Managing Systems with Puppet

Eric Eisenhart
<freiheit@socosa.org>

SoCoSA
August 7, 2007

Overview

- Overview
- Who's Eric?
- What Is Puppet?
- Comparisons
- Client, Server, and Protocol
- Language
  - More Language...
  - Some more Language...
- Real Examples
- Q&A

Who's Eric?
Why does he want to automate?

- Unix user for 15 years
- Unix sysadmin for 12 years
- Currently: Lead Unix sysadmin at SSU
  - ~80 *nix servers
    - ~half highly customer visible (announce outages, etc.)
    - Mostly RHEL; a little RHL (gone soon), a few Solaris
  - 1.5 admins (other unix admin is also storage admin)
  - Recent purchases: about a dozen more servers
    - Expected this FY: another 9 for sure
    - plus another dozen or so likely?
    - plus more test systems?

What Is Puppet?

- A declarative language for expressing system configuration
- a client & server for distributing it
- a library for realizing the configuration
- an open development community
  - With a 2-person company behind it

Huh? What's that mean?

- A way to manage many computers without touching them
- Make the computers do the boring work and concentrate on interesting things

Comparable Solutions

- cfengine
- Bcfg2
- lcfg
- radmind
- Opsware
- BladeLogic
- Microsoft SMS
- Netdirector
- IBM Tivoli

Incomparable Solutions

- shell script loops
  - ```
    for host in `cat rhel4boxes`; do
      ssh root@$host \
        perl -pie 's/foo/bar/' /etc/baz
    done
    ```
- Installation-time scripting (kickstart)
- Crazy RPM tricks
  - ```
    %post
    perl -pie 's/foo/bar/' /etc/baz
    ```
- Disk imaging (dd, ghost, etc.)

Client, Server, and Protocol

- Protocol: SSL; CA builtin
- Server: listens on one master
- Client: daemon runs on many nodes and wakes up every 30 minutes
- factsync, pluginsync, reports

Client <-> Server

- Client wakes up
- Client connects to server
- Client sends facts to server
- Server compiles configuration
- Server sends configuration
- Client loads configuration
- Client runs needed transactions
- (Optional) Client fetches more stuff from server
- Client sends report to server

Client

- Transactional
- Idempotent
- Modular
  - Resource Types <-> Resource Providers
  - Reusable

Library

- Ruby
- Reusable
- Swappable
- Extensible
  - new types can be just one ruby file
  - plugins!

Language: structure

- nodes
- inheritance
- classes
- types
- definitions
- Lions, Tigers and Bears!

Language: Types

- Any Unix:
  - cron, exec, file, group, host, mount, package, service, sshkey, tidy, user
- Exclusive:
  - yumrepo, zone
- Special:
  - schedule, filebucket

Metaparameters

- name
- alias
- before & require
- notify & subscribe
- schedule
- tag

Type examples

- ```
  cron { logrotate:
    command => “/usr/sbin/logrotate”,
    user    => root,
    hour    => 2,
    minute  => 0,
  }
  ```
- `package { “kernel”: ensure => latest }`
- `host { “foo.org”: ip => “10.2.5.2” }`
- `user { “eric”: ensure => present }`
  ```

Type examples: exec

- ```
  exec { “make stuff”:
    cwd     => “/nfs/example/foo”,
    creates => “/nfs/example/foo/stuff”,
    require => Mount[“/nfs/example/foo”],
  }
  ```
- command, creates, cwd, env, group, logoutput, onlyif, path, refresh, refreshonly, returns, timeout, unless, user
  ```

Conditionals

- ```
  file { "/some/file":
      owner => $os ? {
          sunos   => "adm",
          redhat  => "bin",
      },
      mode => 0755, owner => root,
  }
  ```
- ```
  case $operatingsystem {
    sunos:   { include solaris }
    redhat:  { include redhat  }
    default: { include generic }
  }
  ```

Definitions

- ```
  define apache::virtual_host($docroot, $ip, $order =
  500, $ensure = "enabled") {
    $file = "/etc/sites-available/$name.conf"
    # The template fills in the docroot, ip, and
  name.
    file { $file:
      content => template("virtual_host.erb"),
      notify  => Service[apache]
    }
    file { "/etc/sites-enabled/$order-$name.conf":
      ensure => $ensure ? {
        enabled  => $file,
        disabled => absent
      }
    }
  }
  ```
  ```

Language: classes

```
class ntp {
  file { "/etc/ntp.conf":
    source => [
      "puppet://$puppetserver/ntp/ntp.conf.$hostname",
      "puppet://$puppetserver/ntp/ntp.conf"
    ],
    notify => Service[ntpd],
  }

  service { "ntpd":
    ensure => running,
    enable => true,
  }

  package { “ntp-server”: ensure => installed }
}
```

Templates (ERB)

- ```
  $backupserver = [ “foo”, “bar” ]
  $backupclient = “baz”;
  file { "/opt/openv/.../bp.conf":
    content => template("nbp/bp_conf.erb")
  }
  ```
- ```
  # HEADER: Do not edit on live system.
  # HEADER: Look in puppet instead.
  <% backupservers.each do |server| -%>
  SERVER = <%= server %>
  <% end -%>
  CLIENT_NAME = <%= backupclient %>
  ```
- See also: generate()

Modules

- ```
  # cd /etc/puppet/modules/netbackup_client
  # find . | grep -v CVS
  .
  ./templates
  ./templates/bp_conf.erb
  ./README
  ./manifests
  ./manifests/init.pp
  ./files
  ./files/NET_BUFFER_SZ
  ```
  ```

Bringing it all together

```
node obsidian inherits typicalserver {
    include apache
    virtualhost { “www.example.org”:
        ip      => 10.2.5.7,
        docroot => “/var/www/example.org/htdocs”,
    }
}

node default inherits typicalserver {}

node typicalserver {
    include $operatingsystem
    include security
    include ntp
    include ssh
}
```

Show and Tell

Q&A