Expanded Introduction to Puppet

2011-04-03
Flourish
Chicago, IL

Garrett Honeycutt
Professional Services Consultant
garrett@puppetlabs.com
http://linkedin.com/in/garretthoneycutt
Puppet Open Source Ecosystem

Puppet Distribution
Bundled with major OS

Puppet Community
Active participation

Puppet Contributors
Framework enhancements

3,000 person mailing list
2,000 messages a month
300 people at all times in IRC
(#puppet on freenode.net)

100+ people contributing to documentation and code
100+ modules contributed to Puppet Module Forge

Ubuntu, RedHat, Debian

100+ people contributing to documentation and code
100+ modules contributed to Puppet Module Forge
Puppet Enterprise

What it is:

• Puppet and related components packaged and integrated in one install:
  • Puppet
  • Puppet Master
  • Dashboard
  • Facter
  • Ruby
  • Apache
  • Passenger, etc.
Puppet Enterprise

• Fully QA’d stack of Puppet and dependencies
• Simplified installation
• Ease of maintenance
• Pre-configured for scalability and performance
• Predictable enhancement delivery
• Enhanced enterprise class Support
Puppet Enterprise

What you can expect in the future:

• Pre-loaded set of commonly used modules
• Direct integration with public module-forge
• Support for additional platforms
• Integration with MCollective
MCollective adds orchestration

MCollective sequences actions based on data in Puppet

**Scenario 1 (Complete Process)**
- **Step 1**: Remove server group 3 from load balancers
- **Step 2**: Upgrade server on group 3
- **Step 3**: Verify monitoring status of group 3
- **Step 4**: Put group 3 back in load balancers

**Scenario 2 (Across Server Groups)**
- **Step 1**: Prevent Puppet runs for machines of type 5
- **Step 2**: Restart Apache on all machines of type 2
- **Step 3**: Update machines of type 4 in batches of 50 every 2 hours
- **Step 4**: Schedule mail delivery of type 4 in batches of 50 every 2 hours
Puppet is Pervasive

Web
- Google
- eBay
- digg
- twitter
- zynga

Entertainment
- Disney
- Turner
- HBO
- PlayStation

Technology
- Apple
- Nokia
- Motorola
- Juniper Networks
- Siemens
- Dell
- Rackspace

Financial
- Barclays Capital
- TransUnion
- Bank of America
- J.P. Morgan
- NYSE Euronext
- D.E. Shaw & Co

Defense
- Lockheed Martin
- Northrop Grummam
- Los Alamos National Laboratory
How Puppet Works

1. Define: With Puppet’s declarative language you design a graph of relationships between resources within reusable modules. These modules define your infrastructure in its desired state.

2. Simulate: With this resource graph, Puppet is unique in its ability to simulate deployments, enabling you to test changes without disruption to your infrastructure.

3. Enforce: Puppet compares your system to the desired state as you define it, and automatically enforces it to the desired state ensuring your system is in compliance.

4. Report: Puppet Dashboard reports track relationships between components and all changes, allowing you to keep up with security and compliance mandates. And with the open API you can integrate Puppet with third party monitoring tools.

- Current State
- Desired State
Use Puppet to create composable configurations and manage the enterprise infrastructure

1. Define Your Resources in Modules.
   With Puppet, you define your modules by node classifications, such as Web Server or Database, allowing you to define relationships between resources and configure thousands of servers at once.

2. Assign resource relationships automatically.
   You can then assign and deploy configurations via Puppet Dashboard, or with your own customized CMDB tools.

3. Reusable, composable configurations.
   With Puppet you can re-use modules across multiple nodes, in whatever combination you need, reducing repetitive tasks and eliminating error-prone scripts.

Multi Node
Puppet Assigns and Maintains a Node’s Desired Role
Managing Configuration Drift

Diagram showing the relationship between Node State, Desired State, Convergence, Drift, Report, and Report Server.
How Puppet Manages Data Flow for Individual Nodes

1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   - Puppet’s open API can also send data to third party tools.

SSL secure encryption on all data transport
Facts

Automatically Maintained Asset Inventory
domain => local
facterversion => 1.5.8
fqdn => sliver.local
hardwaremodel => i386
hostname => sliver
interfaces => lo0,gif0,stf0,en0,en1,fw0,vmnet1,vboxnet0
ipaddress => 192.168.174.1
ipaddress_lo0 => 127.0.0.1
ipaddress_vmnet1 => 192.168.174.1
kernel => Darwin
kernelmajversion => 10.6
kernelrelease => 10.6.0
macosx_productname => Mac OS X
macosx_productversion => 10.6.6
netmask => 255.255.255.0
netmask_lo0 => 255.0.0.0
netmask_vmnet1 => 255.255.255.0
network_lo0 => 127.0.0.0
network_vmnet1 => 192.168.174.0
operatingsystem => Darwin
operatingsystemrelease => 10.6.0
path => /opt/local/bin:/opt/local/sbin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin: /
usr/X11/bin:/Users/gh/bin:/Users/gh/.gem/ruby/1.8/bin/
ps => ps auxwww
puppetversion => 2.6.4
rubysitedir => /opt/local/lib/ruby/site_ruby/1.8
timezone => PST
uptime => 1 day
rubyversion => 1.8.7
sp_bus_speed => 1.07 GHz
Custom Facts

# role.rb
require 'facter'
Facter.add("role") do
  setcode do
    Facter::Util::Resolution.exec("cat /etc/role")
  end
end
How Puppet Manages Data Flow for Individual Nodes

1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **SSL secure encryption on all data transport**

3. **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   - Puppet’s open API can also send data to third party tools.

   **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.
Catalog

- Automatically maintained comprehensive resource list
- Easily validated against compliance requirements prior to client configuration
How Puppet Manages Data Flow for Individual Nodes

1. **Facts**
   - The node sends normalized data about itself to the Puppet Master.

2. **Catalog**
   - Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.

3. **Report**
   - The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.

4. **Report Collector**
   - Puppet’s open API can also send data to third party tools.

SSL secure encryption on all data transport.
Report

- Comprehensive report of every change ever made, correlated to every resource being managed
- Easily validated against compliance requirements after each run
Report

- http
- log
- rrdgraph
- store
- tagmail
Report
The one-off myth

Your systems are not beautiful snowflakes

photo from http://beesknees67.deviantart.com/
The one-off myth

- Only temporary
The one-off myth

- Only temporary
- Replicas for pre-production environments
The one-off myth

- Only temporary
- Replicas for pre-production environments
- Disaster recovery
Why?

Hosts on demand
Why?

- reduce entropy
Why?

• reduce entropy
• disaster recovery
Why?

• reduce entropy
• disaster recovery
• change management
Why?

• reduce entropy
• disaster recovery
• change management
• infrastructure as code
What not How

```ruby
package {  
  "ntp":
  ensure => installed,
}
```
What not how

```bash
$ ls puppet/lib/puppet/provider/package
aix.rb  blastwave.rb  hpux.rb  ports.rb  up2date.rb
appdmg.rb  darwinport.rb  nim.rb  portupgrade.rb  urpmi.rb
apple.rb  dpkg.rb  openbsd.rb  rpm.rb  yum.rb
apt.rb  fink.rb  pkg.rb  rug.rb  yumhelper.py
aptitude.rb  freebsd.rb  pkgdmg.rb  sun.rb  zypper.rb
aptrpm.rb  gem.rb  portage.rb  sunfreeware.rb
```
Example Resource Types

- cron
- exec
- file
- group
- host
- zfs
- mount
- package
- service
- sshkey
class ntp {
  package {  "ntp":
    ensure => installed,
  }

  file {  "/etc/ntp.conf":
    owner    => "root",
    group    => "root",
    mode     => "0644",
    require  => Package["ntp"],
  }

  service {  "ntpd":
    ensure   => running,
    enable   => true,
    subscribe => File["/etc/ntp.conf"],
  }
}
class motd {
    file { 
        "/etc/motd": 
        owner  => "root",
        group  => "root",
        mode   => "0644",
        source => "puppet:///modules/motd/generic_motd",
    }
}
Templates

# motd.erb

Welcome to `<%= fqdn %>`
My uptime is `<%= uptime %>`
Templates - Advanced

search <%= dnssearchpath %>
options ndots:2 timeout:3
<% nameservers.each do |nameserver| -%>
  nameserver <%= nameserver %>
<% end -%>
Syntax Checking

```
# puppet template syntax checking
function pt()
{
  if [ -z $1 ]; then
    echo "usage: pt <puppet_template_file.erb>"
    return;
  fi
  /usr/bin/erb -P -x -T '-' $1 | /usr/bin/ruby -c
}

# puppet manifest syntax checking
function pc()
{
  if [ -z $1 ]; then
    puppet --ignoreimport --parseonly init.pp
  else
    puppet --ignoreimport --parseonly $1
  fi
}
```
Storeconfigs

Ability to pass data between nodes, via a database acting as a proxy
Storeconfig

Ability to pass data between nodes, via a database acting as a proxy

• MySQL
• SQLite3
• PostgreSQL
• Oracle
Storeconfigs

- Puppet node
  - exporter
- Puppet node
  - collector 2
- Puppet node
  - collector 1

- @@host{}
- Host<<ll>>
- Host<<ll>>

- storeconfigs
- Puppet Master
External Node Classifier

- Puppet Dashboard
- Your own CMDB
External Node Classifier

A script that takes $certname as an argument and outputs YAML to STDOUT
External Node Classifier

```
node "somenode.puppetlabs.net" {
    $puppet_server = "puppet.puppetlabs.net"
    $dns_server = "ns.puppetlabs.net"
    $datacenter = "slicehost"
    include common, puppet, dns, ntp
}
```
External Node Classifier

classes:
- common
- puppet
- dns
- ntp

parameters:
puppet_server: puppet.puppetlabs.net
dns_server: ns.puppetlabs.net
datacenter: slicehost