Our Puppet Story

Martin Schütte

DECK36

LINUX TAG
May 5 2014
About DECK36

• Small team of 7 engineers
• Longstanding expertise in designing, implementing and operating complex web systems
• Developing own data intelligence-focused tools and web services
• Offering our expert knowledge in Automation & Operation, Architecture & Engineering, Analytics & Data Logistics
Common Problem

Jason Antman
@j_antman

We have the word "iff". Can we start using "inn", as in "this works inn the test environment"?

12:48 PM - 15 Nov 2013

Zvi 'Viz' Effron @CtrlZvi
@j_antman @tom_forsyth And it's better known cousin "onn." As in, "Works onn my machine."
“We hired someone. How can we reproduce our dev environment?”
Vagrant
Vagrant

Configuration tool for VMs and Provisioning.

“Local cloud”
- Self service
- Instant provisioning
- Cost efficient
- Elastic
- Pay per use
Vagrant

VM Providers:
- VirtualBox: “default”, works offline, resource hungry
- Docker: lightweight, requires Linux, good for testing
- AWS EC2: remote VMs, good for automation (Jenkins)
- 3rd party plugins for KVM, libvirt, ESXI, …

Provisioning:
- Shell script
- Puppet, apply manifest or run agent
- Chef, solo or client
- Ansible playbooks
- Docker containers
“Synced folders are too slow.”
Synced Folders

Shared folders, mounted from host into guest.

Options:

• VirtualBox slow!
• NFS often the best choice
• SMB for Windows support
• rsync new and promising
“But our QA needs many VMs and their machines are slow.”
vagrant-aws

Vagrant.configure("2") do |config|
  config.vm.box = "dummy"

  config.vm.provider :aws do |aws, override|
    aws.access_key_id = "YOUR KEY"
    # ...

    region = "eu-west-1"
    aws.ami = "ami-20414854"

    aws.tags = {
      'Role' => 'TestVM',
      'Net' => 'Devnet'
    }
  end
end
“How can we configure all those VMs?”
Puppet
Puppet

- Configuration Management
- Declarative: Resources and Dependencies
“How should we manage write access for multiple Ops/DevOps?”
git workflows

- use git!
- use git hooks
- use per-user environments for easy testing
- repos for testing/production
git hook: Syntax Check

Git pre-commit hook with puppet-lint to syntax check Puppet, ERB templates, YAML files (http://github.com/gini/puppet-git-hooks)

Example Output:

$ git commit -m 'test' modules/graylog2/templates/server.conf.erb
-:5: syntax error, unexpected $undefined
...rd_sha2 = "\n; _erbout.concat(( @ root_pwd_sha2 ).to_s); _erbo...
...
ERB syntax error in modules/graylog2/templates/server.conf.erb
environments

- per user env + production
  - easy testing with `puppet agent -t --environment=user`
- two servers for testing/production

Config File Environments:

```
[mschuette]
modulepath = $confdir/environments/mschuette/modules
manifest = $confdir/environments/mschuette/manifests/site.pp
pluginsync = true
```

Directory Environments (Puppet >= 3.5.0):

```
[main]
environmentpath = $confdir/environments
```
environments

Dev/Test

user1

user2

user3

...

dev-master

Prod

prod-master
“But we cannot write and maintain all those modules.”
“How do we use inventory and EC2 metadata in Puppet manifests?”
Facter

Gather information from system.

- standard values
- extensible via Puppet plugins

Example Output:

```
# facter -p
architecture => i386
operatingsystem => CentOS
operatingsystemrelease => 5.5
...
ipaddress => 172.16.182.129
...```
stdlib facts.d

- puppetlabs-stdlib reads facts from /etc/facter/facts.d
- simple data inputs
- e.g. ec2metadata, inventory lookup

-------------------------- custom_facts.sh --------------------------

```
#!/bin/sh

which ec2metadata >/dev/null 2>&1 || exit 1

echo "ec2_ami_id=$(ec2metadata --ami-id)"
echo "ec2_instance_id=$(ec2metadata --instance-id)"
echo "ec2_instance_type=$(ec2metadata --instance-type)"
echo "ec2_public_ipv4=$(ec2metadata --public-ipv4)"
echo "ec2_public_hostname=$(ec2metadata --public-hostname)"
```
“There has to be a way to split modules and config parameters.”
Hiera
Hiera

- banish top scope variables
- use Hiera!
- structure with roles & profiles
Without Hiera (Puppet 2.x legacy code)

node "mydev\d+.vagrantup.com" inherits basenode-vagrant {
  $vmEnv = "development"
  include sysadmin
  include ntp

  if $::fqdn = "mydev01.vagrantup.com" {
    class { 'vpn':
      version => latest,
      ca_crt => '....',
      usr_crt => '....',
      usr_key => '....',
    }
  } else {
    class { 'vpn':
      version => "2.3.2-7~bpo70+1",
      ca_crt => '....',
      usr_crt => '....',
      usr_key => '....',
    }
  }
}

# ...
Explicit Hiera Usage

$vpn_version = \text{hiera}('vpn\_version', 'latest')$
$vpn\_ca\_crt = \text{hiera}('vpn\_ca\_crt')$
$vpn\_usr\_crt = \text{hiera}('vpn\_usr\_crt')$
$vpn\_usr\_key = \text{hiera}('vpn\_usr\_key')$

class { 'vpn':
  version => $vpn\_version,$
  ca\_crt => $vpn\_ca\_crt,$
  usr\_crt => $vpn\_usr\_crt,$
  usr\_key => $vpn\_usr\_key,$
}
class vpn($version = hiera('vpn::version', 'present'),
    $ca_crt = hiera('vpn::ca_crt'),
    $usr_crt = hiera('vpn::usr_crt'),
    $usr_key = hiera('vpn::usr_key')) {

    package {
        'openvpn':
            ensure => $version;
    }
    # ...
}

class { 'vpn': }
# or "include vpn"
Puppet 3.x with Hiera

```yaml
site.pp
hiera_include('include_classes', ['sysadmin'])

node default {
}

profile_vpn.yaml
include_classes:
  - ntp
  - vpn

vpn::version: present
vpn::ca_crt: ...
vpn::usr_crt: ...
vpn::usr_key: ...
```
“Our modules and manifests grow too complex. How can we structure them?”
Module Design Pattern: Roles & Profiles

Roles: Business Logic

Profiles: Implementation

Components: Resource modelling

Resources

from: Craig Dunn, Advanced Puppet Design
“What other pitfalls will we encounter?”
Puppet Problems

• some tasks require two agent runs
• apt-get upgrade and package dependencies
• version mismatch between apt (or yum) and package
• scoping and namespaces
• exec is the new eval
Namespace problems

# this does not work, cf. #PUP-1073

package { 'memcached':
    ensure => present,
    provider => apt,
}

package { 'memcached':
    ensure => present,
    provider => gem,
}
exec tricks

Both source and solution to a great many problems. You can do (and break) everything with exec and a shell script.

But of course you should not.
exec tricks

# pkg name collision
exec { 'npm install -g less':
    creates => '/usr/lib/node_modules/npm/node_modules/less',
}

# abuse puppet as cron, and hide the change
exec { 'zabbix_update.sh':
    command => 'false',
    onlyif => "'/opt/zabbix_update.sh $api_url && false",
    logoutput => on_failure,
}
“How can we monitor Puppet changes?”
Integration
Puppet Dashboard

Daily run status
Number and status of runs during the last 30 days:

- 12 Unresponsive
- 1 Failed
- 0 Pending
- 3 Changed
- 9 Unchanged
- 0 Unreported

25 All

Export nodes as CSV

<table>
<thead>
<tr>
<th>Node</th>
<th>Latest report</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Failed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unchanged</td>
</tr>
<tr>
<td>------</td>
<td>---------------------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3813</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>410</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2065</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:42 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:34 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:33 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:20 UTC</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:20 UTC</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:19 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:19 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:17 UTC</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>2014-04-16 16:13 UTC</td>
<td>91</td>
</tr>
</tbody>
</table>
# External Monitoring

![Chart showing Puppetmaster stats over time](chart.png)

- **Puppet: unchanged**
  - (avg) 61 nodes, 0 nodes
  - avg 54.59 nodes, max 65 nodes

- **Puppet: pending**
  - (avg) 0 nodes, 0 nodes
  - avg 1 node, max 1 node

- **Puppet: changed**
  - (avg) 0 nodes, 0 nodes
  - avg 4.68 nodes, max 62 nodes

- **Puppet: unresponsive**
  - (avg) 6 nodes, 4 nodes
  - avg 9.88 nodes, max 19 nodes

- **Puppet: unreported**
  - (avg) 54 nodes, 33 nodes
  - avg 45.04 nodes, max 54 nodes

- **Puppet: failed**
  - (avg) 0 nodes, 0 nodes
  - avg 3.88 nodes, max 67 nodes
git hook: E-Mail Notification

Git post-receive hook to notify team on push
(http://git.kernel.org/cgit/git/git/git.git/tree/contrib/hooks/post-receive-email?id=HEAD)

Example E-Mail:

- Log

commit 5df04ee883b8de8a37bf0ac97eec068cd1f3a414
Author: N. N. <n.n@deck36.de>
Date: Tue Jan 7 08:57:17 2014 +0000

    fixed path to csync2 executable

Summary of changes:
    modules/user/files/etc/sudoers.d/support | 2 +-
    1 file changed, 1 insertion(+), 1 deletion(-)
“How do we coordinate a cluster restart?”
MCollective

“multissh deluxe”

AMQP client/server framework to
• orchestrate actions
• control puppet agents
• run commands
• query resources
• …

Alternatives: Ansible, serf, …
“Why do we still manually configure DNS and monitoring?”
Hooks to other systems

- include in provisioning process
- provide normative data as facts
- register or update DNS name → e.g. Route 53
- register or update host in Zabbix monitoring → API
Questions?

class presentation {
    package { 'questions':
        ensure => 'answered',
    } } 

Links:

- Vagrant
- Puppet Language: Visual Index
- Puppet Type Reference
- Puppet Ask
Thank You