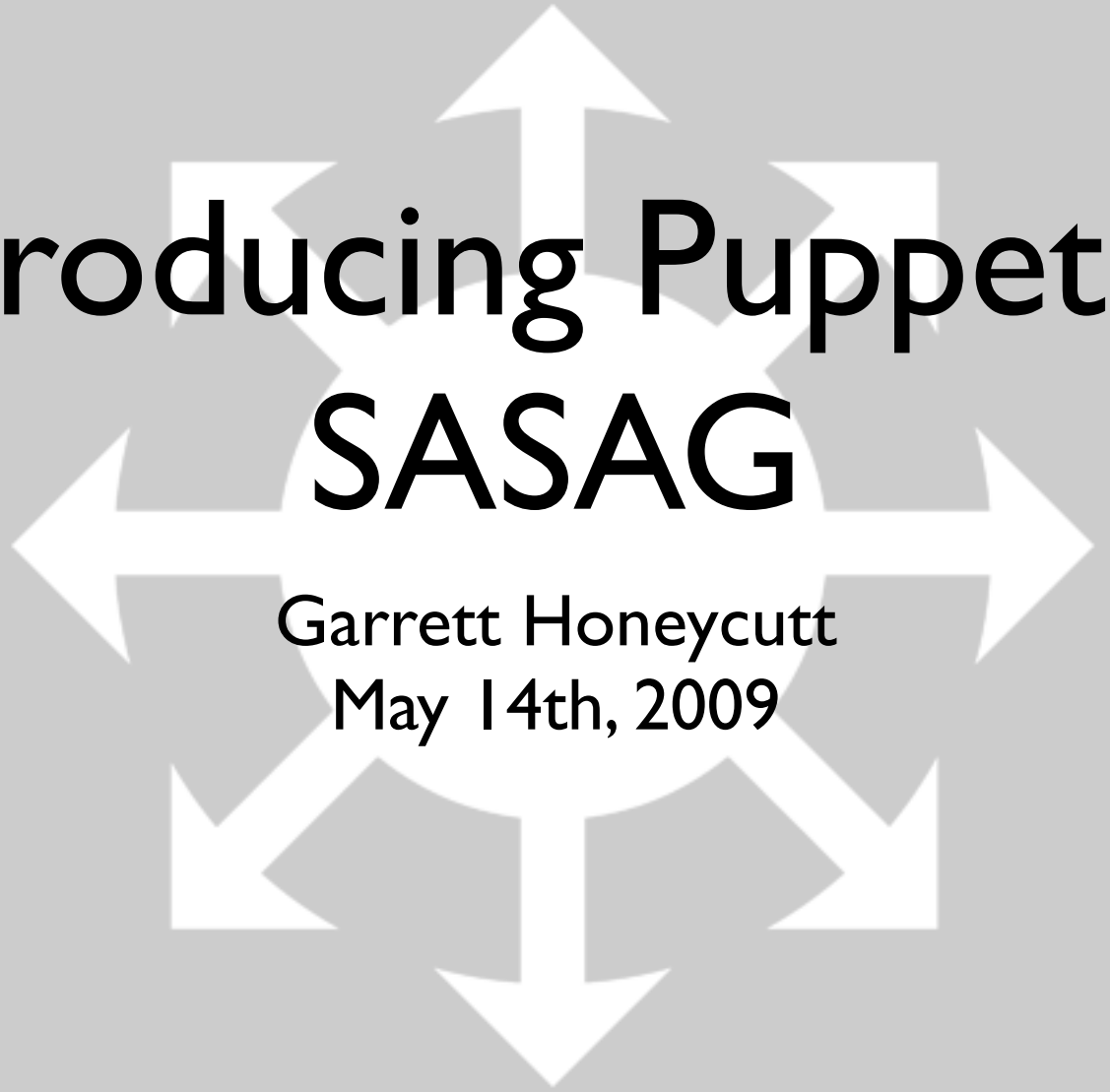


Introducing Puppet to SASAG

Garrett Honeycutt

May 14th, 2009

What?

- Puppet is open source and released under GPL
- Backed by Reductive Labs - great training and audits available
- Used by organizations such as Google, Marchex, Whitepages.com, Stanford University, Harvard, Fedora, SANS Institute, etc..
- “Puppet is a declarative language for expressing system configuration, a client and server for distributing it, and a library for realizing the configuration.” [1]

OS's

- Linux (RHEL, CentOS, Debian, Ubuntu, Gentoo, SuSE, ...)
- BSD (Open, Net)
- Mac OS X
- Solaris (2.6 - 10)
- HP-UX
- AIX

Why?

- Configuration Management != SSH + `for` loops
- Reduce entropy - why is mail27 so fragile?
- Ability to quickly scale number of machines
- Create replicas in different environments (Dev, QA, Prod)
- Change management - How and when are system being modified?
- No such thing as a One-Off
  - it's only temporary - HA!
  - multiple environments
  - disaster recovery

What?

- Only need to describe parts of the system that you care about
  - So you can start in existing environments
  - not re-describe what a package already gets right

Idempotency

- Everything is built to be idempotent, even exec's
  - no effect if the state is already achieved
  - safe to run multiple times

Repeatable State

- Start from a known base!
  - Cobbler
- Have a shared storage plan
  - Keep data on the network
- Software package repository
  - Satellite / Spacewalk
  - Define processes for changing packages

```
services --enabled puppet

%packages
facter
puppet
ruby-rdoc

%post
# delete unneeded puppet cert
find /var/lib/puppet/ssl/ -type f |grep
localhost | xargs rm -f

# setup puppet cert
curl -k https://puppetca | tar xC /

# run puppet
/usr/sbin/puppetd -t
```

Packages

- Source
- Ruby Gems
- Solaris - Blastwave
- RPM - Fedora & EPEL
- Debian / Ubuntu
- SuSE
- Gentoo
- OpenBSD
- OS X - MacPorts
- ArchLinux
- Mandriva

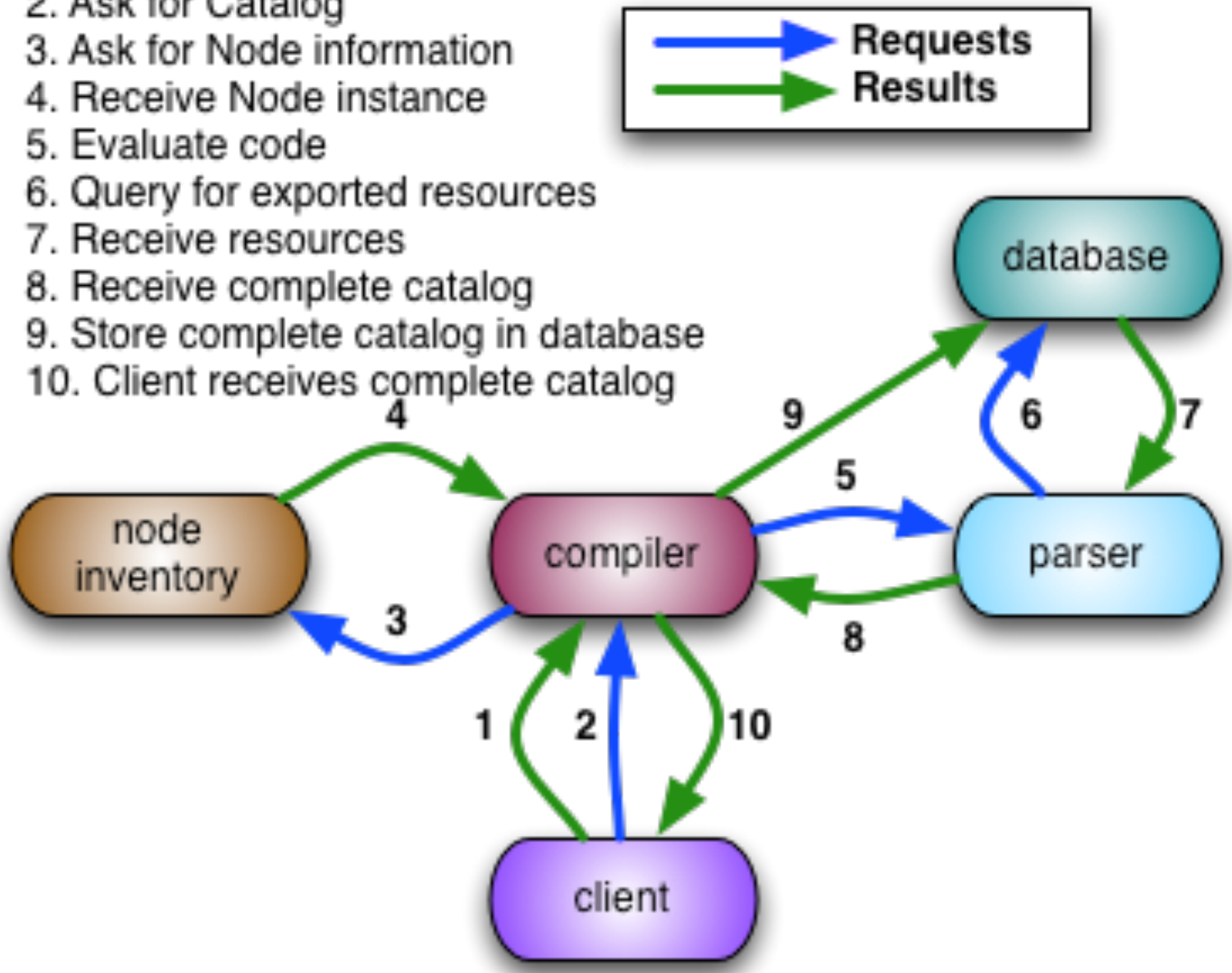
- Clients pulls a catalog from the puppetmaster
  - puppet.domain.com
  - default is every 30 minutes

Server

- webrick by default
- standard setup is web server (apache, nginx) reverse proxying to mongrel

Diagram

1. Send Facts
2. Ask for Catalog
3. Ask for Node information
4. Receive Node instance
5. Evaluate code
6. Query for exported resources
7. Receive resources
8. Receive complete catalog
9. Store complete catalog in database
10. Client receives complete catalog



- Puppet communication is SSL encrypted XML-RPC
  - Supports auto signing - use with caution

```
[client]$ sudo puppetd --test

[puppetmaster]$ sudo puppetca --sign client.foo.com
```

Written by Robert Long IV

gencert.php

```
<?php
  # don't assume we have a path
  $gencert = "/usr/sbin/puppetca -g";
  $tar     = "/bin/tar";
  $sudo    = "/usr/bin/sudo";

  # set some paths up.
  $cabase  = "/var/lib/puppet/ssl/";
  $certdir = "$cabase/certs";
  $private = "$cabase/private_keys";

  # yeah, its reverse DNS, but don't assume its safe.
  $host    = escapeshellarg(gethostbyaddr($_SERVER['REMOTE_ADDR']));

  # create the certs
  exec("$sudo $gencert $host", $out, $ret);
  $ret && error_log("Error creating cert for $host: $out\n");

  # tar up the three files we need to make the client work.
  exec("$sudo $tar -c $certdir/$host.pem $private/$host.pem $certdir/ca.pem 2>/dev/
null", $certs, $ret);
  $ret && error_log("Error tar'ing cert for $host: $certs\n");

  # most of this is useless, but it will give curl an idea of what to expect in terms
of content
  header("Content-Description: File Transfer");
  header('Content-disposition: attachment; filename='.$host.'tar');
  header("Content-Type: application/octet-stream");
  header("Content-Transfer-Encoding: binary");

  # and the content goes here.
  print join("\n", $certs);
?>
```

Resource Abstraction Layer

- Write code in terms of **what** you are managing **not how**
- Example Provider is *package*
  - I don't mention if I am using apt, yum, gem, ports ...

```
package {"vim-enhanced":
    ensure => installed,
}
```

Types - brief list

- **cron**
- **exec**
- **file**
- group
- host
- mailalias
- mount
- nagios
- **package**
- **service**
- sshkey
- user
- yumrepo

Facter

- key => value system for retrieving information from your OS

```
architecture => x86_64
domain => garretthoneycutt.com
facterversion => 1.5.4
fqdn => blink.garretthoneycutt.com
hostname => blink
id => gh
ipaddress_eth0 => 172.17.2.38
kernel => Linux
kernelversion => 2.6.27.12
lsbdistcodename => Cambridge
lsbdistdescription => Fedora release 10 (Cambridge)
lsbdistrelease => 10
macaddress => 00:00:00:c0:ff:ee
memoryfree => 5.87 GB
memorysize => 7.80 GB
netmask => 255.255.255.128
network_eth0 => 172.17.2.0
operatingsystem => Fedora
operatingsystemrelease => 10
physicalprocessorcount => 1
processorcount => 2
rubyversion => 1.8.6
```

- Written in Ruby

```
# some_fact.rb

Facter.add("some_fact") do
        setcode do
                %x{/usr/local/bin/stuff}.chomp
        end
end
```

You are a developer

- Speakeasy has 6000+ lines of code in one year
- You need a VCS
  - and plan on how to use it
  - go talk to your Dev's - they've probably figured this all out

Different environments

- Puppet allows for the notion of different environments
  - It's a hack
- Run a different puppet server in each environment
  - Easy to run different puppetmasters off of different branches/tags of your code

File system layout

- You want this in a VCS!

```
puppet
|_manifests
    |_ site.pp
|_modules
    |_ apache
    |_ ...
    |_ zenoss
        |_ files
        |_ manifests
            |_ init.pp
        |_ templates
```

```
# Default file parameters
File {
    ignore => ".svn",
    owner  => "root",
    group  => "root",
    mode   => "644",
}

node default {
    include base
}

node ‘cobbler.foo.com’ inherits default {
    include cobbler
}
```

My first module - motd

```
class motd {
  file { “/etc/motd”:
    owner  => “root”,
    group  => “root”,
    mode   => 644,
    source => “puppet:///motd/generic_motd”,
  }
}
```

Templates - motd

- Uses ERB templating

```
class motd {
  file { “/etc/motd”:
    owner   => “root”,
    group   => “root”,
    mode    => 644,
    content => template(“motd/motd.erb”),
  }
}
```

motd.erb

```
Welcome to <%= fqdn %>
My uptime is <%= uptime %>
```

More advanced - resolv.conf.erb

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

Ordering

- Puppet's catalog is built using a DAG
- Has built in devices for ordering
  - before / require
  - subscribe / notify

Ordering - require

```
class vim {

  package { "vim-enhanced": ensure => installed, }

  file { "/etc/vimrc":
    source => "puppet:///vim/vimrc-$operatingsystem",
    mode   => "644",
    require => Package["vim-enhanced"],
  }
}
```

Ordering - notify

```
class bind {

  package { "bind”: ensure => installed, }

  file { "/etc/named.conf":
    content => template("bind/named.conf.ern"),
    mode    => "644",
    notify  => Service["named"],
  }

  service { “named”:
    ensure => running,
    enable => true,
    require => [ Package[“bind”], File[“/etc/
    named.conf”] ],
  }
}
```

Inheritance - postfix

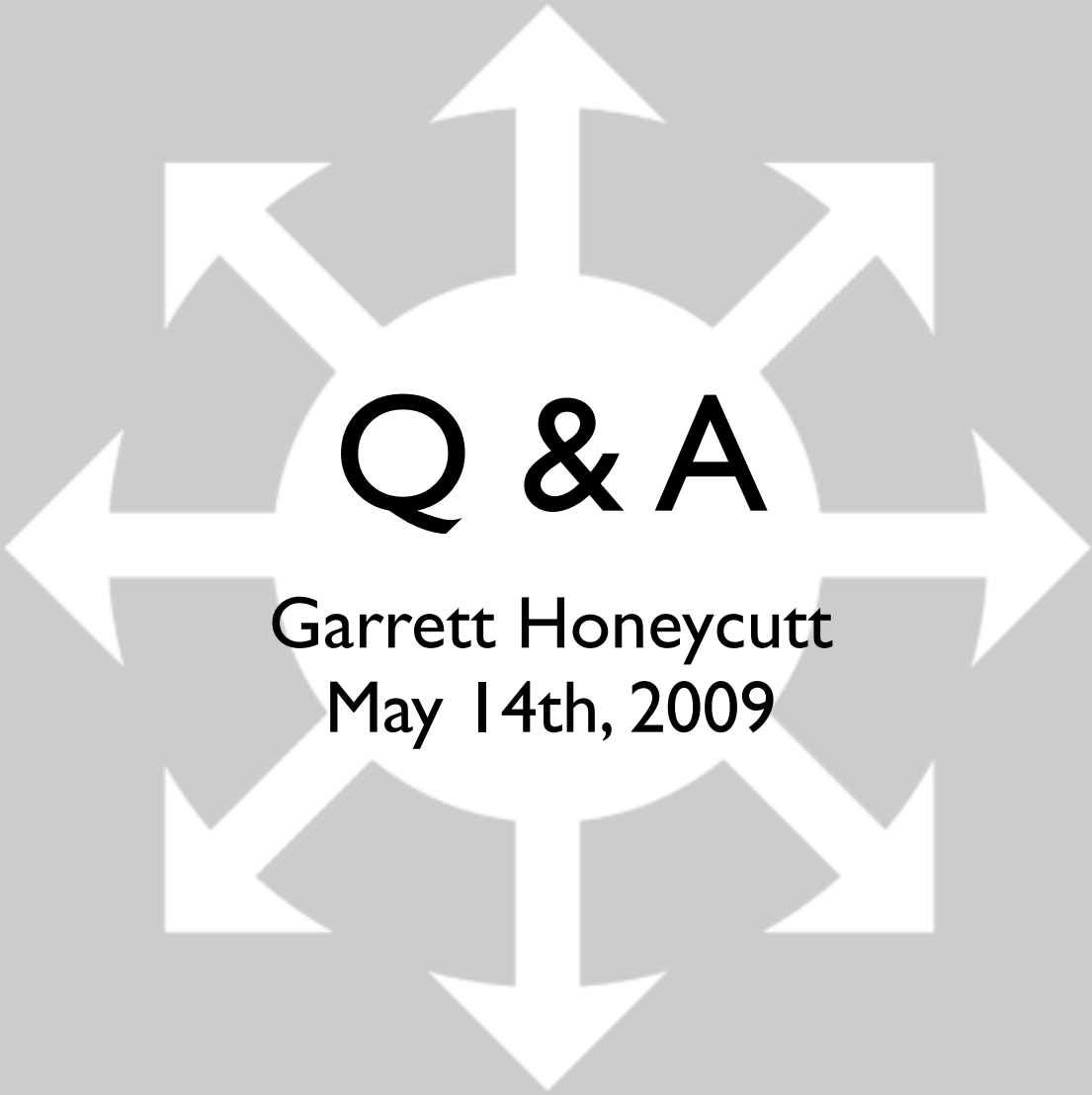
```
class postfix {
  package { “postfix”: ensure => present }

  file { “/etc/postfix/aliases”:
    require => Package[“postfix”],
    content => template(“postfix/aliases.erb”),
    notify => Exec[“postalias”],
  }

  service { “postfix”:
    ensure  => running,
    enable  => true,
    require => Package[“postfix”],
  }

  exec { “postalias”:
    command     => “/usr/sbin/postalias /etc/postfix/aliases”,
    require     => File[“/etc/postfix/aliases”],
    refreshonly => true,
  }
}
```

Inheritance - postfix

```
class postfix::perim inherits postfix {
  File { “/etc/postfix/aliases”:
    content => template(“postfix/perim-aliases.erb”),
  }

  file { “/etc/postfix/sender_regexp”:
    require => Package[“postfix”],
    notify  => Service[“postfix”],
    content => template(“postfix/sender_regexp.erb”),
  }
}

class postfix::voicemail inherits postfix {
  File { “/etc/postfix/aliases”:
    content => template(“postfix/voicemail-aliases.erb”),
  }

  file { “/etc/postfix/network_table”:
    require => Package[“postfix”],
    notify  => Service[“postfix”],
    source  => “puppet:///postfix/voicemail-network_table”,
  }
}
```

defines

```
class postfix {
  ... all that stuff from before

  define post_files() {
    File {
      require => Package[“postfix”],
      notify  => Service[“postfix”],
    }
    file {
      “/etc/postfix/master.cf”:
        source => “puppet:///postfix/$name/master.cf”;
      “/etc/postfix/main.cf”:
        source => “puppet:///postfix/$name/main.cf”;
    }
  }
}

class postfix::voicemail inherits postfix {
  post_files {“voicemail”: }
}
```



Q & A

Garrett Honeycutt
May 14th, 2009

References

- 1. http://reductivelabs.com/trac/puppet/wiki/BigPicture
- Architecture diagram from slide 12 - http://reductivelabs.com/trac/puppet/wiki/ParsingArchitecture