Creating a Mature Puppet System

github.com/rkhatibi/puppetcampla2013
Thanks Scale 11x & PuppetLabs

● Attended 3x (or was it 4?)
● Discovered PuppetLabs at 8x
● I like Puppet (more sleep, better work)
Hi, My name is: Ramin

- Sysadmin for seventeen years
- Currently at SnappyTV
- Yahoo!, Netzero
- Half dozen startups
- Not a ninja or rockstar
SnappyTV

- Cloud based video editing
- Immediate publishing
- Real time social media data consumption and analysis
Mature?

- Operable (by more than 1 person)
- Consistent (updates and fresh installs)
- Flexible (hold on while I refactor, again)
- Enjoyable (the opposite of frustrating)
Getting There

● Process (remember less, do more)
● Technique (sneaky tricks)
● Documentation (words, boring words)
● Experimentation (aka failure)
Are you ready to write code?

- Software development
- Your environment is important
- Take a few hours to set it up
Choose Your Weapon (Editor)

- No need to change
- Add plugins
- post commit tools
- Some choices are more mature
Syntax Highlighting

- `git commit -m 'missing comma'`
- `git commit -m 'missing quote'`
- `git commit -m 'I hate my life!!!'`
- Defense in depth
Code Snippets

- Add code **easily**
- Reminders for resource **types**
- **Config** to your usage
It's puppet-lint

- It's opinionated, use it anyway
- Chokes on complex quoting
- .puppet-lint.rc
- When in doubt, do what it says
Code Style is Important

- Try to decide on one early
- Use the Puppet Style Guide
- aka puppet-lint
- Consistency is always good
Validate Your Code

- puppet parser validate some.pp
- Doesn't catch everything
- Or work on templates
All Together With VIM

- vim + pathogen
- syntastic, tabular
- vim-puppet, puppet-lint
- mv-vim-puppet
Exists for Other Editors Too

- Emacs (for terrible people)
- Sublime (didn't look mature)
- Eclipse (very nice)
- Anything else?
Vagrant for VMs

- Your experimentation system
- spin VMs up, test, destroy
- Cost of failure is very loooow
Testing in Puppet

- Test from a fresh install
- Easy to miss dependencies
  Nothing worse than discovering ordering problems in prod
Puppet Environments

- Use them, use them, use them
- stage and production
- directories on the master
- --env stage from client
Promote Code to Each Env

- Standard development practice
- devel -> stage -> prod
- There are some caveats
Environment

Caveats

- Providers and facts **leak**
- Best to **have** an env per Puppet master instance in adv usage
- This may change (I **hope**)

SnappyTV™
Setup Simple Environments

- puppet.stage cname puppet02
- puppet cname puppet01
- Push to one, then the other
Clients to Env

- Just add to **puppet.conf**
- **Ideally** part of template
- production env is **default**

```
[agent]
<% if @fqdn =~ /(.*)stage(.*)/ -%>
environment = stage
<% end -%>
```
Watch Paths

- Might need to **rearrange** your repo or push **process**
- Where will **auth.conf** live?

```
./puppet/production/modules/
./puppet/stage/modules/
./puppet/auth.conf
./puppet/hiera.yaml
```
Pushing Your Code

- puppet_push stage
- puppet_push prod
- rsync, fabric, capistrano, etc
- **Steal** from your Developers or reuse your normal process
Sync your plugins

- `--pluginsync` from cli
- `pluginsync=true` under [main]
- Default in 3.x
Puppet Master

- Is it ready for production traffic?
- Apache/Passenger is common
- **Upgrade** to Passenger 3.0.x
- debs/rpms **available**
Tuning Passenger

- MaxPoolSize = CPU Cores × 2
- MinInstances = CPU Cores
- RAM may limit this
- Each Puppet/Rack = 200MB (ish)
More Tuning Passenger

- Use `vhost` or `passenger.conf`
- `vhost` if `sharing` machine
- `PassengerPreStart <url>`
- multi Ruby instances in 4.x
Apache Tuning

- mpm-worker > prefork
- should "just work"
- more threads if > 8 cores
- nginx/passenger also an option
Other App Servers

● Little personal experience
● Not worth it in my opinion
● Use what you know best
Isolate the Master

● Easier to manage
● Quite easy to do
● Less likely to make mistakes
Like These Problems

- `certname = hostname (no! no!)`
- `rm -rf /var/lib/puppet/ssl`
- `puppet:puppet vs root:root`
Split Your Modules Too

- include *puppet*
- include *puppetmaster*
- shared *nothing* (almost)
Create Dir Structure

- `mkdir -p ./puppet/{etc,rack,var}`
- `./puppet/pm.conf`
- All in one tree
For Masters,

pm.conf

- no complicated concat
- config.ru is the entry point
- ARGV << "--config=pm.conf"
- Also takes other arguments
[main]
confdir=/home/$some_user/puppet/etc
logdir=/home/$some_user/puppet/logs
vardir=/home/$some_user/puppet/var
ssldir=$vardir/ssl
rundir=/home/deploy/puppet/run
factpath=$vardir/lib/facter
templatedir=$confdir/templates
Simple to Backup

- `sudo tar -czvf p.tgz ./puppet/`
- **that's it**
- **ignore** reports
- **always backup** certs
Can Re-Use Locally

- `rvm`, `ruby`, `gem install puppet`
- `mini` puppet environment
- test new setups `without` affecting the rest of the server
Master Monitoring

- https:8140
- At least one Rack process
- logwatch
- ask for a catalog
Client Monitoring

- Daemon **running** (or not)
- `last_run_summary.yaml`
- **Easy** to parse
- simple **check** in my github
You Can't Escape Crons

- **delete** those reports
- couple of **days** is fine
- **prune** nodes in Dashboard
- PuppetDB (**not sure yet**)
Mysql Tuning

● Default `my.cnf` is useless
● Do `at least` the following
● Also prune tasks (`rake -T)

```
innodb_buffer_pool_size = 512M
innodb_file_per_table = 1
key_buffer = 32M
```
Certs, not that complicated

- **Master** cert
- **Client** cert
- **Application** cert
- `/etc/hosts` is not a solution.
Master Cert

- **Multiple** names
- Your **clients** don't care
- Migrations are **easy**

```plaintext
[master]
certname = puppet.example.com
dns_alt_names = puppet, puppet.new, puppet.old, spam, puppet.localdomain, baked_beans, puppet, puppet, spam, puppet.localhost
```
App Certs

- Dashboard, PuppetDB, etc
- $your_app ?
- auth.conf matters

```bash
$ curl --cert $my_app_cert.pem --key $my_app_private_key.pem -k -X DELETE -H "Accept: pson" https://puppet.example.com:8140/production/certificate_status/$myhostname
```
Useful Cert Commands

- `client`, `$ rm -rf /var/lib/puppet/ssl`
- `$ puppet cert list --all`
- `$ puppet cert clean $fqdn`
Invoking Puppet

- sudo service puppet restart
- not too useful in testing
- or provisioning
- need something ad hoc
Puppet Agent

- Pass environments, hostname
- Change facts too
- Useful for troubleshooting

```
$ sudo puppet agent
$ sudo puppet agent --server puppet --pluginsync

$ sudo FACTER_role=database_master puppet agent --certname dbm01 -tv

$ sudo puppet agent --server puppet.new --environment stage --certname test01
```
Puppet Apply

- good for **development**
- testing **without** a puppet master
- aka **masterless** Puppet

```
$ puppet apply -l ./test.log manifest.pp
$ puppet apply --modulepath=~/.puppet/modules -e "include ntp"
$ puppet apply --catalog catalog.json
```
Your multi-tool puppet-stdlib

- Does a bit of *everything*
- validate, replace, convert
- **Should** be a talk in its own right
puppet-stdlib validation

- One simple example
- Or I'll never finish this talk
- Really

```bash
if $order != " and !is_integer($order) {
    fail('Only integers are allowed in the apt::pin order param')
}
```
Towards a Better Module

- No god modules
- Each module is a discrete chunk of functionality
- Apply functionality as needed
Code vs Data

- Data and code separation
- `wordpress` => `db`
- `puppet` => `hiera`
- Manipulate `data`, not code
Why Separate?

- Your system will change
- versions, vhosts, aliases
- change code as little as possible
- portability and shared code
Write Less Code

- Default values in your modules still useful (if Debian do..)
- if { if { if { if { if { gah!
- Write once, feed data
Hiera, as in hierarchical

- **yaml** by default, json available
- **redis**, mongo, mysql, others
- your hierarchy **will** take a few tries to get right
Hiera, How does it work?

- Data position matters, it's hierarchy
- start at the top
- work your way down
- First match or collect
Hiera, the Mistakes

- hiera_array is not for arrays
- hiera_hash is not for hashes
- Just `hiera('some_var')`
Hierarchy

---

:hierarchy:
- %{fqdn}
- %{environment}/{%{role}}
- %{role}
- %{environment}
- common

hieradata/stage/frontend.yaml
hieradata/production.yaml
Hiera, a data example

---

apache_address: '127.0.0.1'
apt_server: 'apt.build.example.com'
facter_version: 'latest'
mysql_innodb: '256MB'
puppet_master: 'puppet.build.example.com'
puppet_version: 'present'
ruby_version: '1.8.7-p371'
Hiera, in a module

class facter::install {

    $version = hiera('facter_version','present')

    package { 'facter':
        ensure => $version,
        notify => Class['puppet::service'],
    }
}

Where to Concentrate?

- Execs < 5%
- Services ~ 10%
- Packages ~ 25%
- Files ~ 60% (Best use of time)
Manipulate files with...

- Ruby **ERB** templates
- Puppet **concat** module
- **Augeas**
Templates

- <%= I'm ruby, I execute code %>>
- <%= I'll print the output %>>

```ruby
server_id = <%= @ipaddress.split('.').inject(0) { |total, value| (total << 8) + value.to_i } %>
expire_logs_days = <%= scope.lookupvar('mysql::data::expire_logs_days') %>
<% if (scope.lookupvar('mysql::data::slaves')).include? @clientcert then %>
read_only = 1
<% end %>
```
puppet-concat

- **Download** from the Forge
- **remember** `pluginsync = true`
- Useful for daemons that don't support `configdirs`
- `sshd`, `rsync`, `haproxy` (sorta)
haproxy example

```erb
concat { '/etc/haproxy/haproxy.cfg': }

concat::fragment { 'haproxy_01_main':
  target  => '/etc/haproxy/haproxy.cfg',
  order   => '01',
  content => template('haproxy/haproxy.cfg.erb'),
}

define haproxy::configs ( $order = '10',) {
  concat::fragment { "haproxy_${order}_${name}":
    order  => $order,
    target => '/etc/haproxy/haproxy.cfg',
    source => "puppet:///modules/haproxy/${name}",
  }
}
```
Augeas

- Single line replacement
- usage is less common
- install the cli tools in devel
- make sure you have installed a recent version
Augeas is best for...

- Files you can't fully control
- Files you don't want to control
- Your last resort
- grub.conf, sysctl.conf
Start by reminding yourself

cat /etc/ntp.conf
# PUPPETHEADER: This file is owned by Puppet.

ls -a /etc/apache2/sites-enabled/
   .00_puppet_will_delete_files
   .01_that_are_not_directly_managed
   .02_by_puppet_you_have_been_warned
Advance Documentation

● Readme files in your modules
● with actual examples
● rdoc too
Thank You for Coming

https://github.com/rkhatibi/
https://twitter.com/Ramin_DK
http://www.snappytv.com/
Appendix A

- http://www.slideshare.net/PuppetLabs/130208-puppet4-sysadminsbelblirefinal
- Craig Dunn - http://www.slideshare.net/PuppetLabs/roles-talk
- http://blog.mozilla.org/it/2013/01/30/liveblog-how-to-use-puppet-like-an-adult/
Appendix C

- Passenger rpms - http://passenger.stealthymonkeys.com/
- Passenger debs - http://apt.brightbox.net/