Puppetize It!

An Introduction to Puppet

Mike Seda
CEO, Seda Systems, Inc.

Meet the Guru

Profile, Experience, Preface

- Mike Seda
  - CEO, Seda Systems, Inc.
  - System Administrator since 2005
  - Accumulating Puppet skills since 2008
- The goal of this presentation is to provide an overview of Puppet. Advanced topics are mentioned, but not explored in detail.

SedaSystems™ We Know IT™

Introduction

Puppetize It?

In the past, I've been asked to perform various system and application tasks. After reviewing the requirements for many of these tasks, I found that they could often be "Puppetized" instead of being performed manually.

Eventually, I found myself and others around me saying "we can just Puppetize it" or "we can just Puppetize that". Essentially, we were saying "we can use Puppet to automate this task for us."

Introduction

Trends

Data Center Automation and Configuration Management have become increasingly popular lately. This recent trend may be due to the increased adoption of Virtualization and Cloud Computing.

This trend is definitely due to new and powerful tools that are now available to do this work. Without these tools, System Administrators would be forced to keep writing their own tools from scratch, which are often inferior and rarely portable, sharable, or scalable.

Introduction
Benefits

Benefits to Puppet users include increases in:

- Consistency
  - Caused by decrease in human error.
- Efficiency
  - Caused by increase in automation.
- Availability
  - Caused by decrease in downtime.

Introduction
Testimonials

Puppet has been called the "System Admin's best friend". Once you start using it, you'll see why.

A small list of organizations currently using Puppet is provided below:

- Google
- Stanford University
- Fedora
- Twitter
- SANS Institute
- Sun Oracle

Basic Knowledge

Supported Platforms

- Linux
  - Many popular flavors.
- Unix
  - Many popular flavors.
- Mac OS X
- Windows
  - ETA 2010.
    - Limited functionality currently exists as recently noted on the Puppet Labs Wiki.

Basic Knowledge
Installation

- Packages are available for the various flavors of Linux/Unix.
  - On CentOS/RHEL, install is as simple as the following command (once the machine has been pointed at the EPEL repository):
    - yum install puppet (for clients)
    - yum install puppet-server (for server)
- Mac OS X packages are available, as well.
- You can always install from source to get the bleeding edge version.

Basic Knowledge
Configuration

- Open TCP/UDP 8140 on server.
- Edit/create the necessary files:
  - /etc/puppet/puppet.conf
    - Main Puppet daemon(s), both client and server, configuration file.
  - /etc/puppet/manifests/site.pp
    - Central manifest capable of configuring an entire site.
  - /etc/puppet/manifests/nodes.pp
    - Contains node definitions.
  - /etc/puppet/manifests/templates.pp (optional)
    - Contains template class definitions.
- Start the central daemon (Puppet Master):
  - /etc/init.d/puppetmaster start

Basic Knowledge
Configuration (Cont.)

puppet.conf

```
[main]
    # Where Puppet stores dynamic and growing data.
    # The default value is '/var/puppet'.
    vardir = /var/lib/puppet

    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    syslogfacility = local5

[puppetd]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be loaded in
    # the separate ``puppet`` executable using the ``--loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt

    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig
```

Basic Knowledge
Configuration (Cont.)

site.pp

```
# site.pp

import "templates"
import "nodes"

filebucket { main: server => puppet }

# global defaults
File { backup => main }
Exec { path => "/usr/bin:/usr/sbin/:/bin:/sbin" }

Package {
    provider => $operatingsystem ? {
        debian => aptitude,
        redhat => up2date
    }
}
```

Basic Knowledge
Configuration (Cont.)

nodes.pp

```
# nodes.pp
#

node 'someserver.domain.com' inherits basenode {
    $web_fqdn = 'www.domain.com'
    include genericwebserver
    include some_other_service
}

node 'ldapmaster.domain.com' inherits basenode {
    include s_ldap::master
}

node 'humanresources.domain.com' inherits basenode {
    include c_humanresources
}
```

Basic Knowledge
Configuration (Cont.)

templates.pp

```
# templates.pp
# All server templates for various flavors of templates defined here

class baseclass {
    include $operatingsystem,
        afs,
        cron,
        dns,
        puppetclient,
        ssh,
        unixadmin_users,
        user_root,
        virt_all_users

    package { "ourcompany-package": ensure => present }
}
```

Basic Knowledge
Configuration (Cont.)

- Verifying installation
  - On client, run:
    - puppetd --server myserver.domain.com --waitforcert 60 --test
  - On server, run:
    - puppetca --list
    - puppetca --sign myclient.domain.com

Intermediate Knowledge

Puppet Language

- Resources
  - file
  - service
  - package
- Classes
  - Use once.
  - Support inheritance.
- Defines
  - Use many.
  - Support arguments.

Intermediate Knowledge

Puppet Language (Cont.)

- Modules
  - Contain **collections** of classes, definitions and resources.
    - Apache
    - Sendmail
  - Portable
    - Every module should be able to just work after being dropped into any installation.
- Nodes
  - The nodes concept is how we map a particular configuration to a particular machine or set of machines.

Intermediate Knowledge

Best Practices - Style

In organizations that have numerous System Administrators managing hundreds of Puppet manifests that contain thousands of classes, it may be necessary to adopt a formal convention for how Puppet syntax is written/edited.

The organization credited with coming up with the current Puppet style guidelines is Stanford University.

Intermediate Knowledge

Best Practices - Style (Cont.)

When these guidelines are adhered to, all manifests have a similar look and feel, and the need for folks to refactor syntax due to readability or other issue is eliminated.

Intermediate Knowledge

Best Practices - Modules

Make sure to use modules. It's very much frowned upon not to.

Intermediate Knowledge

Best Practices - VCS

Using a VCS (Version Control System) with Puppet is highly recommended. If Puppet is to be used to manage the configurations of your entire enterprise environment, then there must be a facility in place to revert changes and view a change history.

Advanced Knowledge
Templating

Templating allows you to manage the content of template files. Such files are usually not yet managed directly by a built-in Puppet type. Good examples are Apache or Samba configuration files.

- # Use name-based virtual hosting.
  NameVirtualHost <%= ipaddress %>:80
  NameVirtualHost <%= ipaddress %>:81

Template **files** should not be confused with template **classes**, which were discussed previously.

Advanced Knowledge
Scaling

- Web Server
  - By default, the Puppet Master uses WEBrick to serve files. This method can be replaced with that of Passenger (recommended) or Mongrel (legacy) to increase performance.

Advanced Knowledge
Scaling (Cont.)

- Delayed Check-in
  - By default, Puppet clients check in with the master once every 30 minutes. This could potentially bog down the server if too many clients check in at the same time. Therefore you can do the following to delay check-in:
    - Edit /etc/puppet/puppet.conf and add the following:
      - splay = true
    - Run puppetd out of cron at different intervals across all of your boxes. Make sure you pass the following option to puppetd:
      - --onetime

Advanced Knowledge

External Nodes

- Advantages:
  - Removes the need for nodes.pp
    - The same information contained in nodes.pp could be pulled from an Asset Management and/or Provisioning system.
    - You can just point Puppet at this external nodes source, which will be the central location for node information.

Advanced Knowledge

External Nodes (Cont.)

- Sources
  - Asset Management system
  - Provisioning system
    - Cobbler
      - Allows for integration with Configuration Management systems such as Puppet
      - Goal is to link Provisioning with Configuration Management
  - LDAP

Questions?

Try to stump the Puppet guru

SedaSystems™ We Know IT™

Contact

Mike Seda
CEO, Seda Systems, Inc.

P: 919.627.1260
E: mike@sedasystems.net
W: www.sedasystems.net

SedaSystems™
We Know IT™