Puppet for Developers

“Intermediate” SOE

That other talk

- Setup an SOE based on:
  - Repo Server with Kickstart; and
  - Puppetmaster;
- Write some modules for every day things:
  - files, users, some templates, etc;
  - very simple fact;
- Discouragement of using define to create types.
This talk

- Leveraging existing SOE to help other teams;
- Puppet to assist in deployment and maintenance of developing/evolving projects;
- Manage “complex” Puppet installations;
- Creating a real fact;
- Leverage Puppet’s programming features;
- Encouragement to use define for custom types.

Leveraging the existing SOE

- The SOE should allow quickly deployment of alike systems that meet a set specification.
- Others in organisation can benefit from access.
- Best to offer before “others” re-invent wheel.
Reuse, replicate and expand

• Kickstart environment:
  • build to closely matches yours;
  • but not so close that it restricts.
• Puppetmaster:
  • use environments; and
  • be involved in the evolution to production

The “Others”

• Primary focus is working with “Developers”; 
• Differ from most clients:
  • more than a handful of scripts;
  • not just a few service configuration;
  • not just one canned application;
  • bring own (custom) application(s) that need to integrate with the OS.
Help them!

• Developers are not System Administrators:
  • Trust system defaults:
    • Consistent UID / GID; and
    • Security (DAC) is optional or random;
    • Anything more serious (e.g.: SELinux and firewalls) are restrictive and thus disabled.
  • SOE probably has preferences or policies.
  • Changing their assumed settings late may break their work.

Assisting developing projects

• “Standard” Puppet is geared to make System Administrator work easier, right now;
• “Complex” Puppet is about making everyone’s life a little harder (use puppet = change), with a longterm payoff.
• It will pay off: this is standardising how to deploy custom applications to SOE hosts;
• Results in “appliances”.

“Standard” Puppet

- Puppet can easily manage:
  - servers that have a defined role;
  - adjustments to that role.
- ... and just as easily many different roles.
- Puppet maintains; it does not automatically evolve servers / projects.

“Complex” Puppet

- Supporting projects that are still evolving.
- Currently favours very dynamic practices.
- Virtual hosts can be rapidly:
  - modified;
  - multiplied;
  - torn down and
  - rebuilt.
Developers’ needs

• Multiple stages or classes of hosts;
• The ability to rapidly test new builds;
• Progress builds from test through to production;
• Make things work.

Sys Admins’ needs

• Hosts and services to keep running;
• Servers that can be maintained with confidence;
• Less people to have root access.
• Everything to keep running, no matter what.
These clearly conflict

• In short:
  • Developers need hosts that are flexible;
  • Sys Admins need hosts that are predictable.

Classes of hosts

• testing - anything goes, managed by “others”;
• development - “manything” goes, used for dev work. Not mission critical...
  • ... though sometimes part of the DR strategy.
• production - hands off.
define “testing”

- Desktop VMs;
- Proof of concept;
- very disposable;
- internal use only;
- “not your problem”;

simplified kickstart

- Using the production kickstart environment is usually not simple:
  - Restricted access;
  - Limited customisation;
  - Not something responsive to individual needs.
- Thus, build a cut down kickstart system.
SOE-like Desktop VMs

• Ideally uses the same:
  • partitioning;
  • base packages; and
  • repositories;

• But might use:
  • DHCP rather than fixed IP;
  • (possibly) reduced security settings;
  • (possibly) local users only;

These are for them

• Remember everyone’s expertise:
  • Developers’ deliverables will need to work on production hosts one day.
  • Communicate and work together.
Our solution: differences

- kickstart via url (rather than ISO);
- mostly identical ks.cfg but uses php;
- relies on DHCP rather than fixed IP and hostname.

Our SOE-like Desktop VMs

- same partitioning, base packages and repos;
- uses LDAP for authentication;
  - service users are always local;
- optional iptables and SELinux (as in production);
- some disabled SOE Puppet modules;
  - own Puppet modules sub-tree;
- on-host network with PFsense, DHCP and NAT.
KS via http

- Requires:
  - PHP installed (on the repo server);
  - CentOS 6.2 netinstall ISO (on target desktop);
  - ks.cfg which is copied to somewhere httpd can serve it and named index.php
- remember to restart httpd after installing php.

```php
<?php
$hostname = $_GET['hostname'];

install
#url --url http://192.168.1.5/mrepo/rhel6-server-x86_64/
url --url http://192.168.1.5/mrepo/CentOS6-x86_64/disc1
key --skip
lang en_US.UTF-8
keyboard us

network --device eth0 --bootproto dhcp --hostname <?php echo $hostname . PHP_EOL ?>

# password is kickstart
rootpw --iscrypted $1$5YF630$HDlrn.YYFUvtfPVwHDmdun0
firewall --enabled --port=22:tcp
authconfig --enablesystem --enablenetwork
selinux --enforcing
timezone Australia/Brisbane
```
bootloader --location=mbr --driveorder=sd0 --append=" rhgb crashkernel=auto
quiet"
clearpart --all --initlabel --drives=sd0
part /boot --fstype=ext4 --fsoptions="defaults,strictatime" --size=128 --ondisk=sd0
part pv.1 --size=100 --grow --ondisk=sd0
volgroup VolGroup00 --pesize=32768 pv.1
logvol / --fstype=ext4 --fsoptions="defaults,strictatime" --name=LogVol_root
--vgname=VolGroup00 --size=2048
logvol /usr --fstype=ext4 --fsoptions="defaults,strictatime" --name=LogVol_usr
--vgname=VolGroup00 --size=3072
logvol /home --fstype=ext4 --fsoptions="defaults,strictatime" --name=LogVol_home
--vgname=VolGroup00 --size=1024
logvol /var --fstype=ext4 --fsoptions="defaults,strictatime" --name=LogVol_var
--vgname=VolGroup00 --size=100 --grow

%packages
@Base
@Core
- NetworkManager
- NetworkManager-glib
- arts
@end

%post --nochroot
mkdir /mnt/sysimage/mnt/dvd
mkdir /mnt/sysimage/mnt/nfs
mkdir /mnt/sysimage/mnt/samba
%post
## Setup /opt
mkdir /var/root-opt ; chmod 755 /var/root-opt
mkdir /opt ; chmod 755 /opt
echo " /var/root-opt /opt none bind" >> /etc/fstab
/bin/mount /opt

## Setup /tmp
mkdir /var/root-temp ; chmod 1777 /var/root-temp
rm -fr /tmp ; mkdir /tmp ; chmod 1777 /tmp
echo " /var/root-temp /tmp none bind" >> /etc/fstab
/bin/mount /tmp
# install repo releases (keys and repo files)
rpm -i http://192.168.1.5/mrepo/CentOS6-x86_64/RPMS.epel-x86_64/epel-release-6-5.noarch.rpm
rpm -i http://192.168.1.5/mrepo/puppetlabs/puppetlabs-products/puppetlabs-release-6-1.noarch.rpm

# disable repofiles
for repos in `ls /etc/yum.repos.d/` ; do > /etc/yum.repos.d/$repos ; done
chattr +i /etc/yum.repos.d/*repo

# get local configuration
wget http://192.168.1.5/local_repo/LocalMirror.repo -O /etc/yum.repos.d/
LocalMirror.repo
wget http://192.168.1.5/hosts/hosts -O /etc/hosts
wget http://192.168.1.5/resolv_conf/resolv.conf -O /etc/resolv.conf

# install puppet
yum clean all
yum clean metadata
yum install puppet -y

wget http://192.168.1.5/puppet/puppet.conf -O /etc/puppet/puppet.conf

echo "127.0.0.1 <?php echo $hostname . PHP_EOL ?>" >> /etc/hosts

# grub-install fails consistently
grub-install /dev/sda

network install - escape

CentOS 6
Community ENTERprise Operating System

Welcome to CentOS 6.2!
Install or upgrade an existing system
Install system with basic video driver
Rescue installed system
Boot from local drive
Memory test

Press [Tab] to edit options
Automatic boot in 47 seconds...

press “ESC”
network install - set ks.cfg

```
linux ks=http://<server>/<path>/?hostname=<hostname>
```

network install - finishing

```
Congratulations, your CentOS installation is complete.

Please reboot to use the installed system. Note that updates may be available to ensure the proper functioning of your system, and installation of these updates is recommended after the reboot.
```

remove disk and reboot.
Encourage the user to change the root password.

Your solution

- glossed over networking;
  - or at least our DHCP / DNS management.

- (local) cloud with automatic provisioning;
- centrally hosted, full SOE;
- ... lots of options.
now back to the good part

- Manage “Complex” Puppet installations;
  - example server / service layout;
  - environments;
  - hazardous changes;
    - file overrides;
  - swapping Puppetmasters;
- a real example fact;
- programming in Puppet.

“Complex” Puppet

- Puppet can not solve the conflict between Sys Admins and their clients on its own;
- Communication and co-operation are key (in production AND on the road there).
- In testing and development, isolation can go a long way ...
- but the closer to production the more involved other team members and teams need to be.
Assumptions

• Developers:
  • administer VMs on their desktop;
  • tweak development instances of their software on production hosts, possibly even have root access; but
  • do not (generally) touch production instances of their software;

Assumptions ... continued

• There needs to be a code repository;
• The developers should probably be the code repository administrators;
• The repository should be accessible from every host the developers work on.
Before we begin

- Node configurations are essential, irrespective of which of the following options will be implemented.
- Doing away with node files that make a (group of) server(s) unique is unlikely to be beneficial.
- Easy to retrofit (see slides about migrating Puppetmasters).

Node configurations

- Node configurations are not enough to separate projects being actively managed with Puppet:
  - There is a risk of contaminating unrelated hosts because projects will need reusable modules;
  - Node files no longer affects unlisted hosts.
- Developers should be involved in tuning theirs hosts’ node configurations;
  - ... but this is a Sys Admin area of expertise.
Working with Puppets

• Physically separate Puppetmasters:
  • Pro: others can have access to their own Puppetmaster instance;
  • Pro: little chance of cross contamination;
  • Con: more painful to migrate Puppet configuration from test through to production;
  • Con: if you lose a Puppetmaster, remaining cannot "just" take over;

Shared Custody Puppets

• Same Puppetmaster, different "environments":
  • Pro: cheaper;
  • Pro: reasonably simple to maintain (to a point);
  • Con: best administered by a Puppet expert;
  • Con: reduces flexibility in maintaining the SOE;
  • Con: access to select files by select people;
  • Con: ...what is your Puppet DR strategy?
The best of both worlds

- Multiple hosts with multiple environments:
  - Pro: SOE system stays clean;
  - Pro: Easy to migrate changes;
  - Pro: Modules from one stage are unlikely to contaminate another stage;
  - Pro: if you lose a Puppetmaster any of the remaining can take over with a little tweaking;
- Con: possible extra costs / definitely requires more resources;

The key is this:
Project Puppet Code becomes SOE Puppet Code.
Environments

- Ideally someone audits Puppet code before it becomes SOE / Production.
- Since code is in two filesystem locations any host can talk to the same Puppetmaster;
  - Preferably only in a DR situation.
- Requires environments and node files.
- “Environment” is a Puppet Agent (client) setting which allows deviation / override from standard configuration.

why won’t this contaminate?

- Node files identify the hosts, and sets environment by affecting the client’s puppet.conf;
- The puppet.conf sets environment and thus includes additional module path; and
- Module path contains different stages (on different Puppetmasters) of the project Puppet code;
better than just environments

- If anything breaks it will not impact:
  - other groups’ projects;
  - other hosts at different stages;
  - when people get to go home;
    - especially the expert who has to make N systems not have a fault;
  - ...even though Puppet is not a critical service!
- test hosts should not be (as) monitored.

Puppetmaster’s puppet.conf

```
[main]
  logdir = /var/log/puppet
  rundir = /var/run/puppet
  ssldir = $vardir/ssl

[agent]
  classfile = $vardir/classes.txt
  localconfig = $vardir/localconfig

[development]
  modulepath = /etc/puppet/modules:/opt/dev/puppet-modules

[testing]
  modulepath = /etc/puppet/modules:/opt/test/puppet-modules
```
node files

- Sample test node file (used real life):

  ```
  node /\.*.vm.test$/ { 
    $puppetd_environment = "testing"
    include defaultnode
    include control
  }
  ```

- what I used in testing:

  ```
  node "c6pagent.example.org" { 
    $puppetd_environment = "testing"
    include defaultnode
    include control
  }
  ```

puppet.conf.erb

```ruby
[main]
  logdir = /var/log/puppet
  rundir = /var/run/puppet
  ssldir = $vardir/ssl
  pluginsync = true

[agent]
  classfile = $vardir/classes.txt
  localconfig = $vardir/localconfig
  server = c6pmaster.example.org
  splay = true
  runinterval = 1800
  environment = <%= puppetd_environment %>
```

- deploy to puppet_conf/templates/puppet.conf.erb
new puppet_conf module

class puppet_conf {
   file { "/etc/puppet/puppet.conf":
      owner => root,
      group => 0,
      mode => 644,
      content => template("puppet_conf/puppet.conf.erb"),
      notify => Service["puppet"]; }

   service { "puppet":
      name => $operatingsystem ? {
         darwin => "com.reductivelabs.puppet",
         default => "puppet",
      },
      ensure => running,
      enable => true;
   }
}

SELinux

[root@c6pmaster ~]# semanage fcontext -a -t puppet_etc_t /opt/dev/puppet-modules \(/.*/\)?
[root@c6pmaster ~]# semanage fcontext -a -t puppet_etc_t /opt/test/puppet-modules\(/.*/\)?
[root@c6pmaster ~]# semanage fcontext -a -t puppet_etc_t /var/root-opt/dev/puppet-modules\(/.*/\)?
[root@c6pmaster ~]# semanage fcontext -a -t puppet_etc_t /var/root-opt/test/puppet-modules\(/.*/\)?
[root@c6pmaster ~]# restorecon -Rv /opt/*/puppet*
restorecon reset /opt/dev/puppet-modules context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules/control context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules/control/templates context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules/control/files context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules/control/manifests context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
restorecon reset /opt/test/puppet-modules/control/manifests/init.pp context unconfined_u:object_r:usr_t:s0->unconfined_u:object_r:puppet_etc_t:s0
[root@c6pmaster ~]#
Control

- Test module which echoes into /root/purpose

```bash
[root@c6pmaster ~]# vi /opt/test/puppet-modules/control/manifests/init.pp
[root@c6pmaster ~]# cp -R /opt/test/puppet-modules/control /opt/dev/puppet-modules/
[root@c6pmaster ~]# cat /opt/test/puppet-modules/control/manifests/init.pp
class control {
    file {
        "/root/purpose":
            content => $puppeted_environment;
    }
}
```

Control

- Can not use modules in new environment until the client configuration is updated:

```bash
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
err: Could not retrieve catalog from remote server: Error 400 on SERVER: Could not parse for environment main: Syntax error at 'control' at /etc/puppet/manifests/nodes/c6pagent.node:1 on node c6pagent.example.org
warning: Not using cache on failed catalog
err: Could not retrieve catalog; skipping run
[root@c6pagent ~]#
```
Deploy new puppet.conf

```
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '1333712437'
notice: /File[/etc/puppet/puppet.conf]/content:
--- /etc/puppet/puppet.conf 2012-04-06 02:57:21.259390010 +1000
+++ /tmp/puppet-file20120406-7723-1eph5x8-0 2012-04-06 02:58:14.175704006 +1000
@@ -10,4 +10,4 @@
    server = c6pmaster.example.org
    splay = true
    runinterval = 1800
-  environment = main
+  environment = testing

info: FileBucket adding {md5}4d6895c9ff7f6f45d042d04a5baef45f
info: /File[/etc/puppet/puppet.conf]: Filebucketed /etc/puppet/puppet.conf
to puppet with sum 4d6895c9ff7f6f45d042d04a5baef45f
notice: /File[/etc/puppet/puppet.conf]/content: content changed '{md5}4d6895c9ff7f6f45d042d04a5baef45f' to '{md5}52d66941298f8abca8a3f4b8afca5cf3'
info: /File[/etc/puppet/puppet.conf]: Scheduling refresh of Service[puppet]
notice: /Stage[main]/Puppet_conf/Service[puppet]: Triggered 'refresh' from 1 events
notice: Finished catalog run in 7.27 seconds
[root@c6pagent ~]#
```

Deploy control

```
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '1333713217'
notice: /Stage[main]/Execute/Exec[echo top into /tmp/puppet.top]/
returns: executed successfully
notice: /File[/root/purpose]/ensure: defined content as '{md5}ae2b1fca515949e5d54fb22b8ed95575'
notice: Finished catalog run in 3.02 seconds
[root@c6pagent ~]# cat /root/purpose
testing
```

- and switched to “development” ...:
That should not have worked

- The catalog is compiled before the new puppet.conf is deployed;
- Once Puppet is running it does not adjust to the new environment listed in the puppet.conf.
why’d that work?

• It actually did not:
  • used the original (testing) control module;
  • but both use a variable to set the content;
  • test it by changing the content to a string;
  • ... or just trust me.

more node files

• Sample development node file

```plaintext
node "wsadev1.example.org", "wsadev2.example.org" {
  $service_group = "wsa_dev"
  $puppetd_environment = "development"

  include defaultnode
  include control
}
```

• Sample production node file:

```plaintext
node "wsaprod1.example.org", "wsaprod2.example.org" {
  $service_group = "wsa_prod"

  include defaultnode
  include control
}
```
Automate Puppet Module Replication

- Two aspects:
  - SOE Puppet code - which is next;
  - Project Puppet code - not dealt with...
    though our setup allows Developers to check project Puppet Code out to the development Puppetmaster without Sys Admin involvement.

Replicate SOE Puppet Code

- “Automatic” means “break everything at once”;
- “Manual” means “a ‘change’ causing an ‘incident’”;
- “Delayed” means you have to wait before you break all Puppetmasters at once;
  - Though implementing a delay is neither simple;
  - ... nor will it help.
Automatic Replication

- Test your change;
- Fix your typos;
- Worst Case: affected nodes’ catalogue will not build and thus change will not be applied until the next run.
- IF your change can cause worse, you should be following your hazardous change procedure (see “Hazardous Changes”).

Auto Replicate module

- rsyncd.conf

```ini
[modules]
  use chroot = false
  read only = true
  path = /etc/puppet/modules

[manifests]
  use chroot = false
  read only = true
  path = /etc/puppet/manifests

[fileserver]
  use chroot = false
  read only = true
  path = /etc/puppet/fileserver
```
Auto Replicate module

• Cron Job

* * * * * root /usr/bin/rsync --delete --rsh="/usr/bin/ssh -2 -l puppetsync -i /opt/puppetsync/.ssh/id_rsa" --exclude-from=/home/puppetsync/excludelist -a c6pmaster.example.org::modules /etc/puppet/modules/ > /dev/null 2>&1

* * * * * root /usr/bin/rsync --delete --rsh="/usr/bin/ssh -2 -l puppetsync -i /opt/puppetsync/.ssh/id_rsa" --exclude-from=/home/puppetsync/excludelist -a c6pmaster.example.org::manifests /etc/puppet/manifests/ > /dev/null 2>&1

* * * * * root /usr/bin/rsync --delete --rsh="/usr/bin/ssh -2 -l puppetsync -i /opt/puppetsync/.ssh/id_rsa" --exclude-from=/home/puppetsync/excludelist -a c6pmaster.example.org::fileserver /etc/puppet/fileserver/ > /dev/null 2>&1

Auto Replicate module

• exclude list

##
## don't copy rsa keys or .svn
##
*id_rsa* .svn
##
## Server & Cert name will be different
##
/shared-puppetd/templates/puppet.conf* /shared-puppetd/files/puppet.conf*
class auto_replicate_puppet {
  Group["puppetsync"] -> User["puppetsync"] -> File["/home/puppetsync"]
  File["/home/puppetsync"] -> File["/home/puppetsync/.ssh"]
  File["/home/puppetsync"] -> File["/home/puppetsync/excludelist"]
  File["/opt/dev"] -> File["/opt/dev/puppet-modules"] -> Exec["dev puppetmodules"]
    -> Exec["dev puppetmodules real location"]
  File["/opt/test"] -> File["/opt/test/puppet-modules"] -> Exec["test puppetmodules"]
    -> Exec["test puppetmodules real location"]
  user {
    "puppetsync":
      uid => 5000,
      gid => 5000,
      comment => "Puppet synchronization user",
      shell => "/bin/bash",
      home => "/home/puppetsync";
  }
  group {
    "puppetsync": gid => 5000;
  }
}

file {
  ["/home/puppetsync","/home/puppetsync/.ssh"]: 
    owner => 5000,
    group => 5000,
    mode => 700,
    ensure => directory;
  
  "/home/puppetsync/excludelist": 
    owner => 5000,
    group => 5000,
    mode => 700,
    source => "puppet:///modules/auto_replicate_puppet/exclude";
  
  "/etc/cron.d/auto_replicate_puppet": 
    owner => root,
    group => root,
    mode => 644,
    source => "puppet:///modules/auto_replicate_puppet/cronjob";
  
  ["/opt/dev","/opt/test","/opt/dev/puppet-modules","/opt/test/puppet-modules"]: 
    owner => root,
    group => root,
    mode => 755,
    ensure => directory;
}

# continued on next slide
Auto Replicate module

exec {
"dev puppetmodules":
    command => "/usr/sbin/semanage fcontext -a -t puppet_etc_t /opt/dev/puppet-modules\(\/.\*\)?",
    cwd => "/",
    unless => "/usr/sbin/semanage fcontext -l | grep '/opt/dev/puppet-modules'";

"dev puppetmodules real location":
    command => "/usr/sbin/semanage fcontext -a -t puppet_etc_t /var/root-opt/dev/puppet-modules\(\/.\*\)?",
    cwd => "/",
    unless => "/usr/sbin/semanage fcontext -l | grep '/var/root-opt/dev/puppet-modules'";

"test puppetmodules":
    command => "/usr/sbin/semanage fcontext -a -t puppet_etc_t /opt/test/puppet-modules\(\/.\*\)?",
    cwd => "/",
    unless => "/usr/sbin/semanage fcontext -l | grep '/opt/test/puppet-modules'";

"test puppetmodules real location":
    command => "/usr/sbin/semanage fcontext -a -t puppet_etc_t /var/root-opt/test/puppet-modules\(\/.\*\)?",
    cwd => "/",
    unless => "/usr/sbin/semanage fcontext -l | grep '/var/root-opt/test/puppet-modules'";
}

returned ... nothing?

err: /Stage[main]/Auto_replicate_puppet/Exec[dev puppetmodules]/returns: change from notrun to 0 failed: /usr/sbin/semanage fcontext -a -t puppet_etc_t /opt/dev/puppet-modules\(\/.\*\)? returned instead of one of [0] at /etc/puppet/modules/auto_replicate_puppet/manifests/init.pp:69

• commands / puppet can be very memory hungry;
• VM used for test testing could not cope on only 512MB.
Auto Replicate module

[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '133372948'
note: /File[/opt/test]/ensure: created
notice: /Stage[main]/Auto_replicate_puppet/Group[puppetsync]/ensure: created
notice: /Stage[main]/Auto_replicate_puppet/User[puppetsync]/ensure: created
notice: /File[/home/puppetsync]/ensure: created
notice: /File[/home/puppetsync/excludelist]/ensure: defined content as '{md5}737dadfe1586ed07603c849c71ce849e'
note: /File[/etc/cron.d/auto_replicate_puppet]/ensure: defined content as '{md5}c0e2cc2b6b05ce51242a6c4a5a0ec793'
note: /File[/opt/test/puppet-modules]/ensure: created
notice: /Stage[main]/Auto_replicate_puppet/Exec[test puppetmodules]/returns: executed successfully
notice: /File[/home/puppetsync/.ssh]/ensure: created
notice: /File[/opt/dev]/ensure: created
notice: /Stage[main]/Auto_replicate_puppet/Exec[test puppetmodules real location]/returns: executed successfully
notice: /File[/opt/dev/puppet-modules]/ensure: created
notice: /Stage[main]/Auto_replicate_puppet/Exec[dev puppetmodules]/returns: executed successfully
notice: /Stage[main]/Auto_replicate_puppet/Exec[dev puppetmodules real location]/returns: executed successfully
notice: Finished catalog run in 35.42 seconds
[root@c6pagent ~]#

manually on replicating host

[root@c6pagent ~]# su - puppetsync
-bash-4.1$ ssh-keygen -b 1024 -t rsa -f ./.ssh/id_rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./.ssh/id_rsa.
Your public key has been saved in ./.ssh/id_rsa.pub.
The key fingerprint is:
puppetsync@c6pagent.example.org
The key’s randomart image is:
+-[ RSA 1024]-----
  o.*+= .
  + B o
  o E = .
  . B B
  . * S
  .
  .
  .
  .
  +-----------------+
-bash-4.1$
manually on Puppetmaster

[root@c6pmaster ~]# groupadd -g 5000 puppetsync
[root@c6pmaster ~]# useradd -u 5000 -g 5000 -c "Puppet synchronization user" -s "/bin/bash" -d "/home/puppetsync" -m puppetsync
[root@c6pmaster ~]# su - puppetsync
[puppetsync@c6pmaster ~]# vi ~puppetsync/rsyncd.conf
[puppetsync@c6pmaster ~]# mkdir .ssh ; chmod 700 .ssh/
[puppetsync@c6pmaster ~]# vi .ssh/authorized_keys
[puppetsync@c6pmaster ~]# chmod 600 .ssh/authorized_keys

manually on replicating host

-bash-4.1$ ssh c6pmaster.example.org -i .ssh/id_rsa
The authenticity of host 'c6pmaster.example.org (192.168.1.9)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'c6pmaster.example.org,192.168.1.9' (RSA) to the list of known hosts.
[puppetsync@c6pmaster ~]$ exit
logout
Connection to c6pmaster.example.org closed.
automatically on replicating host

```
[root@c6pagent ~]# ls -l /opt/{dev,test}/puppet-modules/ /etc/puppet
/etc/puppet:
total 20
-rw-r--r--.  1 root   root   2552 Mar 13 02:30 auth.conf
drwxr-xr-x.  3 root   root   4096 Apr  6  2012 fileserver
drwxr-xr-x.  3 root   root   4096 Apr  6  2012 manifests
drwxr-xr-x. 13 puppet puppet 4096 Apr  6  2012 modules
-rw-r--r--.  1 root   root    266 Apr  6 03:26 puppet.conf

/opt/dev/puppet-modules/:
total 4
drwxr-xr-x. 5 puppet puppet 4096 Apr  6  2012 control

/opt/test/puppet-modules/:
total 4
drwxr-xr-x. 5 puppet puppet 4096 Apr  6  2012 control

[root@c6pagent ~]#
```

To make this a Puppetmaster

- install puppet-server;
- set to start on boot;
- remove and recreate /var/lib/puppet;
- restore the SELinux context;
- this is a bad idea once host is a Puppetmaster;
- update firewall;
- Misc /etc/puppet configs are not explicitly replicated.
Swapping Puppetmasters

- Setup new Puppetmaster;

- On the client to be moved:
  - delete /var/lib/puppet;
  - run `puppetd -vt --server=<new master>`;

- On the new master, sign certificates;

- On the client run puppetd again.

On the client

[root@c6repo ~]# puppetd -vt --server=c6pagent.example.org
info: Creating a new SSL key for c6repo.example.org
warning: peer certificate won't be verified in this SSL session
info: Caching certificate for ca
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
info: Creating a new SSL certificate request for c6repo.example.org
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
Exiting; no certificate found and waitforcert is disabled
[root@c6repo ~]#
On the Puppetmaster

[root@c6pagent puppet]# puppetca --sign c6repo.example.org
notice: Signed certificate request for c6repo.example.org
notice: Removing file Puppet::SSL::CertificateRequest
c6repo.example.org at '/var/lib/puppet/ssl/ca/requests/c6repo.example.org.pem'
[root@c6pagent puppet]#

On the client

[root@c6repo puppet]# puppetd -vt --server=c6pagent.example.org
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6repo.example.org
info: Applying configuration version '1334642468'
notice: /File[/etc/pam.d/system-auth-local]/ensure: defined content as '{md5}f1d3f40734136a98d16ade24066ee042'
info: FileBucket adding {md5}e8aee610b8f5de9b6a6cdba8a33a4833
info: /File[/etc/pam.d/system-auth]: Filebucketed /etc/pam.d/system-auth to puppet with sum e8aee610b8f5de9b6a6cdba8a33a4833
### ... trust me, it worked
info: /File[/etc/puppet/puppet.conf]: Scheduling refresh of Service[puppet]
notice: /Stage[main]/Puppet_conf/Service[puppet]/ensure: ensure changed 'stopped' to 'running'
notice: /Stage[main]/Puppet_conf/Service[puppet]: Triggered 'refresh' from 1 events
notice: /File[/home/t.durden]/ensure: created
notice: /Stage[main]/Local_users/Deploy_user[Tyler Durden]/User[t.durden]/ensure: created
info: Creating state file /var/lib/puppet/state/state.yaml
notice: Finished catalog run in 12.90 seconds
[root@c6repo puppet]#
Swapping issues

• Most problems due to certificates:
  • Remove the client certificate from old master;
  • Ensure client certificate not on new master;
  • Stop puppetd before deleting /var/lib/puppet;
  • Time of hosts must be in sync;

Swapping issues ... continued

• Client should not be newer than master;
  • 2.7.x client talking to 2.6.x master likely to fail.
• Do NOT delete the puppet.conf ;
  • can affect the client’s directory structure;
Hazardous Changes

- And thus file overrides

Hazardous Changes

- Always tell the service owner you are about to do something that may ruin their day.

- Occasionally things go wrong, if others do not know in advance it will be worse.

- Sometimes this might not be a scheduled outage or require a change request, but that depends on your site.
Sample Hazardous Change

- Imagine:
  - Using LDAP to look up and NSCD to cache user information on hosts.
  - (Service users are on host accounts);
  - What could go wrong?

Things go wrong

- Network could drop out; or
- LDAP service could disappear; or
- Power or hardware failure on any of the components;
Pardon my paranoia

In the past three years (2010 - 2012) UQ has had (at least) one of each:
- DC fire;
- 100 year (30 year?) flood;
- DC power failure;
  - which badly affected the SAN.

... but the service is up! ...

Through all of these, LDAP stayed up.

But at least:
- one gateway failed (fire);
- half of the VMware farm disappeared (SAN);
- some intermittent networking issues arose (other than missing a gateway) (fire, SAN and changes).
Segue - SOE DR

LDAP Service revisions

- LDAP is not using Multi Master setup:
  - Version 2 of our setup went:
    - from one master and slave;
    - to one master, two primary slaves and ucarp;
  - Version 3
    - one master, two primary slaves and ucarp;
    - and LDAP on our other core DR hosts (Repositories and Puppet Masters);
LDAP Master

• LDAP master vulnerable because one of a kind;
• Can be rebuilt in ~20 minutes because deployed entirely automatically via Puppet;
• Slapcat backups are done daily;

designed for failure

• On host authentication used to comprise of:
  • LDAP +
  • NSCD +
  • pam_ssh (slightly hacked for on-host auth) +
  • two sets of centrally pushed out ssh keys:
    • one for pam_ssh; and
    • one for traditional ssh;
Not great because

- pam_ssh for on-host auth is flakey;
- NSCD times out; ...
  - PAM won’t meet its configured requirements;
  - reconfiguring PAM’s base requirements can be a bad idea;
- building tarballs of /home and ssh keys is:
  - CPU intensive;
  - does not deploy unless LDAP already works;

because ... continued

- NSCD does not:
  - cache authentication related information;
  - retain information indefinitely;
- Does (depending on version):
  - leak memory;
  - hang on network access if network is down;
- NSCD was not designed to be used this way.
meet SSSD

- Can use different authentication methods;
- Will cache:
  - passwd and group info for anything it sees;
  - shadow for users it has authenticated;
  - pam_sss and sssd replaces pam_ssh + keys.

SSSD downsides

- Does not cache information it has not needed;
- Very occasionally the cache gets corrupted and needs to be reset;
- Server rebuild procedure does not include restores by default;
- People who had logged into the destroyed server are not cached on the rebuilt one.
So, SSSD is great

- Thus, change all hosts to use it.
- This is a major change:
  - If it worked on a few hosts it should work every time;
  - sssd and nsld running together leads to a corrupt sssd’s cache;
  - Bad idea to deploy to 100+ servers in one go.

Relax

- Probably would not spend a lot of effort tweaking NSCD’s Puppet module;
- Ensuring absence of NSCD remnants is always good when deploying SSSD, so put this in the new SSSD Puppet module.

- Can you disable your modules?
disable modules?

class shared-USG_internal_ldap2010
{
    if ($skip_USG_internal_ldap2010 != "true") {

        ## Deploy client certificate - needed on all hosts
        file {
            "/etc/ssl", "/etc/ssl/certs":
                owner  => root,
                group  => root,
                mode   => 755,
                ensure => directory;

            "/etc/ssl/certs/cacert.pem":
                owner  => root,
                group  => root,
                mode   => 644,
                source => "puppet://modules/shared-USG_internal_ldap2010/cacert2010.pem",
                require => File["/etc/ssl/certs"];
        }

    }

    ###...

So then

- In the individual node files set:

  $skip_USG_internal_ldap2010 = "true"

- when migrating them to the new SSSD based solution;

- If the Puppet modules are modular, may need to retrofit this to several ; or

- Work out which ones to disable and what dependencies this will affect.
Conversely

• In new modules set something like:

```
class redhat-sssd
{
    if ($deploy_sssd == "true") {
        #...
    }
}
```

• ... to selectively enable for nodes being migrated;
• careful with that logic:
  • skip uses !=
  • deploy uses ==
  • might accidentally deploying something.

File Overrides

• Disabling the old method is a start;
• /etc/pam.d/system-auth also needs replacing.
class system-auth {
    if ($skip_system_auth != "true") {

        file {
            "/etc/pam.d/system-auth-local":
                owner => root,
                group => root,
                mode => 644,
                source => "puppet:///modules/system-auth/system-auth.conf";
            "/etc/pam.d/system-auth":
                ensure => "/etc/pam.d/system-auth-local",
                require => File["/etc/pam.d/system-auth-local"];
        }
    }
}
Caveat / Retraction

- Updated code allows override of files, configured in node file;
- Unless the configuration structure relies on inheritance:
  - 2011 talk included this.
  - If implemented, here is the required change:

```plaintext
defaultnode.node

- was:
  node default {
    ## lots of includes
  }

- now:
  class defaultnode {
    ## lots of includes
  }
```

- Remember:
  - can not name default class “default”;
  - do not need to change the file extension;
Individual node files

• was:

node "c6pagent.example.org" inherit default {
}

• now (including override):

node "c6pagent.example.org" {
  $file_system_auth = "puppet:///modules/system-auth/system-auth.sssd"
  include defaultnode
}

Where to store the file

• Sample system-auth file for using SSSD’s will become default in system-auth module;
  • logical to keep in the module.
  • Consider node specific overrides.
$service_group

- A site specific variable, can be called anything.
- Used to differentiate between:
  - individual hosts and
  - collections (i.e.: a “service group”);
- Set variable in every node file;

overrides and service groups

- Configure via /etc/puppet/fileserver.conf

```plaintext
[modules]
  allow *.example.org

[puppettest]
  path /etc/puppet/fileserver/puppettest
  allow c6pagent.example.org

- In the node file:

node "c6pagent.example.org" {
  $service_group = "puppettest"
  $file_system_auth = "puppet:///$service_group/system-auth"

  include defaultnode
}
```
Drop throughs

- The file resource type supports definition of multiple sources.
- Starts with first source, and stops on first match:

```yaml
file {
  "/etc/sysconfig/iptables":
    owner   => root,
    group   => root,
    mode    => 600,
    source  => [
      "puppet:///modules/iptables/iptables.$fqdn",
      "puppet:///modules/iptables/iptables.$service_group",
      "puppet:///modules/iptables/iptables",
    ],
    notify  => Service["iptables"];
}
```

Be careful though

- Non-generic items in modules is generally bad:
  - decommissioned hosts’ files linger;
- Divide:
  - Generic files via Modules;
  - Specific files and settings via Node file and custom fileserver shares.
- Things might remain but are out of the way.
Drop Through a better way

- Check the service group’s custom files first;
- or else deploy module default:

```plain
drop /etc/sysconfig/iptables:
    owner => root,
group => root,
mode  => 600,
source => [
    "puppet:///$service_group/iptables",
    "puppet:///modules/iptables/iptables",
],
notify => Service["iptables"];
```

The story so far

- Manage “complex” Puppet installations;
  - Server / service layout and implementation;
  - Puppet configuration’s environments;
  - Hazardous changes;
    - File overrides and
    - Drop through;
  - Swapping Puppet clients’ Puppetmasters;
Next

- Puppet Configurations files revisited;
- A real fact using Ruby;
- Programming with Puppet;
- Creating Puppet configurations via Python;
- Adding Passenger to Puppetmaster;
- Lot’s of SELinux related joy;
  - Classes to collect defines.

Config files ... revisited

- puppet.conf
- fileserver.conf
- autosign.conf
- auth.conf
fileserver.conf

- Discussed above in file overrides;
- Works with
  - FQDN (including * wildcard);
  - IP addresses, CIDR or * wildcard);
- Some changes require a Puppetmaster restart;
- http://docs.puppetlabs.com/guides/file_serving.html

fileserver.conf

- May break if it contains trailing spaces / tabs;

[root@c6pmaster ~]# service httpd stop
Stopping httpd: [ OK ]
[root@c6pmaster ~]# service puppetmaster start
Starting puppetmaster: [ OK ]
[root@c6pmaster ~]# service puppetmaster status
puppetmasterd (pid  7215) is running...
[root@c6pmaster ~]# vi /etc/puppet/fileserver.conf
[root@c6pmaster ~]# service puppetmaster restart
Stopping puppetmaster: [ OK ]
Starting puppetmaster: [ OK ]
[root@c6pmaster ~]# service puppetmaster status
puppetmasterd dead but pid file exists
[root@c6pmaster ~]#
**autosign.conf**

- Very handy for your Developer’s Test VMs
- Tells Puppetmaster to always sign the client;
  - also updates the certificate if it changes;
- There are security issues;
- Might contain:

  ```
  [root@c6pmaster ~]# cat /etc/puppet/autosign.conf
  *.example.org
  ```

**auth.conf**

- Authentication config for REST API
- [http://docs.puppetlabs.com/guides/rest_api.html](http://docs.puppetlabs.com/guides/rest_api.html)
- [http://docs.puppetlabs.com/guides/rest_auth_conf.html](http://docs.puppetlabs.com/guides/rest_auth_conf.html)
Facts

- Collected before the main Puppet run and used in building the client specific catalog;
- Useful to extract information;
- There are size constraints (e.g.: `rpm -qa | sort` returns too much);
- Simple fact that executes a bash script:

```ruby
Facter.add("rh_release") do
  setcode do
    %x{ /bin/cat /etc/redhat-release | /bin/sed 's/[^0-9.]*//g' | /bin/cut -d . -f 1}.chomp
  end
end
```

A real fact

- As discussed, LDAP used for managing groups;
- Dev’s needed to get some information for deploying mercurial configurations;
getGIDs.rb

# getGIDs.rb
require 'ldap'

$HOST = 'usgldap.example.org'
$PORT = LDAP::LDAP_PORT
$SSLPORT = LDAP::LDAPS_PORT
$BIND = 'cn=unprivuser,dc=example,dc=org'
$PASSWORD = '!53cr37'

groups = {
  'usg' => 'nsysadm',
  'ss' => 'wdu',
}

myfilter = '(|
groups.each { |key, val|
  myfilter += "(cn=#{val})"
} 
myfilter += ')'

## to be continued next slide

getGIDs.rb ... continued

base = 'ou=Group,dc=example,dc=org'
scope = LDAP::LDAP_SCOPE_SUBTREE
attrs = ['cn', 'gidNumber']
results = {}

begin
  conn = LDAP::Conn.new($HOST, $PORT)
  conn.bind($BIND, $PASSWORD)
  # this preserves the existing mappings, a single query
  group_lookup = groups.invert
  conn.search(base, scope, myfilter, attrs) { |entry|
    results[group_lookup[entry.vals('cn')[0]]] = entry.vals('gidNumber')
  }
  conn.unbind
rescue
  LDAP::ResultError
  conn perror("search")
  exit
end

results.each { |key, val|
  Facter.add("#{key}_gid") { setcode { val[0] } } }
}
resulting facts

```
[root@c6pmaster node]# pwd
/var/lib/puppet/yaml/node
[root@c6pmaster node]# grep gid something.example.org.yaml
  usg_gid: "902"
  ss_gid: "923"
[root@c6pmaster node]#
```

- rather than hard code the GID, use the fact:

```yaml
file {
  "/home/chakkerz":
    ensure => directory,
    owner => chakkerz,
    group => 902,
    group => $usg_gid,
    mode => 700,
    require => User["chakkerz"]; }
```
Why Puppet?

- Already existed and better understood than the older CCMS;
- No installation scripts;
- No installation procedures;
- No packaging applications;
- Just a configuration of what to do...
  - and lots of support from friendly SysAdmins.

What is Puppet (again)?

- Puppet tries hard to offer features developers are familiar with:
  - branching execution;
  - inheritance;
  - scope; but
  - sequential execution is limited;
  - variables are constants / different; and
  - for loops are only sort-of do-able.
Let’s re-word that

- Puppet offers:
  - an uncertain execution path; and
  - an unfamiliar approach to loops;
  - with variable constants where you:
    - define how they are set;
    - can append to already set values;
  - a familiar concept of scope for “functions” / “variables”; and
  - inheritance (with overrides).

Segue inheritance

- [http://docs.puppetlabs.com/guides/language_guide.html](http://docs.puppetlabs.com/guides/language_guide.html) lists the following (abridged):

```ruby
class unix {
  file {
    "/etc/passwd":
      owner => root,
      group => root,
      mode  => 0644;
  }
}

class freebsd inherits unix {
  File["]/etc/passwd"] { group => wheel }
}
```
class system-auth {

    if ($skip_system_auth != "true") {
        if ($file_system_auth == "") {
            $file_system_auth = "puppet:///modules/system-auth/system-auth"
        }

        file {
            "/etc/pam.d/system-auth-local":
            owner => root,
            group => root,
            mode => 644,
            source => $file_system_auth;

            "/etc/pam.d/system-auth":
            ensure => "/etc/pam.d/system-auth-local",
            require => File["/etc/pam.d/system-auth-local"];  
        }
    }
}
}

class sssd-system-auth inherits system-auth {
    File["/etc/pam.d/system-auth-local"] {
        source => "puppet:///modules/system-auth/system-auth.sssd"
    }
}

Our way

node "c6pagent.example.org" {
    $file_system_auth = "puppet:///modules/system-auth/system-auth.sssd"

    include defaultnode
}

Inheritance’s way

node "c6pagent.example.org" {

    include execute
    include local_users
    include packages
    include puppet_conf
    include rh_release_if
    include sshd_config
    include sysadmins
    include sssd-system-auth
}

Disclaimer

- Did not test the inheritance code;
- Default node would still be a node (rather than a class; see “File Overrides”);

Puppet is first and foremost

- A system administrator’s tool.
- Deploy various bits and pieces;
- ... not necessarily in a particular order;
  - though that can be achieved;
- Configuration is mostly applied again and again.
- Programmability is handy, but code visually differs from configuration.
What went wrong?

- Both developers who wrote this code moved on;
- Other developers had never become familiar with Puppet, or the modules the project relied on;

Top three issues

- Order of execution;
  - inter-dependencies defined wrong or not at all;
- Defines;
  - some identical “functions” in every module;
  - some four levels of indirection removed;
- Extremely slow;
Extremely slow

• Obvious:
  • every run Puppet would reset permissions;
• Red herring:
  • recursive directory deployments - already stopped using built-in file server in favour of mongrel and passenger.

Defines

• $N$ modules implementing the same function;
• giving $N$ implementations of the same function in $N$ files;
• where $N = ...$

[root@tangelo]# ls -dl ss-app* | wc -l
28
[root@tangelo]# grep "$define environment" ss-app*/manifests/init.pp | wc -l
13
Let me show you

class ss-application-<something> {  
  ## ... snip ...
  $user = <something>
  ## ... snip ...

  ss-application-<something>::environment {  
    ["local","development","test","staging","production",]:
  }

  define environment() {  
    include ss-platform-php

    $type = $name

    ss-platform-php::zend_environment { "${$user}_env_${type}":  
      basedir => $home,
      type => $type,
      user => $user,
      require => [File[$home],Ss-util::Set_group_fac1["$home-wdu"],],
    }

    # this bit ties us to the repo layout
    file { "$home/www/$type":  
      target => "$home/$type/php",
      ensure => "link",
      require => [Ss-platform-php::Zend_environment["{$user}_env_${type}"],],
    }
  }

(slightly formatted to fit on slide)

Things to note

• This is about showing that different mindsets resulting in different code.

• Yes, that is a for loop;

• $type is set to each element of the array;
Order of execution

Puppet looks for chaining statements to determine order;

Wrong or missing chaining means Puppet needs to run repeatedly / does not run at all;

Requiring an entire Class means everything in the class AND their requirements must be satisfied;

This is calculated every time.

You’re not alone

It is bad when your code depends on someone else’s;

you require nscd service being configured;

and they switch from nscd to sssd.

...and they don’t know that you depend on it...

Better to use a fact that talks to LDAP directly, irrespective of the host’s running configuration.

Not always an option.
class shared-users::create_home_link {
  if ($operatingsystem == "solaris") {
    file {
      "/export/home":
        ensure => directory;
      "home_directory":
        path => "/home",
        force => true,
        ensure => "/export/home",
        require => File["/export/home"];
    }
  } elsif ($operatingsystem == "freebsd") {
    file {
      "/var/home":
        ensure => directory;
      "home_directory":
        path => "/home",
        force => true,
        ensure => "/var/home",
        require => File["/var/home"];  
    }
  }
}

shared-users’s init.pp (continued)

class shared-users {
  require shared-users::create_home_link

  if ($skip_shared_users != "true") {
    ## Always deploy USG, IRT and SB
    include shared-users::nsysadm
    include shared-users::nirtadm
    include shared-users::nsbadm

    if ($enable_un == "true") {
      include shared-users::ndnadm
    }

    if ($enable_wdu == "true") {
      include shared-users::wdu
    }

    if ($enable_is == "true") {
      include shared-users::nsiadm
    }
  }
}
class shared-users::nsysadm {
  if ($no_sssd_available == "true") {
    group {
      "nsysadm":
      gid => 902;
    }
    user {
      "chakkerz":
      uid   => 750,
      gid   => 902,
      home  => "/home/chakkerz",
      comment => "Christian Unger",
      shell  => $operatingsystem ? {
        freebsd => "/bin/sh",
        default => "/bin/bash",
      },
      password => '${\$S.tAd0$wLUZe8egCOnyxSIZiLv.M}',
      require  => Group["nsysadm"];
    }
  }
}

file {
  "/home/chakkerz":
  owner => 750,
  group => 902,
  mode  => 700,
  ensure => directory;

  "/home/chakkerz/.ssh":
  owner => 750,
  group => 902,
  mode  => 700,
  ensure => directory,
  require  => File["/home/chakkerz"];

  "/home/chakkerz/.ssh/authorized_keys":
  owner => 750,
  group => 902,
  mode  => 600,
  content => 'ssh-rsa .....',
  require  => File["/home/chakkerz/.ssh"];
}
Generating classes

- If sssd is not available on a client host it is still possible to rely on LDAP to centrally manage users;
- Ruby DSL on the client is one way;
- Getting a script to generate class files is another;

```
#!/usr/bin/python
# source http://www.grotan.com/ldap/python-ldap-samples.html
import ldap

## first you must open a connection to the server
try:
    l = ldap.initialize("ldaps://ldap.example.org:636/")
    l.protocol_version = ldap.VERSION3
    l.simple_bind_s("cn=auth_LDAP,dc=usg,dc=example,dc=org ","7h3$3crg37")
except ldap.LDAPError, e:
    print e
    # handle error however you like

## The next lines will also need to be changed to support your search requirements and directory
searchBase = "ou=group,"
baseDN = "dc=usg,dc=example,dc=org"
sScope = ldap.SCOPE_SUBTREE
retrieveAttributes = ['memberUid', 'gidNumber', 'cn']
sSearchFilter = "cn="

groups = []
```
try:
    ldap_result_id = l.search(searchItem + baseDN, searchScope, searchFilter, retrieveAttributes)
    result_set = []
    while 1:
        result_type, result_data = l.result(ldap_result_id, 0)
        if (result_data == []):
            break
        else:
            # here you don't have to append to a list
            # you could do whatever you want with the individual entry
            # The appending to list is just for illustration.
            if result_type == ldap.RES_SEARCH_ENTRY:
                result_set.append(result_data)
                groups.append(result_data)
except ldap.LDAPError, e:
    print e

# now, based on the retrievedAttributes split the result:
for group in groups:
    try:
        group_name = group[0][1].get('cn')[0]
        group_gid = group[0][1].get('gidNumber')[0]
        group_members = group[0][1].get('memberUid')
        group_users = ''
        group_homes = ''
        group_useremail = ''
        group_email = ''
        except TypeError:
            print '## Error on', group

if group_members != None:
    for member in group_members:
        # now we need:
        # - Real Name
        # - their password
        # - ssh public key
        # - shell
        # - email address (not on every account right now)
        searchItem = "ou=People,"
        retrieveAttributes = ["uid", 'loginShell', 'uidNumber', 'gecos', 'homeDirectory',
                              'userPassword', 'mail', 'sshPublicKey', 'description']
        searchFilter = "uid=" + member

        try:
            ldap_result_id = l.search(searchItem + baseDN, searchScope, searchFilter, retrieveAttributes)
            result_set = []
            while 1:
                result_type, result_data = l.result(ldap_result_id, 0)
                if (result_data == []):
                    break
                else:
                    # here you don't have to append to a list
                    # you could do whatever you want with the individual entry
                    # The appending to list is just for illustration.
                    if result_type == ldap.RES_SEARCH_ENTRY:
                        result_set.append(result_data)
for user in result_data:
    try:
        username = result_data[0][1].get('uid')[0]
        uid = result_data[0][1].get('uidNumber')[0]
        gid = group[0][1].get('gidNumber')[0]
        groupname = group[0][1].get('cn')[0]
        home = result_data[0][1].get('homeDirectory')[0]
        comment = result_data[0][1].get('gecos')[0]
        shell = result_data[0][1].get('loginShell')[0]
        group = result_data[0][1].get('group')[0]
    except TypeError:
        print '## ERROR ON ', user, '\n'
    try:
        keys = ''
        for publicKey in result_data[0][1].get('sshPublicKey'):
            keys += publicKey + '\n'
    except TypeError:
        keys += '\n issue with key(s) \n'
    try:
        password = '!!'
        passwords = result_data[0][1].get('description')
        for password_temp in passwords:
            if password_temp.startswith('"$1"'):
                password = password_temp
                break
    except TypeError:
        password += 'issue with password'

except ldap.LDAPError, e:
    print e
```python
filename = "/tmp/shared-users/" + group_name + ".pp"
file = open(filename, 'w')

file.write("class shared-users::" + group_name + " {\n"
file.write("\tif ($no_sssd_available == "true") {\n"
file.write("\t\tgrou" + 'p_name + ':\n" + group_gid + ";\n" + "\t\grou" + "\n"
if group_users != ":":
   file.write("\t\tuser { " + group_users + "\t}\n"
file.write("\t\n"
if group_homes != ":":
   file.write("\t\tfi" + "le { " + group_homes + "\t}\n"
file.write("}\n"
file.close()
```

- Probably not the best example of how to do this.
- Not tested in production just yet.

---

**What went right?**

- The ground work (not just Puppet, but Load Balancer configuration etc) made deploying new applications extremely easy and flexible;
- Puppet was (relatively) easy to use to deploy new applications.
- Most issues were not Puppet related, but with generic issues of how to interact with a SOE or Unix in general;
  - primarily `sudo` or `su - <application user>` and thus resulting issues.
What went right ... continued

- The basic framework was good, but:
  - badly documented;
  - clearly rushed;
  - under-used because hard to follow.
- The overall project was clearly NOT a failure, but Puppet required attention.

Trade-offs

- Communication is the biggest issue:
  - Both SysAdmins and Developers need to work together.
  - Need to come to an arrangement where both can work autonomously;
Getting it right(er)

• The original code worked, but had issues;
  • Structure was good: e.g.: php applications included php platform module, which contained re-usable functions and shared requirements;
• Retrofitting fixes == very time consuming:
  • Five days to rewrite 38 modules;
  • and end up with 32.
  • versus hundreds of hours to assist with unfamiliar code.

So, what changed?

• Coding style;
• Naming conventions;
• Duplicated types moved to parent module;
• Chaining much more targeted and pervasive;
• Permissions:
  • set to what the service itself was enforcing;
  • FACLs used more extensively and at a higher level (rather than per application);
What else changed?

- Application modules:
  - call shared parent functions;
  - contain application specific settings only;
- Master control module calls global functions always, instead of using Virtual Resources.

Segue Virtual Resources

- Puppet will let most types be defined only once;
  - Imagine: tomcat is needed for two application;
  - Can only install tomcat in one of them;
    - Or install tomcat with neither application;
    - OR create a virtual function and “realise” it in both application modules.
virtual_tomcat

class virtual_tomcat {
    @deploy_service { "tomcat6":
        service => "tomcat6";
    }
}

define deploy_service($service) {
    package { "$service":
        ensure => installed;
    }

    service {
        "$service":
            enable => true;
            ensure => running;
    }
}

Note the @

realize

- realize it in another module / node file:

node "c6pagent.example.org" {
    $service_group = "puppettest"

    realize Deploy_service["tomcat6"]

    include defaultnode
}
... and deploy

```bash
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '1333677567'
notice: /Stage[main]/Virtual_tomcat/Deploy_service[tomcat6]/
  Package[tomcat6]/ensure: created
notice: /Stage[main]/Virtual_tomcat/Deploy_service[tomcat6]/
  Service[tomcat6]/ensure: ensure changed 'stopped' to 'running'
notice: Finished catalog run in 197.08 seconds
[root@c6pagent ~]#
```

- There is a little more to this, see:
- [http://docs.puppetlabs.com/guides/virtual_resources.html](http://docs.puppetlabs.com/guides/virtual_resources.html)

---

Should you virtualize?

- Never seen a Virtual Resources that was not realized.
- For example: a web servers hosting PHP applications will always want PHP installed.
- Virtual Resources offer alternative to on/off switches, though with semantic difference:
  - default off, selective on, multiple invocations in various places;
  - skip_ (default on) or deploy_ (default off) in node file only.
Any other changes?

- Documentation of the overall layout generated;
- Developers maintain the modules, so they should maintain their documentation.
Finally ... some code

- Application modules (eg fcgi, mercurial and tomcat)
- Platform modules (eg php and python)
- Service modules (eg nginx)
- Test VM’s limited deployment

deploy hgrc to /home

- ${homedir_chakkerz} and ${fullname_chakkerz} are custom facts filled by querying LDAP;
- $hgUsername, $realName and $email are used in the template;
- replace => false
- require targets specifically what needs to exist (rather than an entire class),
- homedir_deployment; the “usg” should be a passed argument;
deploy hgrc to /home

class rhel6-ss-util-homedirs {
    hgrc {
        "chakkerz":
        home => "${homedir_chakkerz}",
        username => "chakkerz",
        hgUsername => "uqcunge2",
        realName => "${fullname_chakkerz}",
        email => "c.unger@its.uq.edu.au";
    }

    define hgrc($home, $username, $hgUsername, $realName, $email) {
        file { "${home}/.hgrc":
            content => template("ss-util-homedirs/hgrc.erb"),
            owner => $username,
            replace => false,
            require => [Homedir_deployment["usg"], Service["sssd"]];
        }
    }
}
}

parts of php-platform 1 & 2

• package using an array to install;
• specific require;
• exec chaining;
• returns to avoid failed dependencies;
class rhel6-ss-platform-php {
    package {
        ["php-5.3-pdo-oci-zend-server",
         "php-5.3-oci8-zend-server",
         "php-5.3-pdo-mysql-zend-server",
         "php-5.3-mysqli-zend-server",
         "php-5.3-mbstring-zend-server",
         "php-5.3-gd-zend-server",
         "php-5.3-ctype-zend-server",
         "php-5.3-curl-zend-server",
         "php-5.3-memcached-zend-server"]:
            ensure => installed,
            require => [ File["/etc/yum.repos.d/ITS-Zend6.repo"],
                         File["/etc/pki/rpm-gpg/RPM-GPG-KEY-zend"], ],
            before => Exec["fix zend extensions"];

        "httpd-devel.$arch":
            ensure => installed;

        "autoconf.noarch":
            ensure => installed;
    }
}

exec {
    "fix zend extensions":
        command => "${variables::sed_cmd} -i -e 's|\^extension=\([^/\]+.so\)|extension=/usr/local/zend/lib/php_extensions/\1|' /usr/local/zend/etc/conf.d/*.ini";
}

exec {
    "clear php-5.3 pear cache":
        command => "/usr/local/Zend/bin/pear clear-cache",
        require => Package["php-5.3-dev-zend-server"],
        returns => [ 0, 1 ]; # returns 1 when there was no cache

    "php-5.3-pear":
        command => "/usr/local/Zend/bin/pear update-channels",
        require => Exec["clear php-5.3 pear cache"];
}
$name (pear_channel_discover)

unless attribute

one way of avoiding gcc is installed permanently

though probably not the best way

define pear_channel_discover($channel = "") {
    if $channel == "" { $ch = $name }
    else { $ch = $channel }

    exec {
        "php-5.3 pear channel $ch":
        command => "/usr/local/zend/bin/pear channel-discover $ch",
        unless => "/usr/local/zend/bin/pear channel-info $ch",
        require => Exec["php-5.3-pear"];
    }
}

pear_channel_discover { "pear.phpunit.de":   }
pear_channel_discover { "components.ez.no":    }
pear_channel_discover { "pear.symfony-project.com": }
```php
define pear_install($package = "") {
    if $package == ""               { $p = $name            }
    else                            { $p = $package         }

    exec {
        "php-5.3 pear package $p":
        command => "'/usr/local/zend/bin/pear install $p'",
        unless => "'/usr/local/zend/bin/pear info $p'",
        require => Exec("php-5.3-pear");
    }
}

pear_install {
    "php-5.3-XML_Serializer":
    package => "XML_Serializer-beta";

    "php-5.3-PHPUnit":
    package => "phpunit/PHPUnit",
    require => [
        Pear_channel_discover["pear.phpunit.de"],
        Pear_channel_discover["pear.symfony-project.com"],
        Pear_channel_discover["components.ez.no"],
    ];
}

```
service-nginx

- targeted require's
- refreshonly attribute
- setfacl from a restore file;

parts of service-nginx ...

```python
class rhel6-ss-service-nginx {
    if ($skip_service_nginx != "true") {
        package {  "nginx.x86_64":    ensure => installed;  }

        file {
            "/etc/nginx/conf.d":
                owner => root,
                group => root,
                mode => 755,
                ensure => directory,
                require => Package["nginx.x86_64"];

            "/etc/nginx/nginx.conf":
                owner => root,
                group => root,
                mode => 644,
                source => "puppet:///modules/rhel6-ss-service-nginx/
                nginx.conf",
                require => Package["nginx.x86_64"];
        }
    }
}
```
"nginx_log.perms":
    path => "/etc/nginx/nginx_log.perms",
    owner => root,
    group => root,
    mode => 644,
    require => [ Package["nginx.x86_64"], Service["sssd"],
                  Group["ssapp"], ],
    source => "puppet:///modules/rhel6-ss-service-nginx/
               nginx_log.perms",
}

exec {
    "restore_nginx_log_perms":
        command => "${variables::setfacl_cmd} --restore=/etc/nginx/
                   nginx_log.perms",
        cwd => "/var/log/nginx",
        subscribe => File["nginx_log.perms"],
        refreshonly => true;
}

service {
    "nginx":
        ensure => running,
        enable => true,
        subscribe => File["/etc/nginx/nginx.conf"];
}

# file: .
# owner: nginx
# group: root
user::rwx
user:nginx:rwx
group::r-x
group:wdu:r-x
group:ssapp:rwx
mask::rwx
other::r-x
default:user::rwx
default:user:nginx:rwx
default:group::r-x
default:group:wdu:r-x
default:group:ssapp:rwx
default:mask::r-x
default:other::r-x

contents of nginx_log.perms:
ss-application

- argument passing with default arguments;
- including using passed arguments to act as defaults;
- define calling other defines;
- including from another scope;

class rhel6-ss-application {
    #
    # Setup group, parent directories and facls on parent directories
    #
    group( "ssapp": gid => 765; }

    file {
        ["/opt/apps","/var/log/apps"]:
            owner => root,
            group => root,
            mode => 755,
            ensure => directory;
    }

    exec {
        "set group facl for wdu on /opt/apps/":
            command => "${variables::setfacl_cmd} -R -m
default:group:wdu:rwx /opt/apps & ${variables::setfacl_cmd} -R -m
group:wdu:rwx /opt/apps",
            cwd => "/opt/apps";
    }
}
```perl
define setup_application_account($uid, $gid, $user, $comment, $groups = [ "fcgi", "ssapp"], $log_uid = $uid, $log_gid = $gid) {

  user {
    $user:
    uid  => $uid,
    gid  => $gid,
    groups => $groups,
    comment => $comment,
    home => "/opt/apps/$user",
    shell => "/bin/true",
  }

  group { $user: gid => $gid; }

  file {
    "/opt/apps/$user":
    owner => $uid,
    group => $gid,
    mode => 775,
    ensure => directory,
    require => File["/opt/apps"]; 

    "/var/log/apps/$user":
    owner => $log_uid,
    group => $log_gid,
    mode => 775,
    ensure => directory,
    require => File["/var/log/apps"]; 

    "/opt/apps/$user/www":
    owner => $uid,
    group => nginx,
    mode => 770,
    ensure => directory,
    require => [ Package["nginx.$arch"], File["/var/log/apps/$user"] ],
  }
}
```
define setup_mercurial_application($uid, $gid, $user, $comment, $path) {
    setup_application_account {
        "mercurial setup $user":
        uid    => $uid,
        gid    => $gid,
        user   => $user,
        comment => $comment;
    }

    rhel6-ss-platform-python::setup_mercurial_configs {
        "mercurial configs for $user":
        uid    => $uid,
        user   => $user,
        path   => $path;
    }
}

Selective App deploy

- The biggest issue for the developers using Puppet has been speed;
- Obvious way to speed things up is only deploy what is needed on a particular VM.
- VM’s name matches the application being developed on it.
- e.g.: academicportal-ckz.vm.test
class rhel6-dev-control {
    $apps = [
        # JAVA
        'rhel6-ss-application-cas',
        'rhel6-ss-application-grouper',
        # PHP
        'rhel6-ss-fcgi-application-academicportal',
        'rhel6-ss-fcgi-application-accountactivation',
        'rhel6-ss-fcgi-application-drupaltest',
        # Python
        'rhel6-ss-application-hgitsss',
        'rhel6-ss-application-hgitsusg',
    ]

    define setupApp() {
        notice "requesting ${name}"
        include "${name}"
    }

    if ($skip_rhel6_dev_control != "true") {
        include rhel6-ss-application
        include rhel6-ss-util
        include rhel6-ss-util-homedirs
        include rhel6-ss-service-nginx
        include rhel6-ss-service-fastcgi
        include rhel6-ss-platform-python
        include rhel6-ss-platform-php
    }

    if ($domain == 'vm.test' and $hostname =~ /^(\w+)-(\w+)$/) {
        $app = $1
        if (!$("rhel6-ss-fcgi-application-$app" in $apps) and (!$("rhel6-ss-application-$app" in $apps) {
            notice "No app match VM name, setup all applications"
            setupApp ( $apps :)
        } else {
            if (!$("rhel6-ss-fcgi-application-$app" in $apps) {
                setupApp ( ["rhel6-ss-fcgi-application-$app"] :)
            }
            if (!$("rhel6-ss-application-$app" in $apps) {
                setupApp ( ["rhel6-ss-application-$app"] :)
            }
        } else {
            notice "General setup for all applications"
            setupApp ( $apps :)
        }
    }

    fin

naming conventions

• Production bits code so they clash with the clean examples.
• Everything new starts with “rhel6”
• Application naming:
  • fcgi are PHP apps;
  • things ending in hg* are Mercurial repo apps;
  • everything else is Java.
• Still needs attention and is still evolving to suite.

Execution Order

• include location matters:
  • variables defined below an include, which should use them, notoriously do not work;
  • dependencies also fail;
• have not tried this extensively in 2.7.x
A better example


- **Highlights:**
  - “unless” parameter
  - great use of define
  - fail (function call)

- see [http://docs.puppetlabs.com/references/2.6.8/function.html](http://docs.puppetlabs.com/references/2.6.8/function.html)

---

classes/selinux.class

- in /etc/puppet/manifests create classes/selinux.class

class selinux {
    define fcontext($context, $pathname) {
        if ($context == "") or ($pathname == "") {
            fail ("Context and Pathname must not be empty")
        }

        $semf_cmd = "/usr/sbin/semanage fcontext"

        exec {
            "add $context $pathname":
                command => "$semf_cmd -a -t $context "$pathname"",
                unless => "$semf_cmd -l | /bin/grep "^$pathname.*: $context:""
        }
    }
}
import "nodes/*.node"
import "classes/*.class"

old auto_replicate_puppet

exec {
  "dev puppetmodules":
  command => "/usr/sbin/semange fcontext -a -t puppet_modules_t /opt/dev/puppet-modules\(/.*\)?",
  cwd => "/",
  unless => "/usr/sbin/semange fcontext -l | grep '/opt/dev/puppet-modules'"

  "dev puppetmodules real location":
  command => "/usr/sbin/semange fcontext -a -t puppet_modules_t /var/root-opt/dev/puppet-modules\(/.*\)?",
  cwd => "/",
  unless => "/usr/sbin/semange fcontext -l | grep '/var/root-opt/dev/puppet-modules'"

  "test puppetmodules":
  command => "/usr/sbin/semange fcontext -a -t puppet_modules_t /opt/test/puppet-modules\(/.*\)?",
  cwd => "/",
  unless => "/usr/sbin/semange fcontext -l | grep '/opt/test/puppet-modules'"

  "test puppetmodules real location":
  command => "/usr/sbin/semange fcontext -a -t puppet_modules_t /var/root-opt/test/puppet-modules\(/.*\)?",
  cwd => "/",
  unless => "/usr/sbin/semange fcontext -l | grep '/var/root-opt/test/puppet-modules'"
}
new auto_replicate_puppet

```ruby
selinux::fcontext {
  "dev puppetmodules":
    context => "puppet_etc_t",
    pathname => "/opt/dev/puppet-modules(/.*)?";

  "dev puppetmodules real location":
    context => "puppet_etc_t",
    pathname => "/var/root-opt/dev/puppet-modules(/.*)?";

  "test puppetmodules":
    context => "puppet_etc_t",
    pathname => "/opt/test/puppet-modules(/.*)?";

  "test puppetmodules real location":
    context => "puppet_etc_t",
    pathname => "/var/root-opt/test/puppet-modules(/.*)?";
}
```

chaining updated

- **old:**

  File["/opt/dev"] -> File["/opt/dev/puppet-modules"] -> Exec["dev puppetmodules"] -> Exec["dev puppetmodules real location"]

  File["/opt/test"] -> File["/opt/test/puppet-modules"] -> Exec["test puppetmodules"] -> Exec["test puppetmodules real location"]

- **new:**

  File["/opt/dev"] -> File["/opt/dev/puppet-modules"] ->
  Selinux::Fcontext["dev puppetmodules"] -> Selinux::Fcontext["dev puppetmodules real location"]

  File["/opt/test"] -> File["/opt/test/puppet-modules"] ->
  Selinux::Fcontext["test puppetmodules"] -> Selinux::Fcontext["test puppetmodules real location"]
deployed

[root@c6pagent ~]# semanage fcontext -l | grep puppet-modules
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /etc/puppet/modules/custom/lib/facter/rh_release.rb
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '1337245374'
notice: /Stage[main]/Execute/Exec[echo top into /tmp/puppet.top]/returns: executed successfully
notice: /Stage[main]/Auto_replicate_puppet/Selinux::Fcontext[test puppetmodules]/Exec[add puppet_etc_t /opt/test/puppet-modules(/.*)?]/returns: executed successfully
notice: /Stage[main]/Auto_replicate_puppet/Selinux::Fcontext[test puppetmodules real location]/Exec[add puppet_etc_t /var/root-opt/test/puppet-modules(/.*)?]/returns: executed successfully
notice: /Stage[main]/Execute/Exec[touch a file just once]/returns: executed successfully
notice: /Stage[main]/Auto_replicate_puppet/Selinux::Fcontext[dev puppetmodules]/Exec[add puppet_etc_t /opt/dev/puppet-modules(/.*)?]/returns: executed successfully
notice: /Stage[main]/Auto_replicate_puppet/Selinux::Fcontext[dev puppetmodules real location]/Exec[add puppet_etc_t /var/root-opt/dev/puppet-modules(/.*)?]/returns: executed successfully
notice: Finished catalog run in 33.98 seconds

tested

[root@c6pagent ~]# semanage fcontext -l | grep puppet-modules
/opt/dev/puppet-modules(/.*)? all files
system_u:object_r:puppet_etc_t:s0
/opt/test/puppet-modules(/.*)? all files
system_u:object_r:puppet_etc_t:s0
/var/root-opt/dev/puppet-modules(/.*)? all files
system_u:object_r:puppet_etc_t:s0
/var/root-opt/test/puppet-modules(/.*)? all files
system_u:object_r:puppet_etc_t:s0
[root@c6pagent ~]#
fail

• fail acts only on empty $context or $pathname:

[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /etc/puppet/modules/custom/lib/facter/rh_release.rb
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
err: Could not retrieve catalog from remote server: Error 400 on SERVER: Context and Pathname must not be empty at /etc/puppet/manifests/classes/selinux.class:5 on node c6pagent.example.org
warning: Not using cache on failed catalog
err: Could not retrieve catalog; skipping run
[root@c6pagent ~]#

• omitting either will cause different (built in) error.

Passenger

• see http://projects.puppetlabs.com/projects/1/wiki/Using_Passenger

• Crash course to follow ...
Prepare for Passenger

- Update the puppet.conf

```bash
[puppetmasterd]
ssl_client_header = SSL_CLIENT_S_DN
ssl_client_verify_header = SSL_CLIENT_VERIFY
```

- Install a bunch of packages

```bash
[root@c6pmaster ~]# yum install httpd httpd-devel ruby-devel rubygems gcc mod_ssl
## ...
[root@c6pmaster ~]# yum install mod_passenger
## ...
```

Segue mod_passenger

- Need to mirror a new repo;
  
  - create `/etc/yum.repos.d/passenger.reposync`

```bash
[root@c6repo etc]# cat yum.repos.d/passenger.reposync
[passenger-x86_64]
name=Passenger repository for EL6
baseurl=http://passenger.stealthymonkeys.com/rhel/6/$basearch
enabled=1
check=1
[root@c6repo etc]#
```

- mirror the repository (cronjob)

```bash
15 3 * * * root reposync -n -c /etc/yum.repos.d/passenger.reposync -p /var/www/mrepo/passenger -a x86_64 -r passenger-x86_64 &&
createrepo /var/www/mrepo/passenger/passenger-x86_64
```

- update LocalMirror.repo on the client.
rack.conf and config.ru

• deploy and update rack.conf and config.ru

```bash
[root@c6pmaster ~]# cp /usr/share/puppet/ext/rack/files/apache2.conf /etc/httpd/conf.d/rack.conf
[root@c6pmaster ~]# vi /etc/httpd/conf.d/rack.conf
[root@c6pmaster ~]# mkdir -p /etc/puppet/rack/public
[root@c6pmaster ~]# mkdir -p /etc/puppet/rack/tmp
[root@c6pmaster ~]# cp /usr/share/puppet/ext/rack/files/config.ru /etc/puppet/rack
[root@c6pmaster ~]# chown puppet /etc/puppet/rack/config.ru
```

• restart httpd

```apache
PassengerHighPerformance on
PassengerMaxPoolSize 12
PassengerPoolIdleTime 1500
PassengerStatThrottleRate 120
RackAutoDetect Off
RailsAutoDetect Off
Listen 8140

<VirtualHost *:8140>
  SSLEngine on
  SSLProtocol -ALL +SSLv3 +TLSv1
  SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:-LOW:-SSLv2:-EXP
  SSLCertificateFile /var/lib/puppet/ssl/certs/c6pmaster.example.org.pem
  SSLCertificateKeyFile /var/lib/puppet/ssl/private_keys/c6pmaster.example.org.pem
  SSLCertificateChainFile /var/lib/puppet/ssl/ca/ca_crt.pem
  SSLCACertificateFile /var/lib/puppet/ssl/ca/ca_crt.pem
  SSLCARevocationFile /var/lib/puppet/ssl/ca/ca_crl.pem
  SSLVerifyClient optional
  SSLVerifyDepth 1
  SSLOptions +StdEnvVars
  RequestHeader set X-SSL-Subject %{SSL_CLIENT_S_DN}e
  RequestHeader set X-Client-DN %{SSL_CLIENT_S_DN}e
  RequestHeader set X-Client-Verify %{SSL_CLIENT_VERIFY}e
  DocumentRoot /etc/puppet/rack/public/
  RackBaseURI /
  <Directory /etc/puppet/rack/>
  Options None
  AllowOverride None
  Order allow,deny
  allow from all
  </Directory>
</VirtualHost>
```
SELinux

- `semodule -i /usr/share/selinux/packages/rubygem-passenger/rubygem-passenger.pp`
- `touch ~/.autorelabel ; reboot`

```
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
err: /File[/var/lib/puppet/lib]: Failed to generate additional resources using 'eval_generate: Error 500 on SERVER: <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
title>500 Internal Server Error</title>
</head>
<body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or misconfiguration and was unable to complete your request.</p>
<p>Please contact the server administrator, root@localhost and inform them of the time the error occurred, and anything you might have done that may have caused the error.</p>
<p>More information about this error may be available in the server error log.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at c6pmaster.example.org Port 8140</address>
</body></html>
```

- `no, not SELinux`

- `error on 2.7.12-1 ; downgraded to 2.7.11-2`
- `restart httpd and:

```
[root@c6pagent ~]# puppetd -vt
info: Retrieving plugin
info: Loading facts in /etc/puppet/modules/custom/lib/facter/rh_release.rb
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb
info: Caching catalog for c6pagent.example.org
info: Applying configuration version '1334677530'
notice: /Stage[main]/Execute/Exec[echo top into /tmp/puppet.top]/returns: executed successfully
notice: Finished catalog run in 36.32 seconds
[root@c6pagent ~]#
```
maniacal laughter

• standard SElinux troubleshooting
• follow sealert tickets in /var/log/messages

[root@c6pmaster ~]# cat ruby_puppet.te
module ruby_puppet 1.0.9;
require {
    type httpd_t;
    type puppet_var_run_t;
    type puppet_var_lib_t;
    class file { write rename create unlink setattr };  
    class dir { search read create write getattr rmdir remove_name add_name };  
}

#============= httpd_t ==============
allow httpd_t puppet_var_lib_t:dir read;
allow httpd_t puppet_var_lib_t:dir { write remove_name create add_name rmdir }; 
allow httpd_t puppet_var_lib_t:file { write rename create unlink setattr }; 
allow httpd_t puppet_var_run_t:dir { search getattr }; 

[root@c6pmaster ~]
checkmodule -M -m -o ruby_puppet.mod ruby_puppet.te 
checkmodule:  loading policy configuration from ruby_puppet.te 
checkmodule:  policy configuration loaded 
checkmodule:  writing binary representation (version 10) to ruby_puppet.mod 
[root@c6pmaster ~]
semodule_package -o ruby_puppet.pp -m ruby_puppet.mod 
[root@c6pmaster ~]
semodule -i ruby_puppet.pp 

test entire process

• signing a new agent and standard run:

[root@c6pagent ~]
 puppetd -vt --server=c6pmaster.example.org
info: Retrieving plugin 
notice: /File[/var/lib/puppet/lib/facter]/ensure: created 
notice: /File[/var/lib/puppet/lib/facter/rh_release.rb]/ensure: defined content as 
'(mds)c872f6c6d50139da8034661183d7e1b1' 
info: Loading downloaded plugin /var/lib/puppet/lib/facter/rh_release.rb 
info: Loading facts in /etc/puppet/modules/custom/lib/facter/rh_release.rb 
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb 
info: Caching catalog for c6pagent.example.org 
info: Applying configuration version '1334679591' 
notice: /Stage[main]/Execute/Exec[echo top into /tmp/puppet.top]/returns: executed successfully 
notice: /Stage[main]/Puppet_conf/Service[puppet]/ensure: ensure changed 'stopped' to 'running' 
info: Creating state file /var/lib/puppet/state/state.yaml 
notice: Finished catalog run in 9.24 seconds 
[root@c6pagent ~]
puppetd -vt 
info: Retrieving plugin 
info: Loading facts in /etc/puppet/modules/custom/lib/facter/rh_release.rb 
info: Loading facts in /var/lib/puppet/lib/facter/rh_release.rb 
info: Caching catalog for c6pagent.example.org 
info: Applying configuration version '1334679591' 
notice: /Stage[main]/Execute/Exec[echo top into /tmp/puppet.top]/returns: executed successfully 
notice: Finished catalog run in 5.67 seconds 
[root@c6pagent ~]#
Choices

- clearly you can keep SELinux on:
  
  ```
  [root@c6pmaster ~]# getenforce
  Enforcing
  ```

- it is a bit of effort;

- ultimately worth it.

- see also: [http://wiki.centos.org/HowTos/SELinux](http://wiki.centos.org/HowTos/SELinux)

restop

- Some config changes occasionally do not get picked up;

- Problems with Puppet configuration do not prevent httpd from working. Starting puppetmaster can provide insight into what’s wrong.

- service httpd stop; service puppetmaster start;
  service puppetmaster stop; service httpd start
Conclusion

- Many different ways to do everything covered;
- Remember everyone’s expertise;
  - Sys Admin’s built the SOE;
  - Developers build on it;
- Everyone needs to be happy;
  - Achieved through honest communication and co-operation.