you don't already have it, but you have a server that is configured with the puppet repositories, you can use:

yumdownloader puppetlabs-release
yumdownloader puppet-agent

This will download the RPMS to local disk. There is more information about getting the RPMS available on the puppet agent install page. We used puppetlabs-release-6-11.noarch.rpm for this tutorial because it was the latest release version.

Once you have the RPM downloaded, get it onto the stacki server (if you didn't download it

there), and move it into your carts' RPMS directory:

```
mv puppet*/export/stack/carts/puppet/RPMS
```

Now we have the RPMS where we need them, let's take a look at the node information.
In the nodes directory you will find a file cart-puppet-backend.xml let's take a look at it.

```
<?xml version="1.0" standalone="no"?>
<kickstart>

     <description>
        puppet cart backend appliance extensions
     </description>

        <!-- <package></package> -->

<!-- shell code for post RPM installation -->
<post>
</post>
</kickstart>
```

Here's what this all is. The first line is simply a header like most XML has (and all should have).
The rest of the document is a definition of the kickstart object for this XML file.
The description value is auto-generated, feel free to customize.
The package section is commented out. This is where you tell it what RPM(S) to install.
The post section is what scripts to run after the install is done, but before first boot.

We're going to make a few changes that will tell Stacki what to do. First, uncomment the package tags, and insert the name of your puppet RPM in there. As is usual in installations, you just need the base name:

```
<package>puppetlabs-release</package>
```

But that's not all we need, since there is one package for the repositories and support RPMS, and another for the actual agent, add the following right after that package line:

```
<package>puppet-agent</package>
```

Now we've told it what packages stacki should install (and we could stop there if you were willing to do the configuration step by hand on each machine), but we want to tell it what puppet server to use, and to start and run the puppet service.

(NOTE: If you are using a puppet server whose hostname is puppet, you can skip the next step – puppet uses that name as the default server name)

The place to tell this server where to find its puppetmaster is in the post section. We append the server line to puppet.conf easily, because puppet.conf on first install simply has a [main] section, and we can add an [agent] section with the server command at the end.

Place the following into the <post> section:

```
<file name="/etc/puppetlabs/puppet/puppet.conf" mode="append" perms="644">
[agent]
server=puppet-server.nandgate.com
</file>
```

Of course you want to put your puppet masters' FQDN in the server= line. More information on this config option is available on the puppet website.

The file command does the following:
It opens the file /etc/puppetlabs/puppet/puppet.conf for appending.
It writes out what's between the <file> and </file> tags
Then it closes the file. The perms are used to set permissions. Note this is likely not necessary, but we left it in so as to present as-developed.

Okay, so we have installed puppet agent and told it where to go to get its manifests. The only remaining item is to set puppet up to autostart. Puppet is a little different than most services, because most you can just issue

```
service puppet start
```

and be done with it, but puppet prefers you use their tool, and indeed, using service fails until you do unless you take extraordinary steps. So we will do it their way. We don't want this command to run before first boot (the "post" section runs after install but before first boot) because it is actually expected to be run on a fully up and running system, so we'll introduce a new section: the boot section. Using the tag <boot order="post"> will tell Stacki to run the contents on the first boot after installation. The post part tells it to run the command after everything is up and running. So let's add the following:

```
<boot order="post">
/opt/puppetlabs/bin/puppet resource service puppet ensure=running enabled=true
chkconfig puppet on
service puppet start
</boot>
```

right at the end of the file (just before `</kickstart>`).

The first command is the puppet command to set up the puppet service. It is documented here.

The next command is the linux "make sure it's turned on" command. You can probably leave it out, but it isn't hurting anything, so I use it.
The final command starts the puppet service, since we're already booted, so it won't autostart on

this run. In other words: The first two commands will make certain it starts on reboot, this third one simply starts it right now.

Now save the file, we're done making modifications to it.

We have placed the RPMS into the cart directory, and edited the cart file in the nodes directory to tell Stacki how to configure them, now we need to build and activate our cart.

Enter the command:

```
stack enable cart puppet
```

This will attach the cart to the current distribution.

Followed by:

```
stack create distribution
```

Which rebuilds the distribution and includes our new cart.

If there were errors in the build, it's almost certainly your XML node file. Double-check it, or download the provided one here and modify.
If there were no errors, you can see your handiwork is in place by entering the command

```
stack list distribution
```

and you should see something like this:

```
NAME     OS     GRAPH   PALLETS                   CARTS
default: redhat default os-6.7-6.x stacki-2.0-6.x puppet
```

Notice how puppet is listed under CARTS? That's your cart, and it's ready to go!

Now pick a server and set it to install:

```
stack set host boot backend-0-0 action=install
```

and (re)boot the host in question.

It will install, and will configure puppet agent on the server.
If you have “auto-sign” configured for these servers, then puppet will “just work” after this. If you do not have auto-sign turned on, then shortly after the server has installed and booted into the OS, you should go to your puppet master and sign the node’s certificate.

You may have noticed along the way that we’re simply walking through the by-hand install steps with an automation tool. That is pretty much exactly how Stacki does everything. When I build add-ons for the default distribution, or a new pallet, I start by hand installing everything needed, and take notes about order and configuration, then I simply translate it into XML.