How Puppet fits into your existing architecture

2011-07-23

Melbourne, AU
Devops Down Under

Garrett Honeycutt

Professional Services Consultant

garrett@puppetlabs.com

http://linkedin.com/in/garretthoneycutt

puppet labs

We are hiring

- Professional Services
- QA Engineer
- Core Developer

puppet labs

PuppetConf

9/19 - 23 in PDX

http://puppetconf.com

- Facebook
- Zynga
- DTO Solutions
- Google
- Eucalyptus

puppet labs

Provisioning

Goals:

- One button deploys
- Quick and easy re-provisioning
- No upgrades - just build new systems
    - solves issue of intermediate states

puppet labs

Provisioning

Start from a known base!

- Use the same base install for all nodes
- Smallest footprint of what it means to be a node on your network
- Allows for easier redeploys on other systems (VMWare, bare metal, EC2, Rackspace, Vagrant, etc)

Provisioning

PXE

- Provisions VM's and Physical systems the same way

Cloudy API's

- May not be an option if you have physical hardware

Provisioning

Cobbler

- My favorite provisioning system for PXE
- Handles tftp/dhcp/dns/repo's
- Namely for RedHat-ish systems, also supports Solaris, Debian, and images (ie: memtest, windows, firmware upgrades, etc)

http://github.com/ghoneycutt/puppet-cobbler

Provisioning

Puppet CloudPack

- Provision EC2 systems (others forthcoming)
- Uses fog

Provisioning

Chicken and Egg with Software Repo's

- ensure any modules that use custom repo's include your repo module
- Create repo's at provisioning stage
- Both preferred over run stages for simplicity and portability in modules

Provisioning

Certificate management

- autosigning can be your friend
- pre-generate certs
- gencert.php - uses reverse DNS

https://github.com/ghoneycutt/puppet-puppet/blob/master/files/gencert.php

External Node Classifier

Puppet Dashboard

- source of truth for list of nodes
- Add/Remove hosts through API – ties into provisioning

Package Management

Run your own Software Repositories

- You control when package versions change
- Packages are not mysteriously missing
- Much faster provisioning

puppet labs

Package Management

Version control your repositories

- Does not mean you need to use a VCS
- /data/repos/CentOS_5.5_Base symlink to /data/repos/CentOS_5.5_Base-2011062700
- Use hardlink(1) to deal with duplicate files

puppet labs

Package Management

package {}

- ensure => present or absent
- no version #’s

puppet labs

Package Management

no package { 'foo': ensure => latest }

- not so homogeneous clusters while systems converge
- ideally upgrades happen with rebuilds
- upgrades are triggered en masse during a maintenance window
- increases puppet run times due to an extra query to the packing system

Account Management

Use a directory service

- LDAP
- Active Directory

Account Management

Role based access control

- Groups get access, NOT users
- Who is in what team can be delegated to HR/ management

Account Management

/etc/security/access.conf

- controls groups that may access the system

http://github.com/ghoneycutt/puppet-pam

Account Management

List users as virtual resources sorted by UID and realize as necessary

```
@common::mkuser { ‘apachehup’:
    uid         => ‘32001’,
    gid         => ‘32001’,
    home        => ‘/home/apachehup’,
    managehome => true,
    comment     => ‘Apache Restart User’,
    dotssh      => true,
  }
```

http://github.com/ghoneycutt/puppet-generic

Data storage

Data?

- information that your node serves or creates

Data storage

Keep data stored off node

- SAN / NAS / Cloudy store / bit torrent
- rebuilt machines reconnect to your data

Disposable Architecture

http://www.linkedin.com/in/ericheydrick

I just lost a system.. big deal.

Failure is going to happen, let it.

puppet labs

Disposable Architecture

Develop other metrics to determine system health

- not how many systems are alive
- response times
- % of anticipated capacity

puppet labs

Auto-scaling

Tying it together

- (de)provision based on metrics
    - capacity, response, etc

Change Management with Puppet

...and now a different direction

puppet labs

What?

Change – “an event that results in a new status of one or more configuration items”[1]

[1] – http://en.wikipedia.org/wiki/Information_Technology_Infrastructure_Library#Change_Management

Why?

Environments are the same!

Dev == QA == Staging == ... == PROD

Why?

Compliance with Change Management policies

- CAB – Change Approval/Advisory Board
- Different environments have different criteria for passing to the next one

Different Environments

Puppet Test Area -> Dev -> QA -> Prod

Each environment has different teams and sometimes conflicting goals

Gate Examples

Puppet Test Area -> Dev

- Dev's agree/know of change

Dev -> QA

- Dev's have completed and self tested

QA -> Prod

- QA team has verified systems
- Ops is ready (has runbooks, monitoring setup, ...)

Documentation and Policies

Understand your environments

- What are they?
- What is their order of precedence?
- What are their SLA’s?
- Who owns them?

Documentation and Policies

Understand gating factors for change

- What are the gates between each environment?
- Who approves them?
- In what forum are they approved?

VCS Structure (SVN view)

```
├── branches
|   ├── 644
|   |   ├── manifests
|   |   |   └── site.pp
|   |   └── modules
|   |       ├── apache
|   |       └── zenoss
|   └── 755
|       ├── manifests
|       |   └── site.pp
|       └── modules
|           ├── apache
|           └── zenoss
├── tags
|   ├── 2011041100
|   ├── 2011041101
|   └── 2011041200
└── trunk
    ├── manifests
    |   └── site.pp
    └── modules
        ├── apache
        └── zenoss
```

puppet labs

VCS Structure (git view)

same as SVN except

- you do **not** have separate directories for
  - trunk
  - branches
  - tags

VCS Structure

trunk / master

- New code that is the best known **working code**
- but still not very well tested ...

puppet labs

VCS Structure

branches

- short lived
- use topical branches!
- associate branches with ticket numbers, so you can leverage your ticketing system to capture who is requesting changes and why
- avoid assigning branches to people as they tend to be long lived

VCS Structure

tags

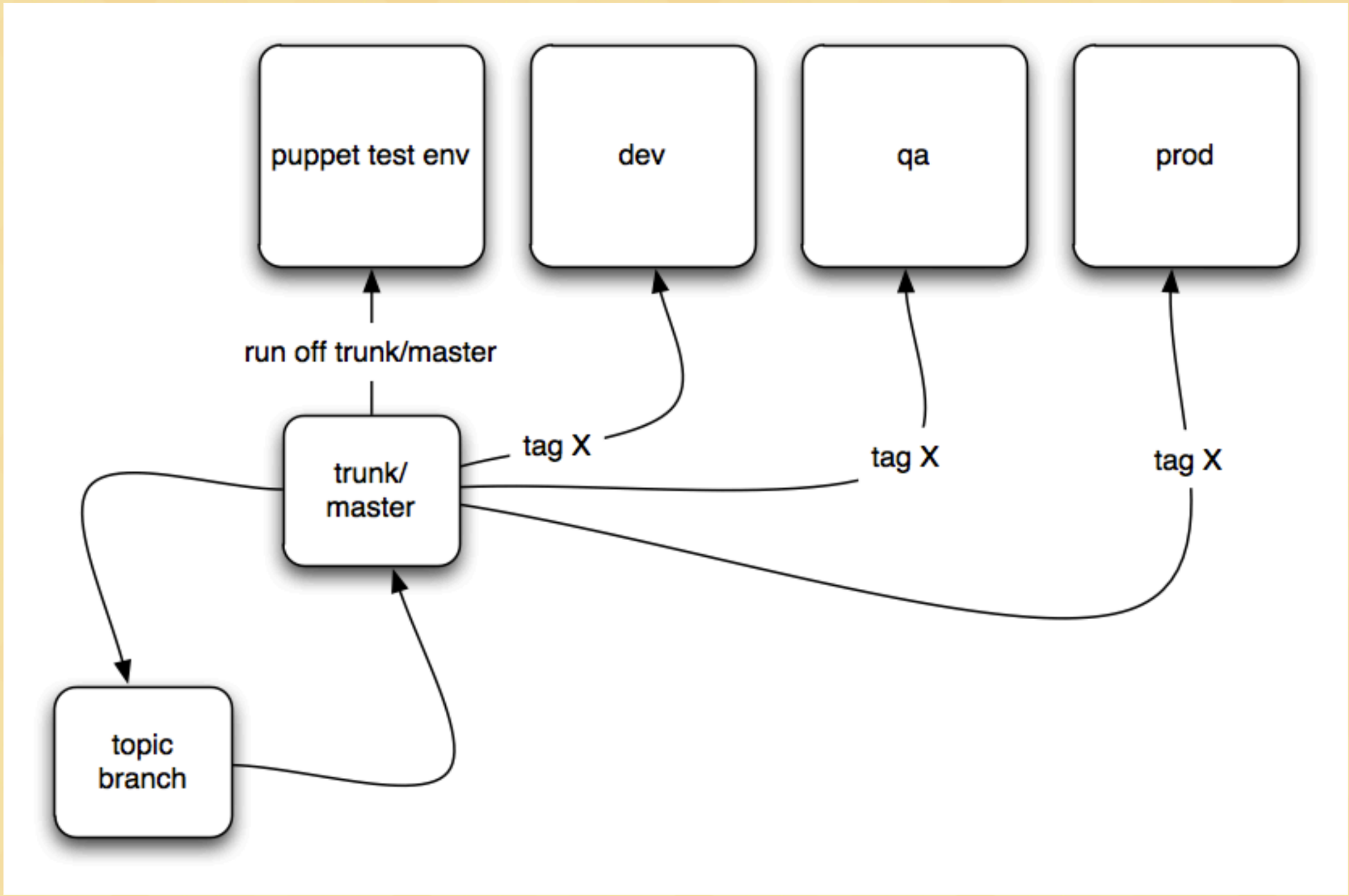
- immutable (even if you can technically make changes)
- found that BIND style serials work quite well for naming tags
- 2011041300 would be the first tag on April 13th, 2011.

puppet labs

Flow

- Change request comes in (from your ticket system)
- You create a branch from trunk/master that corresponds with the request
- Make changes to the branch
- Merge the branch back into trunk/master
- test against trunk/master
- create a tag
- associate that tag with the next environment all the way through to Prod

Flow



Oops, we found a bug

- tags are immutable, remember?
- create a brand new tag off of trunk/master
- start the process from the beginning
- short-cuts are more expensive

Release Management

Multiple people making changes?

- Who is responsible for merging from branches into trunk/master
- Release manager -- if they are more adept than team
- +1 if everyone is at a similar level

Release Management

Multiple teams exchanging code?

- Investigate using multiple module paths
- Communication!
- private github – can facilitate cooperation

puppet labs

Mailing List of changes

Create a mailing list for all changes

- You can always ignore it
- reach out to those writing poor code before they ask you to merge it into trunk
- svnmailer is great

Testing trunk/master

Create at least one representative system for each different type of system you model

- Run these systems off the code in trunk/master
- Before cutting a tag, rebuild all these systems from scratch
  - further tests that relationships between resources are working
  - proves you can actually provision a system from scratch

Approaches to testing branches

- Puppet's understanding of environments is good for this
- Setup a different Puppet master per branch
- Do not rely on a puppet master at all -- use puppet apply and test locally

How Puppet fits into your existing architecture

2011-07-23

Melbourne, AU
Devops Down Under

Garrett Honeycutt

Professional Services Consultant

garrett@puppetlabs.com

http://linkedin.com/in/garretthoneycutt

puppet labs